IL MIO COMPUTER

CORSO PRATICO DI INFORMATICA: MICRO, MINI E PERSONAL COMPUTER



IL MIO
COMPUTER

IL MIO COMPUTER

CORSO PRATICO DI INFORMATICA: MICRO, MINI E PERSONAL COMPUTER

volume sesto

ISTITUTO GEOGRAFICO DE AGOSTINI

Direzione: Achille Boroli - Adolfo Boroli
Direzione editoriale: Mario Nilo; *settore fascicoli:* Jason Vella

Redazione dell'edizione italiana a cura di:
Edizioni Remo Sandron - Firenze
Redazione dell'edizione inglese:
Richard Pawson (*Editor*); David Tebbutt (*Consultant Editor*); Chris Cooper (*Executive Editor*); David Whelan (*Art Director*); Gareth Jefferson, Roger Ford (*Collaborators*)
Designed and Produced by Bunch Partworks Limited
Traduzione a cura di: Daniel Quinn

Coordinamento grafico: Otello Geddo
Coordinamento fotografico a cura del Centro Iconografico dell'Istituto Geografico De Agostini

Titolo originale dell'opera: The home computer advanced course
© Orbis Publishing Limited, Londra, 1983.
© Istituto Geografico De Agostini S.p.A., Novara, 1985.
Stampato in Italia - IGDA Officine Grafiche, Novara, 1985.

Sommario

1201 Percorso a ostacoli
1204 Creare i modelli
1206 Ricerca e sostituzione
1208 Database
1209 Periferiche a confronto
1212 Produrre suoni
1215 Lavorare sui numeri
1218 Una struttura sensata
1220 Deus ex machina
1221 Gli occhi del robot
1224 Variabili variate
1226 Una risposta... a tono
1229 La Snap
1232 Un piano d'azione
1234 Un po' di poesia
1237 Database
1238 Un utile debugger
1240 Flyerfox
1241 Due chiacchiere col robot
1244 Griglia d'acciaio
1246 Spirito d'avventura
1249 Database
1250 Il Beasty
1252 Perso nel labirinto
1254 Giocare in LOGO
1257 Bug sotto controllo
1260 Mugsy
1261 La conoscenza
1264 I Macro comandi del Lotus
1266 Di luogo in... LOGO
1269 Il Commodore 16
1272 Digitaya: i messaggi
1274 Database
1275 Un punto d'arresto
1278 Tiriamo le somme
1281 Simulare un robot
1284 Il TK!Solver
1286 Lo Spectrum+
1288 Fine di un'avventura
1290 Un semplice robot
1293 Digitaya: gli ambienti
1296 Database
1297 Ultimi comandi
1300 Starfinder
1301 Piccoli robot
1304 Indovinare ragionevolmente
1306 Accettare ordini
1309 Database
1310 La Touchmaster
1312 Chi è stato?
1315 Motore fuoribordo
1318 Ritocchi finali
1321 Bracci robotizzati
1324 Il software verticale
1326 Raccogliere oggetti
1329 L'Osborne Encore
1332 Database
1333 Procedure autogeneranti
1336 Riuniamo le parti
1338 Concetti operativi
1340 Summer games
1341 I robot nel futuro
1344 BrainStorm
1346 Abbandonare oggetti
1349 Il Compaq Plus
1352 Database
1353 Grafici in LOGO
1356 Micro elettronica
1358 Il sistema operativo del BBC
1361 Software di taglio europeo
1363 Spiriti e fantasmi
1366 Disegnare cicloidi
1369 Il Link 480Z
1372 Potenza di calcolo
1374 Taratura del robot
1377 Uso di vettori nel BBC
1381 Telematica per tutti
1384 Avventura grafica
1386 Software e didattica
1388 Database
1389 Foto computerizzate
1392 I servomotori
1395 Trasformazioni spaziali
1398 Schermi a volontà
1401 Comunicare tramite computer
1403 Usare i servomotori
1406 Didattica d'oltremanica
1408 Database
1409 Due Sharp da taschino
1412 Più schermi sullo Spectrum
1414 Altre trasformazioni
1417 Movimenti scorrevoli
1420 Software in vetrina
1421 Strumenti per comunicare
1423 Programmi per il robot
1426 Il package "Human Edge"
1428 Database
1429 Memotech RS 128
1432 Più schermi sul CBM64
1434 Fattoriali!
1437 Le chiamate OSBYTE del BBC
1440 Software in vetrina

Indice per settore

Panorami



1201 Percorso a ostacoli
1221 Gli occhi del robot
1241 Due chiacchiere col robot
1261 La conoscenza
1281 Simulare un robot
1301 Piccoli robot
1321 Bracci robotizzati
1341 I robot nel futuro
1361 Software di taglio europeo
1381 Telematica per tutti
1401 Comunicare tramite computer
1421 Strumenti per comunicare

Software



1204 Creare i modelli
1220 Deus ex machina
1232 Un piano d'azione
1240 Flyerfox
1244 Griglia d'acciaio
1260 Mugsy
1264 I Macro comandi del Lotus
1284 Il TK!Solver
1300 Starfinder
1304 Indovinare ragionevolmente
1324 Il software verticale
1340 Summer games
1344 BrainStorm
1372 Potenza di calcolo
1386 Software e didattica
1406 Didattica d'oltremanica
1420 Software in vetrina
1426 Il package "Human Edge"
1440 Software in vetrina

Programmazione BASIC

1206 Ricerca e sostituzione
1224 Variabili variate
1246 Spirito d'avventura
1272 Digitaya: i messaggi
1293 Digitaya: gli ambienti
1306 Accettare ordini
1326 Raccogliere oggetti
1346 Abbandonare oggetti
1363 Spiriti e fantasmi
1384 Avventura grafica
1412 Più schermi sullo Spectrum
1432 Più schermi sul CBM64

Linguaggio macchina

1208 Database
1218 Una struttura sensata
1237 Database
1238 Un utile debugger
1249 Database
1257 Bug sotto controllo
1274 Database
1275 Un punto d'arresto
1296 Database
1297 Ultimi comandi
1309 Database
1318 Ritocchi finali
1332 Database
1338 Concetti operativi
1352 Database
1358 Il sistema operativo del BBC
1377 Uso di vettori nel BBC
1388 Database
1398 Schermi a volontà
1408 Database
1417 Movimenti scorrevoli
1428 Database
1437 Le chiamate OSBYTE del BBC

Hardware

1209 Periferiche a confronto
1229 La Snap
1250 Il Beasty
1269 Il Commodore 16
1286 Lo Spectrum+
1310 La Touchmaster
1329 L'Osborne Encore
1349 Il Compaq Plus
1369 Il Link 480Z
1389 Foto computerizzate
1409 Due Sharp da taschino
1429 Memotech RS 128

Laboratorio

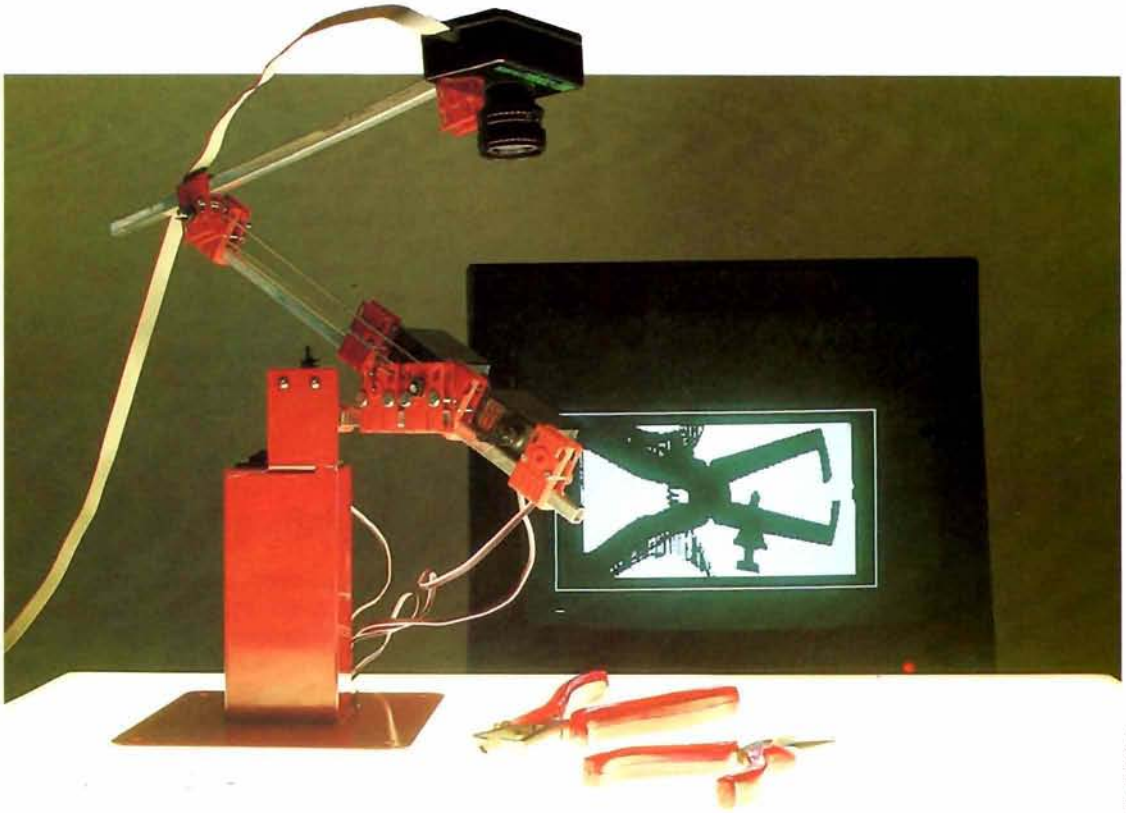
1212 Produrre suoni
1226 Una risposta... a tono
1252 Perso nel labirinto
1278 Tiriamo le somme
1290 Un semplice robot
1315 Motore fuoribordo
1336 Riuniamo le parti
1356 Micro elettronica
1374 Taratura del robot
1392 I servomotori
1403 Usare i servomotori
1423 Programmi per il robot

Note d'Informatica

1215 Lavorare sui numeri
1234 Un po' di poesia
1254 Giocare in LOGO
1266 Di luogo in... LOGO
1288 Fine di un'avventura
1312 Chi è stato?
1333 Procedure autogeneranti
1353 Grafici in LOGO
1366 Disegnare cicloidi
1395 Trasformazioni spaziali
1414 Altre trasformazioni
1434 Fattoriali!

Percorso a ostacoli

Considerazioni sul progetto di un robot capace di muoversi "intelligentemente"



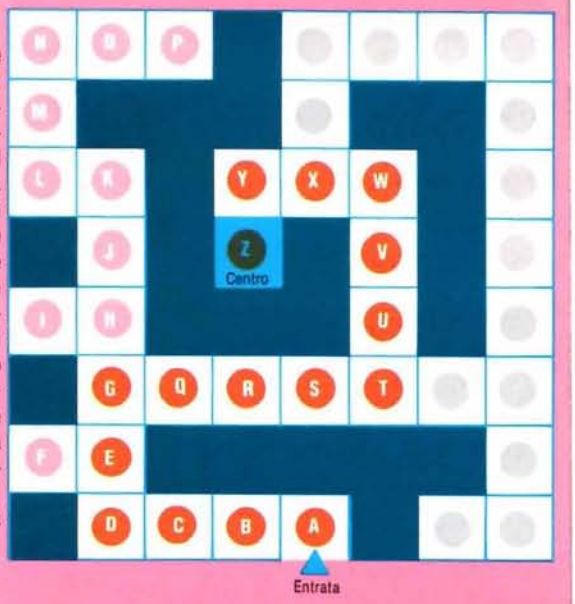
IAN McKINNELL

Apprendere immagini
Se dotati di microprocessori sufficientemente sofisticati, i robot possono "imparare" un repertorio di sagome di oggetti, da impiegare negli algoritmi di riconoscimento e abbinamento di immagini. Il braccio robotizzato "Beasty", qui illustrato, può essere corredato di una telecamera "Snap", che produce immagini digitali. Le informazioni così rilevate vengono poi elaborate dal software venduto assieme all'occhio del computer. Facendo "osservare" un particolare oggetto o attrezzo più volte, con diverse angolazioni, si hanno buone probabilità che il braccio lo sappia poi riconoscere in qualsiasi posizione

Per progettare un robot che si muova in modo "intelligente", bisogna, in primo luogo, escludere il controllo da parte dell'uomo, altrimenti viene meno uno dei requisiti fondamentali di un sistema automatizzato; esistono tuttavia campi (quali il disinnesco di ordigni bellici, ad esempio) in cui il controllo umano è di primaria importanza per l'assoluta precisione richiesta ai movimenti del robot.
Del resto, non ci si può nemmeno limitare a comandare il robot ricorrendo ad una sequenza fissa di istruzioni immesse nel computer di controllo: il robot eseguirebbe una serie di operazioni predeterminate, indipendentemente dalle circostanze ambientali. Ciò non toglie che talvolta sia utile disporre di apparecchi impostati su tale principio: molti bracci robotizzati vengono considerati "intelligenti", sebbene si limitino ad effettuare azioni prestabilite.
La definizione di "robot intelligente" dovrebbe dunque applicarsi ad un dispositivo capace di svegliarci al mattino, servendoci una buona tazza di caffè. Un simile robot non deve essere direttamente controllato dall'uomo, poiché la sequenza di operazioni deve avvenire quando l'uomo sta ancora dormendo, né si può trattare di una sequenza di istruzioni fissa e pre-programmata: basterebbe cambiar letto o lasciare qualche indumento sul pavimento per causare inconvenienti non trascurabili.
Per riassumere, quindi, un robot può esser definito "intelligente" solo se è capace di muoversi nel proprio ambiente, senza esser controllato da un uomo e senza seguire ciecamente una serie predeterminata di istruzioni. Inoltre, nel movimento, il robot deve saper evitare eventuali ostacoli.
Una branca dell'intelligenza artificiale (AI) è dedicata allo studio di giochi, tramite i quali esaminare i problemi connessi al movimento. Infatti, così come il gioco degli scacchi è risultato estremamente utile per sviluppare ed approfondire altri settori dell'intelligenza artificiale, il comportamento di robot all'interno di labirinti serve per studiare movimenti robotizzati "intelligenti". Dalla fine degli anni '70 in poi sono diventate diffusissime le gare tra robot impegnati in labirinti di diversa difficoltà. Tali gare vengono generalmente chiamate "Micromouse contest". L'idea è molto semplice: viene costruito un labirinto

Semplice come l'ABC

In un labirinto funzionano anche algoritmi molto semplici: il robot prosegue su ogni casella "nuova", finché non raggiunge un vicolo cieco (nei punti F, I e P, ad esempio). A questo punto arretra fino all'ultimo bivio percorso (G nell'illustrazione), per scegliere una nuova direzione. Le caselle "inutili" vengono catalogate come tali nella memoria del computer. Se non esistono percorsi intentati, dopo essere arretrato fino a un bivio, il robot indietreggia fino al bivio precedente. Tale comportamento prosegue fino a tornare, eventualmente, alla casella iniziale. Se ciò si verifica, il labirinto è detto "cieco" o senza soluzione



USCIRE DAL LABIRINTO

```
 49 REM  ******* VERSIONE CBM 64 *******
 50 REM *              LABIRINTO          *
 51 REM  ****************************
100 GOSUB 2000          :REM INIZIALIZZA
150 GOSUB 9000          :REM VISUAL. LABIRINTO
200 FOR L=1 TO 1
250 GOSUB 7000          :REM ESPLORA
300 GOSUB 8000          :REM MUOVE
400 NEXT L
450 RI=16:CO=12:GOSUB 9500
500 IF MZ<>HM THEN PRINT "CIECO"
550 IF MZ=HM THEN PRINT "CENTRO"
900 PRINT:PRINT:INPUT A$:RUN
1999 REM ****************************
2000 REM *              INIZIALIZZA        *
2001 REM ****************************
2100 SZ=10:S2=SZ*SZ:GX=SZ/2:GY=SZ+2
2120 DIM MZ(SZ,SZ),R$(4),LX(4),LY(4)
2140 X=RND(-TI)
2150 DEFFNR(N)=INT(RND(1)*N+1)
2160 HM=-1:G0=-2:WL=42:W$=CHR$(WL)
2180 CL$=CHR$(147):H$=CHR$(19)
2200 X=GX-1:Y=GY-2:DR=3
2220 D$=CHR$(17):P$=D$
2240 FOR K=1 TO 5:P$=P$+P$:NEXT K:P$=H$+P$
2400 DATA 0,1,"O",-1,0,".",0,-1,"",1,0,""
2420 FOR K=1 TO 4:READ LX(K),LY(K),R$(K)
2490 NEXT K:RETURN
6999 REM ****************************
7000 REM *              ESPLORA            *
7001 REM ****************************
7110 RI=1:CO=1:GOSUB 9500
7120 L=0:ND=0:FOR S=1 TO 4
7140 NX=X+LX(S):NY=Y+LY(S)
7200 IF NX
7220 IF NY
7230 MZ=MZ(NX,NY):IF MZ=0 THEN ND=S
7260 IF MZ=HM THEN ND=S:S=4:L=1
7280 NEXT S:RETURN
7999 REM ****************************
8000 REM *              MUOVE              *
8001 REM ****************************
8100 MZ(X,Y)=DR:FL=1
8120 RI=Y+1:CO=X+1:GOSUB 9500
8140 IF ND=0 THEN ND=DR-2-4*(DR<3):FL=2
8160 PRINT R$(FL);:X=X+LX(ND):Y=Y+LY(ND)
8180 DR=ND:IF Y>SZ THEN L=1:RETURN
8200 IF FL=2 THEN DR=MZ(X,Y)
8490 RETURN
8999 REM ****************************
9000 REM *        VISUALIZZA IL LABIRINTO  *
9001 REM ****************************
9040 PRINT CL$;:RI=1:CO=1
9050 WL=42:W$=CHR$(WL)
9060 S$=" ":REM 22 SPAZI
9070 FOR K=1 TO SZ+2:T$=T$+W$:NEXT K
9080 E$=W$+LEFT$(S$,SZ)+W$
9100 PRINT T$:FOR J=2 TO SZ+1
9120 PRINT E$:NEXT J:PRINT T$
9140 FOR K=1 TO S2/2
9150 WX=FNR(SZ):WY=FNR(SZ)
9160 MZ(WX,WY)=WL:RI=WY+1:CO=WX+1
9200 GOSUB 9500:PRINT W$:NEXT K
9250 CO=1+FNR(SZ):RI=1+FNR(SZ):GOSUB 9500
9300 PRINT "C";:MZ(CO-1,RI-1)=HM
9350 RI=GY:CO=GX:GOSUB 9500:PRINT "*"
9490 RETURN
9499 REM ****************************
9500 REM *        POSIZIONA IL CURSORE     *
9501 REM ****************************
9600 PRINT LEFT$(P$,RI) TAB(CO-1);:RETURN
```

Basic & Basic

Modifiche per gli altri apparecchi:

Micro BBC

```
49 REM  ******** BBC ********
50 REM *          LABIRINTO          *
51 REM  ********************
9040 CLS:RI=1:CO=1
9600 PRINT TAB(CO-1,RI-1);:RETURN
```

Spectrum

```
49 REM  ******* SPECTRUM *******
50 REM *          LABIRINTO          *
51 REM  ********************
2120 DIM MZ(SZ,SZ):DIM R$(4)
2130 DIM LX(4):DIM LY(4)
2140 RANDOMIZE
2150 DEFFNR(N)=INT(RND*N+1)
9040 CLS:RI=1:CO=1
9600 PRINT AT (RI-1,CO-1);:RETURN
```

BRIAN MORRIS

di adeguate dimensioni e ciascun partecipante deve far correre il proprio robot all'interno di esso. Vince il robot che impiega il minor tempo nell'uscire dal labirinto o, come accade talvolta, nel trovarne la camera centrale. Come in ogni labirinto che si rispetti, alcuni corridoi possiedono più aperture, altri una sola: quella d'ingresso!

Inizialmente, a partecipare alle gare erano pochi appassionati (i cui apparecchi si comportavano spesso in modo erratico dopo appena qualche curva) ma la moda è andata dilagando a macchia d'olio. Uno dei "topi" (così vengono chiamati i piccoli robot) più conosciuti nell'ambiente dei "Micromouse contest" è stato "Stirling Mouse", creato da Nick Smith (l'assonanza col nome del celebre corridore automobilistico Stirling Moss non è casuale!). Dotato di sensori meccanici e di motori passo passo, questo topo è oggi superato da apparecchi molto più sofisticati; in una gara tenutasi nel 1984 a Madrid, il vincitore è riuscito a raggiungere il centro del labirinto in appena 31,4 secondi: un risultato apprezzabile, se si considera che alcuni partecipanti si sono "inchiodati" solo dopo qualche mossa.

Costruzione della mappa

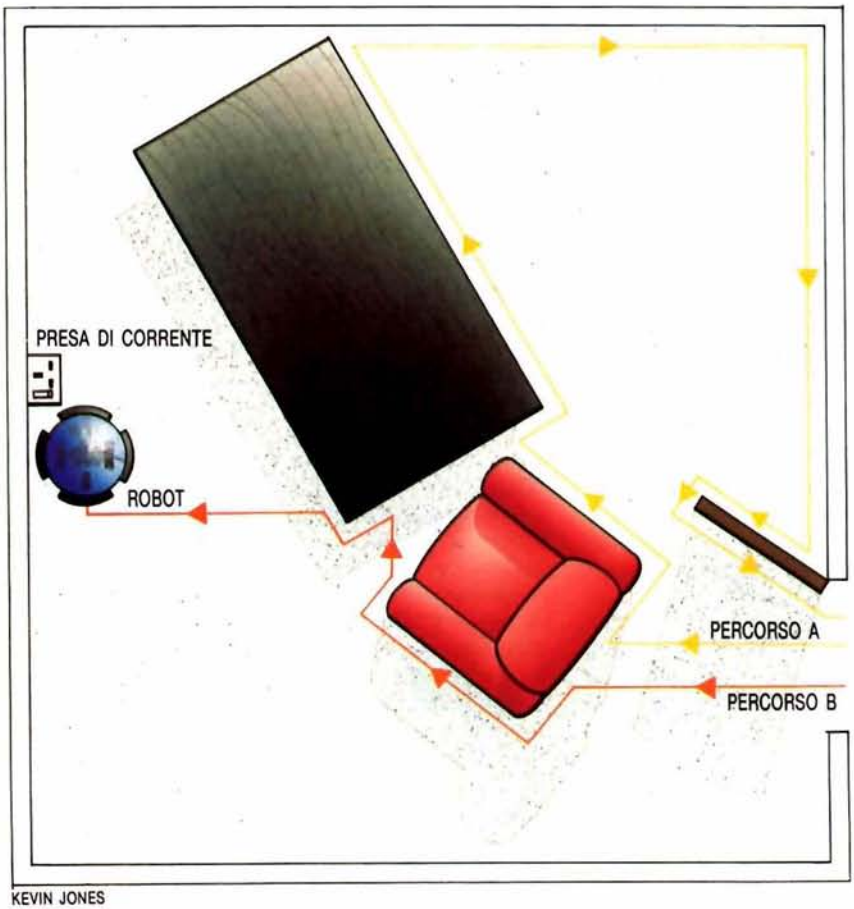
In generale, il topo-robot deve essere in grado di muoversi con precisione nel labirinto, e deve "conoscere" la propria posizione in qualsiasi momento. Ciò si può ottenere montando il robot su ruote, collegate a motori passo passo. Spesso si ricorre a qualche sistema di "feedback", quale un decodificatore assiale. Inoltre, il topo va dotato di sensori, in grado di verificare l'eventuale presenza di aperture o di pareti lungo il percorso e di un sistema capace di "costruire" la mappa del labirinto. A questo scopo, in gara, è consentito fare almeno un paio di giri "di prova". Poi si passa alla gara vera e propria, in cui il tentativo è cronometrato. Sebbene i criteri specifici siano differenti, da topo a topo, una semplice soluzione consiste nel dotarli di un elementare sensore meccanico, collocato sul "muso". Il sensore avverte la presenza di una parete di fronte a sé e, in tal caso, compie una rotazione di 90°, eseguendo una nuova verifica e via dicendo. Dopo un certo tempo, il topo riesce a "sapere" dove si trovano le aperture di ciascun corridoio del labirinto. Tali informazioni sono talvolta conservate sottoforma di numeri binari. Se, per esempio, 1111 e 0000 corrispondono rispettivamente a un corridoio senza uscite e ad uno senza pareti (entrambi i casi limite non si verificano nella realtà), allora 0111 significa un corridoio con una sola uscita (un vicolo cieco).

I dati possono essere conservati, in BASIC, in una matrice bidimensionale e si potrebbe usare DIM A(16,16) per dimensionare una matrice adatta ad un labirinto composto da 16 celle o corridoi. Il robot deve quindi calcolare il percorso adatto a portarlo nella posizione corrispondente al centro del labirinto: A(8,8)

per ipotesi. Spesso, il programma usato dal computer "di bordo" elabora una struttura "ad albero" per ogni possibile percorso nel labirinto. Molti dei rami condurranno ai soliti vicoli ciechi o a posizioni già occupate in precedenza: in tal caso l'intero ramo viene automaticamente accantonato. Infine, l'esame passa ai percorsi possibili, selezionando quello di minor lunghezza (minor numero di passaggi).
Ovviamente, la qualità e il numero di sensori accrescono le possibilità di successo: un semplice rilevatore di contatto obbliga il robot ad "urtare" ciascuna parete per accorgersi della sua presenza. Un rilevatore di prossimità evita tali scontri, mentre un sensore a onda riflessa può rilevare la presenza di una parete fin dall'imbocco di un corridoio. Se, anziché un solo sensore, ne vengono usati quattro, i tempi si riducono notevolmente, non essendo più necessario far ruotare il topo di volta in volta.

In giro per casa

Per molti aspetti, il problema di far muovere intelligentemente un robot in una normale abitazione è simile a quello di farlo spostare in un labirinto: occorrono sensori per rilevare la presenza di pareti e di ostacoli ed il robot deve elaborare un percorso, privo di ostacoli, per raggiungere una data meta. La situazione è però aggravata dal fatto che raramente la stanza di una casa offre percorsi regolari quanto quelli di un labirinto. Per di più, in una stanza, gli oggetti (una poltrona o un qualsiasi mobile) possono non rimanere sempre allo stesso posto, per non parlare di eventuali animali accucciati sul pavimento o la stessa gamba allungata del padrone di casa!
Il robot deve saper scegliere percorsi alternativi, tenendo conto di tutte queste possibilità, aggiornando continuamente la propria mappa interna dei vari ambienti. Tuttavia, i primi incontri col gatto di casa potrebbero risultare veri e propri scontri (il robot non conosce l'esistenza dei cari felini, né del resto ci si può attendere una quieta reazione da parte del gatto!).
La soluzione migliore è quella di applicare al robot dei sensori di movimento, in grado di misurare variazioni nella distanza di un oggetto. Una volta rilevato il movimento di questo, il robot potrebbe arrestarsi del tutto, in attesa che l'oggetto si fermi o si allontani. Un tale comportamento non è del tutto illogico o "stupido" quanto può sembrare, anzi è molto simile a quello adottato da un animale, in circostanze simili. Inoltre, tutto sommato, è meglio un robot fermo piuttosto che un improvvisato quanto indesiderato incontro di boxe nel salotto di casa.
L'intelligenza del robot, dunque, dipende per la maggiore dai "sensi" di cui esso è dotato, che gli permettono di costruirsi, seppure approssimativamente, una propria esperienza sul mondo circostante. Forse, tra non molti anni, ognuno di noi potrà disporre di un fedele maggiordomo (il classico "James") in versione computerizzata.



KEVIN JONES

A caccia di energia

Scegliere un percorso, evitando ostacoli sconosciuti, è un compito molto difficile. In questo esempio, il robot sta cercando di raggiungere una presa di corrente per ricaricarsi. Il percorso A è tipico dei robot più semplici: l'unico criterio seguito è quello di muoversi lungo il contorno degli oggetti o delle pareti. In ambienti non complessi e abbastanza regolari, questo sistema può funzionare, ma non nel caso qui illustrato, dove la disposizione degli ostacoli è tale che il robot esce dalla stanza... senza trovare la presa!
Il percorso B, invece, corrisponde a quello di un robot dotato di un algoritmo di movimento più sofisticato: quando incontra un ostacolo, il robot preferisce ruotare di un angolo il più piccolo possibile. Basta questo piccolo accorgimento per migliorare notevolmente la resa dell'apparecchio

Creare i modelli

Come funziona l'Abacus della Psion, in dotazione al QL della Sinclair



MENU DEI "PROMPT"



MENU DEI COMANDI

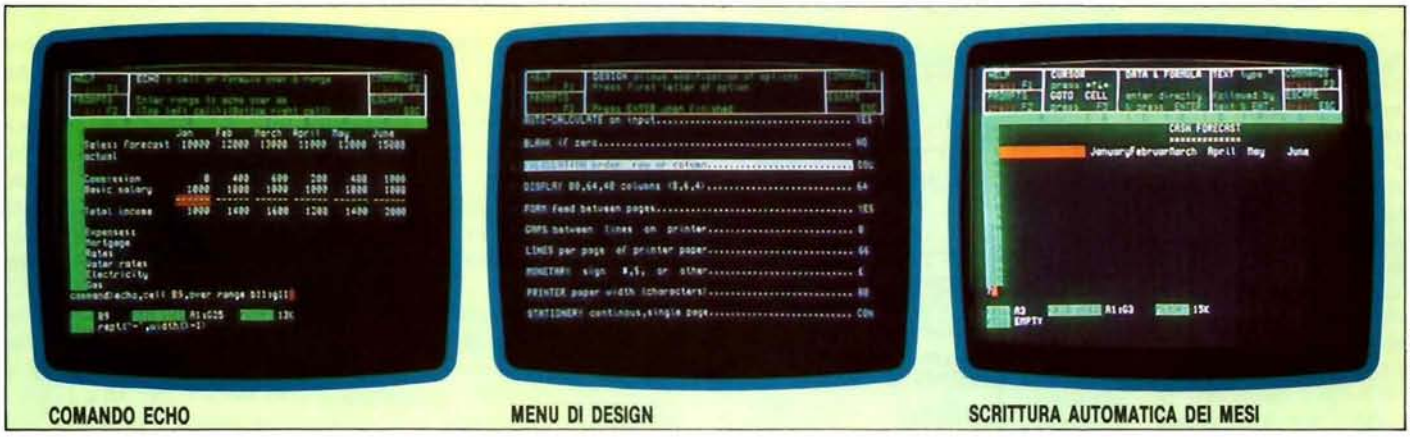
Molto dell'interesse suscitato dal QL Sinclair si deve al gruppo di 4 programmi forniti a corredo: *Quill* (un word processor), *Archive* (un database), *Easel* (un programma grafico) e *Abacus* (uno spreadsheet).
Il package possiede un certo grado di integrazione (vedere a pag. 982), che permette di trasferire con facilità i dati da un programma applicativo all'altro. Ad esempio, i dati calcolati con lo spreadsheet possono essere rappresentati graficamente (con Easel) ed essere poi inclusi in una relazione sviluppata con il word processor. Non solo i programmi si assomigliano "esteticamente", ossia nelle videate, ma hanno anche alcuni comandi in comune. In particolare, è identica la configurazione dei tasti di funzione F1, F2 e F3, usati rispettivamente per ottenere informazioni sul programma (Help), per controllare le richieste che appaiono in cima allo schermo e per richiamare gli altri comandi. Tuttavia, ciascun programma va caricato e lanciato separatamente.
Esistono due modi per caricare in memoria i programmi del package. Il primo consiste nell'inserire la cartuccia del programma nel Microdrive e premere il tasto F1 o F2 (a seconda che al computer sia collegato un monitor o un apparecchio TV): il programma è caricato ed avviato automaticamente. In alternativa, con lo schermo già selezionato, si adopera il comando lrun mdv1_boot (posto che il programma si trovi nel Microdrive 1): così facendo sullo schermo compare la videata iniziale.
Nel caso di Abacus, sullo schermo compare la parte superiore sinistra dello spreadsheet (ossia della griglia): dalla colonna A alla F e dalla riga 1 alla 15. Le dimensioni massime consentite da Abacus sono: 64 colonne e 255 righe (nel *Vu-Calc* i limiti massimi sono 28 colonne e 55 righe). Sopra la griglia appaiono diversi prompt, mentre sotto di essa c'è la linea di immissione dati, assieme ad informazioni sul modello attualmente in uso. I prompt, utili per i principianti in quanto spiegano le varie opzioni disponibili, si possono far scomparire premendo il tasto F2.
Il gruppo di prompt, pur avendone l'apparenza, non costituisce un vero e proprio menu: la selezione non avviene in modo diretto, ma tramite i comandi spiegati, appunto, da queste scritte.
Per modelli relativamente semplici, l'uso di Abacus non comporta problemi, ma in applicazioni più complesse l'uso di alcuni comandi e di certe espressioni matematiche richiede un po' di pratica. Nell'esempio che segue, concernente ancora una volta la preparazione di un bilancio familiare, si vede come opera Abacus.
Per prima cosa, occorre definire un'intestazione. Come nel Vu-Calc, le sequenze alfanumeriche vanno precedute da doppi apici. Usando i tasti di controllo del cursore ci si sposta nella posizione voluta (ad esempio sulla cella D1), poi si digita l'intestazione, preceduta da virgolette. Anche Abacus, come la maggior parte degli spreadsheet, consente che una scritta occupi più di una cella consecutiva.

Un bilancio preventivo

Per meglio evidenziare l'intestazione, possiamo anche sottolinearla: basta spostare il cursore sulla cella sottostante, per immettere l'espressione rept (" = ",len(d1)). Il comando rept (equivalente al REPLICATE del Vu-Calc), provvede a ripetere il simbolo = per tutta la lunghezza (len) della cella D1.
Diversamente dal Vu-Calc, in cui i campi di ogni cella sono fissi (9 colonne), l'Abacus permette di definire la lunghezza dei campi, per meglio adattarli ad una specifica applicazione. Nel nostro caso, è preferibile che la colonna A sia più ampia (per poter inserire le varie "voci" di entrata/uscita) e che le altre colonne siano sufficientemente strette da poter permettere la visione completa di un arco semestrale. Per queste variazioni si preme il tasto F3, facendolo seguire da una G, per selezionare il comando GRID (ossia "griglia") e da una W, per comunicare che intendiamo modificare la larghezza, (comando WIDTH). Nella linea di immissione compare il valore attribuito automaticamente, ossia 10. Supponiamo che la lunghezza voluta, per la colonna A, sia 15. Si digita tale valore e compare una richiesta sull'arco di celle alle quali applicare la lunghezza specificata; basta adesso immettere due volte A, per limitare la modifica alla sola colonna A. Per le restanti colonne si ripete l'operazione, ma specificando stavolta una lunghezza di 6 e un arco di celle da B a G.
Adesso si tratta di inserire le testate relative a ciascun mese. L'immissione può esser fatta in modo analogo a quello usato per la testata; altrimenti, se i nomi dei mesi in inglese non creano problemi, si può ricorrere ad una speciale funzione di Abacus: si posizioni il cursore sulla cella A3 e si scriva row = month(col()−1). A questa immissione segue la richiesta per l'arco di celle interessate: si risponda

digitando rispettivamente B e G e, automaticamente, appaiono i nomi dei mesi. La lunghezza di alcuni di essi, nella nostra particolare applicazione, è maggiore di quella riservata alle rispettive celle: si posizioni il cursore e si accorcino i nomi sovrascrivendo spazi (si rammenti che il testo va preceduto da virgolette).
È ora la volta di immettere le varie "voci" di entrata/uscita; si proceda, previo posizionamento del cursore, come nel caso dell'intestazione. Nell'esempio, abbiamo considerato il bilancio di un rappresentante, le cui entrate sono, oltre allo stipendio, le commissioni sulle vendite. Il modello calcola le future entrate, previste in base allo stipendio più le commissioni su vendite stimate.
Col passare dei mesi, il rappresentante può sostituire le cifre di previsione con quelle effettive, ricalcolando il tutto. Eventuali valori negativi, sulla griglia, vengono visualizzati tra parentesi.
Ecco un elenco di possibili voci. Per le Entrate: Vendite (effettive e previste), Commissioni, Stipendio, Totale Entrate. Per le Uscite: Affitto, Tasse, Luce, Gas, Telefono, Totale Uscite. Inoltre: Totale Netto, Saldo Iniziale e Finale (Banca).

entrate. Per migliorare la presentazione del nosto modello, si può "tirare" una linea di separazione sopra e sotto la riga dei totali delle entrate. Si sposti il cursore in B12 e si scriva row = rept("— ",width( )—2); come risultato, si ottiene una "sottolineatura" di ciascuna cifra dello stipendio. Per ripetere la stessa operazione nella riga 14 (quella per i totali delle entrate), basta ricorrere al comando ECHO. Col cursore in B9, viene chiesto l'arco di celle interessate al "tratteggio": si digiti B14:G14 (qualora si fosse dimenticato di aggiungere righe vuote, durante la creazione, si può usare il comando GRID). Chi ha uno spiccato senso estetico, può infine, adoperare il comando J sull'arco B-G, per allineare a destra le sottolineature, in modo che corrispondano alle cifre visualizzate.
Adesso si possono immettere le spese, semplicemente digitandole nell'appropriata cella oppure usando formule, come visto per lo stipendio. Per il totale delle uscite si usa il criterio già esposto per sommare le entrate.
Sfortunatamente, Abacus sembra ignorare tutto quanto segua uno spazio situato all'interno di un comando, quindi una formula del tipo row = totale entrate—totale uscite non funziona: le testate vanno cambiate in parole singole, per poi immettere row = entrate—uscite, nuovamente



COMANDO ECHO — MENU DI DESIGN — SCRITTURA AUTOMATICA DEI MESI

IAN McKINNELL

Terminata tutta questa fase preliminare, il modello è pronto: basta inserire i dati. Per prima cosa, immettiamo le previsioni di vendita per i sei mesi visualizzati, ad esempio: 10000, 12000, 13000, 11000, 12000 e 150000. I valori (in questo caso espressi in decine di migliaia di lire) vanno immessi senza virgole o punti.
La riga relativa alle vendite effettive va ovviamente lasciata in bianco mentre invece si possono calcolare le provvigioni basate sulle vendite previste. Supponendo che la percentuale sia del 20%, sulle vendite che superano i dieci milioni, la formula da immettere nella cella B10 è row = (vendite—10000)—.2 (di nuovo, l'arco di celle interessato è da B a G); fatto questo, sullo schermo appaiono i valori calcolati.
Se lo stipendio è di un milione, nella cella B11 va immesso row = 1000 (indicando, di nuovo, l'arco di celle da B a G). Adesso passiamo al totale delle entrate, alla cella B13: la formula è row = sum(col), con un arco di righe da 0 a 11 e di colonne da B a G. Ciò completa il calcolo delle

per tutto l'arco B-G. I risultati del calcolo appaiono immediatamente.
Il passo finale riguarda il calcolo del saldo per il conto corrente bancario: si immette il valore iniziale, poi si calcola quello finale di Gennaio usando la formula row = netto + iniziale, col cursore in B28 (sempre sull'arco B-G). Per calcolare i saldi iniziali dei mesi successivi basta immettere, in C27, la formula row = B28. Tuttavia, per far sì che il calcolo si estenda a tutte le colonne, occorre modificare l'ordine dei calcoli da righe a colonne, tramite il comando DESIGN.
Il vantaggio di questo laborioso procedimento è che i vari totali vengono calcolati e aggiornati automaticamente, ogni volta che si modifica una delle cifre all'interno del bilancio.
I valori negativi, come si è detto, vengono racchiusi tra parentesi. Se si desidera invece il segno meno, si può ricorrere al comando UNITS.
Il modello è ora completo e conviene memorizzarlo scegliendo l'opzione SAVE di Abacus, assegnandogli un nome di file appropriato.

Far correre... le "voci"

Una gradevole impostazione video, buone caratteristiche e un'ottima qualità nei messaggi e nei menu fanno di Abacus un programma "spreadsheet" piacevole da usare. Qui è illustrato il comando ECHO (per copiare il contenuto da una cella all'altra), le opzioni del comando DESIGN (per modificare il formato della tabella elettronica) e l'opzione di generazione automatica dei nomi dei mesi (ahimé, soltanto in lingua originale) per le testate delle colonne

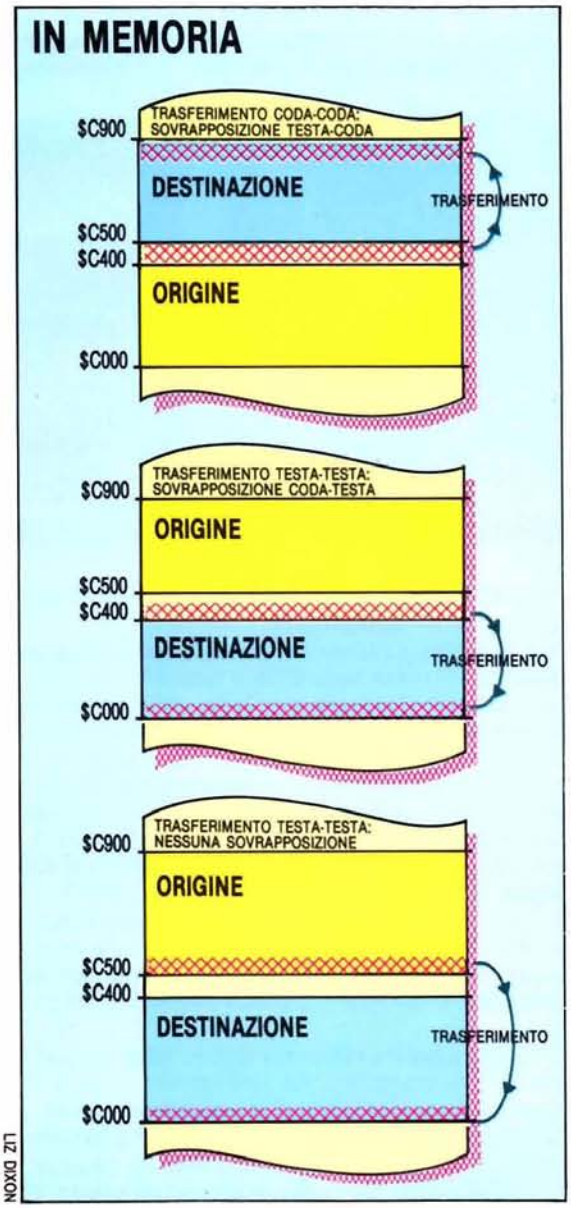
Ricerca e sostituzione

Modifiche alla utility di ricerca, per il BBC ed il Commodore 64

Un programma che permetta di sostituire il nome di una variabile con un altro si dimostra molto più utile della semplice procedura di ricerca dei nomi delle variabili, sviluppata a pag. 1144 e 1180. Il programma di sostituzione deve però essere abbinato ad un programma in codice macchina.
Poiché la CPU 6502 del micro BBC e la 6510 del Commodore 64 usano lo stesso linguaggio Assembly, possiamo scrivere un programma che funziona su entrambi gli apparecchi (la versione per Spectrum è presentata più avanti).

Trasferimento "intelligente"

Quando la routine di sostituzione inserisce nuovi nomi di variabili di lunghezza diversa rispetto ai vecchi, deve contemporaneamente spostare grandi sezioni del programma BASIC all'interno della memoria. In questa fase deve gestire quattro possibili condizioni riguardanti gli indirizzi di partenza e d'arrivo. Se il trasferimento iniziasse sempre dall'inizio del blocco, quando la testa del blocco di destinazione supera l'estremità finale o "coda" del blocco di origine, la procedura di trasferimento finirebbe inevitabilmente per scrivere sopra i dati in esso contenuti. Una routine di trasferimento "intelligente" prevede questa eventualità ed evita la perdita di dati trasferendo il blocco di origine a partire dalla sua coda. Una routine "non intelligente", al contrario inizia sempre il trasferimento dal primo byte o "testa" del blocco di origine.



La prima cosa da fare è studiare una tecnica che permetta di tenere contemporaneamente in memoria due programmi distinti. Sul Micro BBC possiamo raggiungere questo obiettivo modificando le due variabili di sistema PAGE e HIMEM del BASIC. Sul Commodore 64 occorre invece un breve programma in codice macchina che corregga i puntatori in pagina 0 della memoria. Questa operazione viene eseguita dalla prima parte del listato in linguaggio Assembly (quella indicata dall'etichetta SCAMBIO).
La routine SCAMBIO consente di depositare in memoria due programmi BASIC: uno inizia all'indirizzo esadecimale 800 (l'abituale indirizzo iniziale di un programma BASIC), l'altro all'indirizzo esadecimale 9000. La routine SCAMBIO legge il puntatore TXTTAB per sapere qual'è l'area di programma corrente, quindi ne cambia il contenuto per passare all'altra area di programma. Tale modifica porta TXTTAB a puntare all'inizio della nuova area, mentre FRESPC punta all'ultimo byte della nuova area e FRETOP e MEMSIZ al byte immediatamente successivo. Quindi il programma, usando VARTAB come puntatore temporaneo, scorre la sequenza dei puntatori di collegamento d'indirizzo (vedere a pag. 1184) per trovare l'indirizzo finale del programma BASIC. Quando trova i due byte zero, che indicano la fine del programma, incrementa di due il puntatore e trasferisce il risultato in ARYTAB e STREND. In questo modo VARTAB, ARYTAB e STREND puntano tutti al byte successivo alla fine del programma BASIC. Le principali modifiche da apportare al programma BASIC riguardano le subroutine alle linee 30500 e 30600. La prima individua la fine del programma BASIC utilizzando la lunghezza dei byte di linea nella versione BBC e i puntatori alla linea successiva nella versione Commodore 64.
La subroutine alla linea 30600 esegue invece la sostituzione dei nomi delle variabili. Quando il vecchio ed il nuovo nome hanno la stessa lunghezza, il secondo viene semplicemente scritto sopra il primo; in caso contrario la procedura è un po' più complicata. Se i nomi hanno lunghezza diversa, il programma, oltre ad ampliare o ridurre lo spazio sottoposto a modifica, corregge anche le variabili utilizzate per annotarne la posizione nel programma modificato. Infine viene variata la lunghezza del byte di linea nella versione BBC ed i puntatori alla linea successiva nella versione Commodore.
Per ampliare o ridurre lo spazio, il programma usa due subroutine in codice macchina, perché il BASIC non è in grado di eseguire ad

una velocità accettabile spostamenti di centinaia di byte.
Le routine sono: SU, per accrescere lo spazio disponibile; GIU, per eliminare lo spazio in eccesso. Le informazioni riguardanti i blocchi di programma da spostare vengono passate al codice macchina tramite le locazioni di memoria CORR, ULTI e DIFF.
Per la subroutine SU è necessario inizializzare le seguenti locazioni di memoria:

CORR: per l'indirizzo iniziale del blocco
ULTI: per l'indirizzo finale del blocco, diminuito di uno
DIFF: per il numero di byte da aggiungere

mentre per GIU si devono inizializzare:

CORR: per l'indirizzo finale del blocco
ULTI: per l'indirizzo iniziale del blocco, diminuito di uno
DIFF: per il numero di byte da eliminare

Si noti che le due subroutine iniziano a muovere il blocco di programma a partire dalle estremità opposte e si muovono in direzioni opposte per evitare di scrivere sopra i nomi delle variabili, prima che questi vengano trasferiti.
Per poter usare la versione BBC del programma di sostituzione, si deve prima caricare ed eseguire il programma in linguaggio Assembly che trasferisce il codice macchina in memoria, quindi caricare il programma da modificare e poi digitare l'istruzione:

```
P% = PAGE
```

che comunica al programma di sostituzione l'indirizzo iniziale del programma. Infine, si deve assegnare a PAGE un valore superiore a quello di LOMEM. A questo punto si può caricare ed eseguire il programma di sostituzione delle variabili; per ritornare al programma modificato si usano i comandi:

```
HIMEM = PAGE–1
PAGE = P%
```

Anche nella versione Commodore 64 occorre caricare il codice macchina in memoria mediante il programma di caricamento BASIC oppure assemblando il codice sorgente. In entrambi i casi è comunque necessario azzerare gli indirizzi 36864, 36865 e 36866, vale a dire definire una nuova area di programma, alternativa alla prima. Dopo il caricamento del codice macchina, SYS 49152 permette di passare da un'area di programma all'altra. Se ci si dimentica l'area in cui ci si trova, la si può scoprire con l'istruzione PRINT PEEK (44) che indica con 8 l'area di programma normale e con 144 quella alternativa. Una volta che il codice macchina si trova in memoria, si può caricare il programma da modificare nell'area di programma normale e quello che sostituisce le variabili nell'area alternativa; fatto ciò si può eseguire il programma di sostituzione.

ROUTINE DI SCAMBIO E TRASFERIMENTO PER C64

```
  10 INDIR=49152
  20 FOR LINEA=1000 TO 1180 STEP 10
  30 S=0
  40 FOR INDIR = INDIR TO INDIR+7
  50 READ BYTE
  60 POKE INDIR,BYTE
  70 S=S+BYTE
  80 NEXT INDIR
  90 READ CHECKSUM
 100 IF S<>CHECKSUM THEN PRINT "ERRORE NEI
     DATI ALLA LINEA";LINEA:END
 110 NEXT LINEA
 120 POKE36864,0:POKE36865,0:POKE36866,0
1000 DATA162,0,164,44,192,8,208,9,787
1010 DATA160,160,32,72,192,169,144,208,1137
1020 DATA7,160,144,32,72,192,169,8,784
1030 DATA162,1,134,43,133,44,134,45,696
1040 DATA133,46,160,0,177,45,170,200,931
1050 DATA177,45,224,0,208,240,201,0,1095
1060 DATA208,236,230,45,208,2,230,46,1205
1070 DATA136,16,247,165,45,133,47,133,922
1080 DATA49,165,46,133,48,133,50,96,720
1090 DATA134,51,132,52,134,55,132,56,746
1100 DATA202,136,134,53,132,54,96,160,967
1110 DATA0,177,251,164,255,145,251,160,1403
1120 DATA0,196,251,208,2,198,252,198,1305
1130 DATA251,165,253,197,251,208,234,165,1724
1140 DATA254,197,252,208,228,96,164,255,1654
1150 DATA177,251,160,0,145,251,230,251,1465
1160 DATA208,2,230,252,165,253,197,251,1558
1170 DATA208,236,165,254,197,252,208,230,1750
1180 DATA96,0,0,0,0,0,0,0,96
```

SOSTITUZIONE VARIABILI

Commodore 64

Apportare le seguenti modifiche al programma di pag. 1180:

```
30005 INPUT "SOSTITUIRE CON";RS
30006 CORR = 251
30007 ULTI = 253
30008 DI = 255
30035 GOSUB 30500
30036 IF ERR THEN PRINT "NON RIESCO A TROVARE LA
      FINE DEL PROGRAMMA": END
30060 IF NUOVALIN = 0 THEN END
30065 CORR = PUNTXT
```

cancellare la linea 30085

```
30465 IF ERR THEN PRINT "SPAZIO INSUFFICIENTE ALLA
      LINEA";NUMLIN END
```

Micro BBC

Apportare le seguenti modifiche al programma a pag.1145:

```
30005 INPUT "Sostituire con";REPLACEMENT$
30006 CORR = &70
30007 ULTI = &74
30008 DIFF = &78
30055 GOSUB 30500
30056 IF Outofroom THEN PRINT "Non riesco a trovare la
      fine del programma": PRINT "PAGE in posizione
      errata?"
30060 Puntatxt = P%
30070 IF Numlin>32767 THEN END
30105 Llinea = Puntatxt
30460 IF Nome$ = Target$ THEN GOSUB 30600
30465 IF Outofroom THEN PRINT "Spazio insufficiente alla
      linea";Numlin:end
```

ROUTINE DI TRASFERIMENTO PER IL BBC

```
  10 MODE 7
  20 HIMEM=HIMEM-&100
  30 CORR = &70
  40 ULTI = &74
  50 DIFF = &78
  60 FOR PASSO = 1 TO 2
  70 P%=HIMEM
  80[ OPT 1
  90 .SU      LDY #0
 100 .SU1     LDA (CORR),Y
 110          LDY DIFF
 120          STA (CORR),Y
 130          LDY #0
 140          CPY CORR
 150          BNE SU2
 160          DEC CORR+1
 170 .SU2     DEC CURR
 180          LDA ULTI
 190          CMP CORR
 200          BNE SU1
 210          LDA ULTI+1
 220          CMP CORR+1
 230          BNE SU1
 240          RTS
 250 .GIU     LDY DIFF
 260          LDA (CORR),Y
 270          LDY #0
 280          STA (CORR),Y
 290          INC CORR
 300          BNE GIU1
 310          INC CORR+1
 320 .GIU1    LDA ULTI
 330          CMP CORR
 340          BNE GIU
 350          LDA ULTI+1
 360          CMP CORR+1
 370          BNE GIU
 380          RTS
 390]
 400 NEXT PASSO
 410 PRINT~SU,~GIU
```

DATABASE

Ottava scheda delle istruzioni per lo Z80 della Zilog. Istruzioni per la manipolazione di bit

		INDIRIZZAMENTO DEI REGISTRI							REG. INDIR.	INDICIZZATO	
	BIT	A	B	C	D	E	H	L	(HL)	(IX + d)	(IY + d)
VERIFICA SU UN BIT, "BIT"	0	CB 47	CB 40	CB 41	CB 42	CB 43	CB 44	CB 45	CB 46	DD CB d 46	FD CB d 46
	1	CB 4F	CB 48	CB 49	CB 4A	CB 4B	CB 4C	CB 4D	CB 4E	DD CB d 4E	FD CB d 4E
	2	CB 57	CB 50	CB 51	CB 52	CB 53	CB 54	CB 55	CB 56	DD CB d 56	FD CB d 56
	3	CB 5F	CB 58	CB 59	CB 5A	CB 5B	CB 5C	CB 5D	CB 5E	DD CB d 5E	FD CB d 5E
	4	CB 67	CB 60	CB 61	CB 62	CB 63	CB 64	CB 65	CB 66	DD CB d 66	FD CB d 66
	5	CB 6F	CB 68	CB 69	CB 6A	CB 6B	CB 6C	CB 6D	CB 6E	DD CB d 6E	FD CB d 6E
	6	CB 77	CB 70	CB 71	CB 72	CB 73	CB 74	CB 75	CB 76	DD CB d 76	FD CB d 76
	7	CB 7F	CB 78	CB 79	CB 7A	CB 7B	CB 7C	CB 7D	CB 7E	DD CB d 7E	FD CB d 7E
AZZERA UN BIT, "RES"	0	CB 87	CB 80	CB 81	CB 82	CB 83	CB 84	CB 85	CB 86	DD CB d 86	FD CB d 86
	1	CB 8F	CB 88	CB 89	CB 8A	CB 8B	CB 8C	CB 8D	CB 8E	DD CB d 8E	FD CB d 8E
	2	CB 97	CB 90	CB 91	CB 92	CB 93	CB 94	CB 95	CB 96	DD CB d 96	FD CB d 96
	3	CB 9F	CB 98	CB 99	CB 9A	CB 9B	CB 9C	CB 9D	CB 9E	DD CB d 9E	FD CB d 9E
	4	CB A7	CB A0	CB A1	CB A2	CB A3	CB A4	CB A5	CB A6	DD CB d A6	FD CB d A6
	5	CB AF	CB A8	CB A9	CB AA	CB AB	CB AC	CB AD	CB AE	DD CB d AE	FD CB d AE
	6	CB B7	CB B0	CB B1	CB B2	CB B3	CB B4	CB B5	CB B6	DD CB d B6	FD CB d B6
	7	CB BF	CB B8	CB B9	CB BA	CB BB	CB BC	CB BD	CB BE	DD CB d BE	FD CB d BE
PONE AD 1 UN BIT, "SET"	0	CB C7	CB C0	CB C1	CB C2	CB C3	CB C4	CB C5	CB C6	DD CB d C6	FD CB d C6
	1	CB CF	CB C8	CB C9	CB CA	CB CB	CB CC	CB CD	CB CE	DD CB d CE	FD CB d CE
	2	CB D7	CB D0	CB D1	CB D2	CB D3	CB D4	CB D5	CB D6	DD CB d D6	FD CB d D6
	3	CB DF	CB D8	CB D9	CB DA	CB DB	CB DC	CB DD	CB DE	DD CB d DE	FD CB d DE
	4	CB E7	CB E0	CB E1	CB E2	CB E3	CB E4	CB E5	CB E6	DD CB d E6	FD CB d E6
	5	CB EF	CB E8	CB E9	CB EA	CB EB	CB EC	CB ED	CB EE	DD CB d EE	FD CB d EE
	6	CB F7	CB F0	CB F1	CB F2	CB F3	CB F4	CB F5	CB F6	DD CB d F6	FD CB d F6
	7	CB FF	CB F8	CB F9	CB FA	CB FB	CB FC	CB FD	CB FE	DD CB d FE	FD CB d FE

Note:

Nelle istruzioni di verifica (BIT) il flag di Zero rispecchia quello del bit specificato.

RES e SET non hanno alcuna influenza sui flag.

d indica lo scarto (off set), compreso tra —127 e + 128, da aggiungere all'indirizzo di IX o di IY.

Per concessione della Zilog Inc.

Periferiche a confronto

Una panoramica sugli apparecchi da collegare al proprio home computer

Fino a poco tempo fa, la più impor tante periferica a disposizione degli utenti di home computer era un dispositivo ad innesto che conteneva una memoria accessoria. Le memorie erano ancora molto costose ed apparecchi come lo ZX-81 ed il Vic-20 venivano progettati cercando di ridurre al minimo i costi di produzione; in pratica lo ZX-81 non espanso disponeva solo di 700 byte per la redazione dei programmi. Oggi gli home computer possiedono mediamente fino a 64 Kbyte di RAM (alcuni, come il QL, ne hanno molti di più) e solo in rari casi sono necessari pacchetti RAM addizionali. Il mercato offre così una maggior gamma di periferiche: i modem consentono di comunicare con utenti distanti anche centinaia di chilometri, apposite interfacce permettono di controllare veicoli a motore o bracci robot ed i sintetizzatori di parole possono essere usati per giochi o attività didattiche.

Tuttavia molte delle periferiche oggi in commercio sono destinate ai computer di maggior successo. Questa considerazione deve essere tenuta presente al momento dell'acquisto di un nuovo apparecchio: chi decide per uno Spectrum, un Commodore o un Acorn potrà scegliere tra una gamma di prodotti maggiore rispetto a chi opta per un Oric o un Sord. Fino ad ora le periferiche per le macchine più recenti sono comparse sul mercato solo dopo un certo periodo di tempo rispetto all'uscita degli apparecchi. L'introduzione dello standard MSX dovrebbe facilitare notevolmente le cose, rendendo una stessa periferica compatibile con tutti i modelli che aderiscono a tale standard.

Il problema principale per chi acquista una periferica è infatti quello della compatibilità: l'apparecchio deve essere in grado di lavorare con le altre periferiche che verranno acquistate in seguito. Un esempio calzante è quello dello Spectrum Sinclair: gli utenti Spectrum che hanno comperato l'Interfaccia 1 ed il Microdrive si sono accorti con disappunto che alcune periferiche già in loro possesso, oltre ad una parte del software, non funzionano quando viene innestata l'Interfaccia 1. Periferiche compatibili ben assortite permettono di creare sistemi di elaborazione flessibili e perfettamente rispondenti alle necessità dell'utente.



CHRIS STEVENS

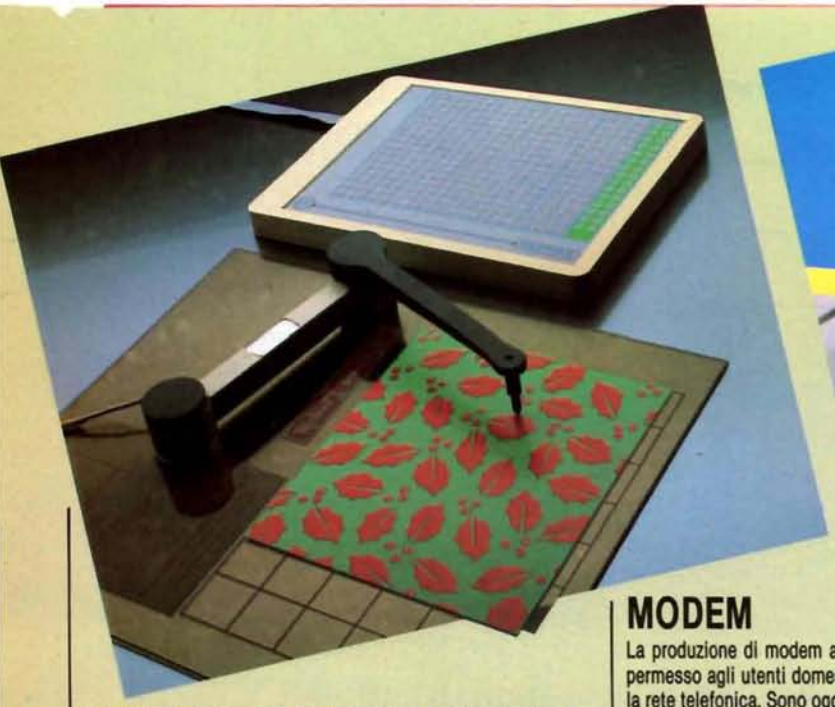
MEMORIE DI MASSA

Le memorie di massa più usate con i microcomputer sono senza dubbio i registratori a cassetta; poco costosi e facili da usare, presentano tuttavia alcuni svantaggi: il caricamento dei programmi è lento e non sempre è facile tenere una precisa documentazione del contenuto di ogni cassetta. Le unità a dischetti sono più veloci e affidabili, ma costano molto di più. La maggior parte degli home computer impiega un solo tipo di unità a dischetti che a volte si dimostra meno veloce di quanto ci si aspetterebbe (ad esempio, i modelli Commodore). I drive maggiormente usati sono da 5 1/4 pollici, ma anche le unità da 3 e 3 1/2 pollici stanno riscuotendo un buon successo. Il drive dell'Oric/Atmos, ad esempio, adotta dischetti da 3 pollici con una capacità di 160 Kbyte per facciata. Il Micro BBC è molto flessibile e può lavorare con modelli di diverso tipo: il Torch Disk Pack lo trasforma in un apparecchio totalmente diverso, perché associa al 6502 del computer un processore Z80 e fornisce al sistema una memoria addizionale di 64 Kbyte. Inoltre il Torch incorpora 4 programmi gestionali ed una versione del BASIC BBC appositamente progettata per lo Z80.

Al contrario, lo Spectrum Sinclair non possiede porte per unità a dischetti ma alcune compagnie indipendenti ne hanno prodotte alcune in proprio. La Sinclair ha realizzato il sistema Interface One/Microdrive, basato su un nastro "a ciclo chiuso" che può contenere circa 85 Kbyte di dati. L'interfaccia è controllata direttamente dal computer ed impiega in media dieci secondi per localizzare un'informazione, ossia un tempo intermedio tra quello di un registratore a cassette e quello di un'unità a dischetti. Il prezzo è inferiore a quello dell'unità a dischetti più economica. L'Interfaccia 1 possiede una porta RS232 (non standard) che permette allo Spectrum di collegarsi "in rete" con un massimo di 64 Spectrum o QL.

Un dispositivo che fa concorrenza all'Interfaccia 1 è il Rotronics Wafadrive; anch'esso si basa su un nastro "a ciclo chiuso", ma include nel prezzo le interfacce RS232 e Centronics ed un word processor. Viene venduto in tre versioni con nastri di diversa lunghezza da 16, 64 e 128 Kbyte; ovviamente il più breve è anche il più veloce. Attualmente viene prodotto un secondo Wafadrive per il Commodore 64 e stanno per essere lanciate nuove versioni per altri micro.

Gli apparecchi in figura sono: il Rotronics Wafadrive, l'unità a dischetti dell'Oric/Atmos, il Torch Disk Pack e l'Interfaccia 1 Sinclair collegata al Microdrive.





PERIFERICHE PER LA GRAFICA

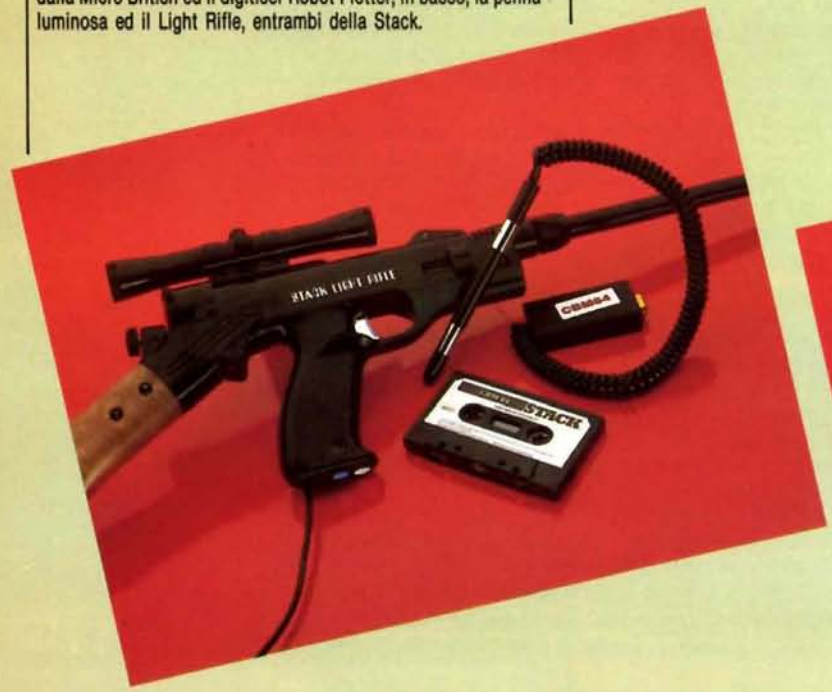
La nascita e lo sviluppo di vari tipi di periferiche grafiche consentono oggi di creare grafica sugli home computer con grande facilità. Il dispositivo più economico è la penna luminosa, che può essere usata per "disegnare" direttamente sullo schermo. Quando la sua punta viene appoggiata al video, una cellula fotoelettrica ne individua la posizione e la trasmette al computer. Questa tecnica ha avuto una diversa applicazione sullo Stack Light Rifle, un fucile giocattolo usato al posto del joystick in alcuni giochi di tiro al bersaglio (vedere a pagg. 710-711).
Per semplificare le cose, sono state realizzate particolari periferiche, dette tavolette grafiche, che vengono appoggiate sul tavolo di lavoro e sulle quali è più facile disegnare: sono collegate ad una speciale penna che trasmette al computer le linee tracciate sulla tavoletta e vengono usate per disegnare a mano libera o per ricalcare immagini stampate. Altre periferiche grafiche sono i "digitiser", composti da un braccio meccanico che, mediante una serie di resistenze variabili, è in grado di individuare la posizione del tracciatore fissato alla sua estremità. Gli apparecchi presentati in figura sono: in alto, la tavoletta grafica Grafpad fabbricata dalla Micro British ed il digitiser Robot Plotter; in basso, la penna luminosa ed il Light Rifle, entrambi della Stack.

MODEM

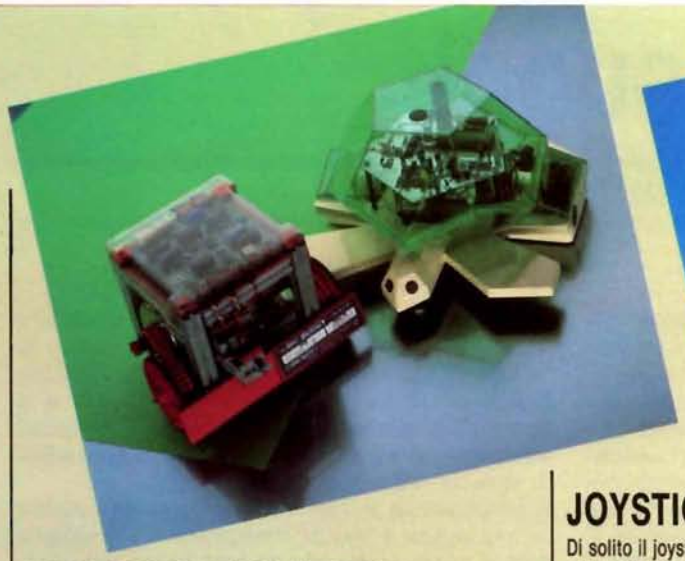
La produzione di modem a basso costo per home computer ha permesso agli utenti domestici di comunicare tra loro attraverso la rete telefonica. Sono oggi disponibili diversi modelli di modem per i computer dotati di interfaccia standard RS232. Grazie ad un apposito software, in Gran Bretagna è possibile collegarsi al Prestel, una banca dati pubblica che dedica un'intera sezione agli utenti di home computer. Esistono tuttavia altri database minori, detti "bulletin boards" gestiti da appassionati di microcomputer, che permettono ai loro utenti di scambiarsi messaggi e informazioni. Permangono tuttavia problemi di compatibilità perché spesso i database privati adottano una propria velocità di trasmissione diversa da quella dei modem collegati al Prestel. Dato che lo Spectrum ed il Commodore 64 non hanno l'interfaccia RS232, i due micro non possono usare modem standard; la Prism ha perciò progettato per lo Spectrum il modem VTX5000, dotato di un software interno per il collegamento con il Prestel e inoltre ha sviluppato un software su cassetta per scambiare programmi tra due Spectrum equipaggiati con modem Prism. La Commodore fabbrica un proprio modem per il Commodore 64 ed ha creato un sistema, il Compunet, che consente di collegarsi in rete. Nell'usare il modem, conviene tener d'occhio l'orologio, infatti la bolletta del telefono cresce in fretta! L'illustrazione in alto mostra il Prism VTX5000 ed il modem della Commodore.

SINTETIZZATORI DI PAROLE

Questi sofisticati strumenti, prodotti per i principali home computer, hanno la capacità di "pronunziare" parole perfettamente comprensibili. I sintetizzatori di parole si suddividono in due gruppi: il primo tipo possiede un "vocabolario" fisso, composto da 100 e più parole (come il sintetizzatore Acorn per il Micro BBC, che ha la voce di un noto speaker radiofonico inglese); il secondo si basa su "allofoni", ossia su una serie di suoni e pause che vengono combinati tra loro per comporre le parole.
La Currah produce una serie di sintetizzatori di parole del secondo tipo per lo Spectrum (Microspeech) e per il Commodore 64 (Speech 64). La colonna sonora di alcuni giochi per questi apparecchi (ricordiamo quelli della Ultimate) contiene parole che vengono generate automaticamente quando il computer viene collegato ad un sintetizzatore Currah. In figura è riprodotto lo Speech 64 della Currah e lo Sweetalker Cheetah per lo Spectrum.







DISPOSITIVI PILOTATI

I computer possono essere usati per controllare o guidare apparecchi di vario genere, come i sistemi di riscaldamento. Molto più divertente è la guida dei numerosi tipi di periferiche che si muovono sul terreno: la Valiant Turtle, ad esempio, è un veicolo a motore elettrico che ricorda vagamente una tartaruga; viene programmata in LOGO per tracciare linee. Può essere pilotata dallo Spectrum, dal Commodore 64 e dal Micro BBC e comunica col computer per mezzo di un fascio luminoso a raggi infrarossi. Il Buggy BBC, simile alla tartaruga, è collegato al computer tramite un apposito cavo.
I due apparecchi in figura sono la Valiant Turtle ed il Buggy BBC.

STAMPANTI/PLOTTER

Chi usa un home computer per scrivere programmi o come word processor non può fare a meno di una stampante. I modelli più usati possiedono testine ad aghi o dispositivi a margherita. Le stampanti ad aghi adoperano una matrice a punti per stampare le lettere, possono creare immagini grafiche e la loro velocità di stampa è piuttosto elevata; la qualità degli stampati è però inferiore a quella delle stampanti a margherita che, fondamentalmente, sono macchine da scrivere controllate da computer.
Un sistema alternativo può essere la piccola stampante/plotter per i computer Tandy, Atari, Commodore ed Oric. L'apparecchio possiede al proprio interno una serie completa di caratteri e stampa su fogli di carta da 4 pollici. La testina di scrittura usa 4 piccole penne a sfera che permettono di scrivere testi e creare grafica a più colori con un unico sistema di stampa. Un'ulteriore alternativa è data dalla stampante termica Epson P40 a batterie ricaricabili, che usa una testina ad elementi termici per stampare su carta sensibile al calore. Questa periferica abbina un prezzo molto contenuto ad una apprezzabile qualità di stampa. Le dimensioni della carta sono piuttosto ridotte ma nel modo "condensato" la P40 può stampare testi a 80 colonne. In figura: la stampante/plotter Tandy/Radio Shack e quella termica P40.



JOYSTICK

Di solito il joystick è la prima periferica che viene acquistata da chi possiede un home computer. Molti apparecchi possiedono apposite interfacce ed alcuni dei modelli più recenti incorporano addirittura un piccolo joystick. I joystick più venduti usano connettori "D" a 9 piedini adottati per la prima volta dai micro Commodore e Atari ed in seguito fabbricati da numerosi produttori indipendenti. I joystick del BBC non sono standard (la scelta è quindi piuttosto limitata) mentre la Commodore ha deciso di abbandonare sui micro Plus/4 e 16 l'abituale formato, costringendo i propri utenti ad acquistare i nuovi joystick Commodore.
La Sinclair ha da poco lanciato per lo Spectrum l'Interfaccia 2, che contiene un'interfaccia per joystick e una slot per cartucce ROM. Lo Spectrum utilizzava un'interfaccia per joystick fabbricata dalla Kempston divenuta uno standard per questo apparecchio e le sue "specifiche" di funzionamento erano state adottate da molti fabbricanti. Ma la nuova porta dell'Interfaccia 2 non è compatibile con la Kempston quindi molti giochi adesso non funzionano più. Per questo motivo la Kempston ha deciso di fabbricare una seconda interfaccia compatibile con il software prodotto per entrambe le porte.
In alternativa si può acquistare un'interfaccia "programmabile" che consente di usare qualsiasi tipo di software, indipendentemente dal joystick per cui è stato sviluppato. Questa speciale interfaccia si serve del joystick per imitare il funzionamento della tastiera ed è l'ideale per quegli utenti Spectrum che già possiedono una vasta biblioteca software.
Dei molti joystick sul mercato, il più singolare è il RAT, prodotto dalla Cheetah in esclusiva per lo Spectrum (vedere pagg. 1070-1071), che è privo di cavo di collegamento e comunica col computer mediante un fascio di raggi infrarossi. Il micro Amstrad possiede invece una porta singola su cui vengono innestati due joystick in parallelo, per giochi a due giocatori.
I joystick in figura sono (da sinistra a destra): il joystick Amstrad, il RAT della Cheetah, il Kempston PRO 5000. L'apparecchio in primo piano è l'interfaccia Kempston per lo Spectrum ZX.

MONITOR

Molti utenti di home computer usano come display un comune apparecchio televisivo. Questa scelta crea numerosi problemi e, tra l'altro, è la causa della mediocre qualità dell'immagine. Se si decide di acquistare un monitor però, si deve fare molta attenzione al tipo di apparecchio che si sceglie: esistono infatti monitor RGB e monitor video compositi (vedere a pag. 509). Questi ultimi vengono usati con i micro Atari e Commodore mentre il BBC, l'Oric/Atmos ed il QL Sinclair adottano il formato RGB che produce immagini di migliore qualità. Alcuni computer generano suoni attraverso l'altoparlante del televisore, quindi hanno bisogno di monitor dotati di altoparlante.
Per supplire a ciò molti fabbricanti di televisori costruiscono oggi apparecchi dotati di interfacce monitor che possono essere usati sia come televisori che come display di alta qualità. La figura in basso mostra il monitor Microvitec Cub.





Produrre suoni

Sfruttiamo l'apparecchio realizzato in laboratorio

```
      LDX #0              2
CICL1
      LDA TABEL,X         4
      STA PORT            4
      INX                 2
      CMP  #PASSI         2
      BNE CICL1           3
      (2 se il ciclo non è
      eseguito)
```

Per controllare se il nostro sistema funziona possiamo usare un breve programma in BASIC. I suoni, in sostanza, vengono generati elettricamente inviando una tensione oscillante all'amplificatore. Perciò, caricando in successione nel registro dati della porta utente valori che oscillano tra 0 e 255, possiamo ottenere una tensione oscillante in uscita dal commutatore D/A. Si inserisca e si esegua il seguente programma, quindi si ruoti verso destra il potenziometro finché non si comincia ad udire il suono.

```
10 REM ***** GENERATORE BASIC DI SUONI PER CBM *****
20 DDR=56579:REGDAT=56577
30 POKEDDR,255
35 N=1
40 POKE REGDAT,0:FORI=1TON:NEXT:POKE REGDAT,255:GOTO40

10 REM ***** GENERATORE BASIC DI SUONI PER BBC *****
20 DDR=&FE62:REGDAT=&FE60
30 ?DDR=255
35 N=1
40 ? REGDAT=0:FOR I=1TON:NEXT:? REGDAT=255:GOTO40
```

Si noti che il programma BASIC ha una struttura ripetitiva, tutta concentrata in un'unica linea (per ottenere una maggior velocità d'esecuzione), e che viene eseguito un ciclo di ritardo quando il registro dati contiene valori compresi tra 0 e 255. La variabile N alla linea 35 definisce la durata del ritardo. Proviamo a cambiarne il valore e ad eseguire nuovamente il programma. Si può notare che l'altezza del tono decresce con l'aumentare del valore di N.

Con questo programma si raggiunge la massima altezza di tono solo quando viene completamente soppresso il ciclo di ritardo. Dopo ripetute prove si può notare che anche una sola esecuzione del ciclo o una minima variazione di N (ad esempio di 1), provocano una variazione udibile del tono della nota.

Purtroppo il BASIC è troppo lento per consentire un adeguato controllo della velocità d'oscillazione. Quindi è necessario usare il codice macchina.

Come controllare il tono ed il volume dei suoni in codice macchina è spiegato più avanti; per adesso ci limiteremo a sviluppare un programma che genera forme d'onda di diverso tipo. Il programma BASIC precedente produceva un'onda quadra ma è possibile creare altre forme d'onda, che modificano la "qualità" del tono. Con una tecnica digitale possiamo sintetizzare onde sinusoidali e a dente di sega partendo da una serie di valori campione della forma d'onda ed inserendoli in una tabella di consultazione.



IAN McKINNELL

Il programma in codice macchina che carica in sequenza i valori campione nel registro dati è tutto sommato abbastanza semplice. La figura alla pagina seguente mostra le tre forme d'onda e le tabelle di consultazione per la creazione di onde sinusoidali e a dente di sega. Se il ciclo d'onda viene suddiviso in una serie di fasi o passi successivi di cui viene fatta una campionatura, è possibile inserire nel programma un ciclo iterativo che trasmetta questi valori campione alla porta utente (vedere riquadro a sinistra).

La generazione dei suoni si basa sulla sincronizzazione. Ogni istruzione è seguita da un numero che indica i cicli macchina necessari per la sua esecuzione. La formula seguente permette di calcolare il numero totale di cicli necessari per produrre un ciclo d'onda completo: numero di cicli macchina = 2 + (4 + 4 + 2 + 2 + 3) x n. passi − 1 = 1 + 15 x n. passi. Se l'onda che si vuol ottenere è suddivisa in 80 passi campione, sono necessari 1.201 cicli macchina per ottenere un ciclo d'onda completo.

Poiché nel codice del 6502 ogni ciclo macchina dura circa un milionesimo di secondo, la seguente espressione permette di calcolare il numero di cicli d'onda che possono essere prodotti in un secondo (ossia la frequenza di una nota): frequenza = 1000000/1201 = 832 Hz. Dato che la frequenza del Do centrale è di 512 Hz, la tonalità della nota emessa dovrebbe essere leggermente superiore (di poche note) a quella del Do centrale.

Ricordiamo che il numero di passi in cui abbiamo suddiviso la forma d'onda che vogliamo generare influisce direttamente sulla frequenza della nota prodotta.

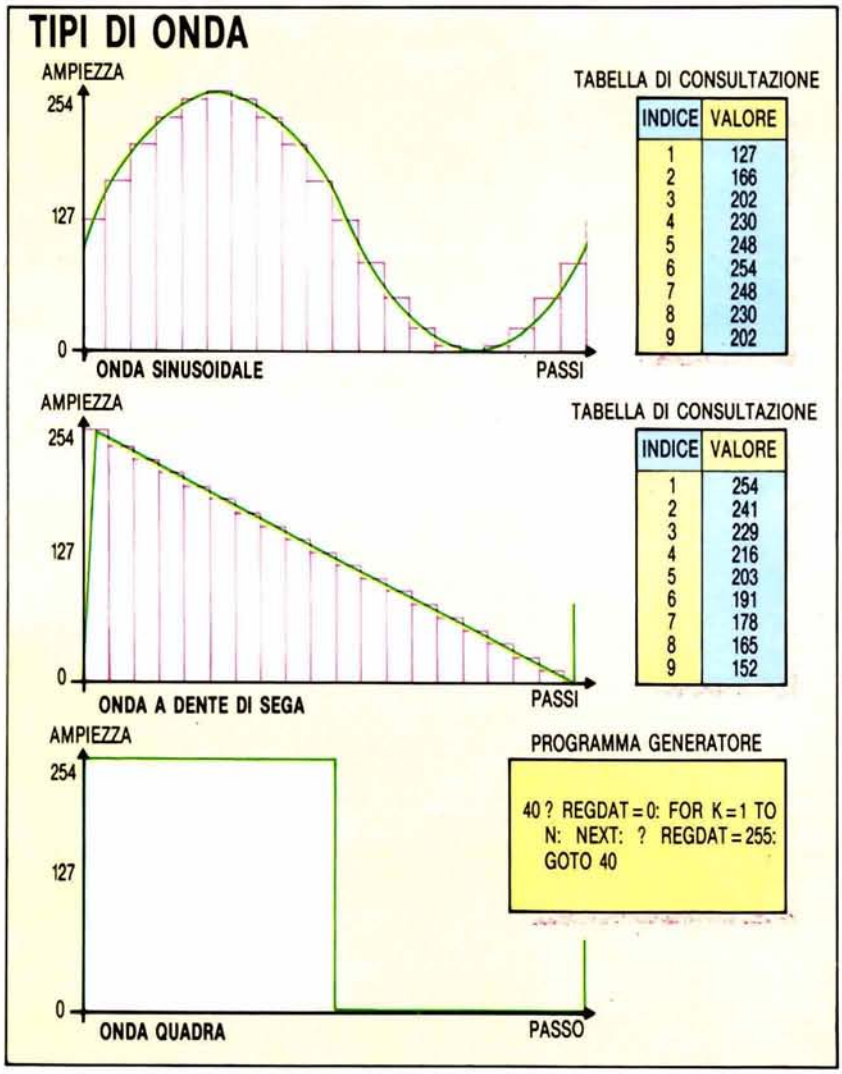
Raddoppiando il numero di passi campione, si riduce della metà la frequenza della nota. Quindi, usando un numero maggiore di valori campione della nota, possiamo ottenere una maggior precisione nella qualità del tono sintetizzato, che tuttavia deve sempre risultare bilanciato rispetto alla frequenza massima finale che si vuol ottenere per un dato numero di passi.

Poiché un unico ciclo d'onda produce un suono così breve da essere difficilmente udibile, il programma deve possedere una routine che ripeta un certo numero di volte la sezione che genera la forma d'onda. A questo

IL PROGETTO

Il segnale audio può essere inviato ad una cuffia o ad un amplificatore stereo attraverso la coppia di prese di output più vicina al potenziometro del commutatore D/A. Per il collegamento con la cuffia si usi un'adeguata spina stereofonica femmina, saldando i due contatti ad un cavetto isolato connesso alla presa rossa del commutatore D/A. Per collegarsi ad uno stereo occorre prima consultare il manuale dell'amplificatore, per individuare l'ingresso audio ed il contatto per la massa. Lo spinotto stereofonico può essere acquistato in un comune negozio di materiali elettrici. A questo punto la procedura è uguale per entrambi i sistemi: si colleghi il convertitore D/A al buffer box e quest'ultimo alla porta utente, quindi si innestino nelle prese del commutatore D/A gli spinotti collegati alla cuffia o all'amplificatore. Si ruoti completamente a sinistra il potenziometro ed infine si dia corrente all'apparecchio.

scopo si carica il numero che indica le ripetizioni da eseguire in un contatore, che viene poi decrementato fino a 0. L'impiego di un contatore a 16 bit, composto da due locazioni adiacenti, permette di usare numeri molto più grandi. All'inizio del programma l'istruzione SEI disabilita le interrupt, mentre l'istruzione CLI le riattiva al termine del programma; una loro comparsa durante l'esecuzione ne danneggerebbe infatti la sincronizzazione. Purtroppo, non è possibile disattivare le interrupt non mascherabili, che possono quindi provocare errori di sincronizzazione.
I valori campione che generano l'onda devono essere allocati nella memoria in forma di tabella di consultazione e ciascun tipo di forma d'onda deve occupare 80 locazioni consecutive.
Nella versione Commodore, la tabella dell'onda sinusoidale viene caricata in memoria a partire dalla locazione $C000, la tabella dell'onda a dente di sega a partire dalla locazione $C050 e quella dell'onda quadra dalla locazione $C0A0. Il programma in codice macchina carica automaticamente i dati contenuti nella tabella dell'onda sinusoidale usando il sistema di indirizzamento indicizzato; tuttavia è possibile passare ad un'altra tabella modificando direttamente il programma mediante un'istruzione POKE del BASIC. La parte LDA dell'istruzione LDA SINE,X si trova alla locazione $C103. Il byte basso dell'indirizzo iniziale della tabella che deve essere caricata è alla locazione $C104, quello alto alla locazione $C105. Per modificare l'indirizzo iniziale dei dati da caricare dobbiamo cambiare il valore contenuto in $C104. Nel caso di un'onda sinusoidale, di solito questo valore è 0; per passare ad un'onda a dente di sega è neccessario inserire il numero 80 nella locazione $C104. Caricando nella locazione il numero 160 si ottiene un'onda quadra.
Anche nella versione BBC, le tabelle di consultazione si trovano all'inizio di una nuova pagina in memoria. Quindi i byte alti degli indirizzi sono identici ed è sufficiente modificare solo i byte bassi.
Sul BBC Modello B, nel MODE 7, HIMEM si trova alla locazione &7C00. Abbassando l'inizio della memoria di tre pagine, si ottiene uno spazio più che sufficiente per le tabelle di consultazione e il programma in codice macchina.



Per il Commodore

```
;••••••••••••••••••••
;••••••••••••••••••••
;••                ••
; GENERATORE DI SUONI CBM 64
;••                ••
;••••••••••••••••••••
;••••••••••••••••••••
;
PORTA       = 56577      ;INDIRIZZO REGISTRO DATI
PASSI       = 80         ;NUMERO VALORI CAMPIONE NEL-
                         L'ONDA
*=$C000
;
;•••• PREDISPONE L'AREA PER LE TABELLE DEI DATI ••••
SINUS       *=*+PASSI
SEGA        *=*+PASSI
QUADRA      *=*+PASSI
NUMERO      *=*+2
CONTO       *=*+2
;
;•••• PROGRAMMA PRINCIPALE ••••
;
78                       SEI
AD F0 C0                 LDA NUMERO
8D F2 C0                 STA CONTO      ;PREPARA VALORE CONTATORE
AD F1 C0                 LDA NUMERO+1
8D F3 C0                 STA CONTO+1
;
                CICL2
A2 00                    LDX #$00
                CICL1
BD 00 C0        LDA SINUS,X  ;CARICA I DATI
8D 01 DD                 STA PORTA      ;LI TRASMETTE ALLA PORTA UTENTE
E8                       INX
E0 50                    CPX #PASSI
D0 F5                    BNE CICL1      ;FINE DI UN CICLO
;
;•••• DECREMENTA IL CONTATORE ••••
AD F2 C0                 LDA CONTO
38                       SEC
E9 01                    SBC #$01
8D F2 C0                 STA CONTO
AD F3 C0                 LDA CONTO+1
E9 00                    SBC #$00
8D F3 C0                 STA CONTO+1
D0 E0                    BNE CICL2      ;SE BYTE ALTO>0
A9 00                    LDA #$00
CD F2 C0                 CMP CONTO
D0 D9                    BNE CICL2      ;SE BYTE BASSO>0
58                       CLI
60                       RTS
```

Dopo aver trascritto questo listato in Assembler, lo si deve assemblare per ottenere un file esadecimale che potrà essere caricato in memoria in qualsiasi momento. Le tabelle di consultazione vengono create dal seguente programma:

```
900 REM •• PROGRAMMA DI CHIAMATA PER IL GENERATORE DI SUONI CBM ••
910:
```

Creazione delle forme d'onda

Prima di generare le forme d'onda sinusoidale e a dente di sega si deve stabilire il numero di passi campione che compongono ogni ciclo d'onda. Quindi vengono calcolati ed immagazzinati in una tabella i valori campione dell'ampiezza dell'onda; questi saranno poi caricati in sequenza nel registro dati della porta utente e inviati al convertitore D/A che li trasforma in livelli di tensione. L'impiego delle tabelle di consultazione permette di eseguire in anticipo calcoli che richiedono una notevole quantità di tempo. Questa tecnica velocizza la creazione delle varie forme d'onda e consente di ottenere numerose ottave nella gamma di frequeze che, in mancanza delle tabelle, sarebbe invece limitata a due sole ottave.
L'onda quadra può essere generata con un programma BASIC poiché il processo di creazione è relativamente semplice. Tuttavia la lentezza del BASIC riduce notevolmente la gamma delle frequenze

```
915 DN=8: REM PER CASSETTA DN=1
920 IFA=0 THENA=1:LOAD"SSOUND.HEX",DN,1
999 :
1000 REM ** PREDISPONE I VALORI CAMPIONE **
1005 S=80: REM NUMERO DI PASSI
1007 TB=12*4096: REM INIZIO DELL'AREA DATI
1008 :
1010 REM ** ONDA SINUSOIDALE **
1020 FOR I=0 TO S-1
1030 Y=127*SIN(X)+127
1040 POKE TB+I,Y
1045 X=X+2/S
1050 NEXT I
1060 :
1065 REM ** ONDA A DENTE DI SEGA **
1070 Y=255:TB=TB+S
1080 FOR I=0 TO S-1
1090 POKE TB+I,Y
1100 Y=Y-255/S
1110 NEXT I
1120 :
1125 REM ** ONDA QUADRA **
1130 Y=255:TB=TB+S
1140 FOR I=0 TO S/2-1
1150 POKE TB+I,Y
1160 NEXT I
1165 Y=0
1170 FOR I=S/2 TO S-1
1180 POKE TB+I,Y
1190 NEXT I
1999 :
2000 REM ** VISUALIZZA LE TABELLE **
2005 TB=12*4096
2010 FORI=TB TO TB+3*S-1
2020 PRINTI,I-TB,PEEK(I)
2030 NEXT
```

Dopo aver eseguito questo programma, battere il comando NEW prima di trascrivere il successivo, che spiega come usare il programma in codice macchina. Durante la sua esecuzione vengono assegnati a SYS e POKE con istruzioni BASIC gli indirizzi necessari per modificare il codice macchina. Il programma chiede all'utente di selezionare la forma d'onda voluta, quindi emette un suono tutte le volte che viene battuto un tasto.

```
10 REM ** EMISSIONE SUONO **
20 REM ** PROGRAMMA D'USO **
30 :
40 DDR=56579: POKE DDR,255: REM TUTTE LINEE IN USCITA
65 CL=49392: REM LOCAZIONE BYTE BASSO DEL CONTATORE
67 TL=49412: REM LOCAZIONE BYTE BASSO DEL PRIMO INDIRIZZO DI UNA TABELLA
   (INDICA IL TIPO D'ONDA)
70 SOUND=49396: REM INDIRIZZO INIZIALE DEL PROGRAMMA
75 REM ** SELEZIONA IL VALORE DEL CONTATORE **
80 NUM=80:NHI=INT(NUM/256):NLO=NUM-256*NHI
82 POKE CL,NLO:POKE CL+1,NHI
83 :
85 PRINTCHR$(147): REM CANCELLA LO SCHERMO
86 INPUT "TIPO D'ONDA (0) SINUSOIDALE (1) A DENTE DI SEGA (2) QUADRA";WT
87 POKE TL,WT*S
88 PRINT: PRINT "BATTERE UN TASTO QUALSIASI (RUN/STOP PER USCIRE)"
90 GETA$: IF A$="" THEN 90: REM ATTENDE LA BATTUTA DI UN TASTO
100 SYS SOUND: REM CHIAMA IL CODICE MACCHINA
110 IF A$="X" THEN 85
120 GOTO 90
```

Se non si possiede un programma assemblatore o non si conosce il linguaggio Assembly si può richiamare il programma in codice macchina usando il seguente programma di caricamento in BASIC. In questo caso si deve eliminare la linea 920 dal programma che prepara la tabella di consultazione.

```
10 REM ** PROGRAMMA DI CARICAMENTO BASIC **
20 REM ** CODICE MACCHINA **
30 FOR I=49396 TO 49449
40 READ A:POKE I,A
50 CC=CC+A
60 NEXT I
70 READ CS:IF CC<>CS THEN PRINT "CHECKSUM ERROR":END
100 DATA120,173,240,192,141,242,192
110 DATA173,241,192,141,243,192,162,0
120 DATA189,0,192,141,1,221,232,224,80
130 DATA208,245,173,242,192,56,233,1
140 DATA141,242,192,173,243,192,233,0
150 DATA141,243,192,208,224,169,0,205
160 DATA242,192,208,217,88,96
170 DATA9115:REM*CHECKSUM*
```

Per il BBC

Poiché il BBC possiede un assembler incorporato, il procedimento che associa il BASIC al codice macchina è più semplice di quello usato sul Commodore 64.

```
 5 REM ** PROGRAMMA DI SUONO BBC **
 8 MODE 7
10 HIMEM=HIMEM-&0301
20 MC%=HIMEM+1
30 DDR=&FE62: ? DDR=255: REM TUTTE LE LINEE IN USCITA
40 porta=&FE60: REM REG. DATI DELLA PORTA UTENTE
50 passi=80: REM N. PASSI IN UN'ONDA
60 iniz_tabella=MC%
70 PROCcrea_tabella
80 PROCcod_macchina
90 PROCprog_dimostrativo
999 END
1000 DEF PROCcod_macchina
1010 :
1020 FOR scelta%=1 TO 3 STEP 3
1030 P%=MC%
1060 sinusoidale=P%:P%=P%+passi
1070 sega=P%:P%=P%+passi
1080 quadra=P%:P%=P%+passi
1090 numero=P%:P%=P%+2
1100 contatore=P%:P%=P%+2
1110 [
1120 OPT scelta%
1130 \** INIZIO PROGRAMMA PRINCIPALE **
1150 .suono
1160         SEI
1170         LDA numero
1180         STA contatore
1190         LDA numero+1
1200         STA contatore+1
1220 .cicl2
1230         LDX #&00
1240 .cicl1
1250         LDA sinusoidale,x
1260         STA porta
1270         INX
1280         CPX #passi
1290         BNE cicl1
1310 \** DECREMENTA IL CONTATORE **
1320 \
1330         LDA contatore
1340         SEC
1350         SBC #&01
1360         STA contatore
1370         LDA contatore+1
1380         SBC #&00
1390         STA contatore+1
1400         BNE cicl2
1410         LDA #&00
1420         CMP contatore
1430         BNE cicl2
1440         CLI
1450         RTS
1455 ]
1460 NEXT scelta%
1480 ENDPROC
2000 DEF PROCcrea_tabella
2020 REM ** ONDA SINUSOIDALE **
2025 x=0
2030 FOR I=0 TO passi-1
2040 y=127*SIN(x)+127
2050 ?(iniz_tabella+I)=y
2060 x=x+2*PI/passi
2070 NEXT I
2090 REM ** ONDA A DENTE DI SEGA **
2100 y=255: iniz_tabella=iniz_tabella+passi
2110 FOR I=0 TO passi-1
2120 ?(iniz_tabella+I)=y
2130 y=y-255/passi
2140 NEXT I
2160 REM ** ONDA QUADRA **
2170 y=255: iniz_tabella=iniz_tabella+passi
2180 FOR I=0 TO passi/2-1
2190 ?(iniz_tabella+I)=y
2200 NEXT I
2220 y=0
2230 FOR I=passi/2 TO passi-1
2240 ?(iniz_tabella+I)=y
2250 NEXT I
2270 REM ** VISUALIZZA LE TABELLE **
2280 iniz_tabella=MC%
2290 FOR I=iniz_tabella TO iniz_tabella+3*passi-1
2300 PRINT ~I, ~(I-iniz_tabella),?I
2310 NEXT I
2330 ENDPROC
3000 DEF PROCprog_dimostrativo
3020 contatore=MC%+3*passi: REM LOCAZIONE BYTE BASSO CONTATORE
3030 tipo_onda=cicl1+1: REM LOCAZIONE BYTE BASSO DEL PRIMO INDIRIZZO DI UNA
     TABELLA (INDICA IL TIPO D'ONDA)
3040 val_contat=80
3050 cont_alto=val_contat DIV 256
3060 cont_basso=val_contat MOD 256
3070 ?contatore=cont_basso
3080 contatore?1=cont_alto
3090 CLS
3100 INPUT "TIPO D'ONDA (0) SINUSOIDALE (1) A DENTE DI SEGA (2) QUADRA";onda
3110 ?tipo_onda=onda*passi
3120 REPEAT
3125 PRINT "BATTERE UN TASTO QUALSIASI (X PER USCIRE)"
3130 A$=GET$
3140 CALL suono
3150 UNTIL A$="X"
3160 GOTO 3090
```

Lavorare sui numeri

Le operazioni aritmetiche in LOGO

Quasi tutti i dialetti LOGO sono in grado di eseguire operazioni aritmetiche intere e reali, impiegando gli operatori aritmetici +, −, * e /. Questi vengono detti operatori "infissi" perché si trovano in mezzo ai numeri su cui operano, ad esempio: 3+4. Alcuni LOGO usano anche operatori chiamati "prefissi"; in tal caso l'esempio precedente verrebbe scritto come SUM 3 4. Questa notazione ha il vantaggio di accordarsi al formato usato per scrivere comandi e altre operazioni LOGO.
Il LOGO MIT possiede solo operatori infissi ma, all'occorrenza, permette di programmare procedure aritmetiche di tipo prefisso come le seguenti SOMMA e PRODOTTO:

```
TO SOMMA :A :B
   OUTPUT :A + :B
END

TO PRODOTTO A: :B
   OUTPUT :A * :B
END
```

La "precedenza" (o "priorità") delle operazioni, ossia l'ordine in cui vengono eseguite, segue le normali regole matematiche: le prime ad essere calcolate sono le operazioni tra parentesi, seguite dalle moltiplicazioni e divisioni ed infine dalle addizioni e sottrazioni:

```
PRINT (3 + 4) * 5
PRINT 3 + 4 * 5
```

Scriviamo le precedenti operazioni in forma prefissa:

```
PRINT PRODUCT 5 SUM 3 4
PRINT SUM 3 PRODUCT 4 5
```

Nelle operazioni scritte in forma prefissa non esistono regole di precedenza, poiché la linea che le contiene viene elaborata come una qualsiasi altra linea di comandi LOGO.
L'operazione di divisione (/) produce come risultato un numero reale. Altre due istruzioni, QUOTIENT e REMAINDER, si dimostrano spesso utili per lavorare coi numeri interi.

```
QUOTIENT 47 5     produce 9
REMAINDER 47 5    produce 2
```

Un metodo molto usato per trasformare in binario un numero in base dieci è dividere il numero per 2 fino ad ottenere come risultato 0: il numero binario è composto da tutti i resti delle divisioni scritti in ordine inverso. Ad esempio, per trasformare il numero 12 nel suo equivalente binario, si scrive:

```
12/2 = 6 (resto = 0)
 6/2 = 3 (resto = 0)
 3/2 = 1 (resto = 1)
 1/2 = 0 (resto = 1)
```

Leggendo i resti in senso inverso si ottiene il binario 1100, che corrisponde al decimale 12. Le istruzioni QUOTIENT e REMAINDER permettono di scrivere semplici procedure LOGO che eseguono tale conversione: ponendo l'istruzione di stampa prima della chiamata ricursiva, i resti vengono stampati in ordine inverso.

```
TO BINARIO :X
   IF :X = 0 THEN STOP
   BIN QUOTIENT :X 2
   PRINT1 REMAINDER :X 2
END
```

Le due istruzioni INTEGER e ROUND permettono di arrotondare i numeri. INTEGER fornisce la parte intera del numero, trascurando le cifre dopo la virgola, ROUND invece arrotonda il valore all'intero più vicino, immediatamente precedente o successivo.
Le seguenti procedure calcolano l'interesse composto di un investimento, in base ad un tasso d'interesse dato. Nella procedura STAMPA.INTERESSE, INTEGER viene usato per calcolare le lire, ROUND per arrotondare i centesimi al numero intero che più gli si avvicina.

```
TO INT.COMPOSTO :CAPITALE :TASSO :ANNI
   IF :ANNI = 0 THEN STAMPA.INTERESSE
   :CAPITALE STOP
   INT.COMPOSTO :CAPITALE*(1 + :TASSO/100)
    :TASSO :ANNI−1
END

TO STAMPA.INTERESSE :DENARO
   MAKE "LIRE INTEGER :DENARO
   MAKE "CENTESIMI ROUND (:DENARO− :LIRE)−100
   (PRINT :LIRE "LIRE :CENTESIMI "CENTESIMI)
END
```

Procedure di verifica

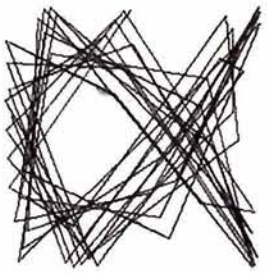
In alcune procedure abbiamo già usato i simboli matematici =, < e > come operatori logici di verifica. Possiamo adesso utilizzare le operazioni logiche ALLOF, ANYOF e NOT per realizzare altri test di controllo. ALLOF dà come risultato "vero" se entrambi i termini sono veri; ANYOF è "vero" se uno dei due termini è vero; infine NOT è "vero" se il termine è falso. Quindi avremo:

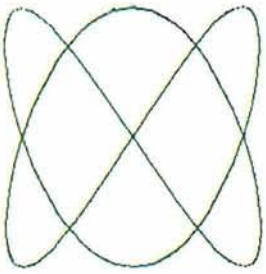
```
IF ANYOF :X > 0 :X = 0 THEN PRINT "POSITIVO
IF NOT :X < 0 THEN PRINT "POSITIVO
IF ALLOF :X > 0 :X < 100 THEN PRINT [BEETWEEN 0 AND 100]
```

L'operazione NUMBER? produce un risultato "vero" solo se il dato in ingresso è un numero. Questa istruzione viene usata nella procedura NUM.PRIMO?, che fornisce una risposta "vero" solo se il numero esaminato è un numero primo. NUM.PRIMO? controlla che il dato in ingresso sia effettivamente un numero e che sia mag-

CURVE DI LISSAJOUS



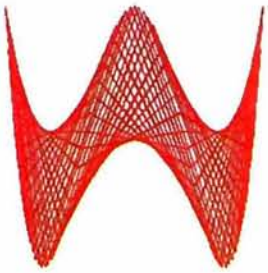






CURVE DI LISSAJOUS

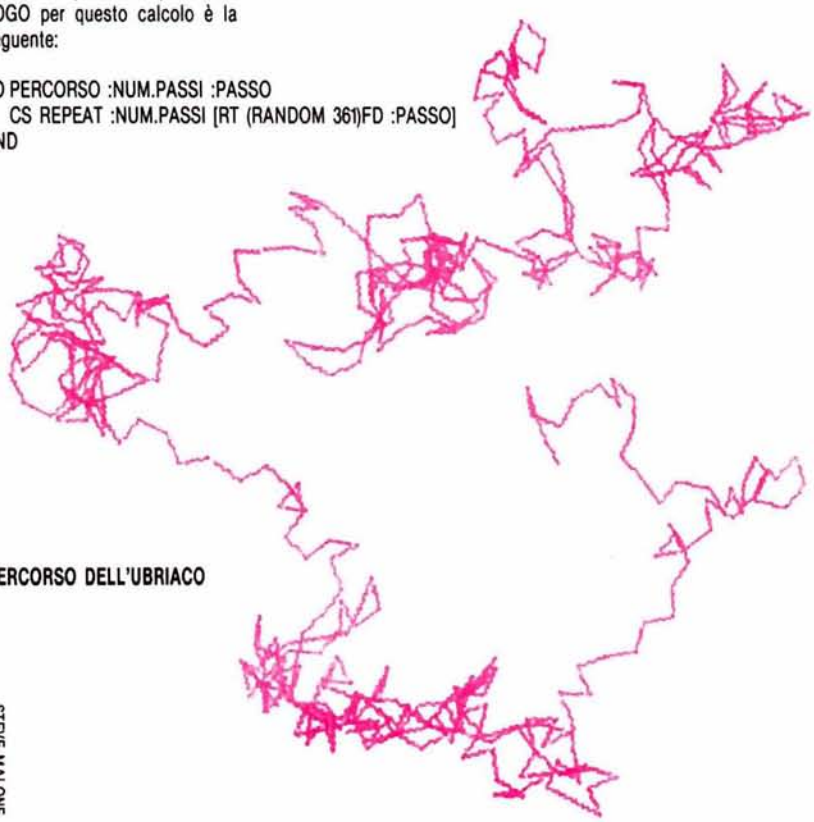




UN PASSO DOPO L'ALTRO

Il teorema del "Percorso dell'Ubriaco" afferma che, dopo un numero N di passi in direzioni del tutto casuali, la probabilità che la distanza dell'ubriaco dal punto di partenza sia minore di SQR(N) passi è maggiore di 0,5. Questa è una previsione statistica basata su un numero molto elevato di passi. La procedura LOGO per questo calcolo è la seguente:

```
TO PERCORSO :NUM.PASSI :PASSO
   CS REPEAT :NUM.PASSI [RT (RANDOM 361)FD :PASSO]
END
```



PERCORSO DELL'UBRIACO

STEVE MALONE

giore di 2. Quindi TEST.NUMPRIMO verifica se il numero è divisibile (senza resto) per un numero intero compreso tra la radice quadrata del numero in esame e 2.

```
TO NUM.PRIMO? :NUM
   IF NOT NUMBER? :NUM THEN PRINT [IL DATO NON È UN
   NUMERO] STOP
   IF :NO<2 THEN OUTPUT "FALSO
   OUTPUT TEST.NUMPRIMO :NUM INTEGER SQRT :NUM
END

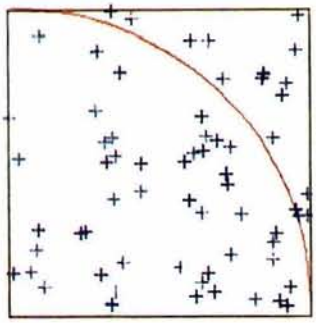
TO TEST.NUMPRIMO :NUM :DATO
   IF :DATO=1 THEN OUTPUT "VERO
   IF (REMAINDER :NUM :DATO)=0 THEN OUTPUT "FALSO
   OUTPUT TEST.NUMPRIMO :NUM :DATO—1
END
```

Numeri casuali

RANDOM n genera un numero casuale intero, compreso tra 0 ed n—1. La procedura UBRIACO fa "barcollare" la tartaruga sullo schermo variando a caso la sua direzione ad ogni passo. La variabile A definisce il massimo angolo di rotazione per ciascun passo della tartaruga. Durante l'esecuzione della procedura, la tartaruga segue traiettorie casuali, spostandosi a destra o a sinistra a seconda del valore assegnato ad A.

```
TO UBRIACO :A
   FORWARD 1
   RIGHT (—:A/2+RANDOM :A)
   DRUNK :A
END
```

Il metodo "Monte Carlo" permette di risolvere problemi matematici e statistici servendosi di numeri generati in modo casuale.



CALCOLO DI PI GRECO COL METODO MONTE CARLO

Proviamo ad usare questo metodo per calcolare, in modo approssimativo, pi greco. La figura mostra un quarto di circonferenza, tracciato all'interno di un quadrato. Il lato del quadrato misura 100 unità quindi la sua area è 100×100, mentre l'area del quarto di circonferenza viene calcolata con l'espressione (1÷4)×pi greco×100×100. Il rapporto tra le aree della circonferenza e del quadrato è pari a pi greco ÷4. Prendiamo adesso 1000 punti a caso interni al quadrato e vediamo quanti di essi "cadono" all'interno del quarto di circonferenza. Si chiami il numero ottenuto IN. Il valore di IN/1000 dovrebbe approssimativamente corrispondere al risultato dell'operazione circonferenza÷quadrato, ad esempio: pi greco÷4. Quindi se moltiplichiamo IN per 4 e lo dividiamo per 1000, dovremmo ottenere un risultato molto vicino al valore di pi greco. Le seguenti procedure eseguono l'operazione:

```
TO MC
   DRAW
   PU
   MAKE "IN 0
   MC1 1000 100 100
   (PRINT [IL VALORE DI PI GRECO E'] 0,004*(:IN))
END

TO MC1 :NUM :XNUM :YNUM
   IF: NUM=0 THEN STOP
   PUNTO.CASUALE :XNUM :YNUM
   IF INTERNO? THEN MAKE "IN :IN+1
   MC1 :NUM—1 :XNUM :YNUM
END
```

La procedura MC seleziona le condizioni, richiama MC1 e visualizza i risultati. MC1 esegue la maggior parte del lavoro, poi chiama la procedura PUNTO.CASUALE per posizionare la tartaruga ed incrementa IN se il punto si trova all'interno del quarto di circonferenza. L'esecuzione prosegue finché la procedura non viene ripetuta il numero di volte stabilito.

```
TO PUNTO.CASUALE :XNUM :YNUM
   SETXY RANDOM :XNUM RANDOM :YNUM
END

TO INTERNO?
   IF (XCOR*XCOR+YCOR*YCOR) THEN OUTPUT "VERO
   OUTPUT "FALSO
END
```

PUNTO.CASUALE assegna alla tartaruga una posizione a caso all'interno del quadrato mentre INTERNO? controlla se questa si trova all'interno dell'arco di circonferenza. L'esecuzione richiede un certo tempo, ma alla fine produce un risultato per pi greco di 3,15999.

Le curve di Lissajous

Le curve di Lissajous sono una particolare "famiglia" di curve: le coordinate x ed y di ciascuno dei loro punti sono determinate rispettivamente dalle funzioni di seno e coseno.

```
TO CURVE.LJ :COEFF1 :COEFF2 :PASSO
   DRAW PU HT
   POS :COEFF1 :COEFF2 0 PD
   CURVE.LJ1 :COEFF1 :COEFF2 0 :PASSO
END

TO POS :COEFF1 :COEFF2 :ANGOLO
   MAKE "X 100—SIN (:COEFF1*:ANGOLO)
   MAKE "Y 100—COS (:COEFF2*:ANGOLO)
   SETXY :X :Y
END

TO CURVE.LJ1 :COEFF1 :COEFF2 :ANGOLO :PASSO
   POS :COEFF1 :COEFF2 :ANGOLO
   CURVE.LJ1 :COEFF1 :COEFF2 (:ANGOLO + :PASSO) :PASSO
END
```

Logo & Logo

Le versioni LOGO LCSI usano operatori aritmetici "prefissi", il LOGO Atari possiede le istruzioni SUM e PRODUCT, i LOGO per Spectrum e per Apple possiedono rispettivamente i comandi DIV e QUOTIENT, che corrispondono all'istruzione QUOTIENT del LOGO MIT.
INT viene usato al posto di INTEGER, NUMBERP al posto di NUMBER?.
Gli operatori logici hanno gli abituali nomi AND, OR e NOT.
IF usa una sintassi diversa, cioèIF :X=0 PRINT "ZERO.
TYPE viene usato al posto di PRINT1 mentre SETPOS (seguito dai parametri delle coordinate) sostituisce SETXY.
CS prende il posto di DRAW.

ESERCIZI LOGO

1) Scrivere una procedura che calcoli la ennesima potenza di un numero. Ad esempio: POWER 4 2 deve dare come risultato 16.
2) Scrivere una serie di procedure che convertano i numeri decimali in esadecimali (usando una tecnica simile a quella adottata nella procedura BINARIO, ma dividendo per 16).
3) Scrivere una procedura PARI? che dia come risultato VERO se il numero è pari e FALSO se è dispari.
4) Servirsi del metodo Monte Carlo per calcolare l'area sottostante alla curva y=x2, per x compreso tra 0 e 10.

RISPOSTE AGLI ESERCIZI

1) Modificare il gioco in modo che sia controllabile da tastiera. Cambiare le procedure DEF.DEMON, CONTROLLO e VERIFICA. Eliminare MOV.JOY. Introdurre MOVIMENTO e LEGGI.TASTO

```
TO DEF.DEMON
   WHEN OVER :PECORA1 :RECINTO[SETSP 0]
   WHEN OVER :PECORA2 :RECINTO[SETSP 0]
   WHEN TOUCHING :PECORA1 :PECORA2[SETSP 0]
   WHEN TOUCHING :CANE :PECORA1[SETSP 0]
   WHEN TOUCHING :CANE :PECORA2[SETSP 0]
END

TO CONTROLLO
   MOVE LEGGI.TASTO
   IF :VELOCITÀ=0[VERIFICA]
   CONTROLLO
END

TO VERIFICA
   IF COND OVER :PECORA1 :RECINTO[ASK :PECORA1[BK
   10 RT 90]]
   IF COND OVER :PECORA2 :RECINTO[ASK :PECORA2[BK
   10 RT 90]]
   IF COND TOUCHING :PECORA1 :PECORA2[COLLISIONE]
   IF COND TOUCHING :CANE :PECORA1[ASK :PECO-
   RA1[RT 90]]
   IF COND TOUCHING :CANE :PECORA2[ASK :PECO-
   RA2[RT 90]]
   DEF.VELOCITÀ
END

TO MOVIMENTO :TASTO
   IF:TASTO="W[ASK:CANE[SETH 0]]
   IF:TASTO="S[ASK:CANE[SETH 90]]
   IF:TASTO="Z[ASK:CANE[SETH 180]]
   IF:TASTO="A[ASK:CANE[SETH 270]]
   IF:TASTO="Q[ASK:TARTARUGA[DRAW.CAGE]]
END

TO LEGGI.TASTO
   IF KEYP[OUTPUT RC]
   OUTPUT""
END
```

2) Nel gioco "meteorite": assegnare alla figura 1 l'immagine del meteorite ed alla figura 2 quella dell'astronave.

```
TO AZIONE
   CS FS
   SET 0 1[—100 80] 180 199
   SET 1 1[0 80]180 199
   SET 2 1[100 90] 180 199
   SET 3 2[0 — 80]90 50
   DEF.DEMON
   MOV.CASUALE 0
END

TO DEF :NUM :FIGURA :POS :DIREZIONE :VEL
   TELL :NUM SETSH :FIGURA
   PU SETPOS :POS
   SETH :DIREZIONE ST SETSP :VEL
END

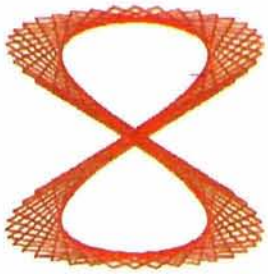
TO DEF.DEMON
   WHEN TOUCHING 0 3[COLLISIONE]
   WHEN TOUCHING 1 3[COLLISIONE]
   WHEN TOUCHING 2 3[COLLISIONE]
   WHEN 15[MOV.JOY]
END

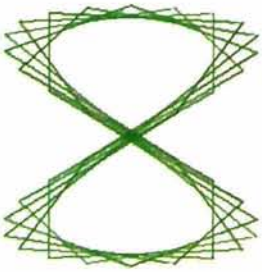
TO COLLISIONE
   TELL[0 1 2 3]
   SETSP 0 SS
   PRINT"PRINT"
   PRINT"DISTRUZIONE
END

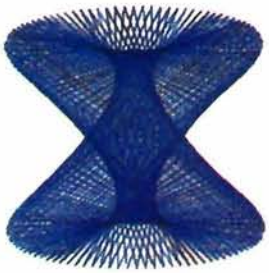
TO MOV.JOY
   IF(JOY1)
   ASK3[SETH 45—JOY1]
END

  MOV.CASUALE :NUM
 IF VELOCITÀ= 0[(PRINT"PUNTEGGIO :NUM)STOP]
 ASK CASUALE 3 [SETH 145+RANDOM 70]
 MOV.CASUALE :NUM+1
END
```

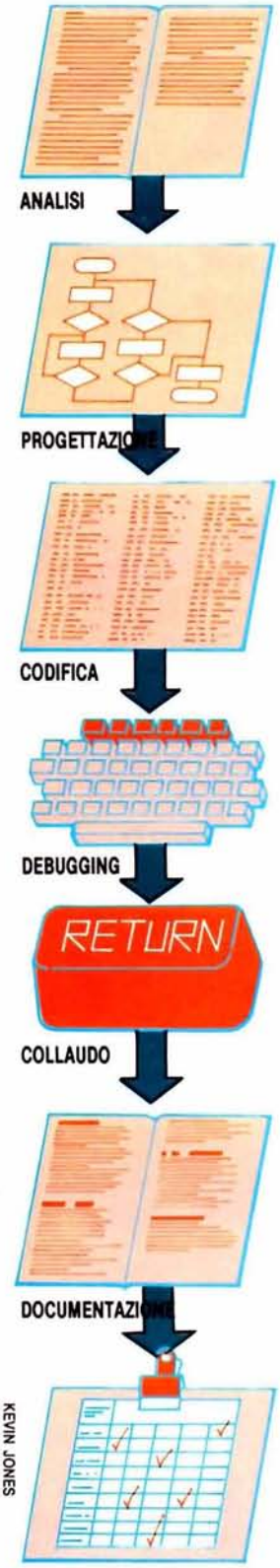
CURVE DI LISSAJOUS











KEVIN JONES

Fasi della programmazione

Osservare le regole che stanno alla base di una buona strutturazione facilita la realizzazione dei programmi in codice macchina, consentendo una maggiore chiarezza di progettazione e un risparmio di tempo in fase di debugging

Una struttura sensata

Riassumiamo i concetti fondamentali della programmazione in codice macchina

La realizzazione di programmi in linguaggi di alto livello risulta più vantaggiosa con l'impiego di tecniche come la corretta progettazione di un programma, la costruzione modulare e la programmazione strutturata. Ma una buona tecnica di programmazione rivela tutte le sue potenzialità nei linguaggi di livello inferiore, che sono i più difficili da programmare. Il linguaggio Assembly non possiede efficaci strutture di controllo, paragonabili alle istruzioni WHILE...WEND e IF...THEN...ELSE del BASIC; inoltre è privo di adeguate notazioni e di rappresentazione interna dei dati delle variabili. Ma l'aspetto più negativo è che i programmi Assembly richiedono un numero di istruzioni da sei a dieci volte superiore rispetto a quelli scritti in linguaggi di alto livello. Inoltre è molto più facile commettere errori dalle conseguenze disastrose: un errore in un singolo byte può causare la completa cancellazione di tutte le informazioni memorizzate su un dischetto! Esaminiamo alcune tecniche che rendono meno difficile la programmazione nel linguaggio Assembly del 6809.

La strutturazione e la progettazione del software non presentano particolari innovazioni (chiarezza e correttezza nell'analisi dei problemi rimangono sempre le regole principali); ciò che sta cambiando profondamente è il mondo dei microcomputer, un tempo popolato di hobbisti ed appassionati ed oggi caratterizzato da una sempre maggiore professionalità. Questa trasformazione balza agli occhi se sottoponiamo a debugging un programma in codice macchina non documentato, non strutturato ed assemblato manualmente, scritto appena qualche mese fa. Una buona programmazione dipende soprattutto da una buona progettazione e dai metodi di lavoro impiegati.

Fasi della progettazione di un programma

- **Analisi del problema.** In questa prima fase, il programmatore in Assembly deve curare con particolare scrupolo la gestione degli ingressi e delle uscite. Spesso le periferiche vengono controllate direttamente (soprattutto la tastiera e lo schermo) quindi è necessario prestare molta attenzione ai segnali di comunicazione ed alla loro sincronizzazione. Non sempre si può disporre di routine che convertono le stringhe di byte in ingresso o in uscita in un formato comprensibile al computer (ad esempio per trasformare una stringa di caratteri ASCII nel corrispondente binario di un numero ecimale). Quindi è importante specificare non solo il formato iniziale dei dati ma anche quello in cui vengono tradotti per poter essere utilizzati dal resto del programma.
- **Progettazione del programma.** Quando è possibile, le procedure attraverso cui i dati in ingresso sono trasformate in dati in uscita devono essere riunite, assieme ai dati che elaborano, in moduli logicamente indipendenti. Le tecniche usate per suddividere un programma in moduli sono essenzialmente due: la prima, detta "bottom-up" (dal basso in alto), consiste nell'individuare e raccogliere assieme tutte le operazioni che svolgono una qualche funzione nel contesto del programma. Nella seconda, chiamata "top-down" (dall'alto in basso), il programma viene via via scomposto in unità sempre più piccole, concentrandosi sulla particolare funzione di ciascun modulo e non sulle modalità di programmazione. La scomposizione termina quando un'ulteriore suddivisione non porterebbe ad alcun concreto vantaggio: solo allora si affronta il problema della codifica (programmazione) dei diversi moduli.

La progettazione bottom-up offre il notevole vantaggio di usare moduli di libreria facilmente combinabili ed il programma finale di solito consente un certo risparmio di memoria. Purtroppo questa tecnica complica le operazioni di debugging e verifica e inoltre rende il programma meno comprensibile. La progettazione top-down produce i migliori programmi strutturati e permette di verificare in modo indipendente ciascuna fase dell'elaborazione per mezzo di "stub" (brevi routine fittizie che sostituiscono sezioni del programma non ancora scritte). Gli stub ricevono dati in ingresso e generano in uscita dati del tipo richiesto, senza eseguire alcuna elaborazione. I programmi finali tendono però ad occupare più memoria dei precedenti e le routine sviluppate raramente possono essere usate in altri programmi.

In ogni modulo devono essere specificate le caratteristiche dei dati, la loro struttura e gli algoritmi. Questi vengono comunemente rappresentati con diagrammi di flusso ma molti programmatori preferiscono usare un particolare linguaggio di alto livello chiamato linguaggio simbolico o "pseudo-codice", di solito basato sul PASCAL (oppure sul BASIC). Lo pseudo-codice semplifica la progettazione di algoritmi e dati che solo in un secondo momento vengono tradotti in lin-

guaggio Assembly. In questo modo non si è costretti a progettare il programma scrivendolo direttamente in linguaggio macchina.

- **Codifica.** Se le routine sono state progettate in modo corretto, la fase di codifica è abbastanza breve e priva di problemi particolari. Quando si traduce un algoritmo di alto livello in un linguaggio di livello basso, le strutture di controllo devono essere trasferite senza cedere alla tentazione di usare indiscriminatamente istruzioni BRA o JMP. Infatti, ogniqualvolta si cerca di "schivare" la scrittura di una sezione di programma non strutturata, spesso si è costretti a lunghe operazioni di ricerca e correzione degli errori in fase di debugging. Nella figura a destra sono riportati alcuni esempi di codifica di alcune strutture di controllo (per semplificare le cose, i dati usati sono ad 8 bit).

Questo tipo di codifica comporta però un aumento delle dimensioni del programma. Quando lo spazio a disposizione è molto vasto, salvare i programmi non è problematico; i programmi brevi non sempre richiedono minor tempo di esecuzione anzi, proprio per la loro compattezza, hanno sovente tempi di sviluppo e debugging molto lunghi. Se lo spazio disponibile è limitato, la prima stesura del codice deve prevedere un tipo di strutturazione libera; solo dopo si cercherà di ridurre il programma riportandolo alla sua struttura essenziale, tenendo conto delle specifiche situazioni in cui si opera.

- **Debugging.** In questa fase il funzionamento di ogni modulo viene controllato separatamente, usando gli stub solo quando è necessario. I programmi di debugging del linguaggio Assembly sono molto diversi da quelli del BASIC. Per controllare l'esecuzione del programma, è necessario esaminare ed eventualmente correggere il contenuto delle locazioni di memoria e dei registri usati. Per fare il debugging di un programma Assembly si ricorre ad una utility che predispone e rimuove punti d'arresto (breakpoint). Questa operazione consente di eseguire il programma fino all'interruzione successiva, quindi di visualizzare e modificare sia i registri sia, se necessario, il contenuto della memoria.
- **Collaudo.** Dopo la verifica ed il debugging dei moduli, si deve ricomporre e collaudare il programma inserendo dati appropriati. Questa operazione è molto più agevole se tutti i moduli funzionano nel modo dovuto.
- **Documentazione.** La documentazione dei programmi Assembly è di importanza capitale perché questo linguaggio è molto più difficile da interpretare rispetto ai linguaggi di alto livello. In particolare occorre documentare l'uso della memoria dello stack (soprattutto quando vengono trasferiti dei parametri) e dei registri all'interno delle subroutine.
- **Manutenzione.** Un programma non usato per un lungo periodo di tempo deve essere revisionato per eliminare possibili imperfezioni o per introdurre miglioramenti. Questo tipo d'intervento evidenzia l'importanza di una corretta progettazione e di una esauriente documentazione, altrimenti la sua manutenzione a volte comporta una vera e propria riscrittura dell'intero codice.

Adesso abbiamo bisogno di un progetto a cui applicare le tecniche di progettazione descritte: un programma di debugger/monitor in codice macchina è senza dubbio quello che meglio si adatta alla nostra prima esperienza di programmazione in linguaggio Assembly. Chi ha già usato un programma assemblatore, ha un'idea di massima del tipo di facilitazioni fornite da un debugger/monitor. Un simile programma offre una serie di funzioni di editing ben note ai programmatori BASIC; permette cioè di esaminare e cambiare i contenuti della memoria.

Tale progetto è l'argomento delle prossime lezioni.

Strutture "in prestito"

Poiché l'Assembly manca di adeguate strutture di controllo, si devono usare metodi provati e verificati, provenienti da linguaggi di alto livello. Le strutture visualizzate nella figura sono chiare e ben costruite in entrambi i linguaggi di alto e basso livello ed il loro impiego esclude qualsiasi altra alternativa

STRUTTURE DI CONTROLLO

La struttura IF...THEN...ELSE

Pseudo-codice	Linguaggio Assembly
	THREE FCB 3
	.....
IF NUM1 = 3	IF LDA NUM1
THEN	CMPA THREE
routine1	BNE ELSE
	THEN .....
ELSE	* routine1
routine2	BRA ENDIF
ENDIF	ELSE .....
	* routine2
	ENDIF .....

La struttura WHILE...WEND

Pseudo-codice	Linguaggio Assembly
WHILE NUM1< =3	WHILE LDA NUM1
routine ripetuta	CMPA THREE
WEND	BGT WEND
	* routine ripetuta
	BRA WHILE
	WEND .....

La struttura REPEAT...UNTIL

Pseudo-codice	Linguaggio Assembly
REPEAT	REPEAT .....
routine ripetuta	* routine ripetuta
UNTIL NUM1<3	LDA NUM1
	CMPA THREE
	BGE REPEAT
	UNTIL .....

La struttura FOR...NEXT

Pseudo-codice	Linguaggio Assembly
FOR NUM1 = 1TO NUM2	LDA NUM2
routine ripetuta	FOR .....
NEXT NUM1	* routine ripetuta
	DECA
	BGT FOR
	NEXT .....

"Deus ex machina"

Un concetto totalmente nuovo nella produzione di software per intrattenimento

Poiché i giochi per computer hanno ormai riproposto quasi tutti i giochi di tipo tradizionale, gli autori ed i programmatori hanno avvertito la necessità di esplorare nuovi argomenti per questo genere di software. La Automata Software, famosa per la serie di giochi che hanno come protagonista Piman, si è mossa in questa direzione sviluppando un pacchetto che abbina ad un programma di gioco una cassetta musicale, contenente una colonna sonora da sincronizzare al programma.

Il gioco si chiama *Deus ex machina* ed ha richiesto sei mesi per lo sviluppo e tre di programmazione. Il protagonista della "storia" è un potente computer del futuro che, ribellatosi ai suoi creatori, partecipa alla creazione di un "essere umano" (qui chiamato "entità"). Per certi versi, il tema assomiglia alla trama del film *Generazione Proteus*. Il giocatore assume il ruolo dell'essere che passa attraverso le varie fasi della crescita (dal concepimento al suo completo sviluppo). Durante tutta la crescita, l'entità viene attaccata a più riprese da quella che viene qui chiamata la "defect police": il giocatore deve difendersi, agendo sulla tastiera o sul joystick.

Il punteggio viene calcolato partendo da un valore percentuale massimo di 99% (corrispondente ad una completa integrità dell'entità), che decresce col procedere del gioco e a seguito degli attacchi subiti. La natura degli attacchi varia nel corso del tempo e il giocatore deve cercare di reagire in modo adeguato ai diversi pericoli.

Dopo aver caricato il gioco, occorre prestare molta attenzione alla sincronizzazione della colonna sonora, in modo che le immagini visualizzate sullo schermo corrispondano esattamente alle parole ed ai suoni.

Il programma è diviso in due parti, una per ciascun lato della cassetta, per un totale di 96 Kbyte. Prima di caricare la seconda parte, il giocatore deve muovere il cursore attorno al corpo dell'entità, attivando i particolari segnali che lo circondano. Il computer non va spento e si deve nuovamente controllare la sincronizzazione della colonna sonora. Nella seconda fase si devono superare una serie di ostacoli per raggiungere un'età "avanzata", pena l'apparizione sullo schermo di chiazze rosse, che il giocatore deve cercare di cancellare.

Alla fine del gioco, indipendentemente dal punteggio ottenuto, l'entità giunge al termine della propria "vita".

L'argomento del pacchetto *Deus ex machina* èmolto particolare e, dato che il punteggio realizzato non porta ad alcuna vincita, si può partecipare al gioco oppure osservarne lo svolgimento senza intervenire. La grafica utilizzata è eccellente e ricca d'immaginazione: le figure sono molto curate e testimoniano l'attenzione profusa nel creare l'intero pacchetto.

Sia la trama del gioco sia la colonna sonora, piacevole ma non eccezionale, sono state scritte e realizzate da Mel Croucher, cofondatore dell'Automata Software. Il brano musicale che accompagna i primi attimi di vita dell'entità è cantato da Ian Dury.

Il *Deus ex machina* affronta indubbiamente un tema nuovo nella storia dei giochi per computer, propone una filosofia non violenta (niente invasori alieni da distruggere o astronavi da abbattere!) e forse non piacerà agli appassionati dei più tradizionali giochi "arcade".

Tuttavia, nonostante che il tono vagamente mistico della colonna sonora sia discutibile, la Automata Software dovrebbe essere elogiata per questo tentativo di innovazione; il *Deus ex machina* rappresenta un audace esperimento ed un passo avanti nel campo del software di gioco.

Lo sviluppo del gioco

Una parte dell'originalità di questo gioco è che il giocatore può partecipare all'azione o semplicemente osservarne lo sviluppo. Il "Deus ex machina" visualizza numerose scene, anche se alcune sono piuttosto simili. Il giocatore deve adottare una tattica flessibile, che gli consenta di preservare l'integrità dell'entità creata. Il punteggio è dato, nell'angolo destro dello schermo, da un valore percentuale che si riduce progressivamente col procedere del gioco.



L'interrogatorio

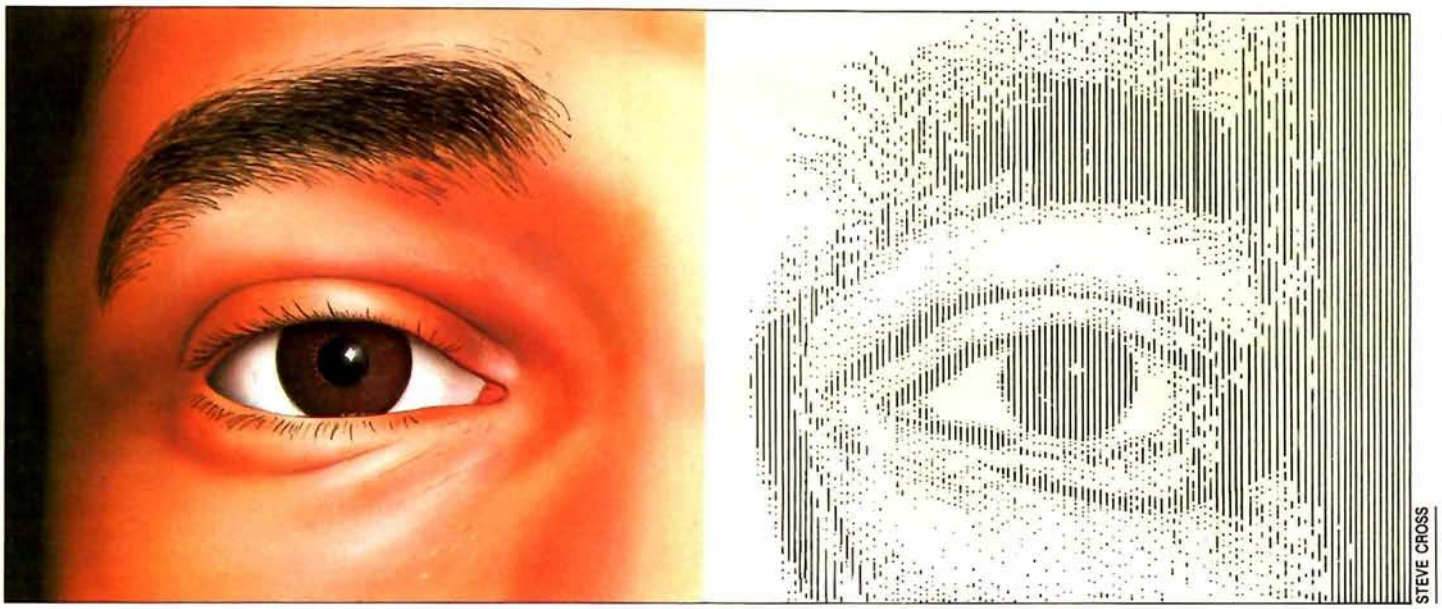
L'incubatrice

IAN McKINNELL

Deus ex machina: Per lo Spectrum a 48K
Editori: Automata Ltd (GB)
Autori: Mel Croucher, Andrew Stagg
Joystick: opzionale
Formato: su cassetta

Gli occhi del robot

Come un robot osserva e interpreta gli oggetti che lo circondano



STEVE CROSS

La vista umana e quella del robot
Il sistema di percezione visiva di cui è dotato l'uomo dipende in massima parte dall'interazione tra le varie componenti nervose controllate dal cervello. Benché un'immagine consista in una serie di sequenze più o meno luminose, impresse sulla retina dell'occhio, l'organo che coordina i segnali e che quindi "vede" nel vero senso della parola è il cervello. Anche nei robot è il "cervello" che sovraintende all'elaborazione delle immagini, che tuttavia hanno una precisione nettamente inferiore

La vista è forse di gran lunga il più importante dei cinque sensi umani, perché consente di ricavare con assoluta immediatezza tutta una serie di preziose informazioni sull'ambiente circostante. La conoscenza di quanto ci circonda diviene estremamente ardua e talvolta addirittura impossibile senza percezioni visive. Quanto detto vale anche per un robot: privo di un apparato visivo, la sua funzionalità viene fortemente limitata. Abbiamo già parlato dell'impiego di sensori di prossimità o di contatto, per rilevare la presenza di ostacoli su di un percorso. L'argomento da affrontare adesso è la possibilità di dotare i robot di un dispositivo di rilevazione visiva, dalle caratteristiche il più possibile simili a quelle umane.

L'iride, nei nostri occhi, agisce sia come lente che come diaframma, regolando la quantità di luce che va a colpire la retina, in modo da mettere a fuoco le immagini percepite. In realtà, però, l'occhio non "vede" niente: si tratta unicamente di un sofisticato dispositivo di conversione, che trasforma alcuni segnali in una forma più accettabile per il cervello. È il cervello che interpreta in modo appropriato i segnali rilevati da questo "sensore".

In modo analogo, la vista di un robot può essere suddivisa in due parti: la realizzazione di un sensore visivo adatto (l'occhio) e la realizzazione di un sistema di controllo (in genere un programma) che metta in grado il computer di interpretare correttamente i segnali ricevuti.

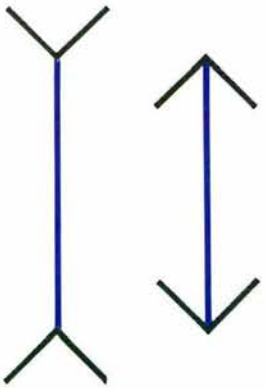
La costruzione di un occhio per robot non è affatto problematica. Al livello più basso basta una semplice cellula fotosensibile: questa può individuare sorgenti di luce o determinare il grado di illuminazione di un locale. Anche il programma di controllo è relativamente facile da realizzare, poiché la quantità e la qualità, sia dei segnali ricevuti sia delle azioni da intraprendere di conseguenza, sono molto limitate.

Tuttavia, ciò è ben lontano da quanto si intende per capacità visiva vera e propria. Più precisamente, l'obiettivo è quello di produrre un dispositivo in grado di costruire un'immagine bidimensionale dell'ambiente circostante, per consentire al "cervello" del robot un esame accurato delle informazioni ed un'azione conseguente secondo il modo di procederre del cervello umano.

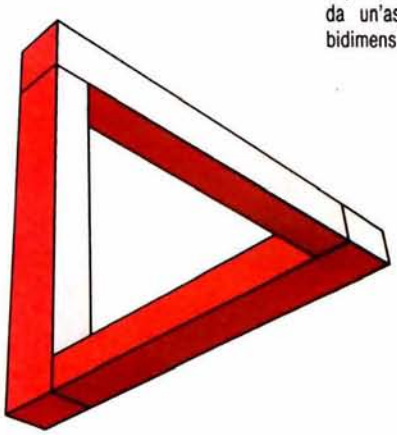
Un sistema, derivato dal precedente, può consistere nel dotare la cellula fotoelettrica di una lente, per eseguire poi la scansione meccanica di tutta una zona dello spazio circostante. Ciò consente di costruire un'immagine in base al criterio adoperato nei normali televisori. L'immagine, sottoforma di segnali digitali, può essere conservata nella memoria del computer ed esaminata in seguito. Tuttavia, questo metodo è molto lento e poco affidabile.

Nella maggior parte dei casi l'occhio del robot viene realizzato con una telecamera, che può essere del tipo comunemente impiegato nelle trasmissioni televisive, oppure progettata appositamente per robot. In questo

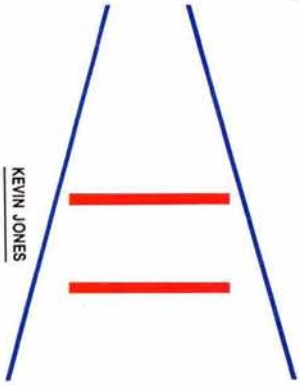
L'OCCHIO TALVOLTA INGANNA



Qual è la linea verticale più lunga? Nell'illusione ottica di Muller-Lyer è la linea a sinistra che appare più lunga, ma in effetti le due linee sono di pari lunghezza

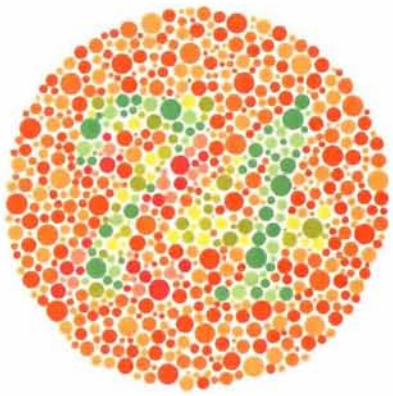


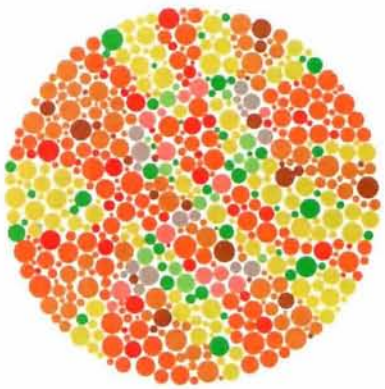
Una figura tridimensionalmente impossibile: l'inganno proviene da un'astuta rappresentazione bidimensionale



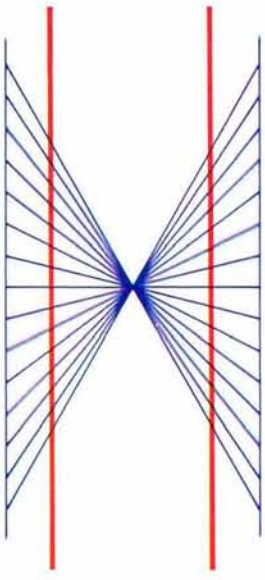
KEVIN JONES

Questa illusione ottica è chiamata "ferrovia": delle due barrette orizzontali, quella superiore sembra più lunga, ma in effetti entrambe hanno la stessa misura





Le due figure, tratte dal test di Ishihara sul daltonismo, servono per misurare eventuali irregolarità nel percepire il rosso e il verde. Nel primo esempio, i soggetti normali vedono il numero 74, mentre chi ha problemi nel distinguere il rosso e il verde vede il numero 21. Nel secondo esempio i soggetti normali non vedono niente (o appena una vaga traccia di un 2), mentre gli altri vedono chiaramente un 2



Nonostante le due linee verticali siano perfettamente diritte, esse paiono curvarsi verso l'esterno nella parte centrale: sono i raggi che partono dal punto centrale a produrre questa deformazione ottica

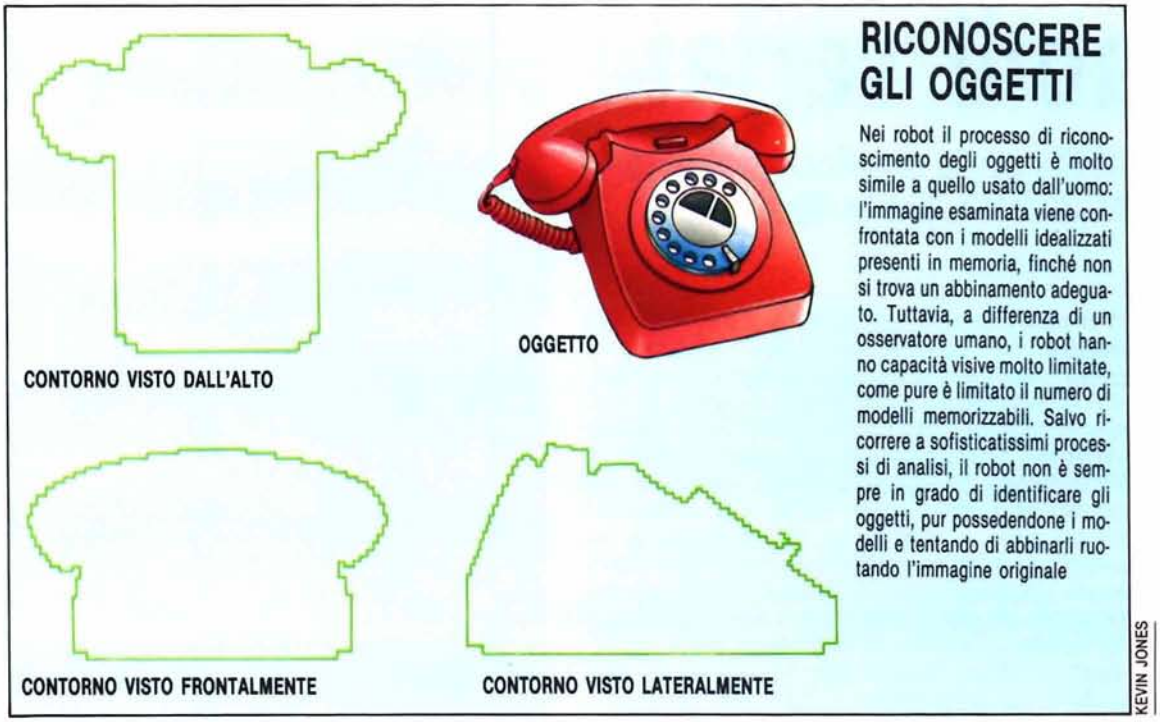
secondo caso, si ricorre all'uso di speciali chip di RAM "ottica", nei quali il valore di ciascun bit è determinato direttamente dalla quantità di luce che lo investe. Il costo per produrre questo speciale genere di memoria è in continua diminuzione, quindi la spesa per realizzare una memoria "visiva" per robot sta divenendo veramente accessibile su larga scala.

In genere, le informazioni ricavate dall'occhio del robot vengono conservate in una matrice bidimensionale, in cui il valore di ciascun elemento corrisponde alla luminosità di una particolare porzione dello scenario "osservato". La risoluzione dell'immagine dipende dalla quantità di elementi nella matrice: maggiore è il numero degli elementi, migliore è l'immagine. La gamma dei valori numerici di ciascun elemento determina le sfumature di grigio rilevabili. In accordo con la terminologia della grafica su computer, ogni elemento d'immagine è chiamato pixel, cosicché una matrice composta da 500 elementi orizzontali e 250 elementi verticali consente una risoluzione di 500 x 250 pixel, ossia 125.000 pixel. Se ad ognuno è associato un byte per il valore della luminosità, si ottengono fino a 256 sfumature di grigio (dal bianco al nero). Una simile risoluzione si avvicina abbastanza a quella offerta dai normali televisori, permettendo al computer di elaborare una nutrita serie di informazioni.

Scansione dell'immagine

I processi che consentono al cervello del robot la visione dell'immagine seguono uno schema ben preciso. Il primo passo consiste nell'aggiustare il livello dei grigi di pixel adiacenti, in modo da uniformare convenientemente l'immagine. Questa fase serve per eliminare eventuali imperfezioni dovute ad irregolarità nel segnale ricevuto. A questo punto, l'immagine viene nuovamente ripercorsa e, se la variazione di livello tra due pixel adiacenti è notevole, essa viene ancor più evidenziata. Il risultato di quest'operazione è che vengono evidenziati i contorni degli oggetti, partendo dal presupposto che è proprio lungo di essi che si ha la massima variazione di luminosità.

Viene poi compiuta un'ulteriore scansione dell'immagine, ma stavolta per comporre le sagome degli oggetti, in base ai contorni evidenziati prima. Nel far ciò il computer tenta di raggruppare logicamente tutti quei punti che possiedono determinati livelli di luminosità. Al termine dell'intero processo il computer ottiene un'immagine omogenea, con le sagome degli oggetti che si stagliano sullo sfondo.



RICONOSCERE GLI OGGETTI

Nei robot il processo di riconoscimento degli oggetti è molto simile a quello usato dall'uomo: l'immagine esaminata viene confrontata con i modelli idealizzati presenti in memoria, finché non si trova un abbinamento adeguato. Tuttavia, a differenza di un osservatore umano, i robot hanno capacità visive molto limitate, come pure è limitato il numero di modelli memorizzabili. Salvo ricorrere a sofisticatissimi processi di analisi, il robot non è sempre in grado di identificare gli oggetti, pur possedendone i modelli e tentando di abbinarli ruotando l'immagine originale

KEVIN JONES

Ma non si può ancora parlare di "vista" in senso stretto: il computer si è limitato a elaborare l'immagine, ma senza comprenderne il significato. In altre parole non "sa" ancora che cosa sta guardando.

A questo punto si possono seguire due strade diverse. La prima consiste nel programmare il computer dotandolo di una serie di regole elementari concernenti l'ambiente che lo circonda. Questo viene indicato come metodo di ricognizione visiva "bottom-up" (dal basso in alto) in quanto il robot esamina le caratteristiche più elementari per poi procedere con ipotesi più complesse.

La seconda strada prevede invece la previa memorizzazione di "modelli visivi", ossia le sagome di vari oggetti. Questo è conosciuto come metodo di ricognizione visiva "top-down" (dall'alto in basso) in quanto le parti dell'immagine vengono confrontate con i complessi modelli conosciuti, per esprimere ipotesi sull'origine di quanto compone l'immagine.

Riconoscimento degli oggetti

Per illustrare la differenza tra i due metodi presi in esame, si consideri un robot che sta osservando un tavolo. Il metodo bottom-up parte con un'analisi dell'immagine, rilevando che l'oggetto consiste di quattro componenti verticali alla cui estremità superiore si trova una vasta superficie piana. Ciò corrisponde ad una definizione pre-esistente di superficie appoggiata su quattro sostegni e alla struttura è abbinato il nome "tavolo".

Nel metodo top-down, invece, il robot esamina la struttura e si chiede se questa è un tavolo, verificando poi i singoli dettagli con il modello idealizzato di tavolo, contenuto nella sua memoria.

In linea generale, il primo metodo consente al robot di riconoscere anche oggetti mai visti prima, cogliendone alcuni aspetti principali. Tuttavia, ciò richiede un notevole impegno di programmazione per arrivare a livelli apprezzabili. Col secondo metodo vengono invece riconosciute soltanto le forme rappresentate dai modelli interni pre-esistenti. Gli oggetti di forma "nuova" vengono semplicemente ignorati.

Entrambi i metodi (e talvolta una combinazione dei due) vengono comunemente usati nel progetto del robot. Questi processi di riconoscimento sono simili a quelli adoperati normalmente dall'uomo, ma in questo caso avvengono in modo così automatico che è difficile rendersene conto.

Il senso della vista, nei robot, è dunque ben lungi dall'essere perfetto e tra i motivi di ciò ha gran peso la notevole quantità di elaborazioni occorrenti per trattare le immagini. Nell'esempio citato, ciascun passaggio implica l'esame di ben 122 Kbyte (si ricordi che ad ognuno dei 125.000 pixel è abbinato un byte). Non solo: i calcoli necessari, oltre ad essere piuttosto complessi, richiedono un certo tempo. Se il robot dovesse osservare in "tempo reale" l'ambiente che lo circonda (ossia essere in grado di rilevare eventi nel momento stesso in cui accadono) dovrebbe esaminare almeno 25 immagini al secondo. Vale a dire 3.050 Kbyte di dati da elaborare al secondo (il contenuto di una decina di floppy disk!).

Per ridurre i tempi di elaborazione non esistono che due alternative: impiegare hardware specificamente dedicato al trattamento delle immagini (permettendo così al computer di occuparsi, nel frattempo, ad altri compiti) oppure diminuire il livello di risoluzione e il numero di sfumature di grigi. In tal caso la qualità delle immagini è peggiore, ma i tempi sono notevolmente ridotti.

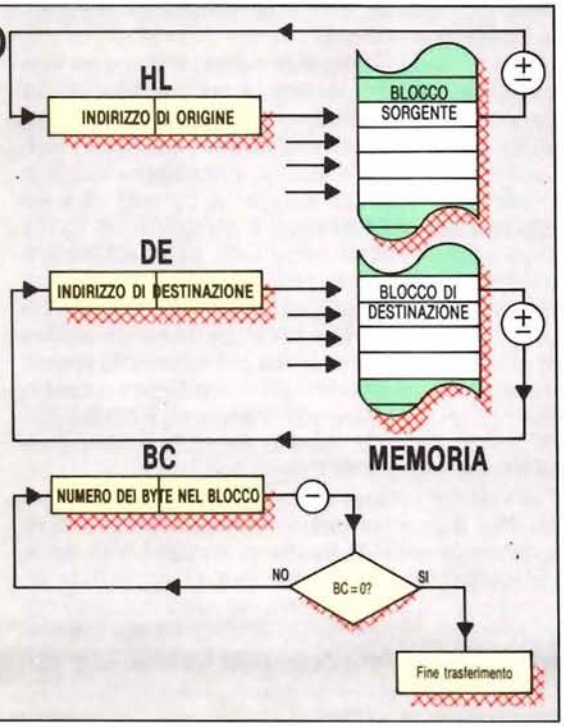
Variabili variate

Utility per la sostituzione dei nomi di variabili, per lo Spectrum

Il programma di sostituzione delle variabili esegue due diverse operazioni: esamina il programma e contemporaneamente trasferisce la versione modificata in un'area di memoria situata al di sopra di RAMTOP. Successivamente, il programma in codice macchina ritrasferisce la nuova versione nell'area del programma principale. Se la lunghezza del programma è cambiata, il programma in codice macchina corregge le dimensioni della memoria disponibile, consentendo alla nuova versione di essere accolta per intero nell'area di memoria BASIC. A differenza delle precedenti versioni, quella dello Spectrum va "accodata" al programma BASIC da trattare.
La prima parte del programma BASIC è simile a quella del programma per la ricerca delle variabili a pag. 1145, tuttavia contiene alcune nuove variabili, tra le quali Progmodif (che punta all'inizio dell'area riservata al trasferimento del programma) e Puntmodif (che indica dove verrà inserito il byte successivo nel programma modificato). Le differenze principali non riguardano la scansione del programma, ma il suo trasferimento. Ciò viene portato a termine dalla subroutine alla linea 9800, che copia il numero di byte indicati dalla variabile q e aggiorna i puntatori per il vecchio ed il nuovo programma.
I nomi delle variabili vengono trasferiti dalla sezione di programma che inizia alla linea

TRASFERIMENTO AUTOMATICO

Gli op-code LDIR ed LDDR dello Z80 sono istruzioni di trasferimento a blocchi che incrementano e decrementano automaticamente i registri. Il registro HL viene inizializzato in modo da puntare all'inizio del blocco "sorgente", mentre in DE va collocato l'indirizzo di destinazione e in BC il numero dei byte contenuti nel blocco. Quindi LDIR e LDDR copiano il byte sorgente in quello di destinazione, incrementano o decrementano automaticamente HL e DE e decrementano BC. Quando BC raggiunge lo 0 il trasferimento è completato. Si rammenti però che LDIR esegue un trasferimento "non intelligente" (vedere a pag. 1206).



KEVIN JONES

PROGRAMMA DI SOSTITUZIONE DELLE VARIABILI

```
9000 INPUT "Nome da cercare ?";LINE t$
9005 INPUT "Sostituire con ?";LINE r$
9010 FOR i=1 TO LEN (t$)
9020 IF t$(i)>="a" AND t$(i)<="z" THEN LET
     t$(i)=CHR$ (CODE(t$(i))-32)
9030 NEXT i
9040 LET SimboloREM=234
9050 LET Virgolette=34
9060 LET Nlinea=13
9070 LET Sottolin=95
9080 LET Numero=14
9090 LET PROG=23635
9100 LET Puntatesto=PEEK (PROG)+256*PEEK
     (PROG+1)
9102 LET Progmodif=46000
9105 LET Puntmodif=Progmodif
9110 LET Numlinea=256*PEEK (Puntatesto)+PEEK
     (Puntatesto+1)
9111 PRINT Numlinea
9120 IF Numlinea>=9000 THEN GO TO 9600
9130 LET q=2: GO SUB 9800
9135 LET Lung1=Puntmodif
9140 LET Lineasucc=Puntatesto+2+PEEK
     (Puntatesto)+256*PEEK (Puntatesto+1)
9150 LET q=2: GO SUB 9800
9160 LET Byte=PEEK (Puntatesto): LET q=1:
     GO SUB 9800
9170 IF Byte=Nlinea THEN GO TO 9110
9180 IF Byte<>SimboloREM THEN GO TO 9220
9190 REM Trasferisce REM senza modifica
9200 LET q=Lineasucc-Puntatesto: GO
     SUB 9800
9210 GO TO 9110
9220 IF Byte<>Virgolette THEN GO TO 9280
9230 REM  Trasferisce  le  stringhe  di
     caratteri  tra  virgolette ma si  ferma
     alla  fine  della linea se  mancano  le
     virgolette di chiusura
9235 LET q=1
9240 IF PEEK (Puntatesto+q-1)=Nlinea THEN GO
     SUB 9800: GO TO 9110
9250 IF PEEK (Puntatesto+q-1)=Virgolette
     THEN GO SUB 9800: GO TO 9160
9260 LET q=q+1
9270 GO TO 9240
9280 REM  Trasferisce un numero binario a  5
     byte
9290 IF Byte=Numero THEN LET q=5: GO SUB
     9800: GO TO 9160
9310 REM  Il  primo carattere del nome  deve
     essere una maiuscola o una minuscola
9320 IF Byte>=CODE ("A") AND Byte<=CODE ("Z")
     THEN LET c$=CHR$ (Byte): GO TO 9370
9330 REM  Usa  le  maiuscole  anziche'  le
     minuscole
9340 IF Byte>=CODF ("a") AND Byte<=CODE ("z")
     THEN LET c$=CHR$ (Byte-32): GO TO 9370
9360 GO TO 9160
9370 LET n$=""
9380 LET n$=n$+c$
9400 REM  Lettere,  cifre  o  car.  di
     sottolineatura  dopo la  prima  lettera
     del nome
9410 IF PEEK (Puntatesto)>=CODE ("A") AND
     PEEK (Puntatesto)<=CODE ("Z") THEN LET
     c$=CHR$ (PEEK(Puntatesto)): LET
     Puntatesto=Puntatesto+1: GO TO 9380
9420 REM  Usa  le  maiuscole  anziche'  le
     minuscole
9430 IF PEEK (Puntatesto)>=CODE ("a") AND
     PEEK (Puntatesto)<=CODE ("z") THEN LET
     c$=CHR$ (PEEK (Puntatesto)-32): LET
     Puntatesto=Puntatesto+1: GO TO 9380
9440 IF PEEK (Puntatesto)>=CODE ("0") AND
     PEEK (Puntatesto)<=CODE ("9") THEN LET
     c$=CHR$ (PEEK(Puntatesto)): LET
     Puntatesto=Puntatesto+1: GO TO 9380
9450 IF PEEK (Puntatesto)=Sottolin THEN LET
     c$=CHR$ (PEEK (Puntatesto)): LET
     Puntatesto=Puntatesto+1: GO TO 9380
9460 REM  Inserisce  $  alla  fine  delle
     variabili stringa
9470 IF PEEK (Puntatesto)=CODE ("$") THEN LET
     n$=n$+"$": LET Puntatesto=Puntatesto+1:
     GO TO 9500
9480 REM ( se matrice o funzione
9490 IF PEEK (Puntatesto)=CODE ("(") THEN LET
     n$=n$+CHR$ (PEEK (Puntatesto)): LET
     Puntatesto=Puntatesto+1
```

```
9500 IF n$=t$ THEN LET n$=r$
9505 LET Puntmodif=Puntmodif-1
9510 FOR p=1 TO LEN (n$)
9520 POKE Puntmodif,CODE (n$(p))
9530 LET Puntmodif=Puntmodif+1
9540 NEXT p
9550 IF n$<>r$ THEN GO TO 9160
9560 LET Lung2=PEEK (Lung1)+LEN (r$)-LEN (t$)
9570 IF Lung2>255 THEN LET Lung2=Lung2-256:
     POKE Lung1+1,1+PEEK (Lung1+1)
9580 POKE Lung1,Lung2
9590 GO TO 9160
9599 REM    Prepara il trasferimento del
     programma modificato nell'area di
     memoria principale
9600 LET Vlung=Puntatesto-(PEEK(PROG)+
     256*PEEK (PROG+1))
9610 LET Nlung=Puntmodif-Progmodif
9620 POKE 45060,Nlung-256*INT (Nlung/256)
9630 POKE 45061,INT (Nlung/256)
9660 POKE 45056,Puntatesto-256*INT
     (Puntatesto/256)
9670 POKE 45057,INT (Puntatesto/256)
9680 IF Vlung=Nlung THEN RANDOMIZE USR
     (45084)
9690 IF Vlung<Nlung THEN LET X=Nlung-Vlung
9700 IF Vlung>Nlung THEN LET X=Puntatesto-
     (Vlung-Nlung)
9710 POKE 45058,X-256*INT (X/256)
9720 POKE 45059,INT (X/256)
9730 IF Vlung<Nlung THEN RANDOMIZE USR
     45062
9740 IF Vlung>Nlung THEN RANDOMIZE USR
     45074
9800 FOR p=Puntatesto TO Puntatesto+q-1
9810 POKE Puntmodif,PEEK (p)
9820 LET Puntmodif=Puntmodif+1
9830 NEXT p
9840 LET Puntatesto=p
9850 RETURN
9900 SAVE "REPLACE" LINE 9910: SAVE
     "REPMC"CODE 45064,37: STOP
9910 CLEAR 45055: LOAD "REPMC"CODE
```

PROGRAMMA CARICATORE DEL CODICE MACCHINA

```
  10 CLEAR 45055
  20 LET a=45062
  30 FOR l=1000 TO 1040 STEP 10
  40 LET s=0
  50 FOR a=a TO a+7
  60 READ b
  70 POKE a,b
  80 LET s=s+b
  90 NEXT a
 100 READ c
 110 IF s<>c THEN PRINT "ERRORE NELLE DATA
     ALLA LINEA ";l: STOP
 120 NEXT l
1000 DATA 42,0,176,237,75,2,176,205,913
1010 DATA 85,22,24,10,42,0,176,237,596
1020 DATA 91,2,176,205,229,25,33,176,937
1030 DATA 179,237,91,83,92,237,75,4,998
1040 DATA 176,237,176,207,255,0,0,0,1051
```

PROGRAMMA ASSEMBLY DI SOSTITUZIONE

```
0000                MKROOM   EQU    $1655
0000                RCLAM1   EQU    $19E5
0000                T0       EQU    $B000
0000                T1       EQU    $B002
0000                T2       EQU    $B004
0000                ALTPRG   EQU    $B3B0
0000                PROG     EQU    $5C53
B006                         ORG    $B006
B006  2A00B0        UP       LD     HL, (T0)
B009  ED4B02B0               LD     BC, (T1)
B00D  CD5516                 CALL   MKROOM
B010  180A                   JR     COPY
B012  2A00B0        DOWN     LD     HL, (T0)
B015  ED5B02B0               LD     DE, (T1)
B019  CDE519                 CALL   RCLAM1
B01C  21B0B3        COPY     LD     HL,ALTPRG
B01F  ED5B535C               LD     DE, (PROG)
B023  ED4B04B0               LD     BC, (T2)
B027  EDB0                   LDIR
B029  CF                     RST    8
B02A  FF                     DB     $FF
```

9500. Se i nomi sono stati cambiati, viene automaticamente aggiornata la lunghezza delle linee di programma, nei due byte iniziali di ciascuna linea. Dopo che l'intero programma è stato modificato e trasferito, la parte in BASIC calcola i valori necessari al codice macchina per ritrasferire la versione modificata e li deposita negli indirizzi in cui il codice macchina si aspetta di trovarli.

Se il nuovo ed il vecchio programma hanno uguale lunghezza, il primo può essere ritrasferito nello spazio precedentemente occupato dal secondo. In questo caso, l'unico valore necessario al codice macchina è la lunghezza del programma.

Se il nuovo programma è più lungo del vecchio, allargare l'area di programma, spostando verso l'alto il programma di sostituzione delle variabili. Il programma in linguaggio macchina crea spazio extra chiamando una subroutine nelle ROM, chiamata MAKE-ROOM, all'indirizzo esadecimale 1665. Al momento della chiamata, la coppia di registri HL deve contenere l'indirizzo successivo alla posizione in cui verrà inserito il blocco e la coppia di registri BC la lunghezza di tale blocco. Il valore per HL è già pronto: è quello contenuto nella variabile Puntatesto. Il valore di BC è dato dalla differenza tra le lunghezze del vecchio e nuovo programma.

Se il nuovo programma è più breve del vecchio, il programma di sostituzione delle variabili va spostato verso il basso, usando la subroutine RECLAIM-1 (sempre in ROM) all'indirizzo esadecimale 19E5. Quando il programma esegue la chiamata, la coppia di registri HL deve contenere l'indirizzo del primo byte da lasciare intatto e la coppia di registri DE l'indirizzo del primo byte da modificare. Il contenuto di HL è ancora il valore finale della variabile Puntatesto, mentre quello di DE viene calcolato sottraendo da Puntatesto la differenza tra le lunghezze dei due programmi.

Il nuovo programma viene trasferito nell'area del programma principale tramite l'istruzione LDIR (per il trasferimento di blocchi di memoria). L'indirizzo iniziale dell'area del programma modificato (sorgente) viene caricato in HL, l'indirizzo iniziale dell'area del programma principale (destinazione) in DE e la lunghezza del programma modificato in BC. L'istruzione LDIR esegue l'intero trasferimento programma, un byte alla volta.

Le ultime due linee del programma in linguaggio Assembly ricorrono a una routine ROM per la visualizzazione di messaggi vari. Essa viene chiamata con l'istruzione RST 8, immediatamente seguita da un byte contenente il codice del messaggio desiderato. Questo valore corrisponde al numero associato al messaggio, diminuito di 1. L'esadecimale FF (—1 in decimale) corrisponde al messaggio OK (programma completato), il valore 0 si riferisce al messaggio d'errore NEXT without FOR e via dicendo. Il programma in codice macchina si conclude con RST 8 e non con l'abituale istruzione RET, evitando così di ritrasferire il controllo al programma BASIC spostato in precedenza.

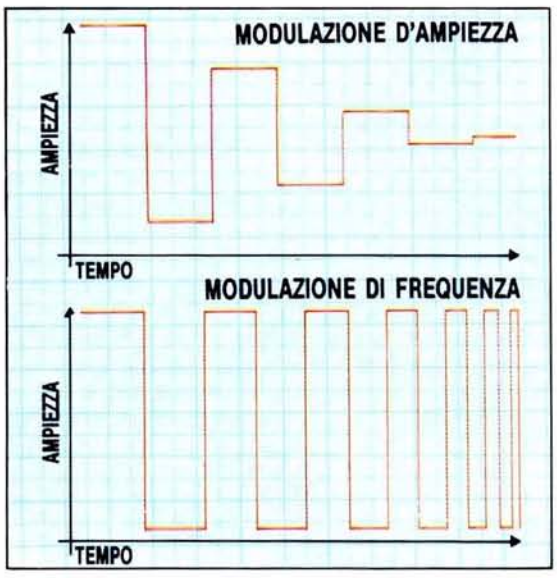
Una risposta a ... tono

Due parametri per il nostro generatore musicale: il volume e l'altezza

Il volume di un suono è determinato dall'ampiezza delle oscillazioni dell'onda che lo generano. In altre parole il volume dipende dallo scarto tra i valori massimo e minimo dell'onda. Questa proprietà di un'onda sonora è detta, appunto, "ampiezza".
Usando un programma BASIC per fare "oscillare" i valori che vengono caricati nel registro dati della porta utente, possiamo dimostrare come sia facile controllare l'ampiezza in un'onda di tipo digitale.

```
10 REM ** VARIAZ. FREQ/AMPIEZZA C64 **
20 :
25 DDR = 56579:REGDAT = 56577
30 POKE DDR,255:REM TUTTE LE LINEE IN USCITA
40 FOR I = 255 TO 0 STEP = -15
50 FOR J = 1 TO 100
60 POKE REGDAT,I:POKE REGDAT,0
70 NEXT J,I
800 REM ** PROGRAMMA DIMOSTRATIVO **
805 :
806 SU$ = CHR$(145)
810 DIV = 49798: REM LOCAZ. FATTORE AMPIEZZA
820 DEL = 49799: REM LOCAZ. FATTORE RITARDO
830 TME = 49800: REM LOCAZ. FATTPRE DURATA
840 CALL = 49801: REM INIDIRIZZO INIZIALE DEL PROGRAMMA
850 :
860 DDR = 56577:POKEDDR,255:REM TUTTE LE LINEE IN USCITA
870 :
880 PRINTCHR$(147): REM RIPULISCE SCHERMO
890 PRINT INPUT"FATTORE AMPIEZZA 0—7";AF
900 IF AF<0 OR AF>7 THEN PRINT SU$:SU$::GOTO930
910 POKE DIV,AF
920 :
930 PRINT:INPUT"FATTORE DI RITARDO 1—101";DF
940 IF DF<1 OR DF>101 THEN PRINT SU$;SU$;:GOTO930
945 POKE DEL,DF
950 :
960 PRINT:INPUT"FATTORE DI DURATA 0—15";TF
970 IF TF<0 OR TF>15 THEN PRINTSU$;SU$;:GOTO960
980 POKE TME,TF
990 :
1000 SYS CALL
1010 GETA$:IFA$ = "" THEN 1010
1020 IFA$ = "X" THEN 880:REM RICOMINCIA DA CAPO
1030 GOTO 1000REM UN ALTRO BIP
```

All'inizio del programma viene generata una semplice onda quadra che oscilla tra 255 e 0. Ciò significa che l'ampiezza dell'onda è 255. Durante l'esecuzione del programma, il valore massimo inserito nel registro dati viene decrementato in scatti di 15: parallelamente decresce l'ampiezza. Questa riduzione porta ad un graduale calo di volume del suono emesso dall'amplificatore o dalla cuffia che riceve il segnale. Perciò il volume di un suono sintetizzato in modo digitale può essere controllato restringendo la gamma dei valori caricati nel registro dati della porta utente.
Il tono (o altezza) di una nota dipende dalla frequenza dell'onda che la genera, ossia dal numero di cicli al secondo: maggiore è il numero di onde prodotte per secondo, maggiore è l'altezza della nota emessa.
Esistono due metodi per controllare la frequenza con una tecnica digitale. Il primo è aumentare la frequenza a partire da un valore base, usando un numero decrescente di valori campione dell'onda. Se l'onda è stata suddivisa in 100 campionature, il programma in codice macchina impiega un certo tempo a caricare in sequenza i loro valori nel registro dati (cioè a generare un'onda), quindi il tempo impiegato determina il numero di onde prodotte in un secondo. Di conseguenza, la frequenza del suono emesso dipende dal numero di campioni impiegati.
Per raddoppiare la frequenza, il programma potrebbe caricare nel registro solo un valore ogni due (il primo valore, saltare il secondo, caricare il terzo, saltare il quarto e così via); per triplicarla, dovrebbe invece saltare 2 valori ogni volta. Ma ci sono due inconvenienti: innanzi tutto è difficile modificare di poco la frequenza, senza distorcere il profilo dell'onda. Inoltre, via via che la frequenza cresce, l'onda generata somiglia sempre meno a quella originale, perché viene usato un numero sempre minore di campioni.
Il secondo metodo inizia con un ciclo iterativo, che legge il più rapidamente possibile i valori campione dell'onda, producendo una frequenza massima, che viene poi messa a punto inserendo nel ciclo brevi fasi di ritardo. Questo sistema consente un più accurato controllo della frequenza, ma prevede l'impiego di un ristretto numero di valori campione dell'onda, in modo da consentire una frequenza massima ragionevolmente elevata.
Con quest'ultimo metodo abbiamo sviluppato un programma in codice macchina, per controllare sia la frequenza sia il volume.
Il sistema migliore per ridurre l'ampiezza di un'onda, senza alterarne il profilo, è dividere per una costante tutti i valori campione contenuti nella sua tabella di consultazione. Si possono seguire due strade: eseguire la divisione



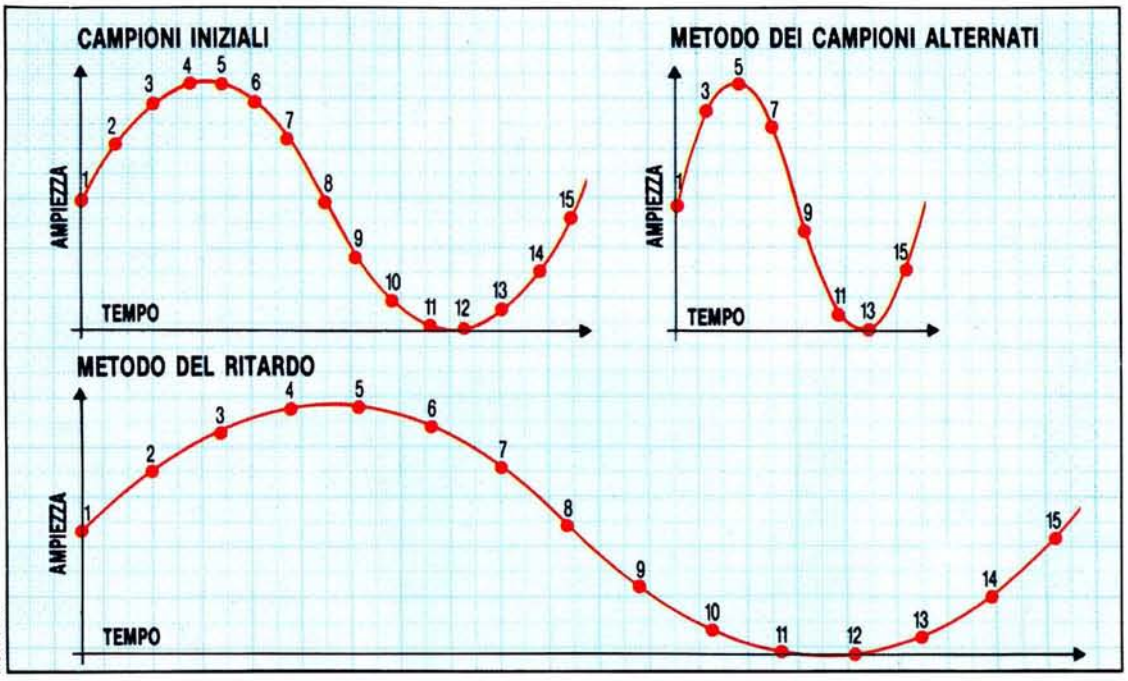
Modulazione dei suoni

Le onde possono essere modulate sia in frequenza sia in ampiezza. La frequenza modifica l'altezza della nota emessa e dipende dal numero di cicli generati ogni secondo. L'ampiezza modifica invece il volume del suono e corrisponde alla differenza tra i valori massimo e minimo all'interno di un ciclo. Il grafico mostra come modulare l'ampiezza, in modo che il volume del suono prodotto diminuisca gradualmente, e la frequenza, in modo da generare un suono di altezza crescente

dopo che i valori sono stati caricati nell'accumulatore, o prima che vengano depositati nel registro dati, oppure prima dell'esecuzione del ciclo principale del programma. Il primo metodo accresce il tempo necessario per eseguire ciascun passaggio del ciclo principale, limitando notevolmente la frequenza massima ottenibile. Il secondo, adottato nel nostro programma, divide i valori nella tabella di consultazione per una costante ed inserisce i risultati in una tabella di nuova creazione. La nuova tabella viene quindi letta dal ciclo principale del programma per generare l'onda. Il metodo di divisione è piuttosto rudimentale: un fattore d'ampiezza indica quante volte il valore in tabella deve essere fatto scorrere verso destra. Poiché uno scorrimento a destra corrisponde a una divisione per due, usando un fattore d'ampiezza "n" ogni spostamento dividerà il valore in tabella per 2n.
Quando il fattore d'ampiezza è 0 si fa in modo che il programma usi l'indirizzo della tabella di consultazione originale e non quello della tabella con i valori divisi.
L'esecuzione del ciclo principale può essere rallentata inserendo un ciclo di ritardo, la cui durata è fissata con precisione. Di solito si usa decrementare un numero a 8 bit (in uno dei registri d'indice) oppure un numero a 16 bit (in memoria) partendo da un determinato valore finché esso arriva a 0. Nel nostro caso, per generare un ritardo adeguato, è sufficiente un numero a 8 bit.
Tuttavia, il problema fondamentale non è quello di ottenere ritardi di lunga durata (che creano le frequenze più basse), bensì ritardi di durata minima per la produzione delle frequenze più alte. A pagina 1212 abbiamo usato un'onda suddivisa in 80 passi. Il ciclo di ritardo di cui abbiamo bisogno deve produrre un rallentamento nel tempo d'esecuzione tale da rendere superfluo un così alto numero di passi. La seguente tabella contiene il listato del codice da inserire nel ciclo principale.

```
MAIN                                  Tempo in cicli macchina
        LDX #$00                      2
PRXVAL
        LDA TABAMP,X                  4
        LDY RITAR                     4
ANCORA
        DEY                           2
        BNE ANCORA                    3 (2 se salto non eseguito)
        STA PORTA                     4
        INX                           2
        CPX #PASSI                    2
        BNE PRXVAL                    3 (2 se salto non eseguito)
```

Il numero complessivo dei cicli macchina necessari viene calcolato con la seguente formula: 2 + (4 + 4 + (2 + 3) x ritardo — 1 + 4 + 2 + 2 + 3) x passi — 1 = 1 + (18 + 5 x ritardo) x passi. Un valore minimo di ritardo di 1 genera la frequenza massima nel modo seguente: frequenza max = 1000000 ÷ (1 + 23 x passi).
Per una frequenza massima di circa 3000 Hz, questa formula dà come risultato 15 passi che corrisponde al numero di campioni d'onda necessari per generare una frequenza massima conveniente. Usando 15 passi, la formula iniziale può essere scritta come segue: num. cicli macchina = 271 + 75 x ritardo.
Il valore di ritardo necessario per ottenere una frequenza massima di 128 Hz (circa due ottave sopra il DO centrale) è 101. Questo all'interno di un registro d'indice.
L'ultimo problema sollevato dalle operazioni di modifica della frequenza è che, per un dato numero di ripetizioni del ciclo di ritardo, la durata prevista del suono emesso decresce al crescere della frequenza. Infatti, col crescere della frequenza, il ciclo principale viene eseguito in un tempo sempre minore. Per bilanciare questa accelerazione occorre introdurre nel programma un'operazione che, indipendentemente dalla frequenza impiegata, calcoli il numero di ripetizioni necessarie per ottenere un suono della durata voluta. Se il parametro di durata è di 1/50 di secondo, il numero



Allungare l'onda

La frequenza di un'onda "campionata" con una tecnica digitale può essere modificata usando un minor numero di campioni o inserendo un ritardo tra ciascun valore e il successivo. Se la tabella di consultazione originale contiene 15 campioni d'onda, si può raddoppiare la frequenza dell'onda in uscita utilizzando i campioni in modo alternato. Al contrario, per dimezzare la frequenza, si possono usare tutti e 15 i campioni, inserendo un ritardo che raddoppi il tempo impiegato per generare l'intera onda. Il primo metodo consente di usare un maggior numero di campioni alle frequenze più basse, ma offre un controllo meno preciso della frequenza. Il secondo consente una regolazione della frequenza più accurata, ma consente l'impiego di un minor numero di valori campione

di ripetizioni necessarie per ottenere il ritardo previsto è dato da: 1000000 ÷ (50 x (271 + 75 x ritardo)). In codice macchina, il calcolo di questa formula si dimostra difficile e lento, quindi è preferibile sostituirlo con una tabella di consultazione programmata in BASIC che conterrà i valori di ripetizione, calcolati con la formula precedente, corrispondenti ai diversi valori di ritardo (per esempio, da 1 a 109). In questo modo la routine in codice macchina estrae l'appropriato valore di ripetizione e lo deposita in un registro, dove verrà poi decrementato. Si ottiene così un suono con durata di 1/50 di secondo.

I possessori di Commodore che hanno un programma assemblatore possono battere sulla tastiera il codice sorgente ed assemblarlo per ottenere un programma oggetto da archiviare. Il programma di chiamata lo ricarica in memoria prelevandolo da una cassetta o da un dischetto e crea le tabelle di consultazione. A questo punto si deve battere il comando NEW, inserire il programma dimostrativo in BASIC (che fornisce i vari indirizzi usati dal programma in codice macchina) ed eseguirlo con un RUN, dopo essersi accertati che il buffer ed il convertitore D/A siano collegati nel modo descritto a pag. 1212. In mancanza un assembler è necessario trascrivere ed eseguire il seguente programma di caricamento BASIC, prima di eseguire il programma di chiamata (dal quale si dovranno eliminare le linee 45 e 50).

I possessori di BBC possono trascrivere ed eseguire la versione BBC senza modifiche.

```
10 REM CARICATORE DEL PROGRAMMA FREQ/AMP
20 :
30 FOR I = 49801 TO 49911
40 READ A:POKE I,A
50 CC = CC + A
60 NEXT I
70 READ CS:IF CC<>CS THEN PRINT "ERRORE DI CHECKSUM":STOP
100 DATA172,134,194,208,8,169,0,141
110 DATA204,104,76,174,194,169,15,141
120 DATA204,194,162,15,172,134,194,169
130 DATA0,194,24,106,157,15,194,136
140 DATA208,248,202,16,239,120,169,0
150 DATA141,133,194,174,135,194,169,30
160 DATA194,172,136,194,240,7,10,46
170 DATA133,194,136,208,249,141,132
180 DATA194,162,0,169,15,194,172,135
190 DATA136,208,253,141,1,221,232
200 DATA224,15,208,239,173,132,194,56
210 DATA233,1,141,132,194,173,133,194
220 DATA233,0,141,133,194,208,218,169
230 DATA0,205,132,194,208,211,88,96
240 DATA16186:REM *CHECKSUM*
```

```
10 REM **      PROGRAMMA CHIAMANTE      **
20 REM **                E              **
30 REM **      PREPARAZIONE TABELLE     **
40 :
45 DN = 8:REM IF CASSETTE DN = 1
50 IF A = 0 THEN A = 1:LOAD"FREQ.HEX",DN,1
60 :
70 REM **    PREPARA TABELLA DELLA FORMA    **
75 :
80 S = 15:TB = 12 * 4096 + 2 * 256
90 FOR I = 0 TO S = 1
100 Y = 127 * SIN(X) + 127
110 POKE TB + I,Y
120 X = X + 2/S
130 NEXT I
140 :
150 REM **    PREPARA TABELLA FREQ/RITARDI    **
160 :
170 TB = TB + 2 * S
180 FOR D = 0 TO 101
190 TV = 10^6/(50 * (271 + 75 * D))
200 POKETB + D,TV
210 NEXT D
```

```
;++++++++++++++++++++++++++
;++                      ++
;++    MODULATORE DI     ++
;++      FREQUENZA       ++
;++    E DI AMPIEZZA     ++
;++    COMMODORE  64     ++
;++                      ++
;++++++++++++++++++++++++++
;
;
PASSI   = 15            ;N.PASSI CICLO ONDA
PORTA   = 56577
;
**      $C200
TBFORM  ***+PASSI       ;TABELLA CONSUL. ONDA
TABMAP  ***+PASSI       ;TABELLA AMPIEZZA
TABRIT  ***+102         ;TABELLA FREQ/RITARDI
CONTO   ***+2           ;CONTATORE
DIVIS   ***+1           ;FATTORE DI DIVISIONE
RITAR   ***+1           ;FATTORE DI RITARDO
DURAT   ***+1           ;FATTORE DI DURATA
;
;++++ DEFINISCE TABELLA AMPIEZZA ++++
;
        LDY DIVISN
        BNE CONT
        LDA #<SHAPTB    ;MODIFICA IL PROGRAMMA
        STA NEXVAL+1    ;PER LEGGERE TBFORM
        JMP INITC
;
CONT
        LDA #<AMPTAB
        STA NEXVAL+1
        LDX #PASSI
NEXT
        LDY DIVISN
        LDA SHAPTB,X
MORE
        CLC
        ROR A
        STA AMPTAB,X
        DEY
        BNE MORE
        DEX
        BPL NEXT
;
;++++ DEFINISCE VALORE CONTATORE ++++
;
INITC
        SEI
        LDA #$00
        STA COUNT+1     ;INIZ. BYTE ALTO CONT.
        LDX DELAY
        LDA LOOPTB,X
        LDY TIME
        BEQ NOMULT
MULT
        ASL A
        ROL COUNT+1
        DEY
        BNE MULT
NOMULT
        STA COUNT
;
;++++ LOOP PRINCIPALE DEL PROGRAMMA ++++
;
MAIN
        LDX #$00
NEXVAL
        LDA AMPTAB,X
        LDY DELAY
MORDEL                  ;LOOP RITARDO
        DEY
        BNE MORDEL
;
        STA PORTA
        INX
        CPX #PASSI
        BNE NEXVAL
;
;+++++ DECREMENTA CONTATORE ++++
;
        LDA COUNT
        SEC
        SBC #$01
        STA COUNT
        LDA COUNT+1
        SBC #$00
        STA COUNT+1
        BNE MAIN
        LDA #$00
        CMP COUNT
        BNE MAIN
        CLI
        RTS
```

```
15REM ****************************
20REM **                        **
25REM **          BBC           **
30REM **  FREQUENZA & AMPIEZZA  **
40REM **                        **
50REM ****************************
90MODE 7
95passi=15:porta=&FE60
97ddr=&FE62:? ddr=255:REM TUTTE LINEE USCITA
100HIMEM=HIMEM-&101:REM RISERVA SPAZIO TABELLA
110tab_forma=HIMEM+1
112tab_ampiezza=tab_forma+passi
114tab_cicli=tab_ampiezza+passi
115PROCprep_tabelle
120PROCassembla_prog
140REM ** PROG DI VERIFICA BASIC **
160CLS
```

```
 180IF AF<0 OR AF>7 THEN 170
 190?fatt_divisione=AF
 200PRINT:INPUT"FATTORE DI RITARDO 1-101";DF
 210IF DF<1 OR DF>101 THEN 200
 220?fatt_ritardo=DF
 230PRINT:INPUT"FATTORE DI DURATA 0-15";TF
 240IF TF<0 OR TF>15 THEN 230
 250?fatt_durata=TF
 265REPEAT
 270CALL freq
 280A$=GET$
 290UNTIL A$="X"
 300GOTO 160:REM RILANCIA
 900END
 999:
1000DEF PROCassembla_prog
1005DIM MC% &FF
1010FOR opt%=0 TO 3 PASSI 3
1020P%=MC%
1030conto=P%:P%=P%+2
1040fatt_divisione=P%:P%=P%+1
1050fatt_ritardo=P%:P%=P%+1
1060fatt_durata=P%:P%=P%+1
1070 [
1075OPT opt%
```

```
1080\** DEFINISCE FATTORE DI AMPIEZZA **
1090\
1095.freq
1100    LDY fatt_divisione
1110    BNE cont
1120    LDA #tab_forma MOD 256
1130    STA prxval+1
1140    JMP initc
1150\
1160.cont
1170    LDA # tab_ampiezza MOD 256
1180    STA prxval+1
1190    LDX #passi
1195.prox
1200    LDY fatt_divisione
1210    LDA tab_forma,X
1220.ancor
1230    CLC
1240    ROR A
1250    STA tab_ampiezza
1260    DEY
1270    BNE ancor
1280    DEX
1290    BPL prox
1300\
1310\** DEFINISCE VALORE CONTATORE **
1320\
1330.initc
1340    SEI
1350    LDA #0
1360    STA conto+1
1370    LDX fatt_ritardo
1380    LDA tab_cicli,X
1390    LDY fatt_durata
1400    BEQ nomolt
1410.molt
1420    ASL A
1430    ROL conto+1
1440    DEY
1450    BNE molt
1460.nomolt
1470    STA conto
1480\
1490\** CICLO PRINCIPALE DEL PROGRAMMA **
1500\
1510.main
1520    LDX #0
1530.prxval
1540    LDA tab_ampiezza,X
1550    LDY fatt_ritardo
1560.ritar
1570    DEY
1580    BNE ritar
1590\
1600    STA porta
1610    INX
1620    CPX #passi
1630    BNE prxval
1640\
1650\** DECREMENTA IL CONTATORE **
1660\
1670    LDA conto
1680    SEC
1690    SBC #1
1700    STA conto
1710    LDA conto+1
1720    SBC #0
1730    STA conto+1
1740    BNE main
1750    LDA #0
1760    CMP conto
1770    BNE main
1780    CLI
1790    RTS
1800]
1810NEXT opt%
1820ENDPROC
1999:
2000DEF PROCprep_tabelle
2005x=0
2010FOR I=tab_forma TO tab_forma+passi-1
2020y=127*SIN(x)+127
2030? I=y
2040x=x+2*PI/passi
2050NEXT I
2060:
2070FOR ritardo=0 to 101
2080val_ciclo=10^6/(50*(271+75*ritardo))
2090tab_cicli?ritardo=val_ciclo
2100NEXT ritardo
2120ENDPROC
```

La Snap

Dotiamo il nostro computer di un sistema visivo, con l'EV1 Video System

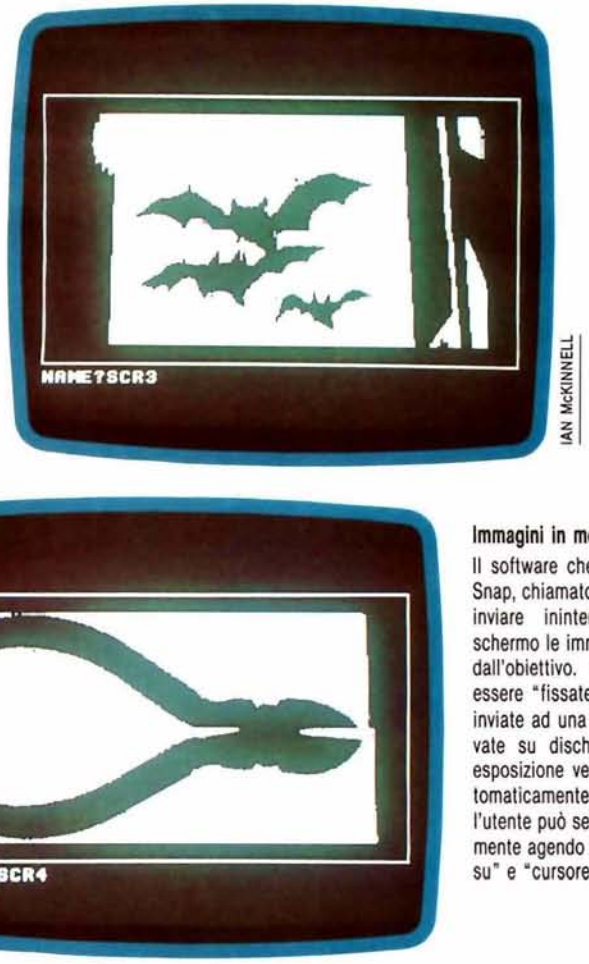
Il sistema EV1 Video, comunemente noto col nome Snap, viene venduto ad un prezzo più che accessibile, anche per l'utente medio di home computer. Poiché è corredato di software, il sistema costa meno della metà di qualsiasi altra interfaccia video dello stesso livello. La camera ottica Snap si compone di una specie di piccola macchina fotografica elettronica, da collegare alla porta utente del BBC tramite un cavo a nastro, e di un pacchetto di software su cassetta o su dischetto. Inserendo il giusto comando, il sistema permette di riprodurre sullo schermo del computer l'immagine di un oggetto o di un paesaggio. La Snap può essere usata come passatempo, ma consente anche applicazioni più serie: può infatti costituire la base di un allarme antifurto "intelligente" o essere usata come "occhio" di un robot per il riconoscimento degli oggetti.

Il funzionamento della Snap si basa su una originale applicazione elettronica: all'interno della macchina fotografica, dietro l'obiettivo, è situato un chip di memoria RAM da 32 Kbyte che, diversamente dai normali chip, possiede nella parte superiore una finestra trasparente. L'immagine, messa a fuoco dall'obiettivo, cade sulla lamina di silicio del chip, che presenta una matrice di 256 x 128 piccole celle di memoria invisibili a occhio nudo. Quando una cella viene esposta alla luce perde rapidamente la propria carica, essendo la velocità di scaricamento direttamente proporzionale all'intensità della luce che la colpisce. Tutte le celle della matrice vengono caricate, inizialmente, depositandovi un particolare valore, in altre parole queste vengono "accese". In seguito, ogni cella viene riesaminata per constatarne lo stato. Quelle che non hanno ricevuto luce contengono lo stesso valore iniziale (livello on), mentre quelle colpite da una luce sufficientemente intensa si sono scaricate e la lettura del loro valore dà come risultato 0 (off). Il software di sistema accende un punto sullo schermo nella posizione corrispondente ad ogni cella scarica. In questo modo l'immagine focalizzata sul chip viene trasferita allo schermo del Micro BBC.

L'intervallo di tempo compreso tra il momento dell'accensione di tutte le celle del chip e la successiva lettura determina il "tempo d'esposizione" dell'immagine. Se la luminosità esterna è sufficiente, la Snap è in grado di generare più immagini al secondo: in una stanza illuminata normalmente, la macchina può fotografare un'immagine in meno di un secondo.

La risoluzione della Snap, che dipende dalla matrice di 256 x 128 celle di memoria del chip RAM, non è particolarmente elevata (è circa la stessa dei Modi 2 o 5 del video BBC). Tuttavia le immagini prodotte sono di qualità discreta, paragonabili alle fotografie stampate sui quotidiani. Un altro fattore che influisce sulla qualità dell'immagine è determinato da una particolarità del chip di memoria. La matrice delle celle è suddivisa in due elementi affiancati e la sottile fessura che li separa provoca la presenza di una linea orizzontale sullo schermo. Infatti, la luce focalizzata sulla fessura non viene catturata dalle celle e non può quindi essere riprodotta sullo schermo. Fortunatamente, questa sottile linea è appena percettibile e in alcuni casi del tutto trascurabile.

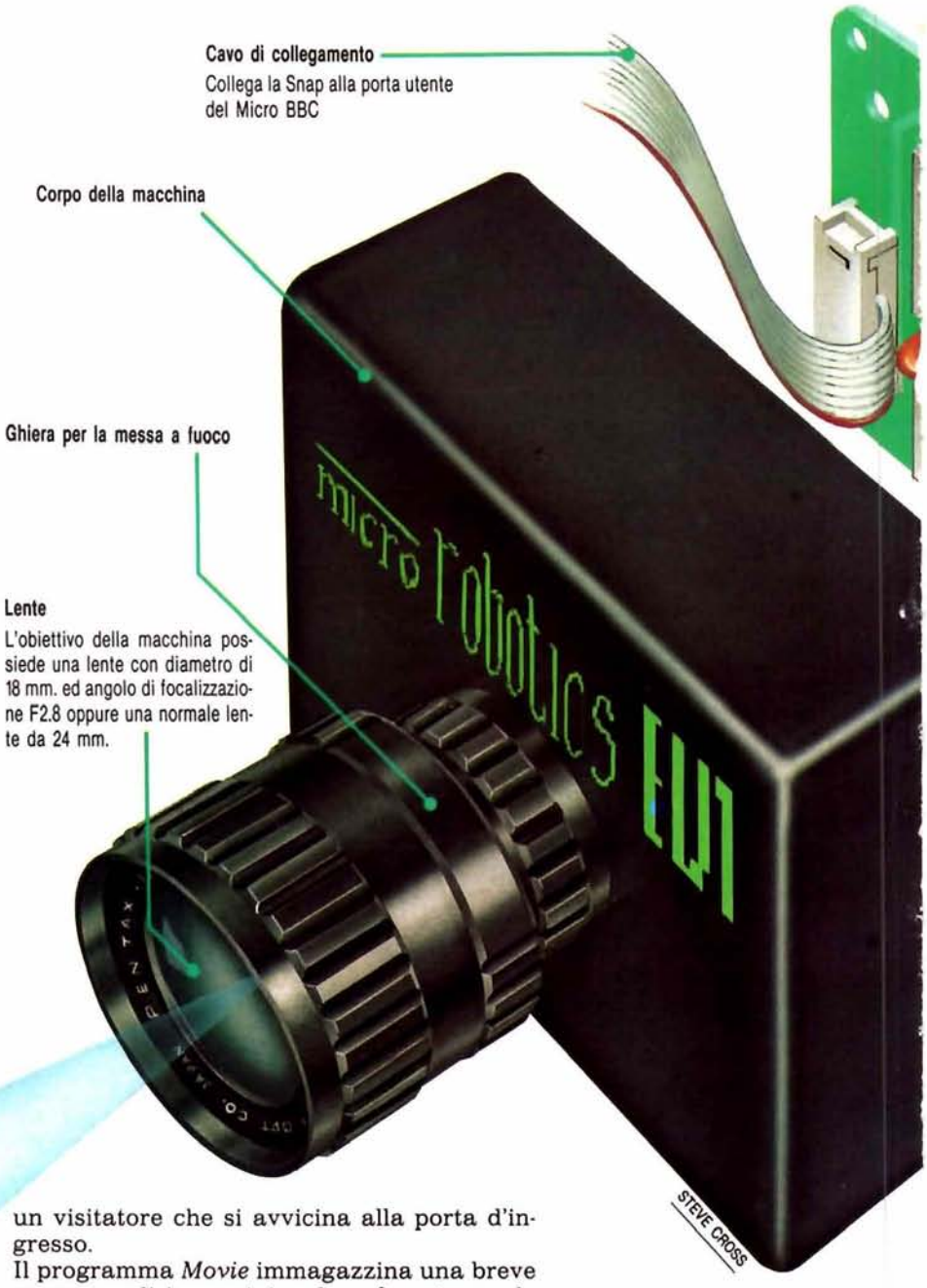
La custodia della Snap è in materiale plastico e le sue dimensioni sono quelle di un pacchetto di sigarette. L'obiettivo si trova sul davanti, mentre alla base è inserito un dispositivo di fissaggio che permette di collocare la macchina su un treppiedi durante le esposi-



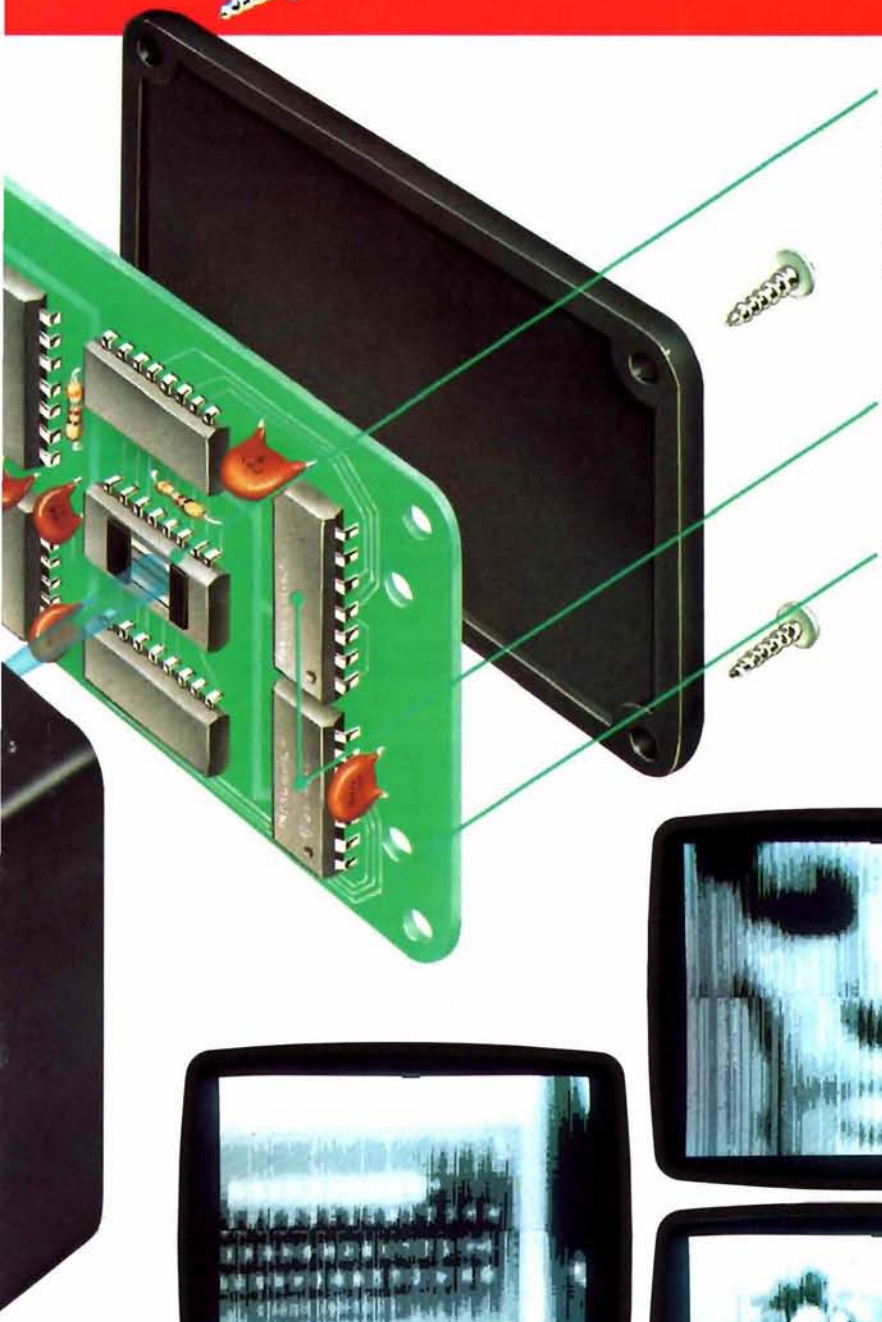
Immagini in movimento

Il software che accompagna la Snap, chiamato EV1, permette di inviare ininterrottamente allo schermo le immagini inquadrate dall'obiettivo. Queste possono essere "fissate" sullo schermo, inviate ad una stampante o salvate su dischetto. I tempi di esposizione vengono fissati automaticamente dal computer, ma l'utente può selezionarli manualmente agendo sui tasti "cursore su" e "cursore giù"

zioni di lunga durata. Alla presa laterale viene innestato un cavo a nastro lungo circa due metri, da collegare alla porta utente del BBC. Non ci sono altri collegamenti quindi la messa a punto del sistema è estremamente semplice.
L'obiettivo "a baionetta" è quello della Pentax 101, una piccola macchina fotografica reflex a lente singola. Simili obiettivi, o altri con focali diverse, possono essere acquistati nei comuni negozi fotografici e sostituiti a quello in dotazione all'apparecchio. Sulla Snap si può anche montare uno zoom, ma per questo (come per gli altri obiettivi Pentax) va tenuto conto della diversa distanza dal chip rispetto all'obiettivo standard. Quindi la distanza focale indicata sui vari obiettivi perde di significato quando vengono usati con la Snap. Poiché la messa a fuoco della macchina è spesso lunga e laboriosa, conviene modificare gli obiettivi ricalibrando le tacche di messa a fuoco.
La qualità dell'immagine video trasmessa dal sistema dipende in gran parte dal software. Il sistema è corredato di numerosi programmi ed il manuale d'istruzioni spiega come usare il software ed il funzionamento dell'intero sistema video.
La Snap può essere adoperata in due modi diversi: può creare immagini televisive a due toni di grigio, basate su una singola esposizione, o immagini multi-tono ottenibili inquadrando più volte un oggetto con diversi tempi d'esposizione. Il primo metodo, adottato dal primo programma dimostrativo del package, utilizza una piccola sezione di schermo, selezionato nel MODE 4, per visualizzare un'immagine costantemente aggiornata. I tempi d'esposizione vengono regolati usando due dei tasti per il cursore.
Questo programma possiede anche una routine di "dump" dello schermo, per riprodurre le immagini su stampanti Epson o Epson-compatibili.
Il secondo programma permette di ottenere riproduzioni più naturali, in varie sfumature di grigio. Sono necessarie 8 immagini separate, prese con diversi tempi d'esposizione, che vengono visualizzate simultaneamente su uno schermo selezionato nel MODE 0. Inoltre, il programma impiega una particolare tecnica di ombreggiatura per evidenziare le immagini a esposizione più breve. In questo modo si ottengono figure con 8 differenti livelli di brillantezza, dal nero al bianco attraverso varie sfumature di grigio. Il risultato è molto realistico ma, con una luminosità ambientale pari a quella di una stanza comune, la produzione di tali immagini può richiedere fino a dieci secondi.
Il programma *Secure* trasforma il sistema Snap in un allarme antifurto computerizzato. Il software prende nota delle diversità presenti in una serie d'immagini successive ed emette un segnale d'allarme solo quando queste superano un determinato valore di soglia. Pertanto, una Snap puntata sul giardino di casa trascurerà i movimenti degli alberi mossi dal vento, ma segnalerà la presenza di un visitatore che si avvicina alla porta d'ingresso.



Il programma *Movie* immagazzina una breve sequenza di immagini a due sfumature e le ritrasmette in rapida sequenza creando l'effetto di un'animazione. Un altro programma, chiamato *Animal*, è una versione video dell'omonimo gioco per computer (vedere a pag. 732). Il giocatore pone di fronte all'obiettivo un'immagine contenente numerosi oggetti. Il programma la analizza, memorizzando i contorni di tutti gli oggetti che riesce a distinguere, quindi invita il giocatore ad assegnare un nome a ciascuno di essi. A questo punto il giocatore dirige l'obiettivo su una seconda immagine ed il programma cerca di indovinare gli oggetti che erano già presenti nell'immagine precedente.
Comunque, il programma più stimolante è probabilmente *Arty* che permette di creare complicate figure sullo schermo, partendo da diverse immagini. Il programma usa l'intero



SNAP/EV1 VIDEO

DIMENSIONI

mm. 70 x 50 x 25

LENTI

Da 24 oppure 18 mm.

INTERFACCE

Connettore a 20 poli

DOCUMENTAZIONE

Manuale d'uso per la Snap

PRO

Consente di memorizzare su disco le immagini riprodotte sullo schermo

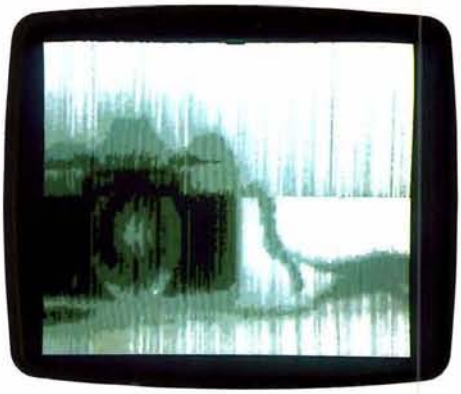
CONTRO

La risoluzione, non molto alta, talvolta provoca difficoltà nel controllo del sistema

Risoluzione verticale

Le immagini prodotte dalla Snap sono interamente formate da linee verticali, con minuscole interruzioni che evidenziano le variazioni di luminosità. La risoluzione dell'immagine è discreta ma non eccezionale





schermo, selezionato nel MODE 1, riportandovi le immagini "catturate" dalla macchina fotografica, che si possono ingrandire o ridurre agendo sui joystick. I colori in primo piano e quelli dello sfondo possono essere cambiati tramite i tasti di funzione del BBC. Sebbene sia necessario un paziente lavoro di composizione, per ottenere discreti risultati, si possono realizzare complesse immagini multicolori usando riproduzioni di oggetti reali.
Complessivamente il software disponibile è abbondante e ben strutturato, ma riesce a coprire solo in parte le possibili applicazioni della Snap. Ad esempio, il programma che crea immagini nelle varie sfumature del grigio può essere usato solo nel MODE 0; l'introduzione di una opzione per l'impiego del MODE 2 avrebbe invece consentito di usare colori diversi.

Il manuale fornisce molte informazioni sulle routine in codice macchina impiegate dal software, ma per poterle utilizzare è indispensabile conoscere il linguaggio macchina; chi conosce solo il BASIC deve accontentarsi del software fornito dalla casa. Inoltre, l'impiego di moduli in codice macchina di tipo più generico avrebbe consentito all'utente di combinarli insieme per ottenere applicazioni personalizzate. Tuttavia il costo di questi perfezionamenti avrebbe senza dubbio inciso sulla principale attrattiva del sistema Snap: il suo prezzo accessibile.

Contrasti di luce

Di solito le immagini vengono visualizzate in bianco e nero. Tuttavia, il programma "Grey" permette di ottenere immagini ombreggiate a schermo intero con 8 diversi livelli intermedi di brillantezza. Questa capacità di evidenziare i contrasti anche di lieve entità accresce il realismo dell'immagine

Un piano d'azione

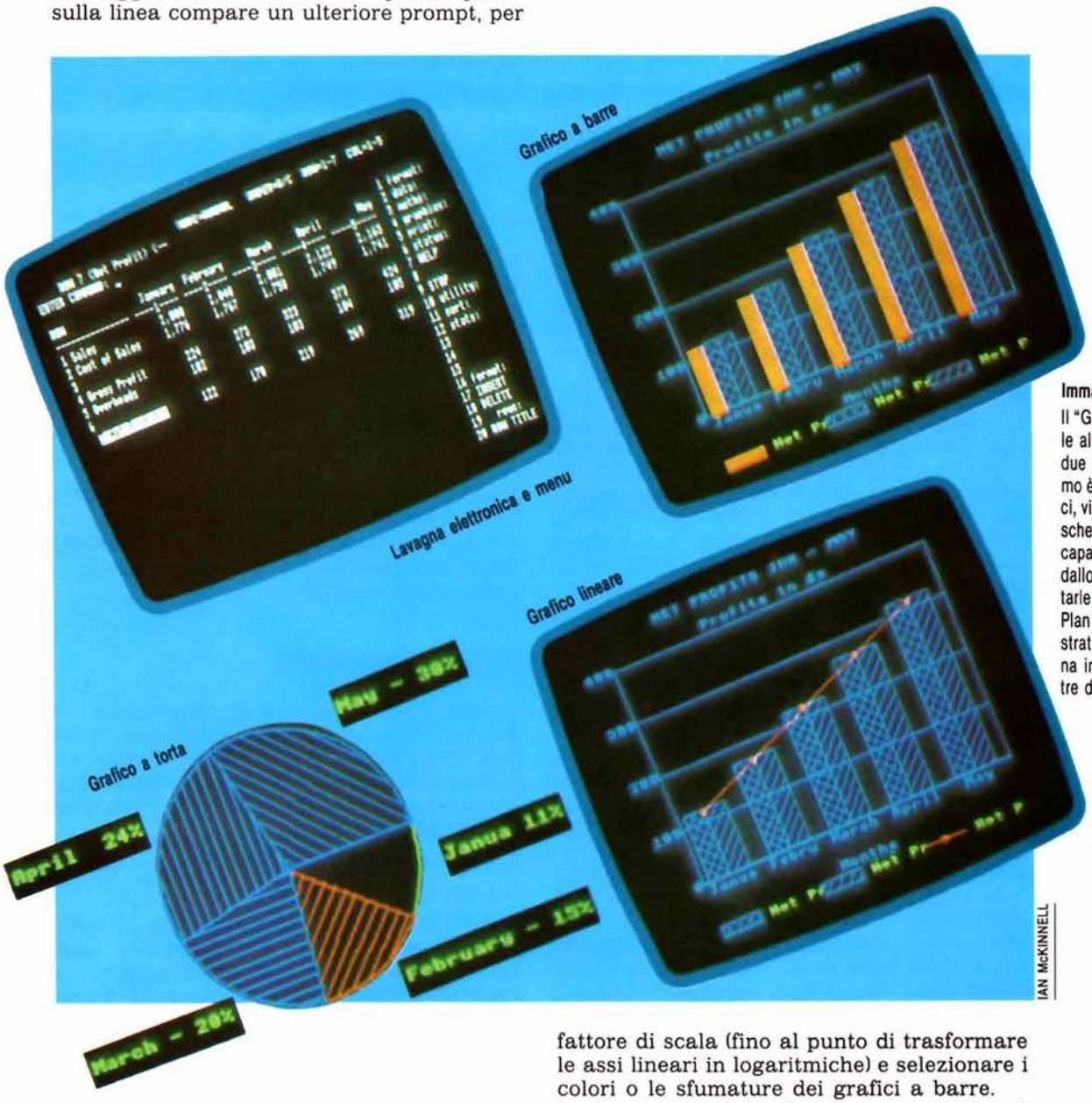
Lo spreadsheet con capacità grafiche "Graph Plan", per il BBC Modello B con scheda Z80

Diversamente dagli altri package finora esaminati, il *Graph Plan* è un programma su dischetto. La Acorn lo distribuisce gratuitamente, abbinato ad altri programmi, a chi acquista il secondo processore Z80 per il BBC Modello B. Come l'altro materiale software in omaggio, il Graph Plan si dimostra un programma utile ed affidabile.
Pur non fornendo le prestazioni dell'*Abacus* della Psion, il pacchetto di grafica/spreadsheet che correda il QL Sinclair (vedere a pag. 1204), il Graph Plan offre una maggior velocità di accesso ed immagazzinamento delle informazioni, tipica dei software su dischetto (i Microdrive del QL danno buoni risultati con i programmi, ma sono notevolmente più lenti delle unità a dischetti). Dato che il Graph Plan viene offerto in omaggio, sarebbe destinato forse a riscuotere un discreto successo, ma l'alto prezzo del processore Z80 e dell'unità a dischetti Acorn probabilmente favoriranno l'Abacus del QL Sinclair.
Esaminiamo adesso le caratteristiche grafiche di questo pacchetto per la realizzazione di modelli finanziari. Come lo stesso nome suggerisce, il Graph Plan possiede estese funzioni grafiche ed incorpora un gran numero di formule matematiche e procedure commerciali. Possiede inoltre un originale sistema di comunicazione, basato su codici numerici: infatti l'utente comunica con il programma tramite 144 comandi numerici. Una volta caricato in memoria, il programma visualizza una lavagna elettronica standard, suddivisa in righe e colonne, che occupa gran parte dello schermo e, sulla destra, i 20 comandi di base con il loro codice di chiamata.
Quando sulla linea di stato (la terza in alto a partire dalla lavagna) compare il messaggio ENTER COMMAND, l'utente seleziona un comando inserendo l'appropriato numero di codice.
Per quanto la selezione dei comandi sia abbastanza semplice (oltre al menu sullo schermo l'utente può consultare una lista completa dei comandi sul manuale d'uso del Graph Plan), questo sistema di chiamata presenta tuttavia alcuni svantaggi. Mentre sui migliori pacchetti per la realizzazione di modelli finanziari, ad esempio il *Lotus 1-2-3* (vedere a pag. 1124), i comandi vengono selezionati battendo la loro lettera iniziale (o indicandoli su un menu che presenta anche una breve descrizione del loro funzionamento), il menu del Graph Plan visualizza solo il nome ed il numero del comando. L'utente deve quindi tenersi a mente le funzioni che essi svolgono. Fortunatamente il Graph Plan possiede diversi menu, per cui sono visualizzati tutti i 144 comandi disponibili. Ad esempio, selezionando il numero di comando 2 (data), il menu iniziale viene sostituito da quello contenente i comandi da 29 a 48 (comandi per l'immissione e la gestione dei dati). Infine, il programma possiede la funzione HELP (numero di comando 7) che fornisce una sintetica descrizione del funzionamento dei comandi.
Oltre a questo sistema di comandi numerici, il pacchetto presenta altre caratteristiche particolari. Mentre la maggior parte delle lavagne elettroniche si basa sul concetto di "cella", intesa come intersezione di una riga e di una colonna, il Graph Plan tratta le righe e le colonne come entità indipendenti. Il suo "data pointer", ossia la seconda linea di stato, non solo indica la posizione del cursore sulla lavagna elettronica, ma comunica anche se il sistema si trova nel "modo" riga o in quello colonna.
Questa distinzione non avrebbe alcun senso negli spreadsheet basati su celle ad indirizzamento unico, ma è fondamentale nel Graph Plan perché permette di tracciare grafici relativi alle righe o alle colonne e non ad entrambe. Per ottenere grafici dell'uno o dell'altro tipo, l'utente deve selezionare il modo desiderato, posizionando il cursore sull'etichetta della linea o della colonna di cui vuole il grafico.

Creazione di un modello

Descriviamo adesso con un semplice esempio la tecnica usata dal Graph Plan per realizzare i grafici. Il nostro modello è composto da cinque colonne, etichettate con i nomi dei mesi da Gennaio a Maggio, e da cinque righe indicate come: Fatturato, Costi, Ricavo Lordo, Spese Generali ed Utile Netto. In questo modello i grafici riferiti alle linee avranno finalità diverse rispetto a quelli delle colonne. Ad esempio, si può produrre un grafico a barre della riga Fatturato per le vendite effettuate da Gennaio a Maggio.
Il grafico riporta lungo l'asse x i nomi delle colonne (da Gennaio a Maggio), mentre le barre rappresentano i valori registrati nella riga Fatturato. Oppure si può ottenere un grafico totalmente diverso, passando al modo

colonna. In questo caso il programma stampa le etichette delle righe (Fatturato, Costi, ecc.) lungo l'asse x e le barre rappresentano i dati di una colonna (ad esempio Gennaio).
Quindi, usando uno stesso gruppo di dati, il Graph Plan può produrre in pochi secondi più grafici di diverso genere e visualizzarli in sequenza, senza ulteriori interventi da parte dell'utente. Quest'ultima funzione viene selezionata inserendo il comando grafico 62 SELECT quando sulla terza linea di stato compare il messaggio ENTER COMMAND. A questo punto sulla linea compare un ulteriore prompt, per l'immissione del numero della riga (se ci si trova nel modo riga) di cui si vuol ottenere un grafico. Eseguita questa operazione, il programma visualizza il seguente messaggio: 'Choose (Bar = 1, Line = 2, Pie = 3)', per la scelta del tipo di grafico, che può essere a barre (1), lineare (2) o a torta (3).
Infine, il comando 61 DISPLAY visualizza il grafico sullo schermo.
Se si desiderano grafici strutturati in modo più flessibile e originale, si può usare il comando 63 OPTIONS che visualizza un menu con sei scelte: Display Chart, Define Chart Options, Define Axes Options, Define Pie Options, Print Chart e Plot Chart. Il manuale del Graph Plan presenta un'esauriente descrizione delle diverse opzioni offerte dal comando. Possiede inoltre un'appendice, molto utile a chi usa i Graph Plan per la prima volta, intitolata *Guida ai menu grafici secondari*. Questa descrive sinteticamente le procedure per intestare i grafici, variarne il fattore di scala (fino al punto di trasformare le assi lineari in logaritmiche) e selezionare i colori o le sfumature dei grafici a barre.
Le funzioni grafiche della lavagna elettronica, come del resto quelle matematiche e statistiche, sono di ottimo livello e rendono questo pacchetto perfettamente adeguato a molte applicazioni scientifiche e di progettazione finanziaria. Inoltre il Graph Plan possiede un eccellente sistema di presentazione dei dati, utile per redigere relazioni, ed è in grado di rispondere alle esigenze non solo di tecnici e scienziati, ma anche di utenti generici.



IAN McKINNELL

Immagini spreadsheet
Il "Graph Plan" si differenzia dalle altre lavagne elettroniche per due aspetti fondamentali. Il primo è il menu di comandi numerici, visualizzato sulla destra dello schermo; il secondo è la sua capacità di estrarre informazioni dallo spreadsheet e rappresentarle in forma grafica. Il "Graph Plan" può visualizzare i dati registrati su una riga o su una colonna in modo indipendente e con tre diversi tipi di grafici

Un po' di poesia

Uso combinato di "liste" e ricursione: il LOGO diventa un poeta

Il termine informatico "lista" rappresenta un insieme di elementi disposti secondo un dato ordine. In LOGO, le liste vengono indicate tra parentesi quadre (ad esempio: [ROMA MILANO NAPOLI]) e compaiono con molta frequenza, poiché questo linguaggio si basa essenzialmente su di esse. Abbiamo già visto che la breve procedura per disegnare un quadrato REPEAT 4 [FD 50 RT 90] presenta come secondo elemento una lista di istruzioni, racchiuse tra parentesi quadre. Ugualmente, la linea MAKE "INP REQUEST assegna alla variabile INP una lista contenente un dato immesso da tastiera.
Le liste possono essere assegnate anche a variabili globali: ad esempio, MAKE "LUOGHI [ROMA MILANO NAPOLI]. L'istruzione PRINT :LUOGHI visualizza la lista, omettendo le parentesi quadre, ossia: ROMA MILANO NAPOLI.
Gli elementi di una lista LOGO possono essere numeri, parole oppure altre liste. Ciò significa che una lista può contenere una seconda lista che a sua volta ne contiene un'altra ancora e via dicendo. Si tratta di un tipico esempio di definizione "ricursiva". [[CARNE PESCE] UOVA VERDURA] è una lista valida, il cui primo elemento è a sua volta una lista: [CARNE PESCE].
Spesso, per elaborare le liste, occorrono procedure ricursive, proprio perché le stesse liste hanno una struttura ricursiva.
La maggior parte delle procedure LOGO finora viste elaboravano un numero o una parola alla volta. Quando invece si devono trattare più elementi contemporaneamente, è necessario raggrupparli all'interno di una singola unità. A questo scopo il LOGO adopera le liste, strumenti estremamente versatili che consentono di organizzare le informazioni in strutture molto complesse.
Le due istruzioni base per l'elaborazione delle liste sono FIRST e BUTFIRST. La procedura FIRST [ROMA MILANO NAPOLI] visualizza la parola ROMA, che è il primo elemento della lista. Al contrario, BUTFIRST [ROMA MILANO NAPOLI] elenca MILANO NAPOLI, ossia l'intera lista, ma omettendo il primo elemento. La seguente procedura visualizza uno sotto l'altro tutti gli elementi della lista:

```
TO ELENCO :LISTA
   PRINT FIRST :LISTA
   ELENCO BUTFIRST :LIST
END
```

Quindi ELENCO [ROMA MILANO NAPOLI] produrrà:

```
ROMA
MILANO
NAPOLI
```

Il primo comando visualizza il primo elemento della lista, quindi trasferisce il controllo a ELENCO, che mostra il resto della lista. Al termine dell'esecuzione (ossia quando la lista è stata visualizzata completamente), sullo schermo compare un messaggio d'errore. La seguente procedura conclude l'elaborazione in un modo più elegante:

```
TO ELENCO :LISTA
   IF EMPTY? :LIST THEN STOP
   PRINT FIRST :LIST
   ELENCO BUTFIRST :LIST
END
```

EMPTY? verifica semplicemente se la lista in questione è "vuota" (ossia []). Alcune versioni MIT del LOGO sono prive della primitiva EMPTY?, ma si può ottenere lo stesso risultato con una procedura del tipo:

```
TO LISTAVUOTA? :LIST
   IF :LIST = [] THEN OUTPUT "VERO
   OUTPUT "FALSO
END
```

Simili a FIRST e BUTFIRST sono LAST e BUTLAST. LAST [ROMA MILANO NAPOLI] stampa NAPOLI (ultimo elemento della lista) mentre BUTLAST [ROMA MILANO NAPOLI] stampa MILANO NAPOLI (tutta la lista escluso l'ultimo elemento).

Dallo psicanalista

Come prima esperienza di elaborazione di una lista, proviamo a visualizzare, con un criterio casuale, alcune delle parole più ricorrenti in una immaginaria seduta psicoanalitica. Innanzi tutto, le parole scelte vanno assegnate alla variabile PAROLE:

```
MAKE "PAROLE [MADRE PADRE SESSO UCCISIONE GELOSIA
FUOCO MARE MORTE SOGNO]
```

L'obiettivo è ottenere una sequenza costante di parole estraendole a caso dalla lista. Queste vengono scelte in base ad un numero casuale n, compreso tra 1 ed il numero totale degli elementi della lista (in questo caso 9). I valori via via selezionati servono per estrarre l'elemento ennesimo della lista, che andrà a comporre la sequenza.

```
TO ENNESIMO :NUM :LISTA
   IF: N = 1 THEN OUTPUT FIRST :LIST
   OUTPUT ENNESIMO :NUM—1 BUTFIRST :LISTA
END
```

Vediamo adesso con alcuni esempi come funziona questa procedura. Se scriviamo ENNESIMO 1 :PAROLE, la condizione contenuta nella prima linea risulta vera quindi la procedura esegue l'istruzione PRINT :PAROLE. Nel nostro caso viene stampata la parola MADRE.

Se invece usiamo ENNESIMO 2 :PAROLE, la condizione risulta falsa quindi viene eseguita l'istruzione ENNESIMO 1 BUTFIRST :PAROLE che ignora il primo elemento della lista e stampa la parola seguente PADRE (ossia il primo degli elementi successivi della lista).
La seguente procedura visualizza una parola estratta a caso da una piccola lista di vocaboli:

```
TO SCELTACASUALE :LISTA
   OUTPUT ENNESIMO ((RANDOM 9) + 1) :LISTA
END
```

Per usare la procedura si digiti SCELTACASUALE :PAROLE.
In questo caso sappiamo che la lista è composta solo da 9 elementi. Se vogliamo conoscere il numero degli elementi che compongono una particolare lista, possiamo utilizzare una procedura di questo tipo:

```
TO LUNGHEZZA :LISTA
   IF EMPTY? :LISTA THEN OUTPUT 0
   OUTPUT 1 + LUNGHEZZA BUTFIRST :LISTA
END
```

Per capirne il funzionamento, digitiamo LUNGHEZZA [SCIENZA FANTASIA]. Poiché la lista contiene più di una parola, la prima condizione non è vera e la procedura invia in uscita 1 + LUNGHEZZA [FANTASIA] e LUNGHEZZA [FANTASIA] dà come output 1 + LUNGHEZZA []. Al contrario, con LUNGHEZZA [] la condizione alla linea 1 risulta vera, perciò la procedura dà in risposta 0. Adesso LUNGHEZZA [FANTASIA] produce in uscita 0 + 1 = 1 mentre il risultato di LUNGHEZZA [SCIENZA FANTASIA] è 1 + 1 = 2. Quindi la procedura più generale per estrarre in modo casuale una serie di parole da una lista di qualsiasi lunghezza è:

```
TO SCELTACASUALE :LISTA
   OUTPUT ENNESIMO ((RANDOM LUNGHEZZA :LISTA) + 1)
   :LISTA
END
```

Molte versioni LOGO possiedono i comandi (o "primitive") ITEM e COUNT, che svolgono lo stesso lavoro delle corrispondenti procedure ENNESIMO e LUNGHEZZA. Usiamoli quindi per riscrivere la procedura precedente:

```
TO SCELTACASUALE :LISTA
   OUTPUT ITEM ((RANDOM COUNT :LISTA) + 1) :LISTA
END
```

La seguente linea visualizza dieci sequenze di parole che susciterebbero l'immediato interesse di uno psicanalista durante una seduta:

```
REPEAT 10 [PRINT SCELTACASUALE :PAROLE]
```

L'impostazione di questi programmi per l'elaborazione delle liste è comune a molte delle procedure ricursive usate per la grafica della tartaruga:

- Se il lavoro da eseguire è molto semplice, lo si può programmare in un'unica procedura.
- In caso contrario, è consigliabile suddividerlo in più procedure (spesso simili alla prima).

Questi principi di programmazione producono ottimi risultati e vengono comunemente impiegati nei programmi che elaborano liste. Confrontando il seguente programma che disegna un poligono:

```
TO POLIGONO :N
   IF: N = 0 THEN STOP
   FD 30 RT (360/ :N)
   POLIGONO :N—1
END
```

con la procedura ELENCO, scritta in precedenza, possiamo notare che la struttura dei due programmi è identica.

ABBREVIAZIONI

BUTFIRST	BF
BUTLAST	BL
SENTENCE	SE

Poesia ... casuale

Dedichiamoci adesso ad una procedura che, invece di visualizzare singole parole, crea brevi brani "poetici".

```
TO POESIA1 :LUNGHEZZA
   IF :LUNGHEZZA = 0 THEN PRINT "STOP
   (PRINT1 " " SCELTACASUALE :PAROLE)
```



La canzone della tartaruga finta

Ecco un estratto della "Canzone della tartaruga finta", da "Alice nel paese delle meraviglie" di Lewis Carrol. La metrica, difficile da riprodurre in un componimento poetico computerizzato, assomiglia a quella di un'antica ballata popolare inglese ed è stata ripresa anche da Mary Howitt nel poema classico "Il ragno e la mosca"

«Vuoi andare un po' più svelta?» disse un merluzzo a una lumaca,
«C'è qua dietro un marsuino che mi pesta la coda.
Guarda con quanto entusiasmo avanzano aragoste e tartarughe!
Aspettano sulla battigia — vuoi venire a ballare anche tu?

Vuoi venire sì o no, sì o no,
vuoi venire a ballare?
Vuoi venire sì o no, no o sì,
vuoi venire a ballare?

Non puoi veramente avere idea di come sarà carino
Quando ci prenderanno e butteranno con le aragoste in mezzo al mar!»
Ma la lumaca rispose: «Troppo lontano, troppo lontano!» e lo guardò storto —
Disse che ringraziava tanto il merluzzo, ma che non sarebbe venuta a ballare.

Che non voleva, non poteva, non voleva, non poteva,
non voleva venire a ballare.
Non voleva, non poteva, non voleva, non poteva,
non poteva venire a ballare.

```
    POESIA1 :LUNGHEZZA-1
END
```

L'istruzione PRINT1" ' inserisce uno spazio tra le parole. Se inseriamo POESIA1 6, la procedura visualizza una frase di sei parole. Invece di scrivere una lista di parole completamente nuova, conviene ampliare la precedente lista PAROLE con l'istruzione SENTENCE, che riceve due dati in ingresso, producendo una sola lista. Per esempio, SENTENCE "OLIO [SALE PEPE] produce la lista [OLIO SALE PEPE].

```
TO NUOVEPAROLE1 :LISTA
   MAKE "PAROLE SENTENCE :LISTA :PAROLE
END
```

Possiamo quindi ampliare la lista PAROLE con NUOVEPAROLE [ANSIETÀ REPRESSIONE [PAURA DI VOLARE]]. Se in precedenza non abbiamo assegnato un valore alla variabile PAROLE, può verificarsi un piccolo problema che può essere prevenuto usando la primitiva THING?, che verifica se ad una data variabile è stato assegnato o meno un valore. La procedura NUOVEPAROLE1 può essere migliorata con le seguenti modifiche:

```
TO NUOVEPAROLE1 :LISTA
   IF NOT THING? "PAROLE THEN MAKE "PAROLE[]
   MAKE "PAROLE SENTENCE :LISTA :PAROLE
END
```

Usando questa procedura con una diversa lista di parole, possiamo ottenere un brano "casuale" vagamente poetico:

```
APPARIZIONE FRAGOROSAMENTE PARLÒ STUPENDO
PARANOICO PIANETA TERRORIZZATO CON VERDE
APPARIZIONE OSCILLANTE PARANOICO ROBOT UOMO
VOLÒ PARLÒ OSCILLANTE FRAGOROSAMENTE
```

Ovviamente, questo brano generato dal computer non possiede alcuna struttura grammaticale. Tuttavia si possono ottenere poesie dal senso più compiuto se le parole vengono ordinate secondo strutture sintattiche del tipo: nome, verbo, nome ecc. Per ottenere questo risultato si devono usare più liste di parole, una per ogni elemento sintattico della frase. In seguito vengono esaminate alcune tecniche che consentono di migliorare le procedure di composizione poetica in LOGO.

Logo & Logo

Alcune versioni del MIT LOGO mancano delle primitive EMPTY?, ITEM e COUNT. Invece tutte le versioni LCSI usano:

EMPTYP al posto di EMPTY?
LISTP al posto di LIST?
TYPE al posto di PRINT1

La primitiva EQUALP verifica se i suoi due operandi sono uguali e viene usata in sostituzione del segno di uguaglianza (=) per confrontare liste e parole. (Solo alcune versioni LCSI prevedono l'impiego di questo segno per il confronto delle liste).

Si rammenti infine il diverso formato dell'istruzione IF:

```
IF EMPTYP :LISTA [OUTPUT 0]
```

Il LOGO Atari usa l'abbreviazione SE al posto di SENTENCE e manca della primitiva ITEM

Esercizi

1) Scrivere una procedura che visualizza una lista in ordine inverso (servirsi delle istruzioni LAST e BUTLAST). Quindi modificarla in modo da ottenere una lista che sia un'inversione di quella originale.
2) Scrivere una procedura che elimina un elemento della lista. Ad esempio, DELETE "PANE [ACQUA PANE] deve produrre [ACQUA] e DELETE "VINO [ACQUA PANE] deve produrre [ACQUA PANE]

Risposte agli esercizi

Ecco le risposte agli esercizi di pag. 1217

1. Calcolo della potenza di un numero:

```
TO POTENZA :A :N
   IF NOT ((INTEGER :N) = :N) THEN PRINT
   [SOLO INDICI INTERI] STOP
   IF: N = 0 THEN OUTPUT1
   OUTPUT :A*POTENZA :A :N *1
END
```

2. Conversione in esadecimale:

```
TO STAMPA.ESADEC :NUM
   IF: NUM < 10 THEN OUTPUT :NUM
   IF: NUM = 10 THEN OUTPUT "A
   IF: NUM = 11 THEN OUTPUT "B
   IF: NUM = 12 THEN OUTPUT "C
   IF: NUM = 13 THEN OUTPUT "D
   IF: NUM = 14 THEN OUTPUT "E
   IF: NUM = 15 THEN OUTPUT "F
END

TO  ESADEC :NUM
    IF: NUM = 0 THEN STOP
    ESADEC QUOTIENT :NUM 16
    PRINT1 STAMPA.ESADEC REMAINDER :NUM 16
END
```

3. Verifica se un numero è pari:

```
TO PARI? :NUM
   IF ((REMAINDER :NUM 2) = 0) THEN OUTPUT "VERO
   OUTPUT "FALSO
END
```

4. Calcola un'area con il metodo Monte Carlo:

```
TO MC
   DRAW PU MAKE "IN 0
   MC1 1000 10 100
   (PRINT [L'AREA È] (:IN))
END

TO MC1 :NUM :XNUM: YNUM
   IF :NUM = 0 THEN STOP
   PUNTO.CASUALE :XNUM :YNUM
   IF INTERNO? THEN MAKE "IN :IN + 1
   MC1 :NUM -1 :XNUM :YNUM
END

TO PUNTO.CASUALE :XNUM :YNUM
   SETXY RANDOM :XNUM RANDOM :YNUM
END

TO INTERNO?
   IF YCOR < XCOR*XCOR THEN OUTPUT "VERO
   OUTPUT "FALSO
END
```

DATABASE

Nona scheda delle istruzioni per lo Z80 della Zilog. Istruzioni di ingresso uscita

ISTRUZIONI DI INPUT

DESTINAZIONE DI INPUT			Indirizzamento della porta: IMMED. n	Indirizzamento della porta: REG. INDIR. (C)
INPUT 'IN'	INDIRIZZAMENTO CON REGISTRI	A	DB n	ED 78
		B		ED 40
		C		ED 48
		D		ED 50
		E		ED 58
		H		ED 60
		L		ED 68
'INI'-INPUT & Inc HL, Dec B	REGISTRI INDIRETTI	(HL)		ED A2
'INIR'-INP, Inc HL, Dec B, ripeti se B≠0				ED B2
'IND'-INPUT & Dec HL, Dec B				ED AA
'INDR'-INPUT, Dec HL Dec B, ripeti se B≠0				ED BA

COMANDI DI INPUT 4 BLOCCHI

ISTRUZIONI DI OUTPUT

			Sorgente: REGISTRO A	B	C	D	E	H	L	REG. IND. (HL)
'OUT'	IMMED.	n	D3 n							
	REG. IND.	(C)	ED 79	ED 41	ED 49	ED 51	ED 59	ED 61	ED 69	
'OUTI'-OUTPUT Inc HL, Dec b	REG. IND.	(C)								ED A3
'OTIR'-OUTPUT, Inc HL, Dec B, ripeti se B≠0	REG. IND.	(C)								ED B3
'OUTD'-OUTPUT Dec HL Dec B	REG. IND.	(C)								ED AB
'OTDR'-OUTPUT, Dec HL Dec B, ripeti se B≠0	REG. IND.	(C)								ED BB

COMANDI DI OUTPUT A BLOCCHI

INDIRIZZO DELLA PORTA DI DESTINAZIONE

Codice mnemonico	Operazione simbolica	Flag S	Z	H	P/V	N	C	Op-code 76 543 210	Hex	N. di byte	N. di Cicli M.	N. di Stati T.	Commenti
IN A, (n)	A ← (n)	•	•	X	X	•	•	11 011 011 ←n→	DB	2	3	11	n in A_0 ~ A_7 Acc. in A_8 ~ A_{15}
IN r, (C)	r ← (C) se r = 110 vengono interessati solo i flag	:	:	X	P	0	•	11 101 101 01 r 000	ED	2	3	12	C in A_0 ~ A_7 B in A_8 ~ A_{15}
INI	(HL) ← (C) B ← B −1 HL ← HL +1	X	① :	X	X	X	•	11 101 101 10 100 010	ED A2	2	4	16	C in A_0 ~ A_7 B in A_8 ~ A_{15}
INIR	(HL) ← (C) B ← B −1 HL ← HL +1 Ripeti finché B = 0	X	1	X	X	X	•	11 101 101 10 110 010	ED B2	2 2	5 (se B≠0) 4 (se B=0)	21 16	C in A_0 ~ A_7 B in A_8 ~ A_{15}
IND	(HL) ← (C) B ← B −1 HL ← HL −1	X	① :	X	X	X	•	11 101 101 10 101 010	ED AA	2	4	16	C in A_0 ~ A_7 B in A_8 ~ A_{15}
INDR	(HL) ← (C) B ← B−1 HL ← HL −1 Ripeti finché B = 0	X	1	X	X	X	•	11 101 101 10 111 010	ED BA	2 2	5 (se B≠0) 4 (se B=0)	21 16	C in A_0 ~ A_7 B in A_8 ~ A_{15}
OUT (n), A	(n) ← A	•	•	X	•	X	•	•	•	11 010 011 ←n→	D3	2	3	11	n in A_0 ~ A_7 Acc. in A_8 ~ A_{15}
OUT (C), r	(C) ← r	•	•	X	•	X	•	•	•	11 101 101 01 r 001	ED	2	3	12	C in A_0 ~ A_7 B in A_8 ~ A_{15}
OUTI	(C) ← (HL) B ← B −1 HL ← HL +1	X	① :	X	X	X	X	1	•	11 101 101 10 100 011	ED A3	2	4	16	C in A_0 ~ A_7 B in A_8 ~ A_{15}
OTIR	(C) ← (HL) B ← B −1 HL ← HL +1 Ripeti finché B = 0	X	1	X	X	X	X	1	•	11 101 101 10 110 011	ED B3	2 2	5 (se B≠0) 4 (se B=0)	21 16	C in A_0 ~ A_7 B in A_8 ~ A_{15}
OUTD	(C) ← (HL) B ← B −1 HL ← HL −1	X	① :	X	X	X	X	1	•	11 101 101 10 101 011	ED AB	2	4	16	C in A_0 ~ A_7 B in A_8 ~ A_{15}
OTDR	(C) ← (HL) B ← B −1 HL ← HL −1 Ripeti finché B = 0	X	1	X	X	X	X	1	•	11 101 101 10 111 011	ED	2 2	5 (se B≠0) 4 (se B=0)	21 16	C in A_0 ~ A_7 B in A_8 ~ A_{15}

Notazione per i flag:

- • nessun effetto
- 0 flag azzerato
- 1 flag posto a 1
- X effetto non definito
- : il flag rispecchia il risultato dell'operazione
- p pari
- ① Se il risultato di B−1 è zero, il flag di zero è posto a 1, altrimenti azzerato

per concessione della Zilog Inc.

Un utile debugger

La realizzazione inizia dal modulo di controllo

Descriviamo brevemente le fasi di analisi e progettazione indirizzate alla creazione di un programma di debugging. L'analisi non presenta particolari difficoltà: abbiamo già visto a pag. 1219 quali siano i compiti di un simile programma.
Le caratteristiche necessarie al suo funzionamento sono:

1. *Un programma da verificare:* quando il debugger viene caricato, il programma da verificare deve già trovarsi in memoria.
2. *Comandi:* bisogna decidere se i comandi verranno inseriti direttamente o faranno parte di un menu. Nel nostro programma i comandi, composti da un solo carattere, vengono selezionati dal menu riprodotto a margine.
3. *Indirizzi:* probabilmente verranno inseriti in notazione esadecimale, quindi il programma deve poter trasformare le stringhe di cifre esadecimali ASCII in numeri binari a 16 bit.

Il debugger deve produrre i seguenti output:

1. *"Eco" dei caratteri digitati sulla tastiera:* si rammenti che le battute dei tasti non producono automaticamente la comparsa dei caratteri sullo schermo: il computer deve essere programmato per eseguire tale operazione.
2. *Numeri ad 8 e 16 bit:* espressi come stringhe di cifre esadecimali.
3. *Stringhe:* usate per "etichettare" i precedenti.

La scomposizione del programma in moduli, e quindi in subroutine, può seguire direttrici diverse ma deve sempre esistere un modulo esterno, detto "shell" (conchiglia), che collega tutti gli altri. Nel nostro debugger, il modulo di controllo avrà la seguente struttura:

Il Modulo Principale

```
Dati:
    Indirizzo iniziale del programma (16 bit)
    Prompt per l'immissione di un comando (il carattere ASCII ">")
    Carattere di Comando: è un singolo carattere ASCII. (Indicare se il programma accetterà anche lettere minuscole)
    Indirizzo d'Interruzione: è l'indirizzo della routine che gestisce le interrupt SWI

Procedura:
    Preparazione interrupt
    Accetta Indirizzo iniziale
    RIPETI
      Visualizza il "prompt"
      RIPETI
        Accetta Comando
      FINCHÉ non viene inserito un comando valido
      Visualizza Comando (Eco del carattere)
      SE Comando = "B" ALLORA Inserimento-Breakpoint
      ALTRIMENTI SE Comando = "U" ALLORA Rimozione-Breakpoint
      ALTRIMENTI SE...
      FINCHÉ Comando = "Q"
Fine del Modulo Principale
```

Adesso abbiamo un'idea più chiara delle routine necessarie. I termini modulo e routine indicano strutture diverse. Il modulo che gestisce i punti d'interruzione (breakpoint) è composto da numerose routine collegate tra loro, che utilizzano gli stessi dati. La fase successiva della programmazione illustra come progettare tale modulo:

Modulo per la gestione dei Punti d'Interruzione

```
Dati:
    Tabella breakpoint: è una matrice composta da indirizzi a 16 bit, che contiene gli indirizzi dei punti d'interruzione
    Valori rimossi: è una matrice di valori ad 8 bit, corrispondente alla tabella precedente. In essa vengono depositati gli op-code originariamente situati nel punto d'interruzione e sostituiti con istruzioni SWI
    Numero breakpoint: è un valore ad 8 bit che indica il numero di breakpoint attivati
    Prossimo breakpoint: valore a 8 bit che indica il successivo punto d'interruzione che si incontrerà durante l'esecuzione del programma
    Op-code SWI: un op-code ad 8 bit per l'istruzione SWI
Procedura 1: Inserimento Breakpoint
    SE Numero breakpoint<MAX ALLORA Accetta indirizzo
       Incrementa di 1 Numero breakpoint
       Deposita l'indirizzo in Tabella breakpoint
       (in posizione corrispondente a Numero breakpoint)
    FINE SE
Fine della Procedura 1

Procedura 2: Inserimento breakpoint(N)
    (N indica quale punto d'interruzione introdurre tra quelli in tabella)
    Estrae indirizzo da Tabella breakpoint(N)
    Estrae l'Op-code a quell'indirizzo
    Lo deposita in Valori-rimossi(N)
    Memorizza Opcode-SWI all'indirizzo
Fine della Procedura 2
```

Adesso possiamo passare alla codifica della Procedura 2 che prevede la manipolazione di 4 valori. Il parametro N, che indica quale punto d'interruzione usare, è un numero ad 8 bit (compreso tra uno e Numero breakpoint - 1), usato come offset all'interno delle due tabelle. Si rammenti che una delle due tabelle contiene valori a 16 bit, l'altra a 8 bit.

Menu dei comandi a singolo carattere

B	Inserimento di un Punto d'Interruzione
U	Eliminazione di un Punto d'Interruzione
D	Elenca i Punti d'Interruzione attivi
S	Avvia l'esecuzione del programma
G	Riprende l'esecuzione dal punto in cui è stata interrotta
R	Visualizza il contenuto dei registri
M	Esame e modifica di una locazione di memoria
Q	Termina il programma di debugging

Diamo per scontato che in A venga caricato il parametro N. In X viene messo l'indirizzo ricavato dalla tabella dei punti d'interruzione e in B l'op-code da depositare in Valori-rimossi. Infine, B può essere usato per inserire l'istruzione SWI nell'indirizzo di destinazione. Il riquadro a destra contiene il listato finale del modulo per l'inserimento dei punti d'interruzione (Procedura 2). Descriviamo adesso il modulo per la gestione dei numerosi dati in ingresso e in uscita dal programma di debugging. Per il momento ipotizziamo l'esistenza di due subroutine: INCH, che carica nel registro A un comando a carattere singolo proveniente dalla tastiera e OUTCH, che visualizza il carattere contenuto in A nella corrente posizione del cursore sullo schermo. Le routine necessarie al funzionamento sono:

1) **Inserimento-comando:** immissione di un carattere (comando) da tastiera.
2) **Inserimento-indirizzo:** accetta un indirizzo esadecimale (da uno a quattro caratteri) battuto sulla tastiera.
3) **Inserimento-valore:** accetta un numero esadecimale (da uno a due caratteri), usato per modificare il contenuto di una locazione di memoria.
4) **Visualizzazione di un valore:** riproduce sullo schermo un numero esadecimale a due caratteri.
5) **Visualizzazione di un indirizzo:** visualizza un indirizzo esadecimale a quattro caratteri.

Questo modo di procedere evidenzia le diversità esistenti tra le due tecniche di programmazione top-down e bottom-up. La prima consente di sviluppare e programmare le varie operazioni in modo indipendente e produce una serie di routine separate, ma che possono contenere duplicazioni di medesime funzioni. La seconda, invece, porta alla scrittura di un ristretto numero di routine utilizzabili in situazioni diverse e permette di risparmiare memoria e di accorciare i tempi della programmazione. Queste routine sono:

GETCH: carica in A un singolo carattere e lo confronta con una lista di caratteri "validi" (lettere alfabetiche di comando o cifre esadecimali). Se il carattere è compreso nella lista lo invia allo schermo, altrimenti lo ignora.
GETHX2: si serve della routine precedente per ottenere due cifre esadecimali che poi trasforma in un numero a 8 bit.
GETHX4: trasforma quattro cifre esadecimali in un numero a 16 bit.
PUTHEX: usa due cifre esadecimali per visualizzare un numero a 8 bit. (Può essere chiamata due volte per stampare un numero a 16 bit).
PUTCR: genera un codice di ritorno carrello (e, se necessario, un avanzamento di linea).

Queste cinque routine devono essere elaborate in sequenza. Per prima cosa, esaminiamo la struttura della routine GETCH.

Routine di immissione caratteri

```
Dati:
    Carattere-comando: è un carattere ASCII proveniente dalla
        tastiera (viene caricato in A)
    Caratteri-validi: contiene l'indirizzo a 16 bit della tabella dei
        caratteri validi
    Num-caratteri-validi: è un numero a 8 bit
    Caratteri-esaminati: è un contatore a 8 bit
Procedura:
    RIPETI
        Accetta un Carattere-comando
        Assegna Num-caratteri-validi a Caratteri-esaminati
        Decrementa Caratteri-esaminati finché
        Caratteri-validi(Caratteri-esaminati)< >
        Carattere-comando AND Carattere-comando > = 0
    FINCHÉ Caratteri-esaminati > = 0
    Visualizza Carattere-comando
```

In questa routine, il registro A viene usato per immagazzinare Carattere-comando ed il registro X per memorizzare Caratteri-validi. La variabile Num-caratteri-validi potrebbe essere caricata in B ma, poiché è necessario immagazzinarla in modo permanente, è preferibile appoggiarla sullo stack. B può quindi essere usato per Caratteri-esaminati. Da notare che B fornisce nella tabella l'offset, che viene poi utilizzato per la decodifica del comando e la conversione esadecimale. Il riquadro presenta il listato finale della routine GETCH. Più avanti vengono esaminate le altre routine necessarie al modulo di I/O.

ROUTINE GETCH

```
GETCH   PSHS    B       Salva il contenuto di B
REPTOO  BSR     INC     Accetta un Carattere-comando
        LDB     1,S     Inizializza contatore Caratteri-esaminati
        DECB            Sottrae 1 all'offset massimo
WHILOO  BLT     ENDWOO  Finché Caratteri-esaminati >=0
        CMPA    B,X     E
        BEQ     ENDWOO  Carattere-comando <> Caratteri-validi
                        (Caratteri-esaminati)
        DECB            Decrementa Caratteri-esaminati
        BRA     WHILOO
ENDWOO  TSTB
UNTLOO  BLT     REPTOO  Termina se Caratteri-esaminati <0
        BSR     OUTCH   Visualizza Carattere-comando
        LEAS    1,S     Incrementa S per cancellare il valore
                        iniziale di B
        RTS
```

SET-UP-BREAKPOINT MODULE

```
                        Dichiarazione dei dati
BPTAB   RMB     32      Tabella breakpoint
REMTAB  RMB     16      Valori-rimossi
NUMBP   FCB     0       Numero breakpoint
NEXTBP  FCB     0       Prossimo breakpoint
SWIOP   FCB     $3F     Opcode-SWI
MAXBP   FCB     16      Numero massimo di breakpoint
                        Procedura 2: Inserimento breakpoint
BP02    PHSH    B,X     Salva il contenuto dei registri usati
        LSLA            Moltiplica l'offset per due
        LEAX    BPTAB,PCR  Indirizzo di base della tabella
        LDX     A,X     Estrae indirizzo da Tabella breakpoint (N)
        LDB     ,X      Estrae l'Op-code a quell'indirizzo
        LSRA            Riporta il registro A al valore iniziale
        STB     A,X     Deposita l'Op-code in Valori-rimossi (N)
        LDB     SWIOP,PCR  Prende l'Op-code SWI
        STB     ,X      Lo deposita all'indirizzo
        PULS    B,X,PC  Ripristina i valori originari nei registri
                        Fine della Procedura 2
```

Flyerfox

Una simulazione di volo, corredata da un combattimento aereo in piena regola

I giochi per computer sono stati molto perfezionati dai tempi dello *Space Invaders*, ma gran parte dei progressi sono dovuti soprattutto alla grafica. I programmatori si sono dedicati allo sviluppo di nuove tecniche di compressione dati, nel tentativo di inserire un numero sempre maggiore d'immagini in un'area RAM di dimensioni limitate. Nel frattempo, però, sono state trascurate quasi del tutto le notevoli capacità audio di molti home computer.

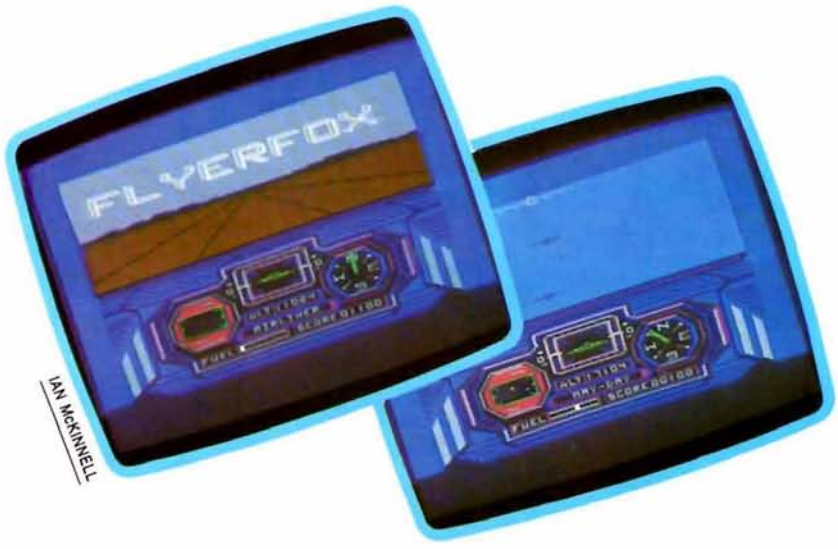
Ultimamente la compagnia americana Tymac ha lanciato una serie di giochi che generano parole sintetizzate pur non usando interfacce per la sintesi vocale. Il primo ad essere commercializzato su vasta scala è stato il *Flyerfox* per il Commodore 64. Questo programma è essenzialmente un simulatore di volo nel quale il giocatore pilota un aereo da caccia che scorta un Jumbo lungo una rotta particolarmente pericolosa. Mentre il nemico tenta di abbattere il Jumbo, l'obiettivo del giocatore è quello di attaccare e distruggere gli aerei avversari. La sintesi delle parole viene usata durante il gioco per lo scambio di messaggi tra il Jumbo ed il pilota/giocatore.

Il sintetizzatore vocale è una parte del software ed usa circa 11 Kbyte di memoria per immagazzinare i dati necessari alla costruzione delle diverse frasi. Il *Flyerfox* adotta un particolare sistema di codifica, grazie al quale le parole sono trasformate in segnali digitali che vengono poi immagazzinati in RAM. Quando occorre una particolare parola, il programma richiama i dati digitali e, attraverso il chip SID del Commodore, li ritrasforma in emissioni vocali.

La grafica del gioco è tutta in alta risoluzione. Le immagini visualizzate riproducono il cielo visto dalla cabina di pilotaggio del caccia e il quadro della strumentazione. Sono visibili vari strumenti di navigazione, tra i quali una bussola ed un radar che segnala i MiG nemici in avvicinamento e consente al giocatore di prepararsi al combattimento. Due luci lampeggianti, poste alle due estremità dell'"orizzonte artificiale", comunicano al giocatore se i MiG si trovano sopra o sotto la cabina di pilotaggio.

Le fasi del combattimento sono rapide e molto realistiche. Quando un MiG compare sullo schermo, il programma emette un segnale d'allarme e il giocatore inizia a manovrare il Flyerfox cercando di inquadrare l'aereo attaccante nel mirino dei propri cannoncini. Questa manovra è piuttosto complicata perché i MiG virano e "picchiano" a grande velocità. Quando il bersaglio è inquadrato nel mirino, il giocatore può lanciare i propri missili a puntamento termico. Tuttavia, la loro precisione è limitata e spesso i caccia nemici riescono ad evitarli. La grafica è discreta, ma piuttosto monotona. L'illusione del movimento è data dal movimento delle nuvole mentre il suolo è rappresentato da una cartina geografica che scorre nella direzione del movimento. La presenza dell'aereo di linea aggiunge ben poco al realismo del gioco. Il Jumbo è sempre rappresentato di poppa, non partecipa in alcun modo all'azione né può essere superato dal Flyerfox all'attacco dei caccia nemici. Infine, diversamente dai veri aerei passeggeri, quando viene attaccato non può compiere manovre diversive.

Il *Flyerfox* rappresenta certamente una novità nel campo dei giochi per computer. L'impiego della sintesi vocale all'interno di un programma era stata prospettata già da tempo, ma la Tymac si è spinta più avanti, producendo un gioco che genera intere frasi senza ricorrere a speciali interfacce o a dispositivi hardware. La è Tymac così considerata una software house all'avanguardia nella progettazione dei giochi per computer.



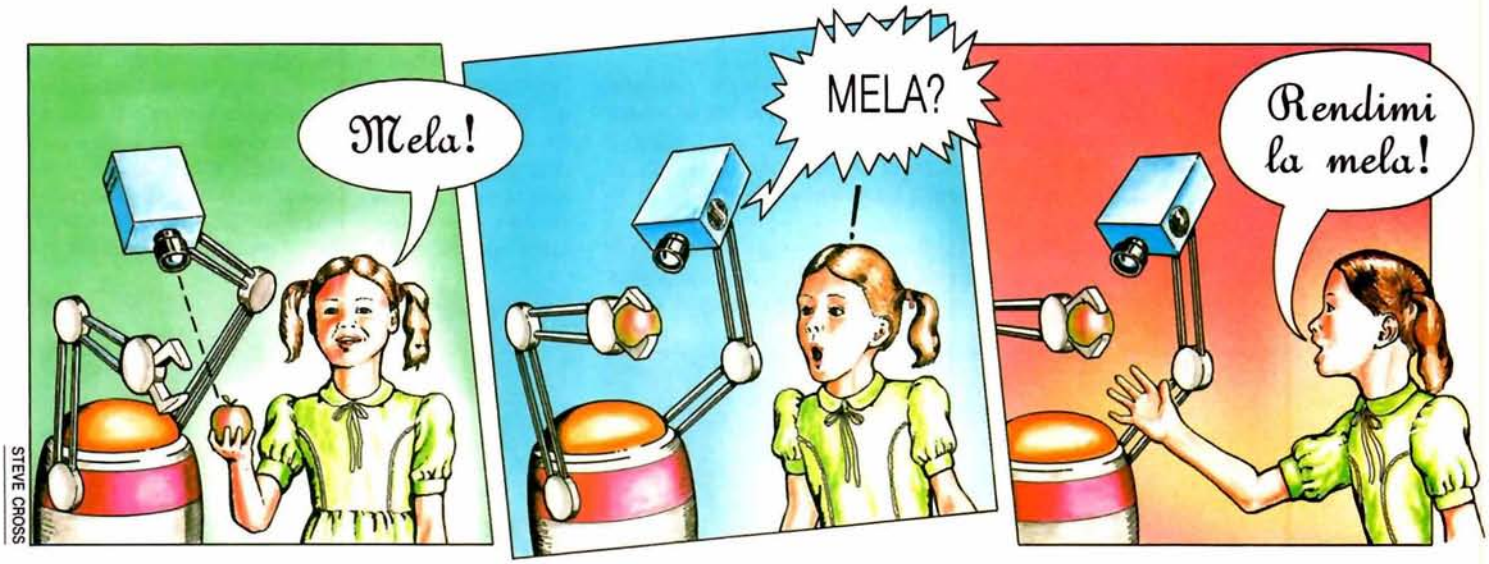
Caccia e distruzione

Il pilota del Flyerfox può ricavare molte informazioni sulla posizione dei caccia nemici tanto dal pannello di strumentazione, quanto dall'osservazione del cielo. I punti sullo schermo del radar rappresentano gli aerei nemici (non tutti però si lanciano all'attacco). L'altitudine relativa dei MiG è indicata dai due quadratini bianchi visibili ai due lati dell'orizzonte artificiale. La bussola aiuta il giocatore a ritrovare il Jumbo

Flyerfox: per il Commodore 64
Editori: Tymac Corporation
Autori: Gregory Carbonaro, Charles Teufert, Ronald Pintus, Arthur Aspromatis
Joystick: necessario
Formato: su disco o cassetta

Due chiacchiere col robot

Acquisire padronanza del linguaggio è un compito difficilissimo per un computer



STEVE CROSS

Il meccanismo che porta l'uomo all'apprendimento del linguaggio parlato non è ancora ben chiaro. Gli esperti si schierano su due fronti: c'è chi afferma che la capacità di parlare è innata e chi, invece, sostiene che tale capacità è di tipo acquisito, ossia deve essere appresa. Gli psicologi che sostengono la prima ipotesi fanno notare che l'uomo è l'unica creatura capace di comunicare col linguaggio. Quelli che aderiscono alla seconda teoria citano i numerosi esperimenti in cui si è riuscito ad insegnare agli animali a comunicare con l'uomo tramite linguaggi simbolici.

Se l'uomo impara a parlare semplicemente perché si trova a contatto con un linguaggio, allora è ragionevole concepire un metodo, applicabile anche ai robot, basato sui medesimi principi.

Per di più, tutto il processo si semplifica notevolmente se si parte dall'idea che un robot possa imparare a parlare semplicemente assistendo a dialoghi umani.

Sono stati fatti alcuni tentativi per accrescere la padronanza grammaticale dei computer, fornendo gli apparecchi di modelli fraseologici precostituiti; in altri casi si è tentato di insegnare ai computer nuove parole e morfemi (elementi di linguaggio) in qualsiasi lingua, semplicemente comunicandoli all'apparecchio. Tuttavia, non è stato ancora individuato un vero e proprio metodo per insegnare la lingua parlata ad un robot.

In pratica, quindi, occorre agire come se l'apprendimento del linguaggio fosse una capacità innata; per far ciò è necessario individuare le regole del linguaggio e depositarle in modo permanente nella memoria dei robot, simulando così una predisposizione innata. In generale, questa materia si suddivide in due branche: l'analisi sintattica e l'analisi semantica.

L'analisi sintattica, che concerne le regole grammaticali del linguaggio, decodifica la struttura superficiale di una frase oppure codifica un messaggio in una corretta forma grammaticale, pronta ad essere adoperata dal robot.

Il più diffuso metodo per raggiungere questa meta consiste nell'impiego di un "albero di scansione", che permette di scomporre (o ricomporre) le frasi in base agli elementi basilari che le compongono. Non è un processo semplice, ma si stanno già ottenendo apprezzabili risultati.

L'analisi semantica, è ancora più difficile ed implica l'estrapolazione del "senso" di una frase, se questa è rivolta al robot, oppure il modo di esprimere un concetto tramite i canali espressivi a disposizione del robot, se è quest'ultimo a dover parlare. Le complicazioni derivano dal fatto che nell'analisi semantica le singole frasi del linguaggio sono quasi sempre legate al contesto in cui appaiono. In altre parole, due interlocutori fissano, più o meno manifestamente, un particolare argomento di conversazione, dando per scontate alcune convenzioni. Un robot non ha sufficiente intuito né conoscenze per abbinare da solo il senso di una frase al contesto generale.

Vedere per credere

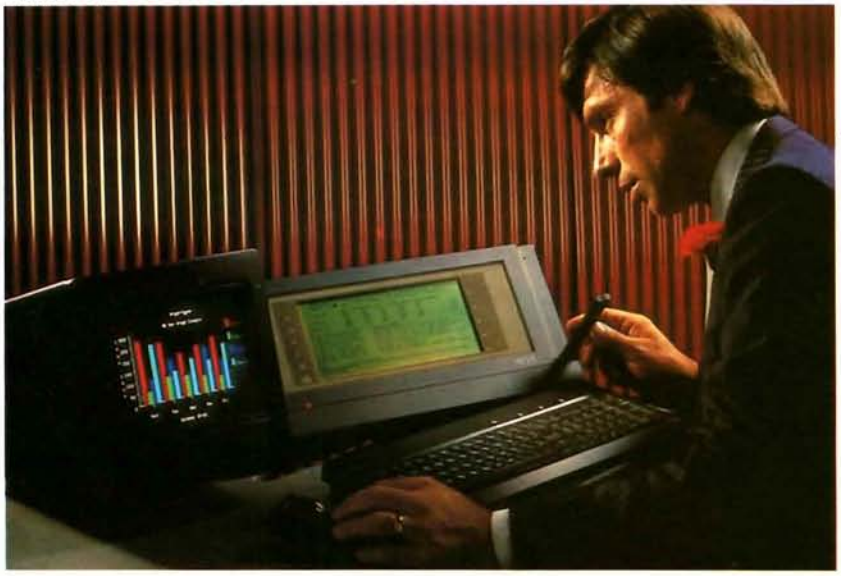
Quando un essere umano vede, ad esempio, una mela ed associa ad essa un nome, si crea il "concetto" di mela. Un robot è in grado di riconoscere visivamente un oggetto, solo se nella sua memoria esiste un modello rispondente a tale forma. Al modello visivo può essere associato anche un modello "fonico", ossia una sequenza di codici che, emessi tramite un sintetizzatore vocale, riproducono il nome dell'oggetto. Tuttavia, il robot non ha alcuna cognizione del fatto che una mela sia commestibile, né (fatto forse più importante ancora) che essa possa "appartenere" ad un essere umano

Un metodo basato sull'analisi semantica è stato adottato in esperimenti condotti da Terry Winograd, uno scienziato che ha elaborato un programma capace di mettere un robot in grado di capire quanto gli viene detto e di agire in conseguenza. Per i motivi appena esposti, però, il robot di Winograd riesce ad operare soltanto all'interno di un "mondo" molto limitato: una serie di blocchetti da costruzione. In tal caso, una buona analisi semantica riesce allo scopo grazie alla limitatezza del mondo in cui si muove e ragiona il robot: nel caos del mondo reale le sfumature di cui occorre tener conto sono troppe.

Per un robot è relativamente facile parlare, perché quanto deve comunicare, nella maggior parte dei casi, riguarda un ristretto numero di situazioni. Al contrario, la comprensione delle frasi ad esso rivolte è molto più complessa.

La tendenza più recente, per superare questo scoglio, consiste in una combinazione di processi alquanto empirici. Il robot esegue un'analisi semantica sul messaggio ricevuto, tentando di desumere una prima approssimazione del significato. Poi, alla luce delle proprie conoscenze sul mondo, il significato viene rivisto e corretto, fino a giungere ad una più precisa identificazione. Tutto ciò, comunque, è ancora allo stadio sperimentale e ben lontano da quanto siano in grado di fare i robot commerciali.

APRICOT F1



Ascoltare e parlare

Far parlare un computer o un robot è relativamente facile: esistono in commercio economici apparecchi (come quello della Currah, illustrato a destra) da collegare anche a piccoli home computer. Al contrario, il riconoscimento vocale comporta problemi molto più complessi, dovuti sia alle considerevoli variazioni di pronuncia tra una persona e un'altra, sia alla notevole quantità di memoria occorrente per l'elaborazione. I sistemi adoperati nel Big Ears (a destra) e nell'Apricot F1 (a sinistra) sono in grado di riconoscere un numero molto limitato di comandi, appena sufficienti per svolgere compiti molto elementari

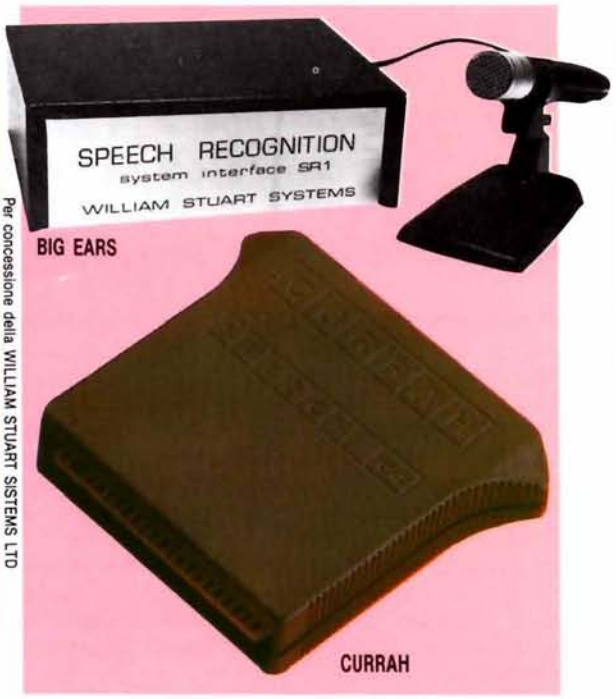
La sintesi vocale

Il più semplice metodo di sintesi vocale si fonda sull'impiego di nastri magnetici, preregistrati dall'uomo, che vengono poi riprodotti dal robot al momento opportuno. Può sembrare un sistema poco efficiente, ma è senza dubbio un punto di partenza abbastanza valido: vediamone i limiti e come fare a superarli.

Le prime limitazioni che vengono alla mente sono l'ingombro del registratore e il fatto che si tratta di un dispositivo meccanico, facilmente soggetto a guasti. Così bisogna anzitutto codificare i vari messaggi in forma binaria e conservarli in appositi chip di memoria all'interno del robot. Ciò si ottiene mediante l'impiego di convertitori analogico/digitali, che convertono le onde sonore in sequenze di numeri facilmente memorizzabili. Questo è lo stesso principio adottato nella registrazione di brani musicali sui dischi "compact".

L'altra faccia della medaglia è che i segnali così convertiti occupano una notevole quantità di memoria: nelle registrazioni su dischi "compact" i segnali vengono campionati circa 44.000 volte al secondo, con una risoluzione di circa 16 bit (vale a dire che l'ampiezza del segnale, in un qualsiasi momento, è memorizzata con un valore a 16 bit, che consente 216 = 65.536 diversi livelli di ampiezza). Con simili parametri, un solo secondo di registrazione occupa ben 88.000 byte. È ovvio che la memoria di un normale microcomputer non è sufficiente neanche per un breve messaggio, ma se ci si accontenta di una riproduzione un po' meno accurata (con una risoluzione di 8 bit e con un tasso di 3.000 campioni al secondo, ad esempio), allora può bastare anche qualche Kbyte di RAM.

Comunque, esistono anche altri sistemi per economizzare memoria: gli esperti di linguistica hanno suddiviso il linguaggio parlato in alcuni elementi di base, chiamati fonemi, e si sono accorti che ne bastano una quarantina per poter comporre con essi qualsiasi parola o frase. Ciò significa che se i fonemi sono memorizzati a parte e individuati tramite numeri di codice, la costruzione di frasi è notevolmente semplificata, con conseguente risparmio di memoria.



Per concessione della WILLIAM STUART SISTEMS LTD

Questo criterio è adottato in molti moduli di sintesi vocale reperibili in commercio: per adoperarli spesso, basta specificare una stringa di caratteri ed emetterla sull'appropriata porta del computer, né più né meno come si fa con le normali stringhe di caratteri da visualizzare sullo schermo. Va detto che, per il momento, i moduli in commercio sono più adatti alla pronuncia di parole inglesi, ma con un po' di pratica si riescono ad ottenere apprezzabilissimi risultati anche con parole italiane.
Il vantaggio di un simile sistema è che i messaggi non sono più preconfezionati, ma possono essere prodotti a piacimento, in base alle proprie esigenze: si tratta unicamente di costruire un vocabolario adatto.
Volendo si possono realizzare programmi che, in base a semplici regole grammaticali, riescono ad estrarre le parole dal vocabolario e comporle assieme in frasi approssimative ma comprensibili.
Tuttavia, per i motivi esposti in precedenza, conviene limitare la fraseologia ad un settore ben preciso.
Chi ha occasione di sentir parlare un robot dotato di un sintetizzatore vocale del genere descritto, si accorge che spesso la pronuncia delle parole non è perfetta: ciò dipende soprattutto dal fatto che nel linguaggio parlato umano il suono dei vari fonemi è in parte alterato da quelli che lo precedono e che lo seguono, mentre il robot è necessariamente costretto ad "appiattire" la pronuncia, essendo incapace di porre enfasi su particolari lettere o sillabe di una frase.
Spesso, è proprio tale incapacità a rendere ambiguo il significato di un messaggio, specialmente quando a precisarlo è l'intonazione della frase. È difficile, dunque, riconoscere una domanda da un'affermazione.

Riconoscimento del linguaggio

Quando vogliamo progettare un sistema in grado di riconoscere le frasi di un linguaggio parlato, ci troviamo di fronte a numerosi problemi; quello maggiore è che quanto noi desideriamo comunicare al robot può essere espresso in un'infinità di modi diversi.
Un sistema potrebbe essere quello di registrare su un nastro tutti i possibili comandi, in modo che il robot ne possa eseguire una scansione per trovare quello che corrisponde al comando impartito. Anche se sembra un metodo complesso, il principio alla sua base è impiegato in gran parte dei sistemi per il riconoscimento del linguaggio parlato. Infatti, nella memoria del robot vengono depositati dei modelli, sui quali viene compiuta una ricerca per trovare quello che più assomiglia al comando impartito. I modelli, in genere, si ottengono tramite sessioni di "apprendimento", ripetendo più volte il comando o la frase, finché il computer del sistema di controllo non è in grado di estrarre una specie di media approssimativa, riconoscibile indipendentemente da particolari inflessioni accidentali. Questo metodo funziona però se il numero di comandi è limitato e se questi vengono sempre impartiti più o meno allo stesso modo (ad esempio: «sinistra», «destra», «avanti», e via dicendo).
Tale sistema è stato definito «riconoscimento vocale di elementi discreti», in quanto il computer è in grado di 'riconoscere' singole parole ben separate tra loro da opportune pause di silenzio.
I problemi più grossi sorgono quando invece si desidera ottenere un sistema in grado di riconoscere il linguaggio parlato "continuo", ossia quello adoperato nelle normali conversazioni tra esseri umani.
In tal caso, il rapido susseguirsi delle sillabe (magari con inflessioni dialettali) può generare non pochi malintesi nell'interpretazione di una frase, stravolgendone completamente il significato.
Per gli esseri umani, i complessi meccanismi che regolano la comunicazione verbale riducono molto il problema, in primo luogo per il fatto che l'ascoltatore tenta inconsciamente d'indovinare quanto sta per comunicargli l'interlocutore, per poi confermare il proprio tentativo via via che riconosce le singole parole. È facile intuire che un robot non ha sufficienti cognizioni per poter tentare una simile strada.
In conclusione, si può dire che la sintesi vocale ha già compiuto sostanziosi progressi, benché richieda ancora perfezionamenti, mentre i processi di riconoscimento del linguaggio parlato, da parte dei robot, sono attualmente in fase di sviluppo ed implicano problemi a tutt'oggi irrisolti.
Anche in questo settore, comunque, le cognizioni aumentano di giorno in giorno e forse tra non molto si troveranno in commercio ubbidienti robot "tuttofare", simili a quelli preconizzati dai più fervidi scrittori di fantascienza, pronti ad esaudire ogni nostro desiderio.

Ascoltare e toccare

Il Voicemate è un braccio robotizzato controllato a voce, per impieghi di laboratorio e industriali, sviluppato dal dipartimento di scienze ingegneristiche del Politecnico di Newcastle (GB)



Per concessione della NEWCASTLE POLYTECHNIC

Griglia d'acciaio

Esaminiamo il "Multiplan" della Microsoft, un eccellente spreadsheet per il Commodore 64

Le caratteristiche del *Multiplan*, il primo spreadsheet prodotto dalla Microsoft, sono senza dubbio le più avanzate tecnologicamente. Una di queste è la possibilità di operare su più celle contemporaneamente, assegnando loro un unico nome o etichetta. Ciò facilita il riordino dei dati, la suddivisione dello schermo in finestre, la ricerca su dati conservati in tabelle e l'impiego di costruzioni condizionali del tipo IF...THEN. Ideato per personal computer di una fascia superiore (il PC della IBM e l'Apple), il Multiplan è stato poi adattato anche al Commodore 64.

Il modello scelto per illustrare le capacità del Multiplan è, stavolta, di carattere statistico-sportivo. I dati si riferiscono a partite di calcio americano, ma l'impostazione può adattarsi ad altre esigenze.

Una volta caricato il programma, viene visualizzata una tabella elettronica di 63 colonne per 255 righe. Sia le righe che le colonne sono numerate, cosicché la cella in alto a destra è indicata da R1C1 (per Riga 1, Colonna 1). Nella parte inferiore dello schermo compare il menu dei comandi e, inizialmente, il cursore evidenzia l'opzione Alpha. La selezione dei comandi può avvenire spostando il cursore, agendo sulla barra spaziatrice e poi premendo RETURN, oppure digitando la lettera iniziale del comando.

Per alcuni comandi viene visualizzato un menu secondario, che offre ulteriori opzioni di elaborazione o di formattazione dei dati. Poiché la pressione di una lettera provoca la selezione di un comando, per immettere testi occorre prima digitare una A (per Alpha). I numeri possono essere immessi liberamente, ma le formule devono essere precedute dal segno più (+) o uguale (=).

Le prime due righe del nostro modello contengono le intestazioni. Per comodità, le celle da R1C1 a R2C5 sono state formattate per un testo continuo, permettendo così alle scritte di superare i margini di ciascuna cella. Ciò si ottiene con:

F(ormattazione) C(elle) R1C1:R2C5

e quindi posizionando il cursore sulla parola Cont e premendo RETURN. I "due punti" (:) servono per indicare un arco di celle. In modo analogo, le altre celle sono state formattate per adeguarsi al contenuto dei dati.

Il foglio di lavoro è suddiviso in due parti: una contiene le informazioni relative ad una particolare squadra, per un arco di tempo di nove settimane, l'altra è una tavola delle partite vinte e di quelle perse durante una stessa "conference" (per una spiegazione dei termini, vedere le note sul Calcio Americano).

Una volta costruito il modello, gran parte delle informazioni settimanali vanno inserite manualmente: qualche formula si occupa di calcolare i totali progressivi per la stagione.

RECORD DELLA LEGA



RECORD DELLE SINGOLE SQUADRE



IAN McKINNELL

La prima porzione del foglio di lavoro contiene una tabella, in cui i totali vanno aggiornati settimanalmente, in base ai risultati delle partite. Conservando i dati sotto forma di tabella, si può sfruttare una caratteristica molto comoda del Multiplan: l'opzione di SORT, permette di riorganizzare l'ordine delle informazioni immesse inizialmente, in base ad uno qualsiasi dei campi che le compongono (in particolare in base al punteggio in classifica). Multiplan è anche in grado di riordinare un dato arco di righe all'interno di una colonna specificata. Ovviamente, le scritte vengono riordinate alfabeticamente.

Per illustrare l'uso dell'opzione SORT chiediamo a Multiplan di riordinare alfabeticamente i nomi delle squadre. All'opzione SORT si accede premendo S e Multiplan visualizza:

SORT by column: between rows: and:order:> <

Il riordino va fatto sulla Colonna 1, tra le righe 7 e 21 ed in ordine ascendente (>). Premendo RETURN si avvia l'operazione. Ovviamente, tutti i dati abbinati a ciascun nome di squadra seguono automaticamente nella

nuova posizione. In modo analogo si può riorganizzare la tabella in base al punteggio di ciascuna squadra, ottenendo così una classifica ben ordinata.

TABELLA RIORDINATA



Adesso, agendo sul tasto "cursore in basso", si provochi uno "scroll" (avanzamento verticale) del foglio di lavoro, per esaminare la seconda tabella, che contiene le prestazioni delle squadre. Qui vengono adoperate due formule, la prima delle quali è una semplice SUM (somma), per tenere aggiornato il totale progressivo settimanale. Si trovi la colonna etichettata con TOTALS (R24C12), nella seconda tabella. Lo scopo è quello di far sommare i dati relativi a ciascuna settimana. Siccome ciò va fatto per tutte le categorie (ossia l'area compresa tra R25C12 e R32C12), occorre impiegare un riferimento relativo alle celle. Con Multiplan, ciò è veramente facile: basta spostare il cursore sopra le celle interessate. La formula si immette digitando:

```
=SUM(
```

e poi premendo il tasto "cursore a sinistra", finché non lo si posiziona sulla cella R25C3. A questo punto si digita un "due punti" (per selezionare un arco di celle). Il cursore torna automaticamente alla cella in cui si sta immettendo la formula. Agendo ancora sull'apposito tasto, si sposta il cursore nella cella R25C11, terminando con la pressione di RETURN. Ora la formula dovrebbe apparire così:

```
=SUM(R[-9]C:R[-1]C)
```

e, simultaneamente, dovrebbero comparire i totali delle celle specificate. Adesso occorre duplicare la formula nell'arco di celle da R26C12 a R32C12. Mantenendo il cursore sulla cella della formula, si usi il comando Copy:

```
C(opy) d(own) 7 rows
```

La formula, in questo caso, viene replicata per 7 righe verso il basso. Lo stesso sistema si usa per calcolare i totali di YDS RUSH e YDS PASS. La formula SUM è collocata nella cella R27C3 (per le yarde conquistate all'attacco) e nella cella R31C3 (per le yarde conquistate dalla squadra avversaria). La formula va poi replicata nelle otto colonne a destra, per coprire l'intero periodo di nove settimane.

La seconda formula, molto comoda ma un po' più complicata, ricorre all'espressione condizionale IF. Nel nostro modello, lasceremo al Multiplan il compito di determinare la squadra vincente, facendo un confronto tra i punteggi conseguiti dalle squadre. Quel che occorre è un'espressione del tipo: Se Punti conseguiti > Punti consentiti allora visualizza WIN, altrimenti LOSS. Di nuovo, è necessario riferirsi alle celle in modo relativo, cosicché si adopera il cursore per immettere la formula. Lo si posizioni in R34C3, la cella relativa alle vittorie/perdite (WIN/LOSS) e si immetta la formula:

```
IF(R[-6]C>R[-2]C,"WIN","LOSS")
```

COSTRUZIONE CONDIZIONALE (IF...THEN)



Si noti che le scritte, all'interno della formula, vanno racchiuse tra doppi apici e che le condizioni vanno racchiuse tra parentesi. Adesso si replichi la formula col metodo descritto in precedenza. Il modello realizzato prevede dati relativi ad un periodo di nove settimane ma, a causa delle dimensioni dello schermo, non si possono vedere simultaneamente le etichette alla sinistra dei dati per la squadra e i totali all'estrema destra. Tuttavia, possiamo suddividere lo schermo in due finestre, che possono scorrere assieme o indipendentemente.

Nel nostro caso, scegliamo di operare la suddivisione alla colonna 3. Si premono in successione: W (per Window) e S (per Split) e V (per Vertical). Multiplan visualizza:

```
WINDOW SPLIT VERTICAL at column: linked YES/NO.
```

SCHERMO SUDDIVISO (SPLIT SCREEN)



Occorre digitare 3 per la colonna, poi portare il cursore sul NO e premere RETURN: se le finestre non sono "collegate" (linked), si possono muovere indipendentemente. Adesso le etichette si possono sempre vedere, qualunque sia la porzione di spreadsheet visualizzato. Per "chiudere" una finestra, si usa la sequenza: W (per Window) C (per Close) e numero della finestra. Il prossimo spreadsheet esaminato è il *Lotus 1-2-3*.



KEVIN JONES

Calcio americano

Per coloro che non conoscono questo sport, ecco qualche concisa spiegazione. Due squadre, di 11 giocatori, tentano di portare la tipica palla ovale oltre la linea del "touchdown" (meta) avversario. Le due mete distano 100 yarde l'una dall'altra. La palla può essere trasportata da un giocatore, lanciata in un passaggio in avanti o calciata direttamente attraverso i pali della porta. Un calcio messo a segno (chiamato "field goal") vale tre punti, mentre se la palla è trasportata o passata oltre la linea (azione "touchdown") i punti sono sei.

Ad ogni squadra sono consentiti quattro tentativi per far avanzare la palla di 10 yarde verso la meta. Se l'azione ha successo, la squadra può compiere altri quattro tentativi per avanzare ancora. Il trasporto della palla, da parte di un giocatore, è chiamato "rush" e il numero di yarde percorse in un "rush" è un'indicazione della distanza coperta da una squadra. Il numero di yarde percorso dalla palla durante i passaggi ("pass") indica quanto lontano è stata lanciata la palla.

La Lega Nazionale di Calcio (la principale lega americana di professionisti) prevede due "conference": la American Conference e la National Conference. Le squadre giocano una stagione di 16 incontri, che inizia a settembre e culmina nel "Super Bowl", in gennaio, al quale partecipano le squadre che hanno vinto le eliminatorie

Spirito d'avventura

Realizziamo un gioco d'avventura completo

I giochi del tipo "adventure" si sono diffusi negli anni '70, quando è ritornato in auge l'ormai noto *Dungeons and Dragons*. Il giocatore si identifica con uno o più personaggi, appartenenti all'immaginario mondo creato da un altro giocatore, ossia il Dungeonmaster. In genere, il mondo immaginario è composto da un'intricata serie di ambienti, nei quali sono collocati oggetti e tranelli che il giocatore deve saper sfruttare a proprio vantaggio. Scopo del gioco, di solito, è quello di uscire dal labirinto salvando la vita di un personaggio o recuperando un oggetto prezioso. Il gioco, inizialmente, è stato programmato su computer di grandi dimensioni (la complessità dei labirinti richiedeva una notevole quantità di memoria e una certa velocità di elaborazione). Il vantaggio della versione computerizzata è che a giocare può essere anche una sola persona, poiché il percorso (e le varie situazioni lungo di esso) vengono create dal computer. Da allora, però, il gioco è stato enormemente migliorato (un ottimo esempio è il *Dungeon* multi-utente, di cui si è parlato a pag. 864) e, soprattutto, ne sono state realizzate versioni adatte anche ai computer più piccoli. Alcune versioni sono esclusivamente testuali, mentre altre prevedono scenari composti, con una grafica a colori altamente realistica. Tuttavia, molti preferiscono le avventure senza grafica, poiché questa consuma molta memoria che può invece essere usata proficuamente per migliorare le caratteristiche dell'avventura. Si dice inoltre che, senza grafica, l'immaginazione e la fantasia del singolo giocatore sono

LA MAPPA DEL GIOCO

Il primo passo nella realizzazione di un gioco d'avventura consiste nel progettare una mappa degli ambienti visitabili dal giocatore. Ogni casella contiene una breve descrizione della scena e indicazioni per determinare quanti e quali oggetti sono presenti in tale locazione e se si tratta di un ambiente dalle caratteristiche "speciali". La numerazione delle varie caselle facilita la codifica dei dati



più libere di spaziare accrescendo così il divertimento. Ovviamente è una questione di gusti: alcune avventure grafiche sono dei veri e propri gioelli di programmazione, paragonando i risultati con la poca memoria a disposizione.
Qui vengono esaminate le tecniche di progetto occorrenti per realizzare i vari moduli che compongono un gioco d'avventura, chiamato *Digitaya*. I vari listati, assieme alle relative spiegazioni, sono suddivisi in più parti, che compongono poi il programma completo. Nel nostro gioco, il protagonista impersona un agente "elettronico" che deve immergersi nei meandri di un microcomputer alla ricerca della misteriosa Digitaya. Il percorso è disseminato di numerosi pericoli e difficoltà ed occorre mettere alla prova le proprie conoscenze informatiche per uscirne vittoriosi. Il programma è scritto, per quanto possibile, in un BASIC standard (eventuali varianti sono riportate, come al solito, in riquadri di *Basic & Basic*), quindi il programma è adatto alla maggior parte dei computer, purché dotati di sufficiente memoria. Visto che sarebbe impossibile descrivere le varie tecniche di programmazione senza svelare i trucchi ed i segreti del gioco, rovinando così il piacere di giocarlo, parallelamente a *Digitaya* viene sviluppato il gioco *Foresta incantata*, in cui vengono più ampiamente discusse le procedure e gli algoritmi impiegati anche nell'altro gioco.

Progetto della mappa

Il primo passo, nella realizzazione di un gioco d'avventura, consiste nella formulazione di una mappa del mondo immaginario da visitare. Su questa vengono messi in evidenza i vari ambienti e la posizione dei vari oggetti all'interno di essi. Alcune locazioni normali prevederanno semplicemente che il giocatore le attraversi, magari raccogliendo o lasciandovi oggetti. Altre, segnalate come speciali, richiedono particolari condizioni o azioni, prima di potervi accedere o uscire.
In primo luogo occorre decidere quanti ambienti si intendono usare: *Foresta incantata* ha 10 ambienti, disposti su una griglia di 5 x 5 (vedere illustrazione), mentre *Digitaya* ne possiede circa 60, inizialmente progettati su di una griglia 10 x 10.
In un primo momento le caselle della griglia non sono numerate e il progettista inizia assegnando brevi descrizioni agli ambienti. Nella mappa di *Foresta Incantata* c'è un sentiero, due gallerie, una palude, una radura e un villaggio. La posizione dei vari oggetti è indicata in basso nella casella che li contiene. Inoltre, queste locazioni sono marcate con un asterisco, a indicare che sono "speciali".
Una volta completata la stesura "di massima" della mappa, possiamo numerare ciascuna locazione. L'unico accorgimento da tenere presente, nella numerazione, è che le locazioni speciali vanno numerate per prime: l'ordine col quale vengono numerate le altre, invece, non è importante. Ovviamente la numerazione, una volta assegnata, non va più modificata.

Programmazione della mappa

Il primo compito di programmazione riguarda la conversione degli elementi che compongono la mappa in dati per il programma. Esistono numerosi metodi, ma qui adoperiamo due matrici monodimensionali per conservare i dati della mappa. La prima, LN$(), contiene le descrizioni di ciascuna locazione. Per la locazione 7, ad esempio, LN$(7) conterrà sul sentiero. In seguito, il programma visualizzerà questa scritta, facendola precedere da Adesso sei.
La seconda matrice, EX$(), contiene i dati concernenti i possibili movimenti per uscire da ciascuna locazione. In entrambi i giochi, sono previste al massimo quattro uscite: Nord, Est, Sud e Ovest. EX$() contiene i numeri delle locazioni "comunicanti" con quella visitata in un certo momento, sotto forma di stringa alfanumerica, composta da otto cifre. Il numero di locazione, per ciascuna direzione, è immesso come numero a due cifre seguendo l'ordine Nord, Est, Sud e Ovest.
La locazione 7, ad esempio, possiede uscite a Nord, Sud e Ovest, ma non a Est, così le prime due cifre di EX$(7) sono 08 (ricordarsi lo zero per le cifre minori di 10), per indicare che a Nord si trova la locazione 8. La seconda coppia di cifre, 00, indica che non esiste un'uscita in direzione Est. Le coppie 03 e 06, infine, indicano le locazioni rispettivamente abbinate a Sud e a Ovest. Questo sistema offre la possibilità di avere un percorso molto complesso, ma espresso in termini semplici.
I tre oggetti nella *Foresta incantata* sono letti e conservati in un'altra matrice: IV$(,). Questa matrice a due dimensioni tiene conto della posizione di ciascun oggetto, durante lo svolgimento del gioco. Ad ogni oggetto è abbinata una descrizione e il numero della locazione in cui esso si trova all'inizio del gioco. Per esempio, IV$(C,1) può contenere PISTOLA e la sua posizione iniziale è data da IV$(C,2). Quando un oggetto viene trasportato lungo il percorso, la posizione specificata nell'elemento di matrice viene automaticamente aggiornata.
In fondo ai dati delle mappe, nei due listati presentati, c'è un valore di "checksum", che serve per verificare che le DATA siano state digitate correttamente: durante la lettura il programma somma i valori dei singoli dati e confronta il totale con la cifra di "checksum". Se c'è discrepanza, il programma si ferma con un messaggio d'errore. Si noti che in *Digitaya* vengono adoperati due checksum: ciò è necessario perché la somma totale dei dati è troppo grande per un solo valore. In pratica, vengono sommate separatamente le quattro cifre a destra e le quattro a sinistra di ciascun elemento di DATA: i due totali vengono poi confrontati con le rispettive cifre di checksum. Nella prossima parte sono presentate le routine per visualizzare ed elaborare i dati qui riportati.

DIGITAYA

```
6090 REM ** S/R LETTURA MATRICE DEI DATI **
6100 REM * LETTURA INVENTARIO *
6110 DIM IV$(8,2),IC$(4)
6120 FOR C=1TO8
6130 READ IV$(C,1),IV$(C,2)
6140 NEXT C
6150 :
6160 REM * LETTURA AMBIENTE E USCITE *
6170 DIM LN$(55),EX$(55)
6180 C1=0:C2=0:REM INIZIALIZZA LE  CHECKSUM
6190 FOR C=1TO54
6200 READ LN$(C),EX$(C)
6210 C1=C1+VAL(LEFT$(EX$(C),4))
6220 C2=C2+VAL(RIGHT$(EX$(C),4))
6230 NEXT C
6240 READ CA:IFCA<>C1THEN PRINT"ERRORE CHECKSUM":STOP
6250 READ CB:IFCB<>C2THEN PRINT"ERRORE CHECKSUM":STOP
6260 RETURN
6270 REM ** DATI PER L'INVENTARIO **
6280 DATA NUMERO INDIRIZZO,45,CHIAVE,34,SCHERMO
     LASER,25
6290 DATA BIGLIETTO PER IL TRI-STATO,26,CARTA DI
     CREDITO PER DATI,28
6300 DATA DIGITYA,30,CIFRARIO,19,DISPOSITIVO DI
     AZIONAMENTO BUFFER,13
6310 :
6320 REM ** DATI SU AMBIENTI E USCITE **
6330 DATA NEL CAVO TV,00000000
6340 DATA NELLA PORTA UTENTE,00090100
6350 DATA NELLA PORTA DEL REGISTRATORE,00110000
6360 DATA NELLA PORTA DEL JOYSTICK,00130000
6370 DATA IN UN DISPOSITIVO TRI-STATO,00170000
6380 DATA "NELL'ULA",00310016
6390 DATA "ALL'INGRESSO DELLA MEMORIA",00490000
6400 DATA "SULL'AUTOSTRADA DI I/O",09000001
6410 DATA "SULL'AUTOSTRADA DI I/O",10000802
6420 DATA "SULL'AUTOSTRADA DI I/O",11000900
6430 DATA "SULL'AUTOSTRADA DI I/O",12001003
6440 DATA "SULL'AUTOSTRADA DI I/O",13531100
6450 DATA "SULL'AUTOSTRADA DI I/O",14001204
6460 DATA "SULL'AUTOSTRADA DI I/O",15001300
6470 DATA "SULL'AUTOSTRADA DI I/O DI UN SEGNALE
     DICE 'S OUT H'",00001400
6480 DATA NEL REGISTRO DEI DATI,00061700
6490 DATA "SU UNA AUTOSTRADA A 8 CORSIE",16001805
6500 DATA "SU UNA AUTOSTRADA A 8 CORSIE",17001900
6510 DATA "SU UNA AUTOSTRADA A 8 CORSIE",18002000
6520 DATA "SU UNA AUTOSTRADA A 8 CORSIE",19292100
6530 DATA "SU UNA AUTOSTRADA A 8 CORSIE",20282200
6540 DATA "SU UNA AUTOSTRADA A 8 CORSIE",21272300
6550 DATA "SU UNA AUTOSTRADA A 8 CORSIE",22262400
6560 DATA "SU UNA AUTOSTRADA A 8 CORSIE",23250000
6570 DATA NELLA MATRICE DEI CARATTERI,26360024
6580 DATA IN ALTO NELLA MEMORIA,27352523
6590 DATA IN MEZZO DELLA MEMORIA,28342622
6600 DATA IN MEZZO DELLA MEMORIA,29332721
6610 DATA IN BASSO NELLA MEMORIA,00542820
6620 DATA NELLA TANA DELL'ACCUMULATORE,00000600
6630 DATA IN UN LUNGO CORRIDOIO,00420006
6640 DATA IN UN REGISTRO INDICE,31000000
6650 DATA BASSO NELLA MEMORIA,54403428
6660 DATA NEL MEZZO DELLA MEMORIA,33393527
6670 DATA IN ALTO NELLA MEMORIA,34383626
6680 DATA NELLA MATRICE DEI CARATTERI,353700025
6690 DATA IN UNA TABELLA DI VETTORI CASUALI,00000000
6700 DATA "IN ALTO NELLA MEMORIA, SOPRA
     UN'AUTOSTRADA",39003735
6710 DATA NEL MEZZO DELLA MEMORIA,40003834
6720 DATA NELLA MEMORIA - AD EST IS A GATEWAY, 41003933
6730 DATA IN BASSO NELLA MEMORIA,00004054
6740 DATA IN UN CORRIDOIO,00430031
6750 DATA IN UN CORRIDOIO,00440042
6760 DATA IN UN CORRIDOIO,00004543
6770 DATA NEL REGISTRO DEGLI INDIRIZZI,00004600
6780 DATA "SU UN'AUTOSTRADA A 16 CORSIE,45004700
6790 DATA "SU UN'AUTOSTRADA A 16 CORSIE,46004800
6800 DATA "SU UN'AUTOSTRADA A 16 CORSIE,47004900
6810 DATA  "SU  UN'AUTOSTRADA  A 16 CORSIE  C'E'
     UN  GRANDE CANCELLO A OVEST,48005007
6820 DATA "SU UN'AUTOSTRADA A 16 CORSIE,49005100
6830 DATA "SU UN'AUTOSTRADA A 16 CORSIE,50005200
6840 DATA "SU UN'AUTOSTRADA A 16 CORSIE,51000000
6850 DATA IN UN VETTORE DI MEMORIA,00290012
6860 DATA IN BASSO NELLA MEMORIA,00413329
6870 REM ** DATI DI CHECKSUM **
6880 DATA 100169,103973
```

FORESTA INCANTATA

```
6000 REM ** LETTURA DATI DEGLI OGGETTI E DELLA MAPPA **
6010 DIM IV$(3,2),LN$(10),EX$(10),IC$(2)
6020 FOR C=1 TO 3
6030 READ IV$(C,1),IV$(C,2)
6040 NEXT C
6050 :
6060 FOR C=1 TO 10
6065 READ LN$(C),EX$(C)
6070 CC=CC+VAL(EX$(C)):REM TOTALE CHECKSUM
6080 NEXT C
6090 :
6100 READ CD:IFCD<>CC THENPRINT"ERRORE CHECKSUM":STOP
6110 :
6120 REM ** DATI PER GLI OGGETTI **
6130 DATA PISTOLA,10,TORCIA,9,CHIAVE,5
6140 :
6150 REM ** DATI PER LA MAPPA **
6160 DATA "VICINO ALL'INGRESSO DI UN TUNNEL,00000900
6170 DATA IN UNA PALUDE,00000000
6180 DATA IN UN VILLAGGIO,07000000
6190 DATA "VICINO ALL'INGRESSO DI UN TUNNEL,05060000
6200 DATA SU UN SENTIERO,00020400
6210 DATA SU UN SENTIERO,02070004
6220 DATA SU UN SENTIERO,08000306
6230 DATA SU UN SENTIERO,09000702
6240 DATA SU UN SENTIERO,01100800
6250 DATA IN UNA RADURA,00000009
6260 REM ** DATO PER CHECKSUM **
6270 DATA 32253121
6280 RETURN
```



Basic & Basic

I listati principali sono scritti per il Commodore 64. Ecco le modifiche necessarie per altri computer:

Spectrum:

Nel listato di Digitaya, operare le seguenti modifiche:

```
6110 DIM V$(8,2,24):DIM I$(4,24)
6170 DIM L$(55,52):DIM E$(55,8)
6200 READ L$(C),E$(C)
6210 LET C1=C1+VAL(E$(C)(TO 4))
6220 LET C2=C2+VAL(E$(C)(LEN(E$(C))-3 TO))
```

Inoltre, ogni valore delle DATA va racchiuso tra virgolette, eccetto il numero di "checksum", alla linea 6880.
Ecco le modifiche per la "Foresta incantata":

```
6010 DIM V$(3,2,5):DIM L$(10,22)
6015 DIM E$(10,8):DIM I$(2,5)
6030 READ V$(C,1)V$(C,2)
6065 READ L$(C),E$(C)
6067 LET CC=0
6070 LET CC=CC+VAL(E$(C))
```

Inoltre, ogni valore delle DATA va racchiuso tra virgolette, eccetto il numero di "checksum" alla linea 6270.

Micro BBC:

Al listato della "Foresta incantata" va aggiunta la seguente linea:

```
6067 CC=0
```

Nessuna modifica a Digitaya

DATABASE

Ultima scheda delle istruzioni per lo Z80 della Zilog. Istruzioni di Chiamata, di Ritorno e di Restart

ISTRUZIONI DI CHIAMATA E DI RITORNO

			CONDIZIONI									
			SENZA COND.	CARRY	NON CARRY	ZERO	NON ZERO	PARITÀ PARI	PARITÀ DISPARI	SEGNO NEG.	SEGNO POS.	REG. B≠0
CHIAMATA, 'CALL'	ESTENSIONE IMMEDIATA	nn	CD n n	DC n n	D4 n n	CC n n	C4 n n	EC n n	E4 n n	FC n n	F4 n n	
RITORNO, 'RET'	REGISTRO INDIRETTO	(SP) (SP + 1)	C9	D8	D0	C8	C0	E8	E0	F8	F0	
RITORNO DA INTERRUPT, 'RETI'	REGISTRO INDIRETTO	(SP) (SP + 1)	ED 4D									
RITORNO DA INTERRUPT NON MASCHERABILE, 'RETN'	REGISTRO INDIRETTO	(SP) (SP + 1)	ED 45									

Note: Alcuni flag hanno più di un significato: consultare il manuale tecnico della CPU Z80

ISTRUZIONI DI RESTART

		OP CODE	
INDIRIZZO DI CHIAMATA	0000_H	C7	'RST 0'
	0008_H	CF	'RST 8'
	0010_H	D7	'RST 16'
	0018_H	DF	'RST 24'
	0020_H	E7	'RST 32'
	0028_H	EF	'RST 40'
	0030_H	F7	'RST 48'
	0038_H	FF	'RST 56'

Codice mnemonico	Operazione simbolica	Flag S	Z	H	P/V	N	C	Op-code 76 543 210	Hex	N. di byte	N. di Cicli M.	N. di Stati T.	Commenti
CALL nn	(SP—1) ← PC_A (SP—2) ← PC_B PC ← nn	•	•	X	•	X	•	•	•	11 001 101 ←n→ ←n→	CD	3	5	17	
CALL cc, nn	Se la condizione cc è falsa prosegue, altrimenti si comporta come CALL nn	•	•	X	•	X	•	•	•	11 cc 100 ←n→ ←n→		3	3	10	Se cc è falsa
												3	5	17	Se cc è vera
RET	PC_B ← (SP) PC_A ← (SP + 1)	•	•	X	•	X	•	•	•	11 001 001	C9	1	3	10	
RET cc	Se la condizione cc è falsa prosegue, altrimenti si comporta come RET	•	•	X	•	X	•	•	•	11 cc 000		1	1	5	Se cc è falsa
												1	3	11	Se cc è vera
RETI	Ritorno da interrupt	•	•	X	•	X	•	•	•	11 101 101 01 001 101	ED 4D	2	4	14	
RETN[1]	Ritorno da interrupt non mascherabile	•	•	X	•	X	•	•	•	11 101 101 01 000 101	ED 45	2	4	14	
RST p	(SP —1) ← PC_A (SP —2) ← PC_B PC_A ← 0 PC_L ← p	•	•	X	•	X	•	•	•	11 ←t→ 111		1	3	11	

cc		Condizione
000	NZ	Non zero
001	Z	Zero
010	NC	Non Carry
011	C	Carry
100	PO	Parità dispari
101	PE	Parità pari
110	P	Segno positivo
111	M	Segno negativo

t	p
000	00H
001	08H
010	10H
011	18H
100	20H
101	28H
110	30H
111	38H

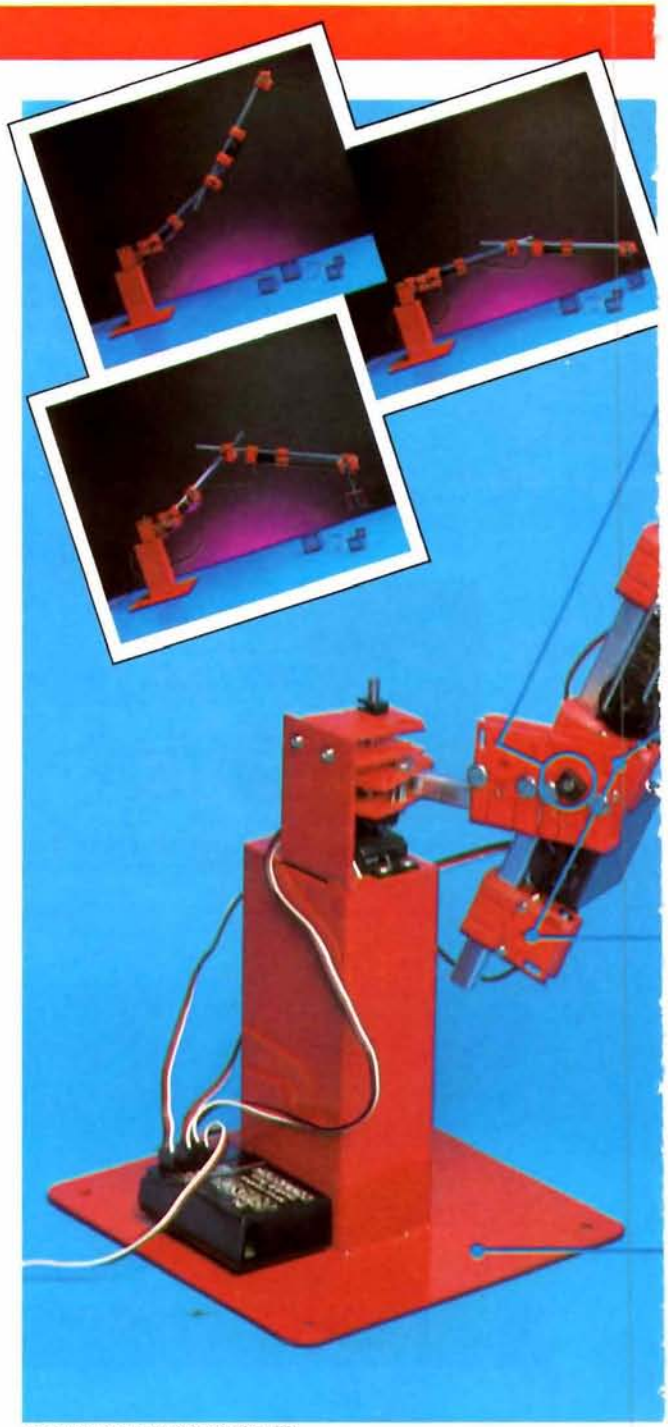
NOTE:
1 RETN carica IFF_2 —> IFF_1

Notazione per i flag:
- • nessun effetto
- x effetto non definito

per concessione della Zilog Inc.

Il Beasty

Uno dei primi bracci robotizzati per home computer, in scatola di montaggio



Per concessione della COMMOTION LTD

Il Beasty, venduto in scatola di montaggio, può essere facilmente montato in poche ore, col solo aiuto di qualche cacciavite. Assieme al robot vengono forniti il software (su cassetta) per farlo funzionare e il Sistema operativo Robol. I manuali in dotazione sono due: uno riguarda le operazioni di montaggio ed è preceduto da una lunga introduzione sulla storia della robotica, l'altro è dedicato al funzionamento del sistema. Per un principiante le istruzioni di assemblaggio risultano un po' scarne e, sarebbe bastato poco per renderle più comprensibili. Comunque, anche i meno esperti riescono a montare il Beasty nel giro di poche ore.

Una volta assemblato, il Beasty consiste in un piedistallo sul quale è montato il giunto che permette i movimenti laterali. A quest'ultimo è collegato il sistema di giunti ed assi che forma il braccio vero e proprio. I movimenti delle giunture sono assicurati da servomotori, ognuno dei quali agisce su due sottili bacchette metalliche, fissate allo "scheletro" del braccio. Ruotando, il servomotore attira una delle due bacchette facendo allontanare l'altra: ciò provoca la rotazione del giunto e lo spostamento delle varie parti del braccio. I servomotori funzionano convertendo segnali digitali in movimenti. Per essere più precisi, il motore riceve ad una partcolare frequenza un treno d'impulsi digitali che vengono interpretati dal microprocessore come angolo di rotazione. Finché la frequenza rimane costante il braccio resta fermo nella stessa posizione. Un cambiamento di frequenza avverte il processore che è richiesta una nuova angolazione, quindi il braccio si sposta.

I servomotori adoperati nel Beasty (si tratta di FP128) sono capaci di generare una trazione di 3,5 Kg/cm, vale a dire che a una distanza di un centimetro dall'asse di rotazione il motore può sollevare 3,5 Kg, ma a una distanza di 10 centimetri il massimo peso sollevabile è di 350 grammi. Questo è un aspetto da considerare attentamente, per non pretendere poi troppo da questo semplice braccio. Il servomotore più lontano dal peso, e quindi quello sottoposto a maggior sforzo, è situato nella "spalla" del braccio.

Il processore necessario ai servomeccanismi è situato in una piccola scatola nera, visibile nell'illustrazione principale. La scatoletta non è fissata in modo permanente al resto del braccio e possiede quattro prese di uscita, per controllare altrettanti servomotori (la quarta è riservata al servomotore, opzionale, dedicato al meccanismo di presa da fissare all'estremità mobile del braccio). Un connettore d'ingresso consente il collegamento alla porta utente del Micro BBC e un cavetto provvede all'alimentazione dell'apparecchio, prelevandola dallo stesso computer.

Il software, una volta caricato e lanciato, inizia con un messaggio di avvertimento, per ricordare all'operatore che è attivo il modo "Edit". Ogni linea di programma in Robol deve contenere un numero di linea, un comando e una serie di numeri, ognuno dei quali corrisponde ad uno dei quattro servomotori.



Cardini
Corrispondono alle "giunture" di un braccio umano

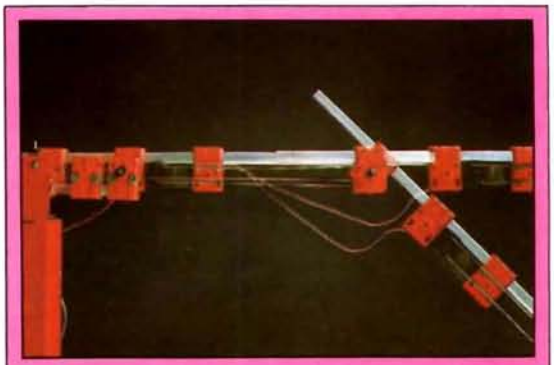
Bacchette di collegamento
Queste sottili coppie di bacchette collegano il servomotore allo "scheletro" del braccio. Quando il motore ruota, una viene attirata, l'altra respinta, facendo muovere il giunto

Servomotori
Questi servono ad imprimere movimento alle varie parti del braccio, tramite coppie di bacchette metalliche

Meccanismo di presa (opzionale)
Una metà del meccanismo è collegata ad un servomotore, l'altra è fissata alla struttura: questa combinazione consente una discreta presa

Piedistallo metallico
È la base di sostegno del braccio. Normalmente non occorre fissarla al piano, purché si presti attenzione nel sollevare e spostare pesi che potrebbero far ribaltare il tutto

IAN McKINNELL



I servomotori
La versione standard del Beasty è venduta con tre servomotori collegati ad un'interfaccia. L'uso di servomotori è molto diffuso nella robotica, poiché questi forniscono una forza di torsione costante, corrispondente alla frequenza degli impulsi applicati

Se la linea di programma contiene il comando MOVE, i numeri si riferiscono alla frequenza degli impulsi che mantengono il braccio in una data posizione. Se si apportano modifiche a tali valori, durante il modo di Edit, il braccio obbedisce immediatamente alle istruzioni, muovendosi nelle varie direzioni. Appena il braccio ha raggiunto la posizione voluta dall'operatore, questi digita un RETURN, chiudendo così la linea ed iniziandone una nuova. Premendo il tasto di funzione F0, si può programmare il braccio in modo che esso esegua un'intera sequenza di movimenti. Tramite il tasto di funzione F1, seguito da F0, si può far iniziare il movimento da qualsiasi punto del programma. Alternativamente, si può modificare il numero di linea tramite il cursore. Al termine del programma la sequenza di azioni viene automaticamente ripetuta, quindi per fermare il braccio occorre sostituire una MOVE con una STOP.

Ritardi cronometrati

Per introdurre delle pause nei movimenti del braccio, esiste una comoda istruzione WAIT, seguita da un valore numerico, da intercalare alle varie MOVE. WAIT funziona generando un segnale d'interruzione sul piedino TIMER1 della porta utente. Poiché gli intervalli sono misurati in 100mi di secondo, una WAIT con un parametro 100 provoca una sosta di 1 secondo tra un movimento è un altro.
La rapidità del braccio può essere notevolmente accresciuta usando istruzioni JUMP (salto), al posto di MOVE (movimento). Inoltre sono fornite due istruzioni, chiamate JDELAY e MDELAY, riferite rispettivamente alle istruzioni JUMP e MOVE, che introducono una pausa nell'esecuzione di JUMP o MOVE, della durata di un quinto di secondo (salvo non venga variata dall'operatore).
L'uso del sistema operativo Robol è molto facile e basta poco tempo per acquisire padronanza del sistema di movimento. Il manuale è succinto, ma perfettamente adeguato allo scopo (i programmatori più smaliziati noteranno la mancanza di informazioni più precise sul modo di programmare il Beasty per azioni complesse). Il controllo del braccio avviene in BASIC, tramite il programma *Drive*, che accede alla porta utente in modo molto simile a quello impiegato nella serie "LABORATORIO" per i nostri progetti.
La ditta produttrice, la Commotion, include anche un breve programma che consente di creare copie di sicurezza del software prodotto col Robol. Sfortunatamente, però, molte unità a dischetti sfruttano l'uscita di alimentazione offerta dal computer, quindi non è possibile azionare il braccio se l'unità a dischetti è collegata al sistema.
Il Beasty, dunque, è poco più di un giocattolo, ma consente di apprendere le nozioni fondamentali che governano l'uso di bracci robotizzati. Può essere molto utile a livello didattico, mentre non consente applicazioni pratiche di rilievo.

BEASTY

INTERFACCE

Il Beasty si collega alla porta utente del Micro BBC, dal quale preleva anche l'alimentazione

SOFTWARE

Il sistema operativo Robol e le routine applicative sono fornite su cassetta

DOCUMENTAZIONE

Sono forniti due manuali: uno riguarda l'assemblaggio delle varie parti, l'altro la programmazione ed il funzionamento del braccio

PRO

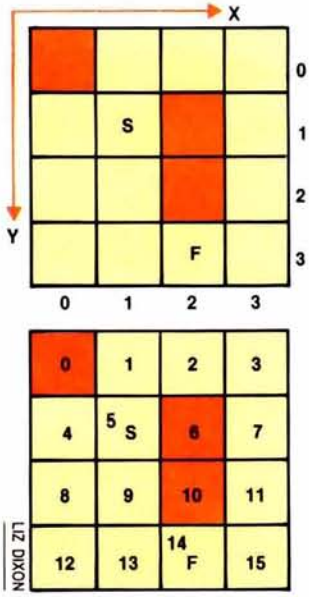
Beasty è facile da montare e da usare e può servire egregiamente a scopo didattico

PRO

I manuali sono un po' carenti. Si tratta di un apparecchio destinato soprattutto alla didattica, quindi non è adatto ad applicazioni pratiche

Perso nel labirinto

Un programma che permette al nostro veicolo di scegliere il percorso più breve



Duplice percorso

Il programma per risolvere il labirinto interpreta quest'ultimo in due modi diversi. I dati che compongono il labirinto vengono letti e depositati in una matrice bidimensionale, assieme alle coordinate del punto iniziale e di quello finale. Per trovare la via d'uscita, il programma deve però costruire una struttura "ad albero", in cui ogni nodo rappresenta una casella del labirinto. Per far ciò, conviene numerare progressivamente ciascuna casella, anziché usare un sistema di coordinate

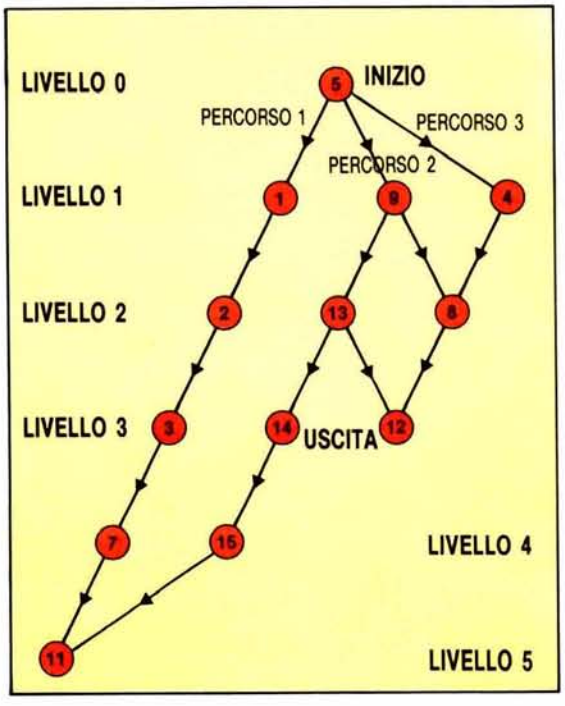
La struttura ad albero

Prima di poter trovare il percorso d'uscita "ottimale", va costruita una struttura ad albero, che rappresenti il rapporto esistente tra le varie caselle del labirinto. Partendo dalla casella iniziale (5 nel nostro esempio) il primo livello di nodi è composto dalle caselle adiacenti accessibili (non occupate da una parete). I nodi non sono disposti a caso: da sinistra a destra corrispondono, rispettivamente, alle direzioni Nord, Est, Sud e Ovest

Il primo passo nella realizzazione di un labirinto per veicoli comandati da computer consiste nello stabilire l'area da esso occupata. Che sia la superficie di un tavolo oppure il pavimento di una stanza, l'area va suddivisa in celle quadrate, le cui dimensioni devono essere adeguate al veicolo usato: quest'ultimo deve poter ruotare comodamente di 360° all'interno di una singola cella. Costruita la griglia, le pareti del labirinto si possono realizzare con qualsiasi oggetto solido che capiti sotto mano: libri, scatole o listelli di legno.

Il programma chiede che vengano specificate le dimensioni del labirinto e fornite le indicazioni sulle celle vuote e su quelle occupate dalle "pareti". Il metodo più pratico per far ciò consiste nell'usare una notazione binaria: 1 indica un quadrato occupato nella griglia, mentre 0 indica una cella vuota. Per evitare di ripetere l'inserimento di questi dati, ogni volta che viene lanciato il programma, è preferibile immetterli sotto forma di frasi DATA. Le ultime due coppie di elementi sono le coordinate iniziali e finali. Supponiamo che il labirinto abbia la propria origine nella cella in alto a sinistra, ossia nella cella individuata dalla riga 0, colonna 0. Ecco le frasi DATA occorrenti:

```
DATA 4,4:REM DIMENSIONI DEL LABIRINTO
DATA 1,0,0,0,0,0,1,0
DATA 0,0,1,0,0,0,0,0
DATA 1,1:REM COORDINATE PUNTO INIZIALE
DATA 2,3:REM COORDINATE PUNTO FINALE
```



Trovare un percorso nel labirinto non è molto difficile: il programma può procedere di una casella per volta, retrocedendo se trova un vicolo cieco, finché non individua l'uscita. Il percorso trovato con questo sistema (anche eliminando i vicoli ciechi) può non essere quello più breve. Se desideriamo invece trovare anche il percorso ottimale, allora va verificata ogni possibile sequenza , alla ricerca di quella che unisce l'ingresso all'uscita attraverso il percorso più breve. Nel nostro caso, il percorso "ottimale" corrisponde a quello che prevede l'attraversamento del minor numero di caselle.

Il compito può essere facilitato costruendo un modello strutturale dei dati, che tenga conto dei rapporti tra i vari quadrati del labirinto. La struttura che meglio si presta a questo tipo di indagine è quella cosiddetta "ad albero ribaltato". La radice dell'albero (situata in alto) rappresenta la casella iniziale. Da essa si dipartono tante diramazioni (terminanti in altrettanti "nodi"), quante sono le caselle accessibili muovendosi di un "passo". Un'ulteriore generazione (o livello) di nodi indica le possibilità per il passo successivo e così via. Qualunque sia il labirinto, possiamo costruire un albero che lo rappresenti: basta numerare le varie caselle ed avere l'accortezza di procedere da sinistra verso destra in base a una sequenza convenzionale (ad esempio: Nord, Est, Sud e Ovest), rispetto al nodo precedente.

Dal nostro semplice labirinto si può uscire in cinque modi diversi, senza mai ripassare per un tratto già percorso. Il diagramma mostra tre possibili percorsi attraverso la struttura ad albero. La soluzione migliore è rappresentata dal percorso 2, ma ciò appare ovvio soltanto perché si è potuto, grazie alla particolare struttura adoperata, analizzare il labirinto nel suo aspetto globale: il computer deve invece esaminare sistematicamente ciascun singolo percorso, per trovare il punto d'uscita o un vicolo cieco. Nel primo caso il percorso va memorizzato come valido, nel secondo esso va marcato come "cieco" per evitarlo in seguito. La ricerca lineare lungo i rami dell'albero si conclude quando si esaurisce ogni possibile combinazione.

Il BASIC non è particolarmente adatto alla formulazione e all'elaborazione di strutture ad albero: questo sono più funzionali il LOGO o l'ALGOL. Comunque, dato che la

maggior parte dei computer possiede il BASIC, il programma viene redatto in tale linguaggio.
La prima cosa da fare è convertire i dati in una forma adatta alla costruzione dell'albero. Per ogni cella, occorrono quattro puntatori, per indicare le possibili direzioni di uscita. Il miglior sistema per conservare i puntatori consiste nell'uso di una matrice bidimensionale TR(N,D), dove N è il numero della cella e D è la direzione (compresa tra 1 e 4). Nel nostro caso, dunque, TR(9,1) vale 5, poiché è questa la casella che si trova a Nord del quadrato 9. Se la cella in una particolare direzione è occupata (a tal fine i contorni del labirinto sono equiparati a caselle occupate) viene usato un valore speciale, ad esempio −1.
Durante l'analisi dell'albero, il percorso viene memorizzato in uno pseudo-stack, realizzato mediante una matrice monodimensionale e una variabile, D, che serve da puntatore alla prossima locazione libera dello pseudo-stack. Anche il percorso considerato via via il migliore viene conservato in una matrice a una dimensione, nel primo elemento della quale si deposita il numero di passi necessari per trovare l'uscita.
Al termine dell'analisi, il miglior percorso è rappresentato da una sequenza di numeri che corrispondono ai quadrati del labirinto. Partendo dal presupposto che il veicolo sia inizialmente rivolto a Nord, esso può essere pilotato usando semplicemente la relazione matematica esistente tra la direzione del



IAN McKINNELL

movimento e la differenza tra due numeri di celle consecutive. Nel nostro caso, ad esempio, una differenza +4 indica Nord, mentre −4 indica Sud. Occorre poi calcolare l'angolo di rotazione per cambiare direzione, prima di passare alla casella successiva. Poiché il veicolo funziona con motorini a corrente continua, sia la rotazione che il tragitto percorso dipendono dal lasso di tempo durante il quale è attiva una particolare combinazione di motori. Per stabilire il tempo occorrente per far ruotare il veicolo di 90° o per farlo avanzare di una casella è necessario compiere qualche esperimento. I rispettivi valori, ricavati empiricamente in questo modo, vanno depositati rispettivamente nelle variabili AF e FF. Nella versione BBC l'unità di misura è in centesimi di secondo, nel Commodore 64 è in sessantesimi di secondo.

CERCA L'USCITA ...

```
900 REM ************************
910 REM ************************
920 REM **    LABIRINTO       **
930 REM **       PER          **
950 REM **   COMMODORE 64     **
960 REM ************************
970 REM ************************
980 :
990 REM ** PROGRAMMA PRINCIPALE **
1000 GOSUB 1600:REM LEGGE I DATI DEL LABIRINTO
1100 GOSUB3700:REM VISUALIZZA IL LABIRINTO
1200 GOSUB5400:REM COSTRUISCE L'ALBERO
1210 GOSUB7700:REM TRAVERSA DELL'ALBERO
1215 IF S(0)=9999 THEN PRINT "NESSUNA SOLUZIONE":END
1220 GOSUB8200:REM DIRIGE IL VEICOLO
1400 END
1500 :
1600 REM ** LEGGE DATI LABIRINTO/INIZIAL. **
1700 READ SX,SY
1800 DIM MZ(SX,SY),TR(SX*SY,4),DR(4),RT(SX*SY),
     LN(SX*SY),CN(SX,SY)
1900 FOR Y=0 TO SX-1
2000 FOR X=0 TO SX-1
2100 READ A:MZ(X,Y)=A
2200 NEXT X,Y
2300 :
2400 READ XS,YS,XF,YF
2500 DATA 4,4
2600 DATA 1,0,0,0,0,0,1,0
2700 DATA 0,0,1,0,0,0,0,0
2800 DATA 1,1:REM COORD. START
2900 DATA 2,3:REM COORD. FINE
3000 :
3100 DR(1)=-SX:DR(2)=1:DR(3)=SX:DR(4)=-1
3110 ID(1)=3:ID(2)=4:ID(3)=1:ID(4)=2
3120 SR(0)=9999:REM PERCORSO PIU' BREVE (INIZ)
3130 DDR=56579:REGDAT=56577:POKE DDR,255
3140 AF=30:FF=45:REM ANGOLAZ. E DURATA (IN AVANTI)
3200 FOR I=1 TO 25:CD$=CD$+CHR$(17):NEXT I
3500 RETURN
3600 :
3700 REM ** VISUALIZZA LABIRINTO **
3800 PRINTCHR$(147):REM RIPULISCE SCHERMO
3900 FOR X=0 TO SX-1
4000 FOR Y=0 TO SY-1
4100 GOSUB 4900:REM POSIZIONE CURSORE
4200 PRINT CHR$(32+MZ(X,Y)*134)
4300 NEXT Y,X
4400 :
4500 X=XS:Y=YS:GOSUB4900:PRINT"S"
4600 X=XF:Y=YF:GOSUB4900:PRINT"F"
4700 RETURN
4800 :
4900 REM ** POSIZIONA CURSORE IN X,Y **
5000 PRINT CHR$(19);:PRINTTAB(X)LEFT$(CD$,Y);:RETURN
5300 :
5400 REM *** COSTRUISCE ALBERO ***
5500 :
5600 REM ** INIZIALIZZA CON VAL. TERMINALI **
5700 FOR P=0 TO SX*SY-1:FORI=1 TO4:TR(P,I)=-1:NEXT I,P
6100 :
6110 REM ** CALCOLA START E FINE **
6120 X=XS:Y=YS:GOSUB9200:S=N
6130 X=XF:Y=YF:GOSUB9200:F=N
6140 :
6200 REM ** COSTRUZIONE **
6300 LC=1:CC=0:REM INIZ. PUNTATORI STACK
6400 LN(LC)=S:REM START POINT
6450 CN=LN(LC):REM ESTRAE NODO CORR. DA STACK
6500 DF=0
6600 FOR D=1 TO 4
6700 N=CN+DR(D):GOSUB 8900:REM CONVERTE IN X,Y
6800 IF(X<0 OR X>SX-1 OR Y<0 OR Y>SY-1)THEN 7300
6900 IF MZ(X,Y)=1 THEN 7300
7000 IF(D=2 AND N/SX=INT(N/SX))THEN7300
7100 IF(D=4 AND CN/SX=INT(CN/SX))THEN7300
7200 IF TR(N,ID(D))=CN THEN 7300
7210 TR(CN,D)=N:DF=1
7220 CC=CC+1:CN(CC)=N:REM APPOGGIA SULLO STACK
7300 NEXT D
7310 IF(DF=0 AND LC=1) THEN RETURN:REM NODO TERMINALE
7320 :
7330 LC=LC-1:REM DECR. ULTIMO PUNTATORE ALLO STACK
7340 IF LC>0 THEN 6450: REM NODO SUCCESSIVO
7345 :
7350 REM ** COPIA STACK CORR. IN ULTIMO STACK **
7360 FOR I=1 TO CC:LN(I)=CN(I):NEXT I
7390 LC=CC:CC=0 GOTO6450:REM NODO SUCCESSIVO
7600 :
7700 REM ** TRAVERSA DELL'ALBERO **
7720 C=0:RN=S:CN=RN:EF=0
7730 C=C+1:RT(C)=CN
7740 IF CN=F THEN GOSUB8100:GOSUB8000:IF EF=0 THEN 7730
7745 IF EF=1 THEN RETURN
7750 DF=0
7760 FOR D=1 TO 4
7770 IF TR(CN,D)<>-1 THEN CN=TR(CN,D):DF=1:DR=D:D=4
7780 NEXT D
7790 IF DF=0 THEN GOSUB8000
7800 IF EF=0 THEN 7730
7810 IF EF=1 THEN RETURN
7820 :
8000 REM ** RIPARTE DALLA RADICE **
8010 TR(RT(C-1),DR)=-1
8020 CN=RN:C=0
8030 IF(TR(CN,1)ANDTR(CM,2)ANDTR(CN,4))=-1
     THEN EF=1
8040 RETURN
8050 :
8100 REM ** SALVA LA MATRICE **
8110 IF C>SR(0) THEN RETURN: REM NUOVO PERCORSO PIU'
     BREVE?
8120 SR(0)=C
8130 FOR I=1 TO C:SR(I)=RT(I):NEXT I:RETURN
8140 :
8200 REM ** DIRIGE IL VEICOLO **
8205 PD=1:REM ASSUME DIREZIONE INIZ = NORD
8210 FOR C=1 TO SR(0)-1
8220 DF=SR(C+1)-SR(C)
8222 :
8225 REM ** TROVA DIREZIONE RICHIESTA **
8230 FOR I=1 TO 4
8240 IF DF=DR(I) THEN D=I:I=4
8250 NEXT I
8260 :
8265 DR=D-PD:PD=D
8270 H=INT(4+DR/4):R=(4+DR)-4*H
8275 :
8277 REM ** ESEGUE SVOLTA **
8280 FOR I=1 TO R
8290 POKE REGDAT=9:REM IN SENSO ORARIO
8300 T=TI
8310 IF (TI-T)<AF THEN 8310:REM ATTENDE
8320 POKE REGDAT=0:REM SPENGE TUTTO
8330 NEXT I
8340 :
8350 REM ** AVANTI **
8360 POKE REGDAT=5
8370 T=TI
8380 IF (TI-T)<FF THEN 8380
8390 POKE REGDAT=0
8400 NEXT C
8410 :
8420 RETURN
8430 :
8900 REM ** CONVERTE N IN X,Y **
9000 Y=INT (N/SX):X=N-SX*Y:RETURN
9200 :
9210 REM ** CONVERTE X,Y IN N **
9220 N=Y*SX+X:RETURN
```

Per il BBC

Apportare le seguenti modifiche:

```
3130 DDR=&FE62:REGDAT=&FE60

8290 ?REGDAT=9
8300 TIME=0
8310 REPEAT UNTIL TIME>=AF
8320 ?DATREG=0

8360 ?DATREG=5
8370 TIME=0
8380 RETEAT UNTIL TIME>FF
8390 ?REGDAT=0
```

Giocare in LOGO

Sviluppare un semplice gioco d'avventura in LOGO è molto facile

In questa prima parte, viene trattato lo sviluppo generico di un programma d'avventura in LOGO; in seguito vengono presentate le routine specifiche per un gioco in particolare. In tutti i giochi d'avventura esistono cinque azioni che il giocatore deve essere in grado di compiere: raccogliere o lasciare oggetti, ottenere un elenco degli oggetti trasportati, "vedere" l'ambiente visitato e potersi muovere da una locazione all'altra. Questi sono, dunque, i comandi che realizziamo per primi. Per semplicità riduciamo i comandi al minimo essenziale: singole parole (come ad esempio: GUARDA) oppure coppie di verbo-sostantivo (come in RACCOGLI ANELLO). Il programma mantiene due liste: una relativa agli oggetti trasportati dal giocatore, chiamata INVENTARIO, e l'altra, chiamata CONTENUTO, con gli oggetti situati nell'ambiente visitato.
Il primo comando definito è INVENTARIO:

```
TO INV
   PRINT [HAI CON TE:]
   IF EMPTY? :INVENTARIO THEN PRINT [NIENTE]
   ELSE PRINT : INVENTRIO
END
```

Si noti che la procedura impiega il formato completo della IF: IF <condizione> THEN <azione1> ELSE <azione2>. Ecco il comando per raccogliere oggetti:

```
TO RACCOGLI :ITEM
   IF MEMBER? :ITEM :CONTENUTO
   THEN PRENDI :ITEM ELSE PRINT [MI SPIACE NON C'È]
END
```

MEMBER? è una primitiva che verifica se un elemento è contenuto in una lista. Per raccogliere un oggetto sono necessarie due azioni: aggiungerlo all'INVENTARIO e rimuoverlo dal CONTENUTO. Ecco le relative procedure:

```
TO PRENDI :ITEM
   AGG.INVENT :ITEM
   RIMUOVI.DA.LUOGO :ITEM
END

TO AGG.INVENTARIO :ITEM
   MAKE "INVENTARIO SENTENCE :ITEM
   :INVENTARIO
END

TO RIMUOVI.DA.LUOGO :ITEM
   MAKE "CONTENUTO DELETE :ITEM
   :CONTENUTO
END
```

L'ultima di queste procedure implica la cancellazione di un elemento da una lista, data come esercizio nella lezione precedente:

```
TO CANCELLA :ITEM :LISTA
   IF :ITEM = FIRST :LISTA THEN OUTPUT BUTFIRST
   :LIST
   OUTPUT SENTENCE FIRST :LISTA DELETE :ITEM
   BUTFIRST :LISTA
END
```

Il comando per abbandonare un oggetto è molto simile:

```
TO LASCIA :ITEM
   IF MEMBER? :ITEM :INVENTARIO THEN ABBAND
   :ITEM ELSE PRINT [NON HAI QUESTO OGGETTO!]
END

TO ABBAND :ITEM
   RIMUOVI.DA.INV :ITEM
   AGG.NEL.LUOGO :ITEM
END

TO RIMUOVI.DA.INV :ITEM
   MAKE "INVENTARIO DELETE :ITEM :INVENTARIO
END

TO AGG.NEL.LUOGO :ITEM
   MAKE "CONTENUTO FPUT :ITEM :CONTENUTO
END
```

Avendo trascritto le procedure, passiamo a verificarne il funzionamento. Prima, però, occorre definire due variabili globali: INVENTARIO e CONTENUTO. Poi possiamo provare i comandi:

```
MAKE "CONTENUTO [SPADA LANCIA TORCIA]
MAKE "INVENTARIO [LANTERNA]
RACCOGLI "SPADA
LASCIA "LANTERNA
```

Adesso esaminiamo CONTENUTO e INVENTARIO, usando questi comandi:

```
PRINT :CONTENUTO
PRINT :INVENTARIO
```

per vedere se tutto è a posto.
Si noti che occorre specificare le virgolette prima del nome degli oggetti: questo può essere fastidioso per chi non è pratico di LOGO. Per consentire l'impiego di comandi più normali, quali RACCOGLI SPADA, è necessario definire SPADA nel seguente modo:

```
TO SPADA
   OP "SPADA
END
```

Ovviamente la stessa operazione va fatta per tutti i nomi di oggetti.
Il comando GUARDA deve visualizzare una descrizione dell'ambiente visitato, assieme ad un elenco di oggetti ivi presenti e di possibili uscite. Per poter produrre scritte che occupino più di una riga di schermo, l'elenco delle descrizioni è definito come una lista di liste. Ad esempio:

```
MAKE "DESCRIZIONE [[SEI IN PIEDI VICINO AD UN INGRES-
SO][DI UNA GALLERIA]]
```

Per tener conto di dove ci si trova, ogni stanza o ambiente è numerato. L'elenco delle uscite è semplicemente una lista di sottoliste, ognuna delle quali consiste in una direzione e in un numero di ambiente. Pertanto:

```
MAKE "LISTA.USCITE [[N 4][E 6]]
```

Adesso possiamo definire GUARDA:

```
TO GUARDA
   PRINTL :DESCRIZIONE
   PRINT "
   PRINT [PUOI VEDERE:]
   IF EMPTY? :CONTENUTO THEN PRINT [NULLA DI
   SPECIALE] ELSE PRINT :CONTENUTO
   PRINT "
   PRINT [PUOI ANDARE A:] PRINT.USCITE :LISTA.USCITE
   PRINT "
END
```

Per facilitare la lettura delle scritte, questa procedura richiama due speciali routine. PRINTL serve per visualizzare più righe di testo:

```
TO PRINTL :LISTA
   IF EMPTY? :LISTA THEN STOP
   PRINT FIRST :LISTA
   PRINTL BUTFIRST :LISTA
END
```

PRINT.USCITE visualizza le possibili uscite, ma non i numeri degli ambienti:

```
TO PRINT.USCITE :LISTA
   IF EMPTY? :LISTA THEN PRINT " STOP
   MAKE "USCITE FIRST :LISTA
   PRINT1 FIRST :USCITE
   PRINT1 " "
   PRINT.USCITE BUTFIRST :LISTA
END
```

Tutto ciò che è possibile conoscere di un ambiente del gioco può essere descritto unendo tre sottoliste: la descrizione, il contenuto e le uscite. Per esempio:

```
MAKE "LUOGO.1 [[[TI TROVI ACCANTO AD UN INGRESSO][DI
UNA GALLERIA]][SPADA][[N 4][E 6]]]
```

Posto che LUOGO.1 sia definito in tal modo, possiamo suddividere le sue componenti con la seguente procedura:

```
TO ASSEGNA.VARIABILI
   MAKE "LUOGO THING "LUOGO.1
   MAKE "DESCRIZIONE DESCRIZIONE :LUOGO
   MAKE "CONTENUTO CONTENUTO :LUOGO
   MAKE "LISTA.USCITE LISTA.USCITE :LUOGO
END
```

THING "LUOGO.1 è un formato alternativo di :LUOGO.1 e significa 'il contenuto della variabile LUOGO.1'. La ragione per usare questo formato viene spiegata tra breve. Le sottoprocedure sono così definite:

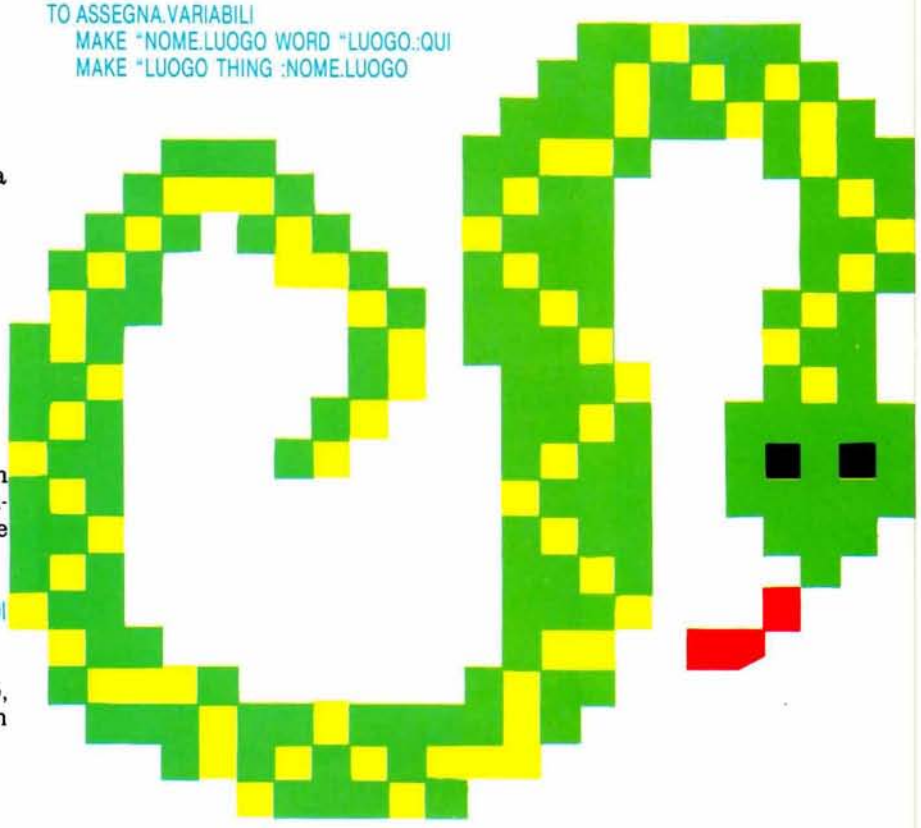
```
TO DESCRIZIONE :LUOGO
   OUTPUT ITEM 1 :LUOGO
END

TO CONTENUTO :LUOGO
   OUTPUT ITEM 2 :LUOGO
END

LISTA.USCITE :LUOGO
OUTPUT ITEM 3 :LUOGO
END
```

Tuttavia, in questa forma, le procedure funzionano soltanto per LUOGO.1, quindi occorre estenderne l'applicabilità ad un LUOGO generico. Ciò si ottiene introducendo una variabile globale, QUI, che contiene il numero dell'ambiente visitato in un dato momento. Supponiamo, ad esempio, che tale numero sia 2. La primitiva WORD del LOGO emette una parola che consiste di una combinazione di due parametri in ingresso (WORD "LUOGO.:QUI produce, dunque: LUOGO.1). Assegniamo quindi questo nome alla variabile NOME.LUOGO: NOME.LUOGO diventa LUOGO.2. Adesso possiamo formulare LUOGO come ITEM :NOME.LUOGO.

```
TO ASSEGNA.VARIABILI
   MAKE "NOME.LUOGO WORD "LUOGO.:QUI
   MAKE "LUOGO THING :NOME.LUOGO
```



```
    MAKE "DESCRIZIONE DESCRIZIONE :LUOGO
    MAKE "CONTENUTO CONTENUTO:LUOGO
    MAKE "LISTA.USCITE LISTA.USCITE :LUOGO
END
```

Adesso è possibile tracciare una mappa degli ambienti per l'avventura ed elencare le loro descrizioni (assieme al contenuto e alle uscite). Nella prossima parte viene spiegato come muoversi da un luogo all'altro e come disseminare il percorso di tranelli e trabocchetti. Inoltre, viene data la traccia per un gioco d'avventura completo.

Logo & Logo

Alcune versioni del MIT LOGO non possiedono EMPTY?, ITEM, COUNT oppure MEMBER?. Le definizioni per le prime tre primitive sono già state date. MEMBER? può essere realizzata con:

```
TO MEMBER? :ITEM :LISTA
    IF :LISTA = [] THEN OUTPUT "FALSE
    IF :ITEM = FIRST :LISTA THEN OUTPUT "TRUE
    OUTPUT MEMBER? :ITEM BUTFIRST :LISTA
END
```

In tutte le versioni LCSI si usi:

EMPTYP per EMPTY?
LISTP per LIST?
MEMBERP per MEMBER?
TYPE per PRINT1

Esiste anche una primitiva, EQUALP, che verifica l'eguaglianza tra due termini. Si usi questa, al posto del segno uguale, per confrontare liste o parole (anche se in alcune versioni LCSI il segno uguale funziona con le liste).
Nel LOGO della LCSI, la sintassi completa per l'espressione IF è illustrata da questo esempio:

```
IF EMPTYP :CONTENUTO [PRINT [NIENTE DI SPECIALE]][PRINT :CONTENUTO]
```

La prima lista, dopo la condizione, viene eseguita se la condizione è vera, la seconda se la condizione è falsa.

RISPOSTE AGLI ESERCIZI

1. Per visualizzare in ordine inverso:

```
TO PRINTR :LISTA
    IF EMPTY? :LISTA THEN PRINT " STOP
    PRINT1 LAST :LIST
    PRINT1 "' '
    PRINTR BUTLAST :LISTA
END
```

La seguente procedura emette in ordine inverso gli elementi della lista, ma senza visualizzarli:

```
TO RIBALTA :LISTA
    IF EMPTY? :LISTA THEN OUTPUT []
    OUTPUT SENTENCE LAST :LISTA RIBALTA
    BUTLAST :LISTA
END
```

2. La procedura per cancellare un elemento da una lista è data nel testo principale.

POESIA CON IL LOGO

Un possibile modo per migliorare le composizioni poetiche con un programma LOGO è quello di creare dei modelli di frasi (ad esempio: sostantivo - verbo - sostantivo) e poi estrarre da sottoliste le parole da immettere nella frase. Ad esempio, usando il modello:

POEMA2 [SOST VERBO SOST]

possiamo avere:

```
TO POEMA2 :MODELLO
    IF EMPTY? :MODELLO PRINT " STOP
    POEMA2.1 FIRST :MODELLO
    POEMA2 BUTFIRST :MODELLO
END

TO POEMA2.1 :PAROLA
    IF :PAROLA = "SOST (PRINT1"' ' GETRANDOM :SOST)
    IF :PAROLA = "VERBO (PRINT2"' ' GETRANDOM :VERBO)
END
```

Questo metodo diventa però molto ingombrante se si introducono molti elementi nella frase. Vediamo se è possibile migliorare le cose intervenendo sulle variabili. Per prima cosa si immetta:

```
MAKE "ROSA "DELICATA
MAKE "ALTRONOME "ROSA
```

Adesso si scoprirà che:

```
PRINT :ROSA visualizza DELICATA
PRINT :ALTRONOME visualizza ROSA
PRINT THING :ALTRONOME visualizza DELICATA
```

THING fornisce il valore associato ad un nome. Nell'ultimo caso, il nome che segue THING è il valore della variabile ALTRONOME, ossia ROSA. Applicando questo sistema, possiamo riscrivere la procedura per le poesie:

```
TO POEMA2.1
    (PRINT1 "' ' GETRANDOM THING :PAROLA)
END
```

Adesso la chiamata POEMA2.1 "SOST assegna il valore SOST alla variabile PAROLA. THING :PAROLA viene quindi abbinato a SOST, ossia la lista di sostantivi.
Usando il modello:

POEMA2 [ART AGG SOST AVV VERBO PREP ART AGG SOST]

assieme ad un'adeguata selezione di vocaboli, possiamo ottenere qualcosa del tipo:

UN VERDE PIANETA SONORAMENTE GIRA SOTTO UN PARANOICO DUOMO
UNA FRASTAGLIATA NAVE LENTAMENTE VAGA VERSO LO SPLENDIDO PIANETA



Bug sotto controllo

Continua lo sviluppo del programma debugger



Ci sono ancora quattro routine da sviluppare per il modulo di I/O: GETHX2, GETHX4, PUTHEX e PUTCR. Le prime due si occupano di accettare e prelevare numeri esadecimali dalla tastiera: GETHX2 accetta un numero di due cifre, mentre GETHX4 ne accetta uno di quattro. La prima cosa da fare, nel progettare queste routine, è quella di decidere se riteniamo obbligatoria l'immissione di due o quattro cifre (più facile da programmare, ma meno professionale) oppure permettere anche l'immissione di meno cifre, seguite da RETURN. Un altro aspetto da considerare è se permettere l'uso del tasto di cancellazione per rimuovere cifre già immesse.
Per GETHX4 usiamo il metodo più semplice, l'immissione obbligatoria di 4 cifre, senza l'uso del tasto di cancellazione. Il valore a 16 bit (in genere un indirizzo) verrà depositato nel registro D.
GETHX2 è un poco più complessa, se si considerano le circostanze in cui viene usata. Occorrono due cifre esadecimali (un valore a 8 bit) per la funzione di modifica della memoria (comando M), che implica l'accesso ad una locazione specificata con un indirizzo. Dopo aver visualizzato il contenuto di tale indirizzo, l'operatore può semplicemente premere RETURN (per passare alla locazione successiva, lasciando il contenuto inalterato) oppure può immettere un nuovo valore esadecimale a due cifre, che sostituisce l'originale. L'ultima possibilità è l'immissione di un carattere (ad esempio di un punto), per terminare l'esame della memoria e tornare al livello dei comandi. Basta aggiungere i due caratteri alla stringa di quelli consentiti. Riassumendo, GETHX2 deve accettare due cifre esadecimali oppure un singolo RETURN o, infine, un singolo punto. Il valore viene depositato nel registro B, usando il registro A per indicare quale delle tre situazioni si è presentata: A=0 se l'immissione è un numero esadecimale, A=1 se contiene un RETURN e A = −1 se si tratta di un punto.
Supponiamo, per il momento, che siano state fatte le seguenti dichiarazioni:

```
CARHEX    FCC '0123456789ABCDEF'
PUNTO     FCB '.'
RETURN    FCB 13 (= codice ASCII per RETURN)
```

La lunghezza della stringa dei caratteri validi risulta così 16 per GETHX4 e 18 per GETHX2.

La routine GETHX2

Dati:
- Prox-caratt è il codice ASCII in A
- Offset è la posizione nella tabella dei caratteri validi, in B
- Val-hex è un valore di 8 bit, costruito nel registro B
- Flag è 0, 1 oppure −1 (registro A)

Processo:
- Accetta Prox-caratt
- SE carattere è un punto (offset = 16) allora pone il Flag = −1
- ALTRIMENTI se il carattere è RETURN (offset = 17) allora pone il Flag = 1
- ALTRIMENTI
  - Salva temporaneamente l'offset
  - Accetta Prox-caratt (solo le cifre sono valide a questo punto)
  - Costruisce il valore esadecimale
- FINE SE

La codifica definitiva di GETHX2 è data a pag. 1259. Per la formulazione di GETHX4 si possono adoperare alcune routine di GETHX2. Assegnando un ingresso secondario, HX4, alla routine GETHX2, possiamo richiamare tale routine ed assicurarsi che vengano accettate soltanto cifre esadecimali valide. Prima, però, il registro B va caricato col valore 16. Il processo per ricavare quattro cifre esadecimali viene così semplificato.

La routine GETHX4

Dati:
- Numero-hex è un valore a 16 bit da depositare in D
- Byte più significativo e
- Byte meno significativo sono entrambi valori a 8 bit, depositati in B

Processo:
- Accetta il Byte più significativo
- Salva temporaneamente il Byte più significativo
- Accetta il Byte meno significativo
- Costruisce il Numero-hex

La codifica della routine GETHX4 è data di seguito alla GETHX2.
Le routine per visualizzare i caratteri sono meno complesse. Per la routine PUTHEX, si presume che il valore a 8 bit si trovi in B.

La routine PUTHEX

Dati:
- Numero è il valore a 8 bit in B
- Offset è lo scarto di 4 bit, in CARHEX

Processo:
Estrae i quattro bit più significativi del numero, usandolo come offset
Visualizza CARHEX(offset)
Estrae i quattro bit meno significativi del numero, usandolo come offset
Visualizza CARHEX(offset)

L'ultima routine necessaria per la gestione dell'I/O è la PUTCR. Questa non presenta alcuna difficoltà ed il relativo codice si spiega da sé. A questo punto possediamo tutte le routine per progettare l'intero modulo di I/O.

Il modulo di I/O

Processo:
AccComando fornisce l'offset, in B, da usare nella tabella dei salti
AccIndirizzo fornisce un indirizzo in D
AccValore fornisce un valore in B e Flag in A
ValoreVis è passato nel registro B
IndirVis è passato nel registro D

La codifica definitiva del modulo di I/O è presentata nel riquadro qui sotto.
Adesso possiamo tornare allo sviluppo del modulo per le Breakpoint (vedere alla pagina 1238).
La codifica del secondo processo di questo modulo (per la definizione dei punti d'interruzione) è già stata presentata.
La codifica del primo processo (per l'inserimento di breakpoint) era stata lasciata da parte, in quanto implicava l'immissione di un indirizzo.
Avendo ora il modulo occorrente, possiamo codificare anche questo processo il quale

LA ROUTINE GETHX2

```
GETHX2  LDB     #18             Numero di caratteri validi
HX4     PSHS    X               Salva il registro usato
        LEAX    HEXCHS,PCR      Indirizzo della stringa di caratteri
                                validi in X
        BSR     GETCH           accetta Prox—carattere
IFOO    CMPB    #16             Se offeset = 16
        LDA     #$FF            Pone Flag = —1 (nel comple-
        BRA     ENDFOO          mento a due)
        CPMPB   #17             Se offset = 17
        LDA     #1              Pone Flag = 1
        BRA     ENDFOO
        LSLB                    Scorrimento di B di quattro posti
        LSLB                    a sinistra, per formare la cifra più
        LSLB                    significativa. B contiene l'offset
        LSLB                    in CARHEX, ossia il valore bina-
                                rio
        PSHS    B               Salva contemporaneamente B
        LDB     #16             Solo cifre hex valide adesso
        BSR     GETCH           Prox-carattere
        ADDB    1,S+            Costruisce il numero a 8 bit,
        PULS    X,PC            perdendo B
```

LA ROUTINE PUTCR

```
PUTCR   PSHS    A               Salva il registro A
        LDA     #13             Codice ASCII per RETURN
        BSR     OUTCH           Visualizza
        PULS    A,PC
```



LA ROUTINE GETHX4

```
GETHX4  LDB     #16
        BSR     HX4             Accetta Byte più significativo
        PSHS    B               Salva temporaneamente Byte più
        LDB     #16             significativo
        BSR     HX4             Accetta Byte meno significativo
                                in B
        PULS    A               Byte più significativo nuovamen-
                                te in A
        RTS                     Valore finale nel registro D
```

LA ROUTINE VISUALIZZA-BREAKPOINT

```
BPLABS  FCC     '1 2 3 4 5 6 7 8 910111213141516'
SPACE   FCB     32              Codice ASCII per spazio
DISPBP  PSHS    A,B,X,Y
        LEAX    BPTAB,PCR       Indirizzo della Tabella-breakpoint
        LEAY    BPLABS,PCR      Indirizzo delle etichette
        CLRB                    Azzera l'offset per Numero-
                                breakpoint
WHIL01  CMBP    NUMBP,PCR       Finché Numero-breakpoint< =
        BGT     ENDW01          Numero di breakpoint
        LDA     ,Y+             Visualizza etichetta
        BSR     OUTCH
        LDA     ,Y+
        BSR     OUTCH
        LDA     SPACE,PCR       Visualizza uno spazio
        BSR     OUTCH
        PSHS    B               Salva temporaneamente B
        LDD     ,X++            Salva indirizzo
        BSR     DSPADD
        PULS    B               Recupera B
        BRA     WHIL01
ENDW01  PULS    A,B,X,Y         Recupera registri e ritorna
```



contiene una chiamata alla subroutine GETADD.
Si noti che, nella codifica, il comando INC NUMBP, PCR somma 1 al Numero-di-breakpoint, ottenendo così il corretto offset nella tabella dei punti d'interruzione. Tuttavia, siccome l'indirizzo è fornito nel registro D (composto dai registri A e B), il contenuto di A andrebbe distrutto, così si usa il registro y per depositare l'indirizzo. Avendo codificato il primo processo per i punti d'interruzione, ne restano altri tre.
Due di questi rappresentano un ribaltamento di due processi già codificati: elimina-breakpoint sostituisce al codice SWI l'opcode originale.
Queste due routine vengono esaminate nella successiva lezione. Della terza ed ultima routine invece, ci occupiamo subito.

Visualizza-breakpoint

Dati:
- Numero-breakpoint è un valore a 8 bit, in B, necessario per la tabella dei punti d'interruzione
- Breakpoint-corrente è l'indirizzo da visualizzare
- Etichette-breakpoint sono numeri a due cifre (decimali) da abbinare agli indirizzi visualizzati
- Spazio è lo spazio che separa un'etichetta dal relativo indirizzo

Processo 3: Visualizza-breakpoint
- Seleziona Numero-breakpoint a 1 (corrispondente ad un offset 0)
- Finché Numero-breakpoint < = Numero breakpoint
- Visualizza Etichetta-breakpoint(Numero-breakpoint)
- Visualizza Tabella-breakpoint(Numero-breakpoint)
- Incrementa Numero-breakpoint
- Fine condizione

Fine del Processo 3

LA ROUTINE PUTHEX

```
PUTHEX  PSHS  A,B,X      Salva i registri usati
        PSHS  B          Salva temporaneamente B
        LEAX  HEXCHS,PCR Indirizzo di CARHEX in X
        LSRB             Quattro scorrimenti a destra per
        LSRB             i bit più significativi
        LSRB
        LSRB
        LDA   B,X        Estrae il carattere, ponendolo in
                         A
        BSR   OUTCH      Lo visualizza
        PULS  B          Recupera B
        ANDB  #%00001111 Maschera i quattro bit più signifi-
                         cativi
        LDA   B,X        Secondo carattere
        BSR   OUTCH
        PULS  A,B,X,PC   Recupera registri e ritorno
```

LA ROUTINE INSERIMENTO-BREAKPOINT

```
BP01    PSHS  A,B,X,Y    Salva i registri usati
        LDX   BPTAB      Indirizzo della Tabella-breakpoint
        LDA   NUMBP,PCR
IF01    CMPA  MAXBP,PCR  Se Numero di breakpoint<Max
        BGE   ENDF01
        INC   NUMBP,PCR  Somma 1 a Numero di Break-
                         point
        LSLA             Moltiplica l'offset per due (la ta-
                         bella è a 16 bit)
        LEAY  A,X
        BSR   GETADD     Estrae indirizzo
        STD   ,Y         Lo deposita nella Tabella-break-
                         point
ENDF01  PULS  A,B,X,Y    Recupera registri e ritorna
```

IL MODULO DI I/O

```
HEXCHS  FCC   '0123456789ABCDEF'
DOT     FCB   '.
RETURN  FCB   13         Codice ASCII per RETURN
COMNDS  FCC   'BUDSGRMQ'
GETCOM  PSHS  A,X        Salva il contenuto di A e di X
        LEAX  COMNDS,PCR Indirizzo dei caratteri di coman-
                         do in X
        LDB   #8         Numero di caratteri validi
        BSR   GETCH      Accetta un carattere
        PULS  A,X,PC
GETADD  BSR   GETHX4
        BSR   PUTCR
        RTS
GETVAL  BSR   GETHX2
        BSR   PUTCR
        RTS
DSPVAL  BSR   PUTHEX
        BSR   PUTCR
        RTS
DSPADD  PSHS  B          Salva temporaneamente B
        TFR   A,B        Byte più significativo in B
        BSR   PUTHEX
        PULS  B          Recupera B
        BSR   PUTHEX
        PULS  B
        BSR   PUTHEX
        BSR   PUTCR
        RTS
```

Mugsy

Un gioco veramente insolito, in cui si assume il ruolo di capo in una "gang" degli anni 30

Questo gioco è prodotto dalla Melbourne House, la stessa che ha realizzato *Hobbit* (vedere a pag. 1020). Il protagonista deve affrontare una serie di compiti decisionali, che riguardano il comportamento della gang. Ad esempio: quanto denaro investire in armi, quanto nella corruzione e via dicendo. Una valutazione errata può causare l'annientamento della banda da parte di avversari o da poliziotti comprati dalle bande concorrenti.
Sullo schermo, il ruolo principale è affidato a Louey, il braccio destro di Mugsy. All'inizio del gioco Louey compare spiegando le regole: il giocatore imposta le proprie decisioni in base ai suggerimenti di Louey e poi attende i risultati. Nel frattempo, viene visualizzata qualche scena animata: la prima mostra una sparatoria tra gangster, la seconda una scorribanda in auto, con un gangster che spara con il suo mitra a più non posso. Dopo questo intermezzo, ricompare Louey con il risultato delle decisioni prese in precedenza. Se gli investimenti sono stati adeguati, l'anno si chiude in attivo e si passa al periodo successivo, immettendo nuove decisioni.
Il punteggio del giocatore è visualizzato, sotto forma di percentuale, al termine del gioco e dipende da fattori quali il numero di "clienti", gli investimenti accantonati e il lasso di tempo in cui Mugsy è riuscito a sopravvivere.

Alcune tecniche di risparmio sono abbastanza evidenti: le immagini sono spesso realizzate con linee diritte e con un'astuta colorazione. Ciò è possibile anche grazie agli economici (in termini di memoria) comandi FILL e DRAW dello speciale package grafico. Il comando DRAW, ad esempio, richiede soltanto due parametri per tracciare linee diritte. Anche per la colorazione delle varie parti di un'immagine animata occorre qualche stratagemma, dettato dal particolare metodo impiegato dallo Spectrum per conservare le informazioni sul colore. Non potendo memorizzare più di due colori per ogni spazio-carattere, occorre una sorta di maschera, che consenta di cambiare colorazione in modo quasi istantaneo. Gli attributi di colore (FLASH, BRIGHT, INK e PAPER) vengono conservati in un unico byte: per cambiare il colore del primo piano va modificato INK, che occupa i tre bit meno significativi del byte. Cosicché sul byte viene prima operato un AND con 248 (per azzerare il valore di INK) e poi un nuovo AND col valore desiderato per INK. Il cambiamento è così rapido da non far notare che in effetti la colorazione è fatta per gradi.
La grafica adoperata in *Mugsy* è senza dubbio eccellente, ma il gioco in sé stanca quasi subito. Inoltre, le scritte (che talvolta appaiono come "fumetti") sono tutte in inglese e, spesso, in uno "slang" poco comprensibile.



Il paese di Gangland
Ecco due scene tratte da "Mugsy". La prima mostra parte della fase dedicata all'immissione delle decisioni, in cui Louey informa Mugsy sulla situazione attuale. La seconda scena si riferisce ad una sparatoria tra gangster e forma parte di una delle sequenze animate usate come intermezzo tra le fasi decisionali

A parte un piccolo riquadro (in basso sullo schermo) nel quale vengono riportati alcuni dati del gioco, il resto è composto da immagini grafiche realizzate mediante il package grafico per lo Spectrum, prodotto dalla Melbourne House. Siccome lo schermo in alta risoluzione dello Spectrum impiega più di 6 Kbyte, è stata evidentemente usata qualche tecnica di compressione dei dati per sfruttare al massimo la poca memoria rimanente.

Mugsy: Per Spectrum 48K
Editore: Melbourne House (GB)
Autori: Phillip Mitchell, Greg Cull, Clive Barrett, Russell Comte
Joystick: Non richiesti
Formato: Cassetta

La conoscenza

L'uso combinato dei sensori fin qui descritti consente ai robot di acquisire conoscenza



Descrivendo i sensori usati dai robot per conoscere il mondo che li circonda, abbiamo considerato i diversi segnali da essi generati come informazioni elaborate in modo individuale. In realtà, un robot ha bisogno di integrare tutte le informazioni ricevute, per poter avere un'idea complessiva di ciò che lo circonda e per costruirsi un modello interno dell'ambiente in cui si muove.

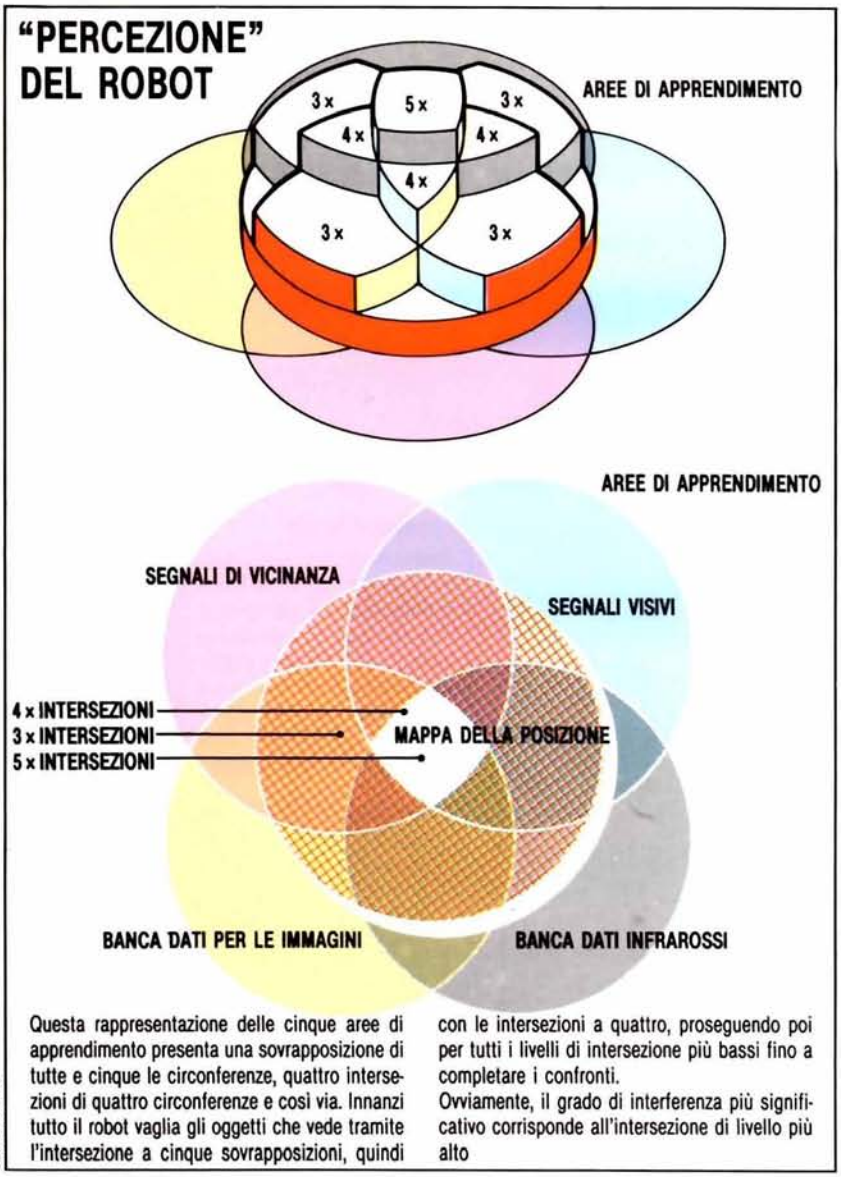
Questo tipo di attività è del tutto simile a quella del cervello umano, che elabora costantemente i segnali provenienti da ciascuno dei cinque sensi, integrandoli e confrontandoli con gli altri. In questo modo l'uomo riesce ad avere una percezione completa della realtà.

Come esempio si può citare la rapidità con la quale persone cieche fin dalla nascita, sottoposte a interventi chirurgici, riescono a recuperare interamente le funzioni visive. Ciò accade perché, nonostante l'infermità, queste persone hanno sviluppato una discreta conoscenza della realtà, basata sull'udito e sul tatto. Una volta recuperata la vista, essi sono in grado di utilizzare la propria esperienza per interpretare ciò che vedono.

È dunque grazie a una buona integrazione dei segnali "in arrivo" che i robot possono usare al meglio i sensori di cui dispongono. Un robot progettato per afferrare oggetti, ad esempio, può eseguire questa operazione "alla cieca", ma fornisce risultati migliori se viene dotato di un sistema visivo. Il robot può allora raggiungere un oggetto posizionato in modo non corretto o con un'angolazione diversa da quella programmata per il suo meccanismo di presa. Ovviamente, la macchina deve prima crearsi un modello interno dell'ambiente in cui opera, integrando poi i segnali provenienti dai sensori. In base al modello il robot può individuare un oggetto, riconoscerlo, posizionare correttamente il proprio "braccio" ed eseguire i calcoli necessari per afferrarlo.

Questa semplice descrizione di un modello interno è tipica dei robot "da labirinto" (vedere a pag. 1252), che usano sensori per individua-

Al nostro servizio ...
Vediamo come un immaginario robot "maggiordomo" potrebbe elaborare le informazioni che riceve dai sensori per eseguire il proprio lavoro. Il problema principale è che gli invitati si muovono continuamente, quindi il robot deve aggiornare di volta in volta il proprio modello interno



Questa rappresentazione delle cinque aree di apprendimento presenta una sovrapposizione di tutte e cinque le circonferenze, quattro intersezioni di quattro circonferenze e così via. Innanzi tutto il robot vaglia gli oggetti che vede tramite l'intersezione a cinque sovrapposizioni, quindi con le intersezioni a quattro, proseguendo poi per tutti i livelli di intersezione più bassi fino a completare i confronti.
Ovviamente, il grado di interferenza più significativo corrisponde all'intersezione di livello più alto

KEVIN JONES

Interfaccia d'intersezione

Di solito i robot ricavano le proprie conoscenze da diverse fonti, tra le quali i database preprogrammati (che contengono i "profili" e le immagini infrarosse degli oggetti più comuni) e le "esperienze precedenti" (ad esempio, il modello dell'ambiente in cui il robot opera). In una situazione ottimale, quando cioè il robot "conosce" alla perfezione la propria posizione e "comprende" la realtà circostante, l'intersezione tra queste aree di conoscenza gli consente di identificare senza possibilità d'errore gli oggetti che "vede"

re la posizione delle pareti nel labirinto e per disegnare una mappa interna bidimensionale dei settori attraversati. Estendendo questo concetto ad un braccio robotizzato, possiamo ottenere mappe tridimensionali. Introducendo un sistema visivo, il robot può creare mappe tridimensionali a più colori e con variazioni di luminosità. Risultati simili sono impossibili da ottenere con un semplice sensore tattile.
Il problema principale, incontrato dai progettisti impegnati nella creazione di robot capaci di interagire con l'ambiente circostante, è che la realtà non è statica, bensì in continuo movimento. Il robot, quindi, deve poterla interpretare per poi adeguarsi a situazioni sempre nuove.
Per chiarire il concetto, si pensi ad un robot programmato per svolgere un semplice lavoro quale la costruzione di una pila di mattoni. Se i mattoni hanno misure diverse, dovranno essere sovrapposti in modo che il loro centro di gravità non cada al di fuori della superficie di base, altrimenti la pila crolla. Sorgono spontanei due interrogativi: quale conoscenza deve avere il robot della legge di gravità e, nel caso la pila crolli, fino a che punto il robot può capire il motivo di quanto accaduto e modificare il proprio comportamento.

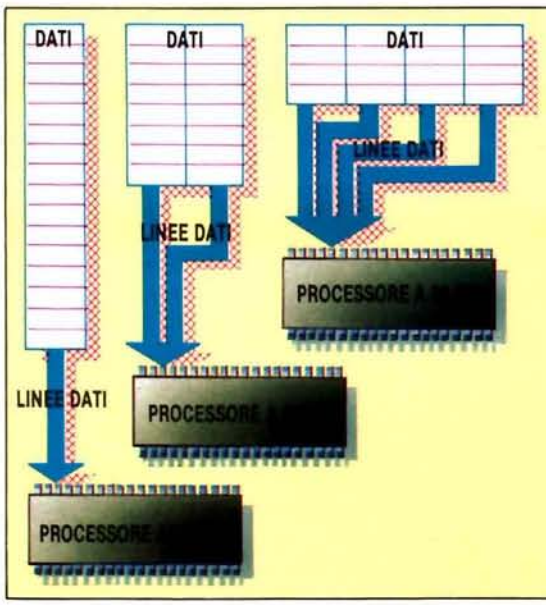
Soluzione del problema

Questi problemi possono essere risolti in due modi. Il primo è programmare il robot con una serie di informazioni che prevedano scelte diverse, a seconda delle situazioni. Questa tecnica ovviamente limita la capacità d'intervento della macchina ad un ben definito campo di applicazioni. In questo caso il robot va quindi programmato in modo da sovrapporre i mattoni esattamente uno sopra l'altro, con tutti i centri di gravità sulla stessa perpendicolare.
Del secondo metodo sono fautori gli studiosi che sostengono il seguente principio: un robot può conoscere l'ambiente che lo circonda solo esplorandolo per conto proprio. Questo campo della scienza dei computer viene comunemente chiamato "apprendimento" o "euristica". In base ai principi di questa scienza, il robot programmato per svolgere una particolare attività viene dotato di segnali di ritorno (feedback), che, generati dai sensori o da un operatore, lo informano su quanto sta facendo. Il robot può quindi migliorare le proprie prestazioni modificando il suo programma interno, ossia il modello dell'ambiente in cui opera, e costruirsi così un "archivio di consultazione" a cui ricorrere per interventi successivi. Questo è il tipo di comportamento adottato dai robot da labirinto, quando cercano la via d'uscita: i sensori consentono loro di individuare i vicoli ciechi, di fare marcia indietro e di tentare una nuova direzione. Sfortunatamente, non esiste ancora un unico programma generico di apprendimento, utilizzabile in qualunque momento si voglia insegnare un nuovo lavoro ad un robot.
Una volta imparata la "lezione" il robot deve memorizzarla sottoforma di programma. Questa operazione viene detta "rappresentazione della conoscenza". Di solito le conoscenze vengono trascritte come linee di codice all'interno di un comune programma, ma i più recenti studi sull'intelligenza artificiale hanno aperto nuove strade. Le tecniche oggi più usate si basano su "regole di produzione", "reti semantiche" e "schede".
Le regole di produzione hanno il formato delle strutture IF ... THEN e sono semplici istruzioni operative. In base ad esse, le conoscenze di un robot possono essere codificate nel modo seguente: se (IF) hai di fronte un muro, allora (THEN) non puoi avanzare. Esiste tutta una serie di tali regole, che il programmatore può facilmente intuire e scrivere ma, affinché il robot possa interpretarle, deve possedere uno strumento d'"inferenza" o di "deduzione" che gli consenta di elaborarle durante lo svolgimento di un'azione. I programmi basati sulle regole di produzione possono essere scritti in un linguaggio convenzionale, come il BASIC, ma di solito i programmatori usano un lin-

guaggio basato su "Dichiarative", quale il LISP, più adatto alla manipolazione di questo genere di informazioni. A differenza degli altri, i linguaggi dichiarativi non eseguono le istruzioni una alla volta ma tengono sotto costante controllo una serie di situazioni correlate ad una o più regole di produzione. Quindi, a seconda delle circostanze, viene attivata ed eseguita una particolare istruzione che a sua volta ne seleziona altre "in cascata".
Le reti semantiche sono particolari strutture di correlazione, paragonabili a reti di collegamento tra le diverse conoscenze di un robot. Sono dette "semantiche" perché le linee di connessione non sono semplici "ponti" tra i nodi della rete, ma spesso indicano invece un particolare tipo di rapporto tra i dati collegati. Perciò il ponte che collega un nodo chiamato "tavolo" ad uno detto "mobilio" può indicare che il tavolo è un particolare tipo di mobile. Al collegamento viene associato il concetto di "tipo".
Questo metodo di rappresentazione può essere programmato in un comune linguaggio come il BASIC, usando variabili stringa per definire i nodi ed i ponti di collegamento ma, come già detto, viene di solito impiegato il LISP, un linguaggio tipico dell'intelligenza artificiale.

Questionario

Le "schede" sono paragonabili a questionari progettati appositamente per le diverse situazioni che un robot deve affrontare e possono essere facilmente programmate in BASIC, ricorrendo ad una serie di matrici stringa a due dimensioni: una per le domande ed una per le risposte.
Con questa tecnica un robot può raggiungere una completa conoscenza della situazione da affrontare, ma solo dopo aver risposto a tutte le domande della scheda: solo allora può decidere quale azione intraprendere. I sistemi più perfezionati prevedono schede di diverso tipo, che vengono selezionate dal robot a seconda delle situazioni.
I linguaggi di programmazione giocano un ruolo fondamentale nelle varie tecniche di rappresentazione della conoscenza. Il LISP, ad esempio, è un linguaggio dedicato alla gestione delle liste e si dimostra particolarmente adatto all'archiviazione e al recupero delle informazioni.
Tuttavia la scelta del linguaggio deve tenere conto del tipo di conoscenza che si vuol rappresentare.
I diversi sistemi di programmazione che stanno alla base dell'apprendimento dei robot sono oggetto di approfondite ricerche in tutto il mondo, soprattutto nelle università. In particolar modo vengono studiate le applicazioni dei sensori, dei sistemi di feedback e delle tecniche di apprendimento e conoscenza.
Il prossimo argomento trattato riguarda la programmazione dei sistemi di simulazione, che permettono ai computer di "imitare" situazioni ed eventi del mondo reale.

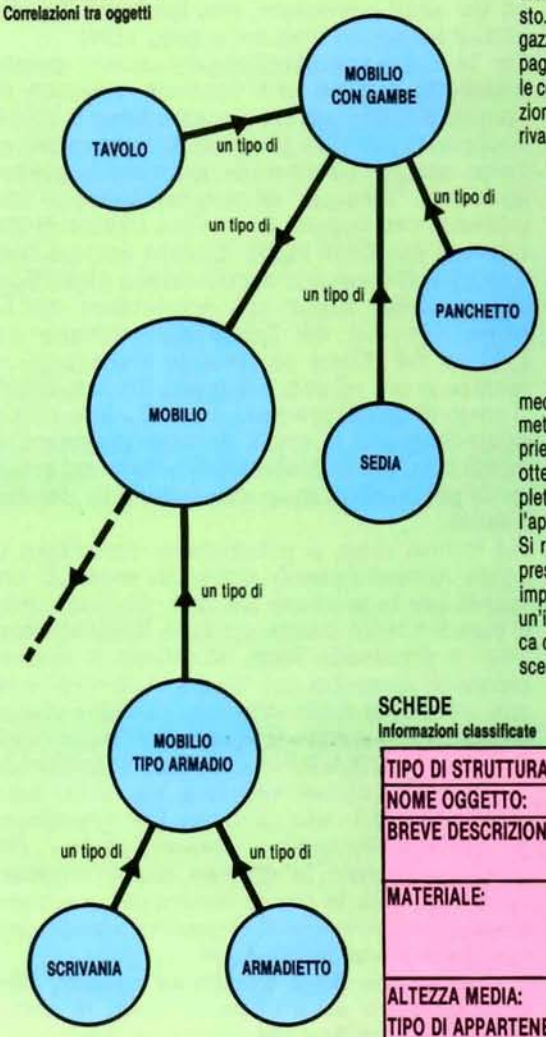


LINEE PARALLELE

Le informazioni che i sensori inviano al sistema d'elaborazione di un robot devono essere analizzate con grande rapidità, per consentire una risposta veloce. La lunghezza della "coda di parcheggio" dei dati in arrivo dipende dalla complessità dell'algoritmo preposto alla loro elaborazione. Se la quantità di informazioni prodotte da un robot sofisticato è superiore alle capacità operative di processori come lo Z80 o il 6502, le code d'attesa tendono ad allungarsi. Per eliminare questo problema si può ricorrere a processori a 16 o a 32 bit, capaci di elaborare nello stesso tempo un numero doppio o quadruplo di informazioni

SENTIERI DI APPRENDIMENTO

RETE SEMANTICA
Correlazioni tra oggetti



Le conoscenze di un robot non sono altro che una serie di informazioni, tutte riferite ad un particolare contesto. I computer sono capaci di immagazzinare dati, ma non risultano equipaggiati in modo specifico per gestire le correlazioni esistenti tra le informazioni e le conoscenze che da esse derivano. È quindi necessario sviluppare apposite strutture per la rappresentazione delle diverse conoscenze e le più usate sono: le "reti semantiche", ossia gruppi di informazioni che vengono archiviate come liste tra loro collegate; le "schede", costituite da semplici matrici a due dimensioni e le "regole di produzione", composte da liste di informazioni mediate da operatori logici. Ciascun metodo possiede applicazioni proprie, quindi è necessario abbinarli per ottenere rappresentazioni il più complete possibile. A fianco è illustrata l'applicazione di una rete semantica. Si rammenti che si tratta di una rappresentazione "statica", cioè senza implicazioni di movimento, basata su un'interazione tra il software di ricerca del robot e il suo archivio di conoscenze

SCHEDE
Informazioni classificate

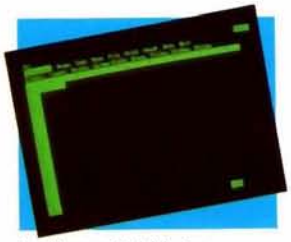
TIPO DI STRUTTURA:	1
NOME OGGETTO:	PANCHETTO
BREVE DESCRIZIONE:	3 GAMBE SEDILE
MATERIALE:	LEGNO METALLO PLASTICA
ALTEZZA MEDIA:	0,5 - 1 m
TIPO DI APPARTENENZA:	MOBILIO CON GAMBE SEDILI

REGOLE DI PRODUZIONE
Linea di programmazione Booleana

IF PANCHETTO THEN (3 GAMBE AND SEDILE) AND (LEGNO OR METALLO OR PLASTICA)

I Macro-comandi del Lotus

Le caratteristiche di uno spreadsheet professionale



La "lavagna" del Lotus
Come sul "VisiCalc", il menu principale del "Lotus" viene visualizzato premendo il tasto /



Risparmio Macro
Un'altra immagine del "Lotus" durante l'introduzione di un Macro-comando. Se necessario, le istruzioni Macro possono occupare più colonne

I programmi per "lavagna elettronica" fin qui esaminati sono stati sviluppati per home computer quali il BBC, lo Spectrum, il Commodore 64 ed il QL della Sinclair. Ovviamente, il loro livello qualitativo è condizionato dalle ridotte dimensioni della memoria a disposizione per i programmi stessi e per i modelli creati. Programmi simili, ma questa volta scritti per il PC della IBM, per l'Apricot ATC e per altri apparecchi gestionali, possono contare su maggiore capacità di memoria e velocità di elaborazione. Di conseguenza alcuni dei migliori programmi per la creazione di modelli finanziari possono girare solo su costosi computer del tipo appena citato. Un tipico esempio è il *Lotus 1-2-3*, un package composto da uno spreadsheet, un database ed un word processor, che lavorano in ambiente integrato (vedere a pag. 1124).
Per la sua particolare impostazione, questo pacchetto occupa una notevole quantità di memoria. Oltre ad un discreto spazio RAM, necessario per i tre programmi applicativi, al Lotus occorre un'ulteriore area per la gestione di una "lavagna" di programmazione che possiede una capacità massima teorica di 256 colonne per 1.028 righe. Questa occupa una quantità di memoria commisurata al particolare modello usato, ma nonostante ciò le prime versioni del Lotus richiedevano un minimo 128 Kbyte per la sola elaborazione, mentre le più recenti occupano almeno 256K! Il costo di un programma come il Lotus 1-2-3 e di un computer in grado di farlo funzionare è molto alto, ma le prestazioni offerte dal gruppo di programmi sono numerose e di elevata qualità.
Per prima cosa, il programma visualizza il Lotus Access System, ossia un menu di comandi per la gestione dei dati. Posizionando il cursore sulla prima opzione (indicata con 1-2-3) e premendo Return, si ottiene il caricamento in memoria del "foglio di lavoro" e la sua comparsa sullo schermo. Le righe che lo compongono sono etichettate con lettere dell'alfabeto, le colonne con numeri. Il Lotus 1-2-3, come il *Multiplan* (vedere a pag. 1244), è un programma pilotato da menu. Per visualizzare il menu principale si preme il tasto /. Da questo momento le diverse scelte vengono fatte digitando la prima lettera del comando desiderato oppure posizionando il cursore sul comando e premendo Return.
Il programma offre numerose opzioni, che corrispondono ad altrettanti livelli di sottomenu. Ciò significa che l'utente può adoperare il Lotus per eseguire centinaia di operazioni ma anche che, per selezionarne alcune, deve digitare una nutrita sequenza di tasti (si veda la figura alla pagina seguente). Chiariamo questo concetto con un esempio. Il Lotus permette di assegnare un nome ad una cella (o gruppo di celle), consentendo poi di usarlo al posto delle coordinate delle celle, nella formulazione di espressioni matematiche. La differenza si apprezza immediatamente:

A3—B3 = C3	VENDITE—COSTI = GUADAGNO
(coordinate di cella)	(nomi di riferimento)

L'attribuzione di nomi a particolari gruppi di celle semplifica e velocizza la ricerca dei dati su una lavagna di grandi dimensioni. Per assegnare un nome ad un gruppo di 4 celle si devono premere i seguenti tasti:

/	visualizza il menu principale;
R(ange)	comunica al programma che si intende operare su un gruppo di celle e non sull'intera lavagna;
N(ame)	permette di assegnare un nome al gruppo di celle interessato;
C(reate)	predispone il programma a ricevere un nuovo nome e ad incorporarlo;

Adesso si digita il nuovo nome, ad esempio: "INIZIO";
Si preme Return per selezionare la prima cella del gruppo;
Si sposta il cursore a destra di 4 celle;
Si preme Return per selezionare l'ultima cella del gruppo.

L'intera operazione implica 10 battute di tasti, più quelle per il nome.
Il Lotus consente però di ridurre il numero di tasti da usare, grazie all'impiego di "Macro-comandi", paragonabili a brevi programmi scritti nel linguaggio operativo dell'1-2-3. Questi vengono creati memorizzando i tasti necessari in una piccola sezione della lavagna ed attribuendo a questa un nome, che viene poi abbinato ad un particolare tasto. Quando questo viene premuto assieme ad un tasto di funzione speciale (etichettato con la sigla ALT sul PC della IBM e sugli apparecchi compatibili), il Lotus esegue automaticamente la sequenza di battute.
L'assegnazione dei nomi può essere automatizzata, assegnando ai Macro-comandi uno specifico gruppo di celle della lavagna, purché si tengano presenti due importanti considerazioni. Innanzi tutto le celle devono occupare un'area sicura (nella quale non verranno mai inseriti altri dati); in secondo luogo va ricordato il criterio adottato dal Lotus per allocare memoria: le celle "attivate", ossia quelle indicate in precedenza con il cursore, e quelle comprese tra loro occupano memoria. Quindi, se un'area Macro viene posta all'estremità destra della lavagna, può accadere che molti Kbyte di preziosa memoria vengano occupati da celle vuote. È consigliabile, dunque, inserire i Macro-comandi a partire dal margine sinistro, ad esempio nelle colon-

ne A, B e C. Se tutte le formule inserite nella lavagna procedono da sinistra a destra, le aree Macro non rischiano di essere alterate. Supponiamo di definire il nostro Macro-comando nella colonna A. Se il numero di caratteri previsto da una singola cella non basta per la definizione, si può tranquillamente "sconfinare" in quelle adiacenti sulla stessa riga. Posizionando il cursore sulla cella A1, battiamo la sequenza di comandi:

'/ R N C

A questo punto, il programma attende l'inserimento del nuovo nome. Si può inserire una pausa, digitando un punto interrogativo racchiuso tra parentesi tonde. L'attesa si prolunga finché l'utente non preme il tasto Return, quindi il Lotus passa al Macro-comando. Il formato tra parentesi tonde è usato coerentemente, nel Lotus, per indicare tutte quelle azioni non selezionabili con un apposito tasto, compreso il posizionamento del cursore, che viene rappresentato da una parola tra parentesi tonde. Il ritorno carrello è indicato invece con una "tilde", ~. Il Macro-comando continua dunque con:

'/ R N C (?)~(right) (right) (right)~

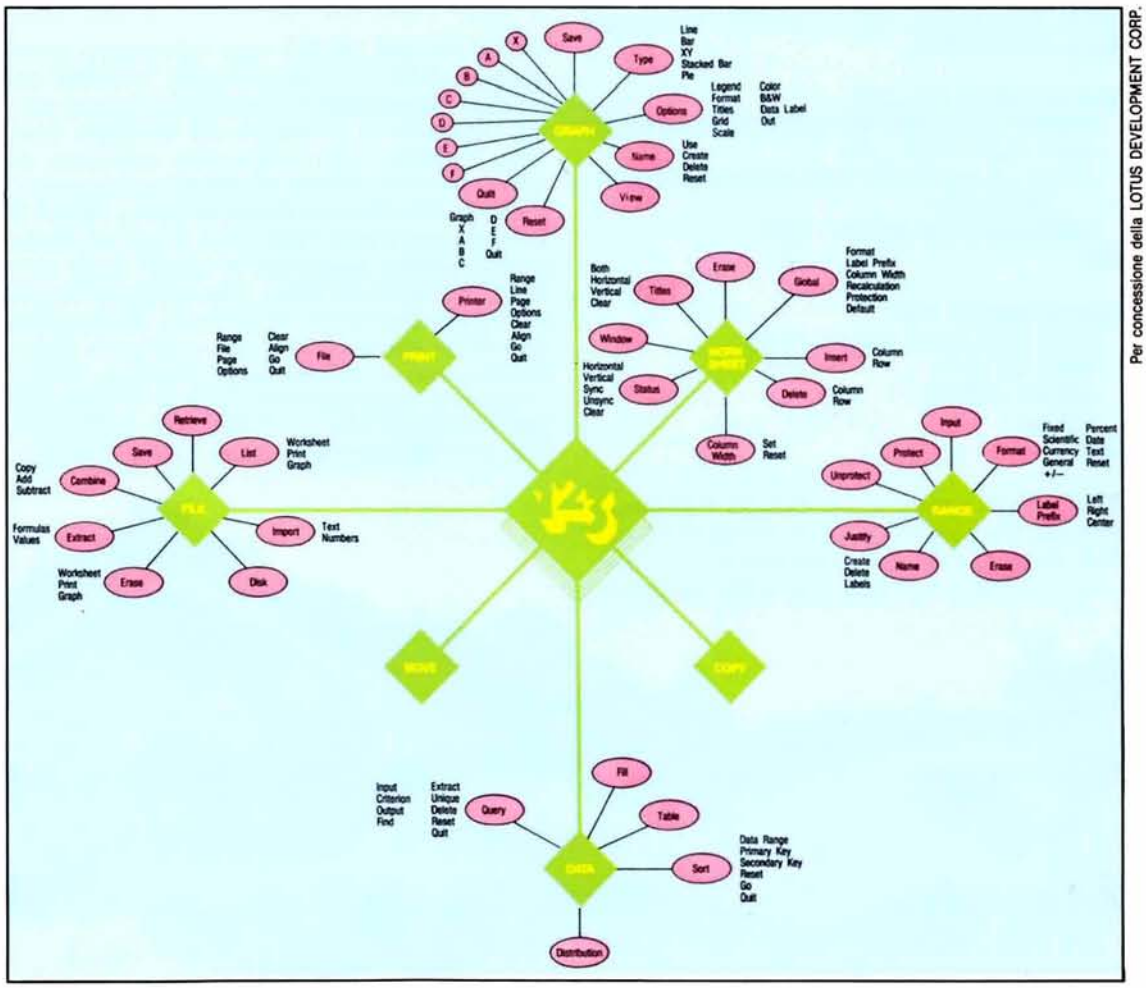
che contiene tutta la sequenza di tasti necessaria per l'assegnazione del nome.
La fase successiva riguarda l'abbinamento del nome ad un tasto, ad esempio: N (per Nome). Così facendo, però, si entra in un circolo vizioso. Infatti si deve ripetere manualmente la sequenza del Macro-comando, per poter assegnare ad esso un nome e archiviarlo! Posizioniamo quindi il cursore sulla cella A1 e scriviamo:

'/ R N C \ N Return Return

Il carattere \ corrisponde alla pressione del tasto ALT, perciò \ N comunica al programma la sequenza di caratteri (o nome) da abbinare al Macro-comando. Avendo definito il Macro-comando "Nome", possiamo premere contemporaneamente ALT ed N per attivare la sequenza di tasti memorizzata. A questo punto possiamo assegnare un nome a qualsiasi area della lavagna scrivendo:

ALT—N NOMEDESIDERATO Return

È evidente il notevole miglioramento, rispetto al procedimento originario.
Quello proposto è un banale esempio delle immense potenzialità dei Macro-comandi. Non ci sono limiti al numero di tasti o sequenze di battute automatizzabili coi Macro. Questa tecnica permette di personalizzare il software applicativo come se si trattasse di programmare in BASIC o in PASCAL. Sebbene il Lotus 1-2-3 sia molto costoso e limitato ai sistemi gestionali, si può applicare il principio dei Macro-comandi al software per home computer.



Per concessione della LOTUS DEVELOPMENT CORP.

Livelli di comando del Lotus

La figura rappresenta le possibili diramazioni di tutti i diversi comandi contenuti nei sotto-menu del "Lotus". Tutti i comandi indicati sono "residenti" (ossia incorporati automaticamente all'accensione) ma, tramite i Macro-comandi, l'utente può crearne di nuovi

Di luogo in ... LOGO

Ecco la procedura per muoversi tra i vari ambienti

Dopo aver creato la struttura di base del gioco, esaminiamo le procedure che controllano gli spostamenti da un ambiente all'altro. Il gioco prevede quattro direzioni di movimento: nord, sud, est ed ovest.

```
TO N
   MOVE "N :LISTA.USCITE
END

TO S
   MOVE "S :LISTA.USCITE
END

TO E
   MOVE "E :LISTA.USCITE
END

TO O
   MOVE "O :LISTA.USCITE
END
```

Una procedura chiamata SPOSTAMENTO verifica la validità della direzione selezionata, quindi affida l'esecuzione del movimento alla procedura SPOSTAMENTO1.

```
TO SPOSTAMENTO :DIR :LISTA
   IF EMPTY? :LISTA THEN PRINT [DIREZIONE VIETATA] STOP
   MAKE "USCITA FIRST :LISTA
   IF :DIR = FIRST :USCITA THEN SPOSTAMENTO1 LAST
      : USCITA STOP
   SPOSTAMENTO :DIR BUTFIRST :LISTA
END

TO SPOSTAMENTO1 :NUM
   MAKE :NOME.LUOGO PARTICOLARI.QUI
   MAKE "QUI :NUM
   ASSEGNA.VARIABILI
   GUARDA
END
```

SPOSTAMENTO1 riceve, come dato in ingresso, il numero d'ambiente. Quindi riorganizza una lista dei vari elementi contenuti nel suo interno e la abbina al nome dell'ambiente (possono essersi verificati dei cambiamenti mentre il giocatore si trova al suo interno). Infine assegna a QUI il nuovo numero d'ambiente e riattribuisce le varie liste. La procedura usata è la seguente:

```
TO PARTICOLARI.QUI
   OUTPUT (LIST :DESCRIZIONE :CONTENUTO :LISTA.USCITE)
END
```

La primitiva LIST crea una lista dei dati che la procedura riceve in ingresso. I seguenti esempi chiariscono la differenza tra i comandi LIST e SENTENCE:

```
LIST [A] [B] [C] dà come risultato [[A] [B] [C]]
SENTENCE [A] [B] [C] dà come risultato [A B C]
```

Dato che vogliamo conservare i diversi elementi sotto forma di sottoliste, si usa LIST e non SENTENCE.

I "pericoli" del gioco

Di solito, nei giochi tipo adventure sono presenti svariati e caratteristici "pericoli" da evitare, come serpenti velenosi o sabbie mobili. Quando il giocatore ne incontra uno, deve eseguire una particolare sequenza di operazioni ed evitare qualsiasi spostamento esterno all'ambiente in cui si trova, finché il pericolo non è stato superato. A questo scopo il programma aggiunge a quella degli ambienti un'ulteriore lista , che contiene i nomi delle varie procedure di pericolo, da eseguire quando si entra in un nuovo ambiente. Quindi possiamo assegnare alla procedura LUOGO.2 i seguenti messaggi [[[PERCORRI UN SOTTERRANEO UMIDO E BUIO] [DI FRONTE A TE BRILLA UNA LUCE]] [SCRIGNO] [[N5] [E6]] [SERPENTE]] dove SERPENTE indica



un pericolo. L'introduzione di questa nuova lista richiede una modifica della procedura GUARDA:

```
TO GUARDA
   PRINTL :DESCRIZIONE
   PRINT "
   PRINT [PUOI VEDERE:]
   IF EMPTY? :CONTENUTO THEN PRINT [NIENTE
      DI SPECIALE] ELSE PRINT: CONTENUTO
   PRINT "
   PRINT [PUOI DIRIGERTI A:] STAMPA.USCITE :LISTA.USCITE
   PRINT "
   IF PERICOLO? THEN RUN :PERICOLI
END
```

RUN è una primitiva LOGO che usa una lista come dato in ingresso ed esegue le procedure in essa contenute. Quindi, se la procedura [SERPENTE] è assegnata a :PERICOLI, l'istruzione RUN :PERICOLI la invia in esecuzione.

```
TO PERICOLI?
   IF EMPTY? :PERICOLI THEN OUTPUT "FALSE
   OUTPUT "TRUE
END
```

L'introduzione delle procedure di pericolo comporta la modifica di numerose altre procedure:

```
TO ASSEGNA.VARIABILI
   MAKE "NOME.LUOGO WORD "LUOGO. :QUI
   MAKE "LUOGO THING :NOME.LUOGO
   MAKE "DESCRIZIONE DESCRIZIONE :LUOGO
   MAKE "CONTENUTO CONTENUTO :LUOGO
   MAKE "LISTA.USCITE LISTA.USCITE :LUOGO
   MAKE "PERICOLI PERICOLI :LUOGO
END

TO
PERICOLI:LUOGO
OUTPUT OGGETTO 4 :LUOGO
END

TO PARTICOLARI.QUI
   OUTPUT (LISTA :DESCRIZIONE :CONTENUTO
      :LISTA.USCITE :PERICOLI)
END

TO SPOSTAMENTO :DIR :LISTA
   IF PERICOLO? THEN PRINT [DIREZIONE VIETATA] STOP
   IF EMPTY? :LISTA THEN PRINT [DIREZIONE VIETATA] STOP
   MAKE "USCITA FIRST :LISTA
   IF :DIR = FIRST :USCITA THEN SPOSTAMENTO1 LAST
      :USCITA STOP
   SPOSTAMENTO :DIR BUTFIRST :LISTA
END
```

Finché la lista PERICOLI non risulta vuota (ossia []), la procedura SPOSTAMENTO impedisce l'effettuazione di qualsiasi movimento. Questo tipo di programmazione dei pericoli consente di inserire e spostare uno stesso pericolo da un ambiente all'altro modificando unicamente la descrizione dell'ambiente.
Adesso possiamo usare le procedure descritte fin qui per creare un gioco d'avventura, *Il Tempio di Zoltoth*, in cui il giocatore deve ritrovare lo scettro di Gilgesh, trafugato dai Sacerdoti di Zoltoth e nascosto nei sotterranei del loro tempio. Iniziamo progettando lo sviluppo del percorso "vincente", quindi sviluppiamo attorno ad esso il resto del gioco (ovviamente il percorso vincente non viene descritto per non privare il giocatore del divertimento del gioco!).
La seconda fase riguarda la progettazione dei sotterranei, ossia dei diversi ambienti di gioco, delle loro posizioni relative e degli oggetti in essi contenuti. Questo mondo fantastico viene infine usato per definire le diverse locazioni all'interno del programma, indicando le uscite possibili da ciascuna di esse. A loro volta i giocatori devono costruire una mappa dei sotterranei che attraversano. Infine, si devono scegliere le parole che compongono il vocabolario dei comandi, cioè le parole che il programma è in grado di interpretare. Queste sono:

1) Sette comandi formati da una singola parola o lettera: INIZIO, GUARDA, N, S, E, O e INVENTARIO (tutte procedure viste in precedenza).
2) Comandi a due parole, composti da un verbo seguito da un nome.

I verbi sono: PRENDERE, ELIMINARE, ESAMINARE, UCCIDERE, STROFINARE e APRIRE.
I nomi sono: SPADA, SCRIGNO, SCETTRO, ANELLO e SERPENTE.

Tutti questi comandi vengono inviati direttamente al LOGO e, se vengono accettati, il programma agisce di conseguenza, altrimenti visualizza un messaggio d'errore LOGO. Tuttavia è preferibile sostituire ai comuni messaggi d'errore del LOGO frasi del tipo: «Non conosco questa parola», e via dicendo.



DAVID HIGHAM

Si deve quindi introdurre una serie di procedure che ricevano i comandi battuti sulla tastiera, ne verifichino la validità e, eventualmente, li eseguano.
Le seguenti procedure sovraintendono alla verifica delle parole di comando precedentemente esposte:

```
TO INIZIO
   MAKE "QUI1
   MAKE "INVENTARIO []
   CREA.LUOGHI
   ASSEGNA.VARIABILI
   GUARDA
   GIOCO
END

TO GIOCO
   PRINT1 "COMANDO:
   MAKE "INPUT REQUEST
   IF VALIDO? :INPUT RUN :INPUT ELSE PRINT [NON
      CAPISCO]
   GIOCO
END

TO VALIDO? :COM
   IF ((COUNT :COM) = 1) THEN OUTPUT VAL1? :COM
   IF ((COUNT :COM) = 2) THEN OUTPUT VAL2? :COM
   OUTPUT "FALSO
END

TO VAL1? :COM
   IF MEMBER? FIRST :COM [INV O E S N GUARDA INIZIO]
   OUTPUT "TRUE
   OUTPUT "FALSE
END

TO VAL2? :COM
   IF ALLOF VERBOVAL? FIRST :COM NOMEVAL? LAST :COM
   OUTPUT "TRUE
   OUTPUT "FALSE
END

TO NOMEVAL? :NOME
   IF MEMBER? :NOME [SPADA SCRIGNO SCETTRO ANELLO
      SERPENTE] OUTPUT "TRUE
   OUTPUT "FALSE
END

TO VERBOVAL? :VERBO
   IF MEMBER? :VERBO [PRENDERE ELIMINARE ESAMINARE
      UCCIDERE STROFINARE APRIRE] OUTPUT "TRUE
   OUTPUT "FALSE
END
```

Il programma

Innanzi tutto si devono inserire tutte le procedure descritte a pag. 1255. Digitare INIZIO per iniziare o rilanciare il gioco.

```
TO INIZIO
   MAKE "QUI1
   MAKE "INVENTARIO []
   CREA.LUOGHI
   ASSEGNA.VARIABILI
   GUARDA
END
```

CREA.LUOGHI predispone gli ambienti secondo la mappa dei sotterranei.

```
TO CREA.LUOGHI
   MAKE "LUOGO.1 [[[TI TROVI DI FRONTE ALL'ENTRATA] [DI
      UN SOTTERRANEO]] [] [[E2]] []]
   MAKE "LUOGO.2 [[[PERCORRI UN SOTTERRANEO UMIDO E
      BUIO]] [] [[S3] [E4] [O1]] []]
   MAKE "LUOGO.3 [[[PERCORRI UN SOTTERRANEO UMIDO E
      BUIO]] [] [[N2] [E5]] []]
   MAKE "LUOGO.4 [[[TI TROVI IN UNA GRANDE CAMERA
      SOTTERRANEA]] [] [[N6] [S5] [O2]] [ATTACCHI
      .SERPENTE]]
   MAKE "LUOGO.5 [[[PERCORRI UN SOTTERRANEO UMIDO E
      BUIO]] [SPADA] [[N4] [O3]] []]
   MAKE "LUOGO.6 [[[TI TROVI IN UNA STANZA SACRA DEL
      TEMPIO] [ALLA PARETE NORD] [C'È UN ALTARE]] []
      [[N7] [S4] [E8]] [PORTA]]
   MAKE "LUOGO.7 [[[SEI ACCANTO ALL'ALTARE] [DELLO
      SPLENDENTE ZOLTOTH] [SOPRA L'ALTARE C'È UNA
      SCRITTA:] ["NON SI AVVICINI ALCUN METALLO"]]
      [ANELLO] [[S6]] []]
   MAKE "LUOGO.8 [[[PERCORRI UN SOTTERRANEO UMIDO E
      BUIO]] [] [[S10] [E9] [O6]] [ATTACCHI.SERPENTE]]
   MAKE "LUOGO.9 [[[PERCORRI UN SOTTERRANEO UMIDO E
      BUIO]] [SCRIGNO] [[S11] [O8]] []]
   MAKE "LUOGO.10 [[[PERCORRI UN SOTTERRANEO UMIDO
      E BUIO]] [] [[N8] [E11]] []]
   MAKE "LUOGO.11 [[[TI TROVI NELLA SACRESTIA DEL] [IL
      SACERDOTE DI ZOLTOTH LO SPLENDENTE]]
      [SCETTRO] [[N9] [O10]] []]
END
```

Logo & Logo

Alcune versioni del MIT LOGO non possiedono i comandi EMPTY?, ITEM, COUNT o MEMBER?. Le procedure sostitutive sono state descritte in precedenza (vedere pagg. 1234 e 1255). Tutte le versioni LCSI usano:

EMPTYP al posto di EMPTY?
LISTP al posto di LIST?
MEMBERP al posto di MEMBER?
TYPE al posto di PRINT1
AND al posto di ALLOF
OR al posto di ANYOF

La primitiva EQUALP verifica l'eguaglianza di due termini: può essere impiegata per confrontare liste e parole in sostituzione del segno " = " (usato solo da alcune versioni LSCI per confrontare liste).
Il formato LOGO LCSI dell'istruzione IF è il seguente:

IF EMPTY :CONTENUTO [PRINT [NIENTE DI PARTICOLARE]] [PRINT :CONTENUTO]]

La prima lista viene visualizzata se la condizione è vera, la seconda se la condizione è falsa.
Il LOGO Atari usa SE e RL al posto di SENTENCE e REQUEST; inoltre è privo del comando ITEM



Il Commodore 16

Anatomia di un microcomputer a diffusione europea: tecnologie a confronto

Destinato con tutta probabilità a sostituire il Vic-20, che possiede una minuscola memoria di 3583 byte, il Commodore 16 condivide col Plus/4 un ottimo dialetto BASIC che integra, con una serie di comandi grafici, di suono e di utilità varie, le istruzioni per la gestione dei dischetti presenti in macchine come il CBM 8296. La presenza delle sole costruzioni DO...WHILE e LOOP...UNTIL...EXIT gioca a favore della programmazione strutturata.
Contrariamente al Plus/4, che possiede una tastiera ed una custodia radicalmente diverse rispetto ai precedenti modelli Commodore, la linea del 16 ricorda molto il Commodore 64 ed il Vic-20. Nuovi sono invece i colori impiegati (grigio scuro per la custodia e grigio chiaro per i tasti) e la disposizione della tastiera. I tasti sono ben distanziati e comodi da usare, come nei modelli che lo hanno preceduto. Una interessante novità è l'introduzione del tasto HELP (il tasto di funzione F8) al quale il programmatore può ricorrere quando un programma si blocca per la presenza di un errore di sintassi (syntax error). Il tasto visualizza la linea contenente l'errore e fa lampeggiare la parte errata del codice. Nelle linee a più istruzioni i caratteri lampeggiano dal punto d'errore sino alla fine della linea (sarebbe stato meglio, però, evidenziare il solo comando errato, ad esempio: PRONT al posto di PRINT).

Aspetti negativi

Una delle novità più discutibili comune tanto al Commodore 16 quanto al Plus/4, è l'adozione di un hardware non compatibile con quello dei modelli precedenti. Di conseguenza, il software sviluppato per il Commodore 64 ed il Vic-20 non può funzionare sui nuovi apparecchi. Le porte per il joystick e per il registratore a cassetta sono infatti diverse dai connettori "D" a 9 piedini, universalmente accettati come uno standard per gli home computer. Fortunatamente, l'interfaccia per l'unità a dischetti e la presa per il monitor sono identiche a quelle del Commodore 64.
La modifica "negativa" che lascia più perplessi (soprattutto i programmatori!) è l'abolizione degli sprite: sebbene il Vic-20 ne fosse privo, gli utenti del Commodore 64 e di altre macchine concorrenti si erano talmente abituati ad usarli per creare immagini grafiche, che la loro eliminazione non ha motivi plausibili.
Quando l'apparecchio viene acceso, sullo schermo compare l'ormai familiare presentazione Commodore: le uniche differenze riguardano la nuova versione del BASIC (3.5) e le dimensioni della memoria disponibile (12277 byte). Il codice 3.5 sta ad indicare la quinta versione del dialetto BASIC adottato sugli apparecchi Commodore. Nella prima, ossia la 1.0, mancavano i comandi per la gestione del dischetto che, pur in un formato di difficile impiego, vennero introdotti nella versione 2, il dialetto BASIC del Commodore 64 e del Vic-20. Dal marzo del 1980 la versione 2 è stata sostituita con la 4, che offre un BASIC molto efficiente scritto per il Pet a 80 colonne, l'8032, l'8096 e il più recente 8296.
La versione 3.5, quasi identica al BASIC 4, possiede un maggior numero di comandi e funzioni, oltre 50 in più rispetto al Vic. Tra questi c'è tutta una serie di comandi di utilità per la codifica e il debugging dei programmi, istruzioni per la programmazione strutturata e comandi grafici e di suono.
Il comando diretto MONITOR richiama il *Tedmon*, un programma di controllo residente in memoria (che sui modelli PET viene abilitato con l'istruzione SYS 4). Questo programma usa comandi a singola lettera: C, ad esempio, confronta due aree della memoria e ne visualizza le differenze, mentre S "salva" un file su nastro o dischetto. Le routine di "kernal" o di sistema sono rimaste le stesse e vengono chiamate dal codice macchina che controlla le routine di input/output. Peraltro è stata introdotta, all'indirizzo $FF81, la funzione IOINIT



Il Commodore 16
Considerato dalla Commodore come successore del Vic-20, questo nuovo apparecchio possiede una memoria di 16 K ed un BASIC più avanzato. È compatibile col Plus/4 per quanto riguarda le interfacce ed il software, ma non coi precedenti modelli Commodore 64 e Vic-20

CHRIS STEVENS

che inizializza le periferiche di I/O (in particolare, le cartucce dei programmi).

La grafica

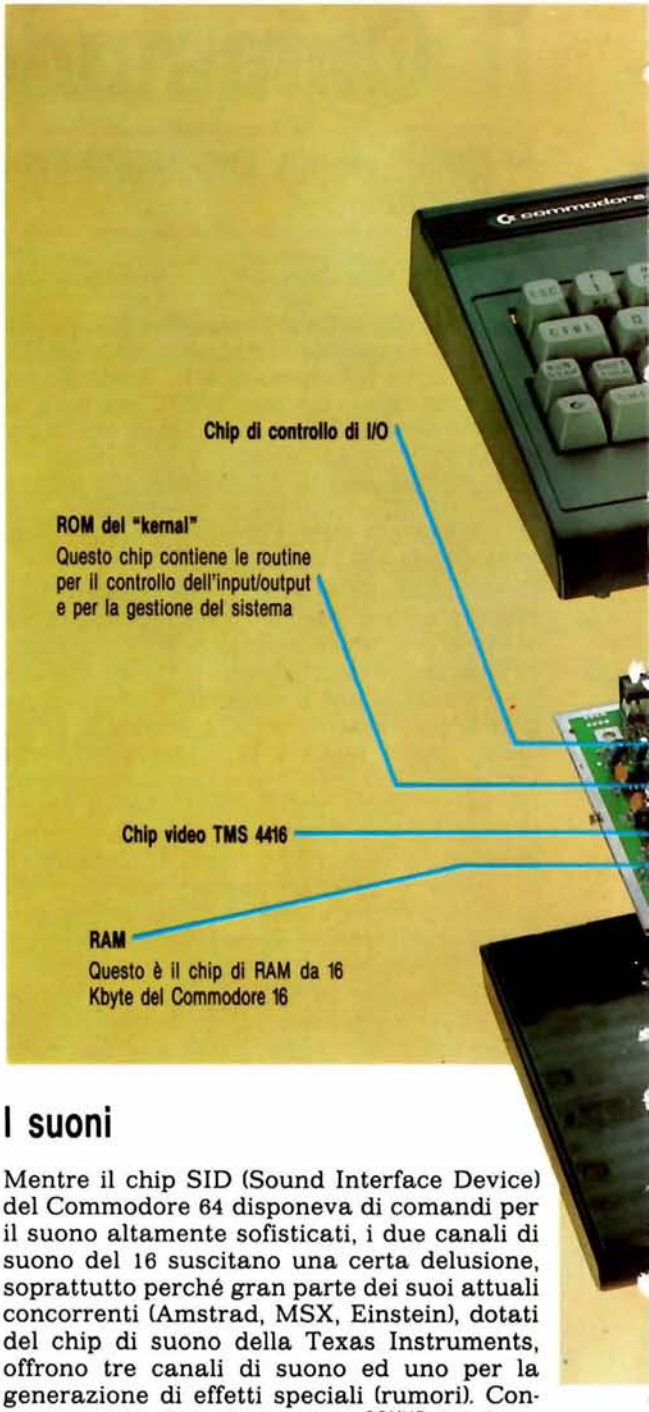
Lo schermo "bit mapped" ad alta risoluzione è composto da 320x160 pixel, mentre quello a più colori ha una risoluzione di 160x160 pixel. Rispetto alle istruzioni PEEK e POKE del Commodore 64, il comando di modo GRAPHIC facilita l'uso della grafica, ma suddivide lo schermo in due sezioni, limitando l'area testuale alle ultime cinque righe del video. Fortunatamente il comando CHAR consente di visualizzare i testi in qualsiasi punto dello schermo grafico, quindi l'istruzione:

```
CHAR 1,0,0, "QUESTA È LA PRIMA LINEA DELLO SCHERMO"
```

visualizza la scritta sulla linea più alta dello schermo grafico. Il numero 1 si riferisce al colore selezionato mentre i due zero indicano il numero di linea e di colonna. La scritta può essere visualizzata in "video inverso" aggiungendo al comando ",1", (",0" consente di tornare al modo normale). Quando ci si trova in uno dei modi grafici, qualsiasi errore di sintassi (syntax error) riporta il sistema a GRAPHIC 0, ossia al normale schermo testuale.
L'istruzione DRAW è, in un certo senso, un compromesso tra il tracciamento di linee diritte, tipico dell'Amstrad, ed il comando DRAW dello standard MSX (simile a quello del LOGO). Ad esempio, la seguente sequenza disegna un quadrato:

```
DRAW,10,10 TO 10,60 TO 60,60 TO 60,10 TO 10,10
```

In questo caso manca il parametro per il colore quindi la figura viene tracciata nell'ultimo colore selezionato (che può essere modificato inserendo un diverso codice di colore). Il comando BOX consente di tracciare rettangoli indicando la posizione sullo schermo dei quattro angoli. La presenza di un parametro di colorazione (fill) permette di colorare l'interno della figura.
A seconda dei parametri specificati, il comando CIRCLE (oltre a disegnare circonferenze) traccia ellissi, ottagoni, rombi e triangoli. Le figure non curve vengono realizzate indicando angoli di 120 gradi per i triangoli, di 90 per i rombi e di 45 per gli ottagoni: in mancanza di altre specifiche, il valore di "default" è 2 gradi. PAINT colora le figure create con lo stesso colore impiegato per i contorni o con un diverso colore di sfondo. Le figure possono essere salvate o richiamate dal dischetto con i rispettivi comandi SSHAPE e GSHAPE.
I colori vengono specificati, in BASIC, selezionando uno dei 16 valori per le aree di sfondo, di primo piano, di margine, e per gli schermi multicolor 1 e 2. È anche previsto un parametro facoltativo di brillantezza, con valori da 0 a 7 (il parametro di default è 7). In tutti i comandi grafici il parametro per il colore deve essere scelto facendo riferimento ad una delle cinque aree prima indicate.



I suoni

Mentre il chip SID (Sound Interface Device) del Commodore 64 disponeva di comandi per il suono altamente sofisticati, i due canali di suono del 16 suscitano una certa delusione, soprattutto perché gran parte dei suoi attuali concorrenti (Amstrad, MSX, Einstein), dotati del chip di suono della Texas Instruments, offrono tre canali di suono ed uno per la generazione di effetti speciali (rumori). Contrariamente al 64, il comando SOUND del Commodore 16 non prevede il deposito di parametri negli indirizzi compresi tra 54272 e 54296. Per conoscere la frequenza di una nota si può consultare una tabella riportata sul manuale ed usare i parametri indicati per ottenere la nota desiderata. Per esempio:

```
SOUND 1,770,60
```

genera un LA sul canale 1, ad una frequenza di 440 Hz e per la durata di un secondo (ossia 60 sessantesimi di secondo).
La più bassa nota ottenibile è un LA (110 Hz) inferiore di due ottave al DO centrale, la più alta un SOL (1575 Hz) superiore di due ottave al DO centrale. La gamma musicale è dun-



CHRIS STEVENS

COMMODORE 16

DIMENSIONI

mm. 76 x 203 x 406

CPU

MOS 7501 (frequenza di funzionamento da 0,89 a 1,76 MHz)

MEMORIA

16 Kbyte di RAM (12 Kbyte per l'utente); 32 Kbyte di ROM (contenente il BASIC ed il sistema operativo)

VIDEO

Testuale: 25 righe di 40 colonne. Grafico: 320 x 160 pixel. Cinque modi: testuale, alta risoluzione, alta risoluzione con 5 linee di testo, multicolore, multicolore con 5 linee di testo, 15 colori per 8 livellli di brillantezza, più il nero (ossia 121 sfumature in tutto)

INTERFACCE

Porta seriale Commodore, slot per espansioni su cartuccia, porta per registratore (a 8 poli), 2 porte per joystick (a 8 poli), uscita per monitor (video composito e segnali di brillantezza, colore, audio ed RF, con selettore di sintonia a scatto). Presa d'alimentazione.

LINGUAGGI DISPONIBILI

Interprete BASIC versione 3.5 su ROM (75 comandi che includono numerose funzioni grafiche)

TASTIERA

Tipo macchina da scrivere, con 66 tasti di cui 7 per funzioni programmabili ed un tasto di HELP

PRO

La versione del BASIC è molto buona, i comandi per dischetti sono eccellenti e l'accesso al programma monitor per codice macchina è facile

CONTRO

Il computer non è compatibile con i modelli precedenti, quindi c'è una certa carenza di software. Non esistono "sprite" e i connettori di ingresso/uscita non sono compatibili con gli altri modelli

que di 4 ottave. Sono disponibili due canali musicali (1 e 2) oppure un canale musicale (1 o 2) ed uno di rumore bianco (3). Poiché i due canali sono combinati e non c'è modo di separarli, il segnale audio d'uscita è monofonico.

Il Commodore 16 è una macchina interessante, dotata di una buona versione BASIC e di eccellenti comandi grafici, ma le funzioni audio appaiono piuttosto primitive, anche se più semplici da programmare, rispetto a quelle del Vic-20.

Inoltre, chi intendesse passare al Commodore 16 da un precedente modello deve ricordare che i programmi posseduti possono non essere compatibili.

UNA MODIFICA ... SOSPIRATA

Finalmente la Commodore, stimolata dalle numerose critiche rivolte dai programmatori alle deficienze del BASIC su ROM, ha prodotto una versione paragonabile a quella degli home computer di maggior successo. Le novità riguardano i comandi grafici, di utilità e di accesso ai dischi, oltre a nuove strutture. Inspiegabilmente mancano le procedure definite dall'utente, strumenti oggi indispensabili in un BASIC quanto meno accettabile. I nuovi comandi, compresi quelli derivanti dal BASIC versione 4, sono:

Funzioni	Utilità	Strutture	Grafica	Dal BASIC 4
RGR	AUTO	DO	GRAPHIC	DIRECTORY
RGCL	TRON	WHILE	SCNCLR	DSAVE
RLUM	TROFF	LOOP	PAINT	DLOAD
JOY	HELP	UNTIL	CHAR	HEADER
RDOT	MONITOR	EXT	BOX	SCRATCH
INSTR	DELETE	ELSE	CIRCLE	COLLECT
DEC	RENUMBER	PUDEF	GSHAPE	COPY
HEX$	KEY	USING	SSHAPE	RENAME
SOUND	ERR$		DRAW	BACKUP
VOL	TRAP		LOCATE	
	RESUME		COLOR	

Digitaya: i messaggi

Presentare ordinatamente le descrizioni e le scritte sullo schermo

Digitaya e *Foresta Incantata* sono due giochi d'avventura di tipo testuale, ossia impiegano una serie di parole per descrivere ambienti e situazioni. Se i testi vengono visualizzati sullo schermo con istruzioni PRINT, può accadere che nelle frasi più lunghe una parola risulti "a cavallo" tra due righe e venga perciò suddivisa in modo poco corretto. Per eliminare questo inconveniente si potrebbe formattare a mano ogni linea di testo, in modo che la parola in questione non venga troncata, ma questa tecnica si dimostra lunga e noiosa. In alternativa possiamo progettare una routine che esegua automaticamente la formattazione delle linee. Le varie frasi da formattare vengono trasferite alla routine mediante una variabile stringa, che, dopo la formattazione, viene visualizzata sullo schermo.

Digitaya e *Foresta Incantata* usano una particolare routine di formattazione. Esaminiamola prima di passare al resto della programmazione. Il seguente listato fa parte del gioco *Foresta Incantata* (la versione per *Digitaya* è riportata più avanti: benché identica nel contenuto, la numerazione delle linee è leggermente diversa):

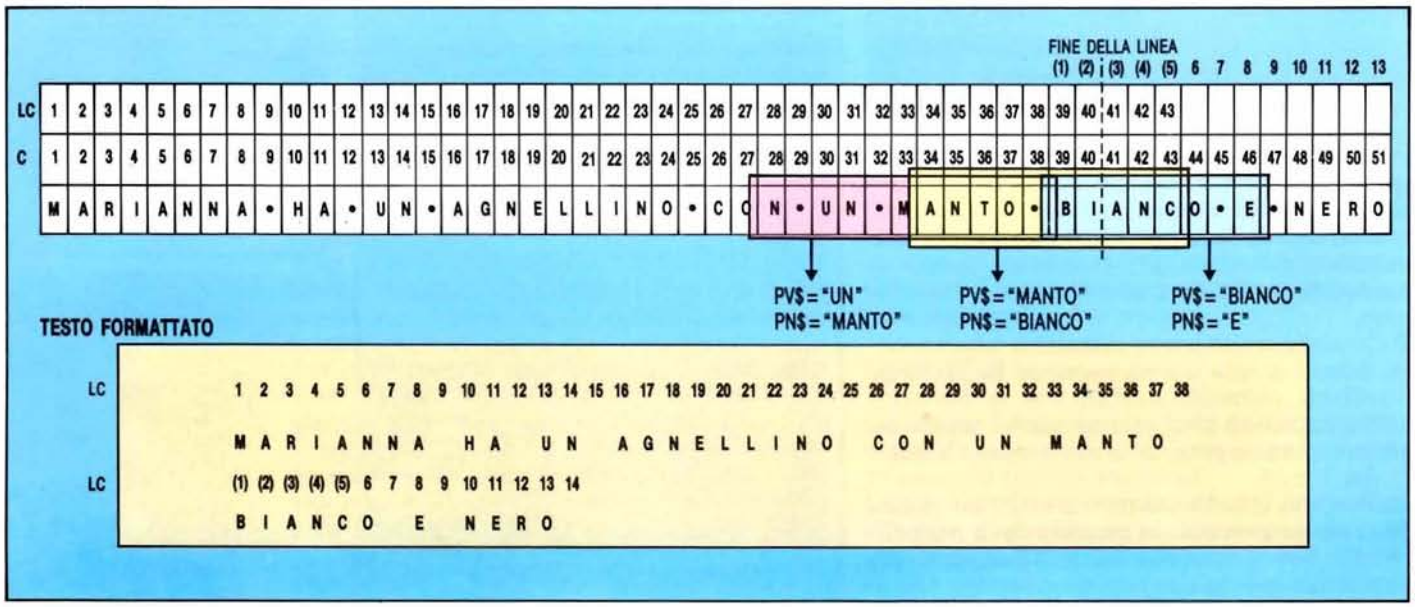
```
5500 REM •••• S/R FORMATTAZIONE LINEA ••••
5510 CL = 0:        REM CONTATORE DI CARATTERI/LINEA
5520 VC = 1:        REM CONTATORE INIZIO DI PAROLA
5530 PV$ = " ":     REM VARIABILE PER VISUALIZZAZIONE
5540 LL = 40:       REM N. CARATTERI SU RIGA DI SCHERMO
5550 SN$ = SN$ + " PIPPO " :REM VARIABILE DI COMODO
5560 PRINT
5570 FOR C = 1 TO LEN(SN$)
5580 CL = CL + 1
5590 IF MID$(SN$,C,1) = " " THEN GOSUB 5800
5600 NEXT C
5605 PRINT
5610 RETURN
5620 :
5800 REM •••• S/R VERIFICA DI FINE LINEA ••••
5810 PN$ = MID$(SN$,VC,C—VC + 1): REM PAROLA NUOVA
5820 IF CL < LL THEN PRINT PV$; : GOTO 5840
5830 PRINT PV$:CL = LEN(PN$)
5840 VC = C + 1:PV$ = PN$
5850 RETURN
```

La frase originale viene "passata" alla routine nella variabile SN$. Innanzi tutto vengono inizializzati alcuni contatori: quello dei caratteri per linea (CL) viene azzerato, mentre VC, che serve per individuare l'inizio di ciascuna parola, viene posto a 1. Alla stringa che conterrà la parola da visualizzare viene assegnato un valore nullo, mentre alla variabile LL si attribuisce il numero di caratteri visualizzabili su di una riga dello schermo. Infine alla frase originaria viene aggiunta una parola in più (PIPPO nel nostro caso), la cui utilità è chiarita in seguito.

Il ciclo FOR...NEXT tra le linee 5570 e 5600, scorre la frase alla ricerca di un carattere "spazio". Ad ogni passaggio nel ciclo, viene incrementata sia la variabile C (contatore di caratteri nella frase) sia VC (il puntatore all'inizio di ciascuna parola). Quando la funzione MID$ individua uno spazio, viene chiamata la subroutine alla linea 5800, che esegue alcune importanti operazioni. Per prima cosa, ricorrendo nuovamente ad una MID$, la subroutine estrae dalla frase la parola immediatamente precedente lo spazio individuato, depositandola in PN$ (Parola Nuova). Quindi controlla se l'inizio della nuova parola (CL) supera il massimo numero di caratteri visualizzabili su di una riga (LL). In base all'esito della verifica, viene deciso se visualizzare la variabile PV$ (Parola Vecchia) sulla stessa riga di schermo, facendola

Formattazione della frase

La routine di formattazione dello schermo, nei giochi "Digitaya" e "Foresta Incantata", evita il troncamento delle parole alla fine delle righe di schermo. Usando le variabili PV$ e PN$, la routine estrae la parola successiva a quella da stampare. Se la nuova parola supera la lunghezza massima della linea, viene omesso il punto e virgola che sopprime il ritorno carrello e la nuova parola viene stampata nella linea successiva



LIZ DIXON

seguire da un "punto e virgola" (linea 5820), oppure su di una nuova riga (linea 5830).
Dopo la visualizzazione, a PV$ viene assegnato il contenuto di PN$ appena estratto dalla frase. cioè, al primo passaggio, PV$ non contiene nulla (poiché PV$ = "" alla linea 5530). In sostanza, la subroutine visualizza sempre la parola "precedente" a quella estratta. Ora spieghiamo la funzione della parola "PIPPO" aggiunta alla frase originaria. Si noti che gli spazi che precedono e seguono PIPPO devono essere mantenuti: il primo separa la parola dalla precedente, il secondo inserisce uno spazio finale che viene individuato dalla routine.
L'intera routine è concepita in modo che venga sempre visualizzata la parola precedente anziché quella appena estratta. Ciò significa che, se non si aggiunge la parola in più, non verrebbe mai visualizzata l'ultima parola della frase (tant'è vero che PIPPO non compare mai sullo schermo).
Facciamo un esempio pratico, con la frase «Marianna ha un agnellino con un manto bianco e nero». Supponiamo che la lunghezza massima di una riga di schermo sia di 40 caratteri. Se la frase non venisse formattata, la parola "bianco" verrebbe spezzata in due e le lettere "anco" comparirebbero alla linea successiva. Poiché la routine di formattazione prende in esame due parole alla volta, i termini "un" e "manto", precedenti a "bianco", vengono assegnati rispettivamente a PV$ e PN$. Dopo aver verificato che il contatore CL non superi 40, la routine visualizza PV$ (seguito da un punto e virgola), poi trasferisce il contenuto di PN$ in PV$ e, ripreso l'esame del testo, trova la parola "bianco". Adesso il contatore CL è maggiore di 40 e perciò "bianco" oltrepassa la fine della riga di schermo; PV$ quindi (che ora contiene la parola "manto") viene stampato senza punto e virgola e al contatore viene assegnato il numero dei caratteri di "bianco". Quest'ultima parola viene poi inserita in PV$, per essere visualizzata, al successivo passaggio, nella riga di schermo sottostante.

Collaudo della routine

Per verificare il buon funzionamento della routine, proviamo a formattare e visualizzare il prologo del racconto. La variabile stringa SN$ viene usata per contenere un testo composto da non più di 248 caratteri, poi viene chiamata la subroutine di formattazione:

```
1000 REM **** S/R PROLOGO DEL RACCONTO ****
1010 SN$ = "BENVENUTO NELLA FORESTA INCANTATA"
1020 GOSUB 5500: REM FORMATTAZIONE
1030 PRINT
1040 SN$ = "TI SEI APPENA SVEGLIATO DA UN PROFONDO SONNO E"
1050 SN$ = SN$ + "TI TROVI IN UNA GRANDE FORESTA. IL SUOLO È"
1060 SN$ = SN$ + "MORBIDO E ASCIUTTO. NON RICORDI COME SEI GIUNTO"
1070 SN$ = SN$ + "IN QUESTO LUOGO, MA SAI CHE PER METTERTI IN"
1080 SN$ = SN$ + "SALVO DEVI RAGGIUNGERE IL VILLAGGIO CHE SI"
1090 SN$ = SN$ + "TROVA AL MARGINE DELLA FORESTA"
1100 GOSUB 5500 :REM FORMATTAZIONE
1110 PRINT
1120 SN$ = "TI GUARDI ATTORNO CERCANDO LA GIUSTA DIREZIONE"
1130 GOSUB 5500: REM FORMATTAZIONE
1140 PRINT:PRINT "PREMI UN QUALSIASI TASTO PER GIOCARE"
1150 GET A$: IF A$ = "" THEN 1150
1160 PRINTCHR$(147): REM CANCELLA LO SCHERMO
1170 RETURN
```

Infine chiamiamo la subroutine "Prologo del Racconto" con le linee:

```
205 GOSUB 1000: REM PROLOGO DEL RACCONTO
990 END
```

LISTATO DEL DIGITAYA

```
1110 GOSUB 1250: REM PROLOGO DEL RACCONTO
1270 END

1290 REM **** PROLOGO DEL RACCONTO ****
1300 SN$="BENVENUTO A 'DIGITAYA'"
1310 GOSUB 5880: REM FORMATTAZIONE
1320 PRINT
1330 SN$="IL TUO APPARECCHIO 'BORBOTTA' SOTTOVOCE.
     TI GUARDI ATTORNO."
1340 SN$=SN$+" DA  NORD A SUD CORRE UNA STRADA DI
     GRANDE COMUNICAZIONE."
1350 SN$=SN$+" LA TUA MISSIONE E' TROVARE LA
     MISTERIOSA DIGITAYA"
1360 SN$=SN$+"E PORTARLA IN SALVO ATTRAVERSO UNA
     DELLE QUATTRO PORTE."
1370 SN$=SN$+" ... MA QUALE ?"
1380 GOSUB 5880
1390 PRINT:PRINT "PREMI UN TASTO QUALSIASI
     PER GIOCARE"
1400 GET A$: IF A$=" " THEN 1400
1410 PRINTCHR$(147): REM CANCELLA LO SCHERMO
1420 RETURN


5880 REM **** S/R FORMATTAZIONE STAMPA ****
5890 CL=0: REM CONTATORE DI LINEA/CARATTERE
5900 VC=1: REM CONTEGGIO PRECEDENTE
5910 PV$=" ": REM PAROLA PRECEDENTE
5920 LL=40: REM LUNGHEZZA LINEA SCHERMO
5930 SN$=SN$+" PIPPO " :REM VARIABILE DI COMODO
5940 PRINT
5950 FOR C=1 TO LEN(SN$)
5960 CL=CL+1
5970 IF MID$(SN$,C,1) = " " THEN GOSUB 6020
5980 NEXT C
5990 PRINT
6000 RETURN
6010 :
6020 REM **** S/R VERIFICA DI FINE LINEA ****
6030 PN$=MID$(SN$,VC,C-VC+1): REM PAROLA NUOVA
6040 IF CL<LL THEN PRINT PV$; : GOTO 6060
6050 PRINT PV$:CL=LEN(PN$)
6060 VC = C + 1:PV$ = PN$
6070 RETURN
```

Basic & Basic

Spectrum:

Inserire le seguenti modifiche nella Routine di Formattazione del listato Digitaya:
Sostituire SN$ con S$, PV$ con O$ PN$ con N$
5920 LET LL = 32: REM LUNGHEZZA LINEA SCHERMO
5970 IF S$ (C TO C) = " " THEN GOSUB 6020
6030 LET N$ = S$ (VC TO C)
Sostituire SN$ con S$ in subroutine Prologo del Racconto
1400 IF INKEY$ = "" THEN 1400
1410 CLS
Per il listato di Foresta Incantata sostituire i nomi delle variabili stringa come suggerito sopra e modificare le linee:
5540 LET LL = 32: REM LUNGHEZZA LINEA SCHERMO
5590 IF S$ (C TO C) = " " THEN GOSUB 5800
5810 LET N$ = S$ (VC TO C)
e
1150 IF INKEY$ = "" THEN 1150
1160 CLS

Micro BBC:

Per quanto riguarda la subroutine per il Prologo del Racconto, apportare le seguenti modifiche a Digitaya:
1095 MODE 1
1400 A$ = GET$
1410 CLS
Per Foresta Incantata, cambiare:
1160 CLS

DATABASE

Prima scheda delle istruzioni per il 6502

CODICE MNEMONICO	DESCRIZIONE	R1 R2	IMPLIC.	IMMED.	PAGINA ZERO	PAG. Z X	PAG. Z Y	X INDIR.	Y INDIR.	INDIRIZZO RELATIVO	INDIRIZZO ASSOL.	X ASSOL.	Y ASSOL.	INDIRETTO	REGISTRO P						
															N	V	•	B	D	I	Z	C
ADC	SOMMA CON CARRY	A		69	65	75		61	71		6D	7D	79		X	X					X	X
AND	AND LOGICO	A		29	25	35		21	31		2D	3D	39		X						X	
ASL	SCORRIM. ARITMETICO A SIN.	A/M	0A(A)		06	16					0E	1E			X						X	X
BCC	DIRAMAZIONE SE CARRY A 0									90												
BCS	DIRAMAZIONE SE CARRY A 1									B0												
BEQ	DIRAMAZIONE SE UGUALE A 0									F0												
BIT	CONFRONTO DI BIT CON L'ACCUMULATORE				24						2C				M7	M6					X	
BMI	DIRAMAZIONE SE SEGNO NEGATIVO									30												
BNE	DIRAMAZIONE SE DIVERSO DA ZERO									D0												
BPL	DIRAMAZIONE SE SEGNO POSITIVO									10												
BRK	BREAK	S	00															X				
BVC	DIRAMAZIONE SE OVERFLOW A 0									50												
BVS	DIRAMAZIONE SE OVERFLOW A 1									70												
CLC	AZZERA IL CARRY		18																			0
CLD	CARRY DECIMALE		D8																0			
CLI	AZZERA MASCHERA DI INTERRUPT		58																	0		
CLV	AZZERA FLAG DI OVERFLOW		B8													0						
CMP	CONFRONTO CON ACCUMULATORE			C9	C5	D5		C1	D1		CD	DD	D9		X						X	X
CPX	CONFRONTO CON REGISTRO X			E0	E4						EC				X						X	X
CPY	CONFRONTO CON REGISTRO Y			C0	C4						CC				X						X	X
DEC	DECREMENTA LOCAZ. DI MEMORIA				C6	D6					CE	DE			X						X	
DEX	DECREMENTA REGISTRO X	X	CA												X						X	
DEY	DECREMENTA REGISTRO Y	Y	88												X						X	
EOR	OR ESCLUSIVO SULL'ACCUMULATORE	A		49	45	55		41	51		4D	5D	59		X						X	
INC	INCREMENTA LOCAZ. DI MEMORIA				E6	F6					EE	FE			X						X	
INX	INCREMENTA REGISTRO X	X	E8												X						X	
INY	INCREMENTA REGISTRO Y	Y	C8												X						X	
JMP	SALTO ALL'INDIRIZZO SPECIFICATO	PC									4C			6C								

Per concessione della Oric Products

Un punto d'arresto

Completiamo le routine per le breakpoint e passiamo a quella per i comandi

Per completare il modulo di breakpoint occorre definire due ultime subroutine: quella per rimuovere i punti d'interruzione inseriti e quella per reintrodurre gli opcode originali al posto di quelli per le SWI. La prima subroutine esaminata si chiama Rimuove-Breakpoint (dalla Tabella-Breakpoint).
La Tabella-Breakpoint (TABBP) può contenere un massimo di 16 punti d'interruzione. Per rimuoverne uno, dobbiamo conoscere il suo numero identificativo (compreso tra 0 e 15) e usarlo come offset all'interno della tabella. L'elemento della tabella viene eliminato facendo scorrere indietro di una posizione (ossia due byte) gli elementi che lo seguono e decrementando Numero-di-Breakpoint.

Rimuove-Breakpoint

```
DATI:
    Numero-di-Breakpoint: è un numero a 8 bit
    Numero-Breakpoint: è un contatore a 8 bit
    Tabella-Breakpoint: è una tabella di indirizzi a 16 bit
    Elemento-da-Rimuovere: è un offset a 8 bit (di valore compreso tra 1 e 16)
Procedura: Rimuove-Breakpoint
    Decrementa Numero-di-Breakpoint
    Se (IF) Elemento-da-Rimuovere < = Numero-di-Breakpoint (penultimo) allora (THEN)
        Per (FOR) Numero-Breakpoint = Elemento-da-Rimuovere fino a (TO) Numero-di-Breakpoint (penultimo)
        Copia Tabella-Breakpoint (Numero-Breakpoint + 1) in Tabella-Breakpoint (Numero-Breakpoint)
        Copia Valori-Rimossi (Numero-Breakpoint + 1) in Valori-Rimossi (Numero-Breakpoint)
        Fine ciclo (FOR)
    Fine (IF)
Fine della Procedura
```

Il parametro Elemento-da-Rimuovere può essere passato nel registro B. In seguito anche il contatore Numero-Breakpoint può essere trasferito in B, che verrà automaticamente riportato al corretto valore iniziale. Dopo averlo confrontato con Numero-di-Breakpoint, deve essere decrementato per formare l'offset all'interno della tabella a 8 bit dei Valori-Rimossi e, infine, viene fatto scorrere (moltiplicandolo così per due) per creare un offset all'interno della Tabella-Breakpoint a 16 bit. Possiamo depositare l'offset a 8 bit in B e quello a 16 bit in A. Gli indirizzi degli elementi contenuti nelle due tabelle vengono messi in X e Y, quindi possiamo ricorrere ad un incremento automatico per esaminare la tabella un elemento alla volta. L'elemento a 16 bit può essere fatto scorrere tramite il comando U, mentre l'elemento a 8 bit dovrà usare nuovamente A.

L'ultima procedura usata in questo modulo elimina fisicamente un punto d'interruzione, sostituendo l'opcode-SWI con l'opcode originale, conservato nella tabella dei Valori-Rimossi.

Elimina-Breakpoint

```
DATI:
    Numero-Breakpoint: è un offset a 8 bit all'interno della Tabella-Breakpoint
Procedura:
    Inserisce un valore in Valori-Rimossi (Numero-Breakpoint)
    Lo immagazzina in un indirizzo della Tabella-Breakpoint (Numero-Breakpoint)
```

Partiamo dal presupposto che il parametro venga caricato in B nel solito modo, come numero compreso tra 1 e 16. Tale valore va convertito per fungere da offset all'interno delle tabelle.
A questo punto possiamo dedicarci alla realizzazione di un modulo per la gestione dei comandi a lettera singola (vedere a pag. 1238). Alcuni comandi vengono eseguiti direttamente dalle routine già descritte ma, per una maggior completezza ed una più corretta strutturazione modulare, il modulo deve contenere anche le chiamate per questi comandi. Il comando B (per l'inserimento di una breakpoint) viene interamente eseguito dalla routine Inserisce-Breakpoint (BP01), quindi per questo comando il modulo conterrà solo l'istruzione di salto:

```
CMDB    BRA BP01
```

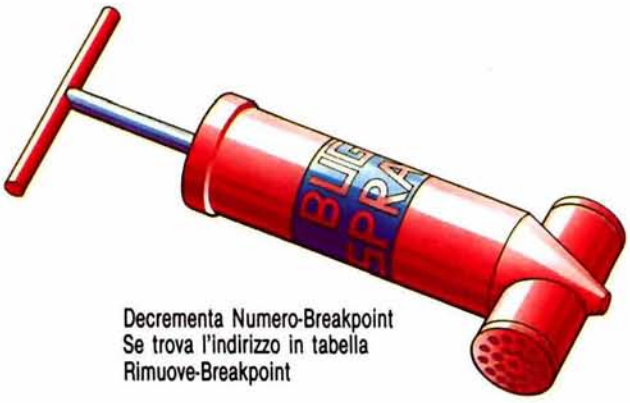
Il comando U, che rimuove una breakpoint, viene in gran parte eseguito da una routine scritta in precedenza (BP04). Questa viene completata dal modulo che riceve l'indirizzo della breakpoint da rimuovere e ne verifica la presenza all'interno della Tabella-Breakpoint. Se l'indirizzo è presente, l'offset viene trasferito alla routine BP02, altrimenti il comando viene ignorato.

Il comando U

```
DATI:
    Prompt: da visualizzare sullo schermo
    Indirizzo-Breakpoint: è il dato in ingresso
    Tabella-Breakpoint
    Numero-Breakpoint
    Procedura:
    Visualizza il prompt
    Riceve l'Indirizzo-Breakpoint
    Carica 16 in Numero-Breakpoint
    Finché (WHILE) Tabella-Breakpoint (Numero-Breakpoint) <> Indirizzo-Breakpoint e Numero-Breakpoint > 0
```



Il debugger inserisce una serie di breakpoint nel programma sottoposto a verifica. Innanzi tutto salva gli opcode da sostituire nella Tabella Valori Rimossi ed incrementa di 1 il Contatore delle Breakpoint, quindi inserisce al loro posto l'opcode di SWI. La Tabella Valori Rimossi funziona come uno stack, ma in realtà è una "pila" dalla quale è possibile recuperare qualsiasi valore (non solo l'ultimo come sullo stack). Quando una breakpoint viene rimossa, il corrispondente valore sostituito viene estratto dalla Tabella e copiato nella Memoria del Programma. Il byte della Tabella rimasto vuoto viene eliminato spostando tutti i byte della Tabella che lo seguono.



Decrementa Numero-Breakpoint
Se trova l'indirizzo in tabella
Rimuove-Breakpoint

Conservando Indirizzo-Breakpoint in Y, X resta disponibile per essere usato come un puntatore alla tabella.
Il comando D, che visualizza le breakpoint, viene eseguito dalla routine indicata dall'etichetta DISPBP (Visualizza-Breakpoint) chiamata dal modulo con l'istruzione di salto:

```
CMDD    BRA DISPBP
```

Il comando S, che lancia il programma sotto test, è più complicato perché durante la sua esecuzione vengono inserite le breakpoint. L'opcode per l'istruzione SWI va inserito in uno degli indirizzi della Tabella-Breakpoint mentre quello sostituito viene depositato in Valori-Rimossi. Eseguita questa operazione, il controllo deve tornare all'indirizzo iniziale del programma. Si ricordi inoltre che la breakpoint successiva è la numero 1. La procedura completa per il lancio del programma è la seguente:

Il comando S

```
DATI:
    Numero-di-Breakpoint: è un numero a 8 bit
    Tabella-breakpoint
    Valori-Rimossi
    Numero-Breakpoint: è un contatore a 8 bit
    Prossimo-Breakpoint: è un valore a 8 bit
    Opcode-SWI: è un valore a 8 bit
    Indirizzo-iniziale: è l'indirizzo di partenza del programma
        sottoposto a debugging
Procedura:
    Copia in Numero-Breakpoint il valore contenuto in Numero-
        di- Breakpoint
    Finché (WHILE) Numero-Breakpoint > 0
    Prepara-Breakpoint (Numero-Breakpoint)
        Decrementa Numero-Breakpoint
        Fine (WHILE)
    Seleziona ad 1 Breakpoint-Succ
    Salta a Indirizzo-Iniziale
```

Questa procedura usa la routine Prepara-Breakpoint, scritta in precedenza, che richiede il caricamento in A di Numero-Breakpoint (diminuito di 1, in modo da poter essere usato come un offset all'interno delle tabelle). Per comodità decrementiamo A prima della chiamata a Prepara-Breakpoint. Il listato completo della routine viene dato più avanti.
La parte finale di questa routine richiede una breve spiegazione. Quando viene eseguito il programma da verificare, le informazioni appoggiate sullo stack non sono più necessarie, quindi ci si deve accertare che questo sia vuoto quando il controllo viene trasferito al programma. Possiamo quindi cancellare lo stack nel modulo principale ma, se la routine viene chiamata con un'istruzione BRA (come accade per gli altri comandi), l'indirizzo di ritorno deve essere appoggiato sullo stack. Se non prevediamo la sua cancellazione, lo stack continuerà a crescere soprattutto durante quegli interventi che richiedono numerosi rilanci del programma. Per eliminare questo problema la routine cancella l'indirizzo di ritorno quando il controllo viene rinviato al programma e lo sostituisce con l'indirizzo di partenza. Quindi RST estrae dallo stack l'indirizzo di ritorno (ossia, in questo caso, l'indirizzo iniziale) e trasferisce il controllo azzerando contemporaneamente lo stack.
Il comando M consente di visualizzare e di modificare il contenuto delle locazioni di memoria. Prima di tutto occorre immettere l'indirizzo della locazione, visualizzandolo assieme al contenuto della locazione stessa. Poi viene data la possibilità di sostituire il valore originale con un nuovo numero esadecimale a due cifre oppure di battere semplicemente un RETURN, senza apportare alcuna modifica. In entrambi i casi il programma passa alla locazione di memoria successiva. Per interrompere la procedura basta digitare un punto. Il comando M sfrutta la routine GETHX2, che prevede questi tre tipi di immissioni.

Il comando M

```
DATI:
    Locazione-Corrente: è l'indirizzo a 16 bit della locazione
        esaminata
    Valore-Corrente: è il contenuto a 8 bit della Locazione-
        Corrente
    Nuovo-Valore: per la Locazione-Corrente. È un numero a 8 bit
Procedura:
    Riceve la Locazione-Corrente
    Ripete (REPEAT)
        Visualizza il Valore-Corrente
        Riceve il Nuovo-Valore
        Se (IF) Nuovo-Valore non è un punto allora (THEN)
            Se (IF) Nuovo-Valore non è Return allora (THEN)
                Registra Nuovo-Valore in Locazione-Corrente
            Fine (IF)
            Incrementa Locazione-Corrente
            Visualizza Locazione-Corrente
        Fine (IF)
    Finché (UNTIL) Locazione-Corrente è un punto
```

In questa routine di comando, Locazione-Corrente viene caricato in X, mentre il registro B viene usato sia per Valore-Corrente sia per Nuovo-Valore. A viene adoperato come un flag per indicare quale scelta (numero esadecimale, Return o punto) è stata fatta.
A questo punto al programma di debugging mancano le routine per gli ultimi tre comandi (G, R e Q), il meccanismo di interruzione che essi adottano e, infine, la progettazione del modulo principale del programma.

ROUTINE "RIMUOVI-BREAKPOINT"

```
BP04    PSHS    A,B,X,Y,U       Salva i registri usati
        DEC     NUMBP,PCR       Decrementa Numero-di-Breakpoint
IF02    CMPB    NUMBP,PCR       Se Elemento-da-Rimuovere <
        BGT     ENDF02          Numero-di-Breakpoint
        DECB                    Converte B in un offset
        TFR     B,A             Copia B in A
        LSLA                    Converte A in un offset
        LDX     BPTAB,PCR       Indirizzo di base della Tabella-Breakpoint
        LEAX    A,X             Tabella-Breakpoint (Numero-Breakpoint)
        LDY     REMTAB,PCR      Indirizzo di base Valori-Rimossi
        LEAY    B,Y             Valori-Rimossi (Numero-Breakpoint)
FOR00   LDU     2,X             Accetta l'elemento della Tabella-Breakpoint da spostare
        STU     ,X++            Lo sposta indietro di una posizione
        LDA     1,Y             Accetta l'elemento di Valori-Rimossi da spostare
        STA     ,Y+             Lo sposta indietro di una posizione
        INCB
        CMPB    NUMBP,PCR       L'ultimo?
        BLT     FOR00           Il successivo
ENDF02  PULS    A,B,X,Y,U,PC    Riporta al valore iniziale e restituisce il controllo
```

ROUTINE ELIMINA-BREAKPOINT

```
PB05    PSHS    A,B,X
        DECB                    Converte in un'offset per Valori-Rimossi
        LDX     REMTAB,PCR      Indirizzo di base Valori-Rimossi
        LDA     B,X             Accetta il valore da spostare
        LSLB                    Converte in un offset per Tabella-Breakpoint
        LDX     BPTAB,PCR       Indirizzo di base della Tabella-Breakpoint
        STA     [B,X]           Immagazzina all'indirizzo in Tabella
        PULS    A,B,X,PC        Riporta al valore iniziale e restituisce il controllo
```

COMANDO U

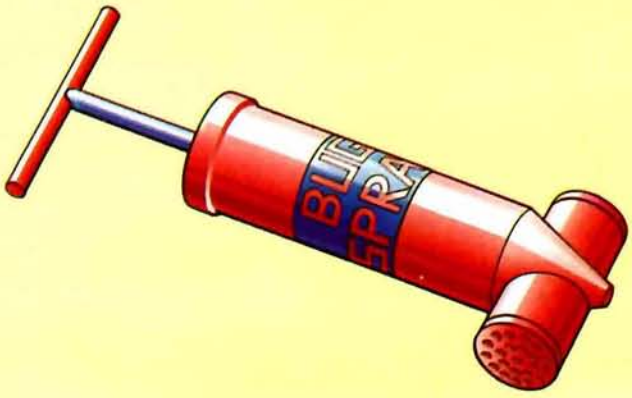
```
PROMPT  FCB     '>
CMDU    PSHS    A,B,X,Y         Salva i registri usati
        LDA     PROMPT,PCR
        BSR     OUTCH           Visualizza il simbolo di prompt
        BSR     GETADD          Accetta l'indirizzo
        TFR     D,Y             Carica Indirizzo-Breakpoint in Y
        LDB     MAXBP,PCR       Num. massimo di Breakpoint (16)
        LDX     PBTAB,PCR       Indirizzo di base della Tabella-Breakpoint
        TFR     B,A
        LSLA                    È un offset per la fine della Tabella-Breakpoint
        LEAX    A,X             X adesso punta oltre la fine della Tabella
        TSTB                    Seleziona i flag sui valori contenuti in B
WHIL02  BLE     ENDW02          Finché B>0
        CMPY    ,--X            (Si ricordi che prima X viene decrementato)
        BEQ     ENDW02          e Indirizzo-Breakpoint non è trovato nella tabella
        DECB                    Decrementa Numero-Breakpoint
        BRA     WHIL02
ENDW02  TSTB                    Indirizzo-Breakpoint trovato se B>0
IF03    BLE     ENDF            Se trovato, allora
        BSR     BP04            Rimuove-Breakpoint
ENDF03  PULS    A,B,X,Y
```

COMANDO S

```
START   RMB     2               Indirizzo-Iniziale
CMDS    LDA     NUMBP,PCR       Copia Numero-di-Breakpoint in Numero-Breakpoint
WHIL03  TSTA                    Controlla il valore di Numero-Breakpoint
        BLE     ENDW03          Finché Numero-Breakpoint>0
        DECA                    Decrementa Numero-Breakpoint
        BSR     BP02            Prepara-Breakpoint
        BRA     WHIL03          Breakpoint-Succ
ENDW03  LDA     #1              Pone Breakpoint-Succ a 1
        STA     NEXTBP,PCR
        STD     1,S
        RTS
```

COMANDO M

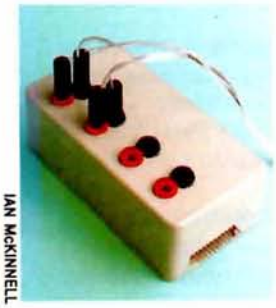
```
PROMPT  FCB     '>
SPACE   FCB     32              Cod. ASCII per "spazio"
CMDM    PSHS    A,B,X           Salva i registri usati
        LDA     PROMPT,PCR
        BSR     OUTCH           Visualizza il simbolo di prompt
        BSR     GETADD          Accetta Locazione-Corrente
        TFR     D,X             La carica in X
REPT01  LDB     ,X              Accetta Valore-Corrente
        BSR     PUTHEX          Visualizza Valore-Corrente
        LDA     SPACE,PCR
        BSR     OUTCH           Visualizza uno spazio
        BSR     GETVAL          Accetta Nuovo-Valore
IF03    TSTA                    Se Nuovo-Valore non è un punto
        BLT     UNTL01
        BGT     ENDF03          Se non è un Return
        STB     ,X              Registra Nuovo-valore in Locazione-Corrente
ENDF03  LEAX    1,X             Incrementa Locazione-Corrente
        TFR     X,D             Visualizza Locazione-Corrente
        BSR     DSPADD
        BRA     REPT
UNTL01  PULS    A,B,X,PC
```



Tiriamo le somme

Un sintetico quadro degli apparecchi di controllo fin qui realizzati

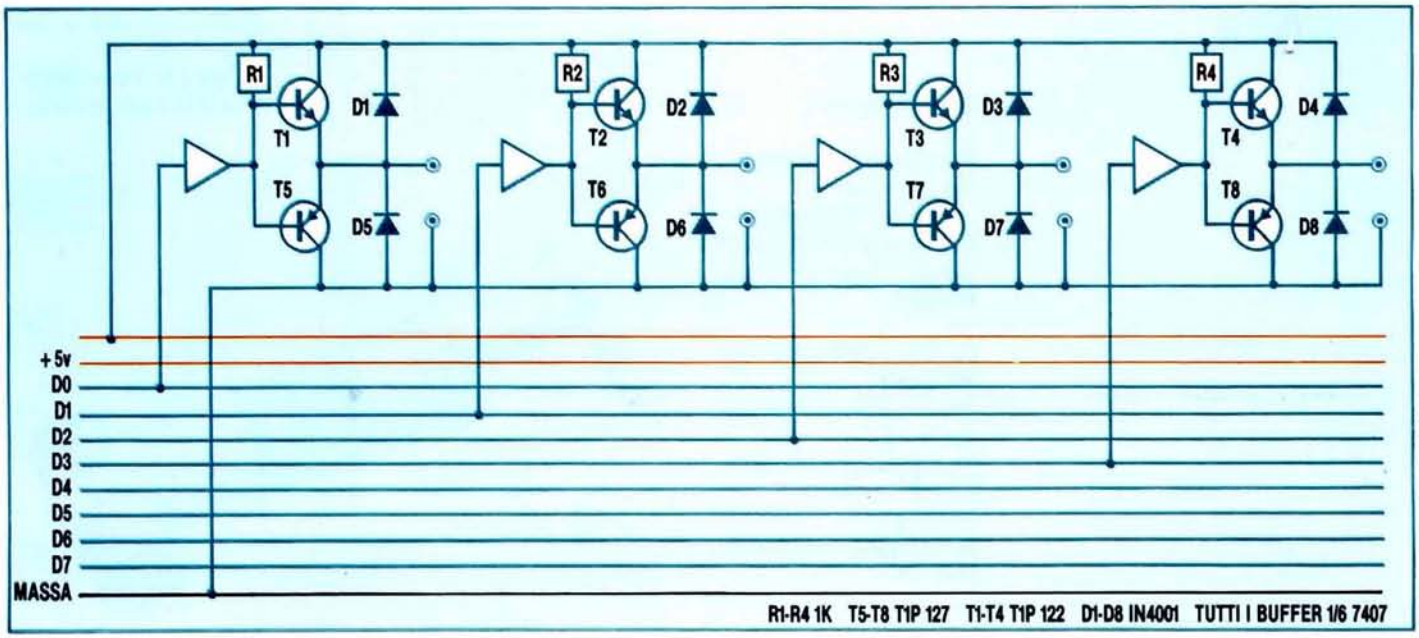
Micro BBC e Commodore 64 possiedono caratteristiche di I/O simili, che consentono loro di comunicare con l'esterno tramite una porta utente; questa possiede otto piedini per i dati ed uno per la massa. Ciascuno degli otto piedini è direttamente collegato ad uno degli otto bit di una particolare locazione di memoria, detta "registro per i dati" (indicato con REGDAT). Una seconda locazione, chiamata registro di direzione dei dati (indicato con DDR), controlla la direzione del flusso delle informazioni che attraversa la porta. Se selezioniamo un piedino per l'uscita (bit DDR = 1), ogni volta che il corrispondente bit dei dati viene posto a uno, il piedino corrispondente riceve una tensione di +5 volt; se invece il bit dati viene posto a zero, il piedino non riceve alcuna tensione. La corrente fornita dai piedini della porta utente non è in grado di alimentare direttamente un apparecchio esterno, tuttavia può essere usata per far scattare un sistema a relais, che a sua volta controlla tensioni di maggiore entità.
Quando un piedino viene posto in ingresso (bit DDR = 0), il funzionamento è notevolmente diverso. In questo caso il bit corrispondente del registro dati viene selezionato a uno e passa a zero soltanto se il piedino viene collegato a massa. Questa particolarità può essere sfruttata per seguire lo svolgimento di eventi che si verificano all'esterno del computer, collegando un interruttore al piedino dei dati da un lato e alla massa della porta utente dall'altro. Quando si aziona l'interruttore, il piedino collegato alla massa ed il corrispondente bit del registro dati passano da una condizione "alta" ad una "bassa" (ossia da 1 a 0). Questo mutamento nel registro dei dati può essere facilmente individuato dal software che sorveglia lo "stato" del registro dati. Un simile sistema permette quindi di modificare, dall'esterno, il flusso di un programma di controllo.
Poiché la massa e le otto linee per i dati devono essere in qualche modo collegate alle periferiche attraverso la porta utente, l'intero sistema si basa su un bus a 9 linee in comune. Ciascuna periferica viene poi connessa alle linee, adatte alla specifica applicazione, tramite un connettore "minicon" a 12 canali. Innestando un connettore "maschio" all'ingresso ed uno "femmina" all'uscita di ogni periferica, possiamo collegare le apparecchiature con una configurazione "a margherita", combinandole a piacere.
Vediamo infine un panorama riassuntivo dei vari moduli del sistema e dei loro circuiti. Per i particolari costruttivi e le liste dei componenti si faccia riferimento alle descrizioni originali.



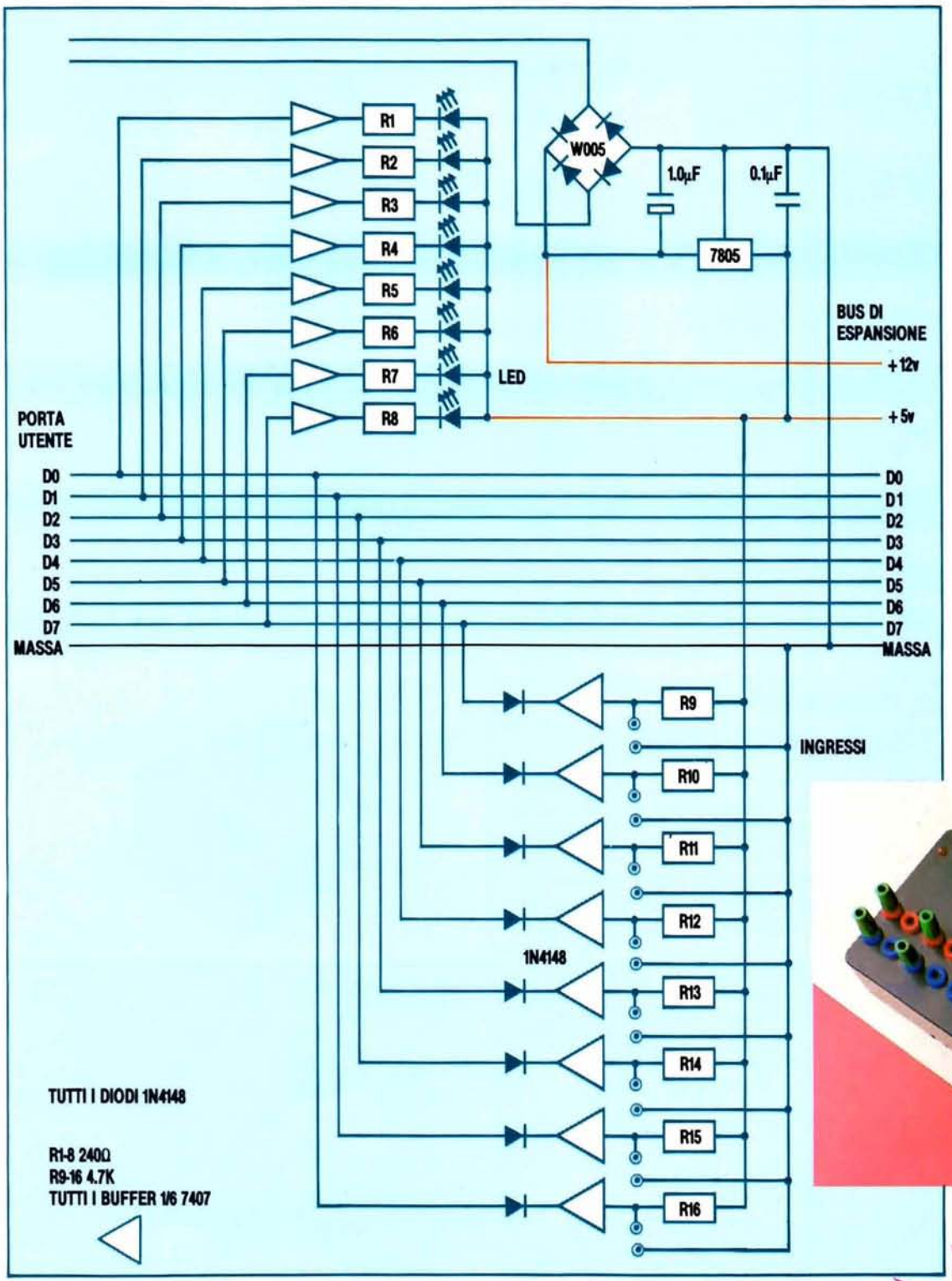
IAN McKINNELL

IL BOX DI USCITA

Il box di uscita a bassa tensione viene collegato al sistema tramite un connettore minicon maschio a 12 canali, da innestare alla presa del buffer box. La corrente fornita da uno dei piedini per i dati è di pochi milliampere, insufficiente per alimentare un apparecchio come un motore elettrico, ma capace di pilotare un transistor. Le linee per i dati numerate da 0 a 3 vengono usate dal box a bassa tensione. Se una di tali linee viene posta a 1, la tensione proveniente dal trasformatore viene commutata da un transistor e diretta alla corrispondente presa rossa del box di uscita. Questa tensione può alimentare contemporaneamente quattro periferiche. A seconda del trasformatore impiegato, ciascuna linea può condurre fino ad un massimo di un ampere (vedere a pag. 1054)

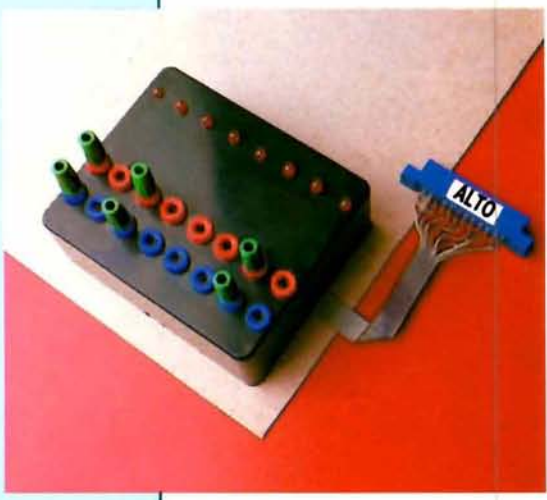


Circuiti e diagrammi di LIZ DIXON



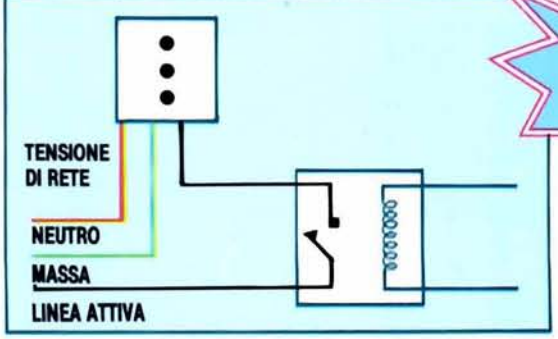
IL BUFFER BOX

Il buffer box è il primo e più importante modulo collegato alla porta utente. Protegge il chip di I/O del computer da eventuali correnti troppo elevate, provenienti dai piedini per i dati, o da una tensione in ingresso. Inoltre è in grado di ricevere e regolare correnti continue o alternate, comprese tra 5 e 21 volt, generate da un trasformatore. Queste raggiungono il bus attraverso una coppia di linee accessorie e vengono utilizzate da altre apparecchiature del sistema. Quando il buffer box è collegato, le tensioni in ingresso possono essere inviate alla porta utente: le otto prese rosse corrispondono alle otto linee per i dati, quelle nere forniscono una massa indipendente per ogni linea. Gli otto LED rossi inseriti sul coperchio del box consentono una "lettura" esterna dello stato di ciascuna linea dati: un LED si accende quando la linea corrispondente viene posta a 0 (vedere pagg. 1003 e 1026)



IL RELAIS DI RETE

Il relais di rete permette di usare la tensione proveniente dal trasformatore per controllare una corrente di rete. Questo dispositivo viene collegato alla rete d'alimentazione e ad una delle quattro linee del box di uscita. Ponendo a 1 un bit del registro per i dati, il trasformatore alimenta la corrispondente presa di uscita, che a sua volta permette alla corrente di raggiungere la presa di rete a tre contatti. È quindi possibile controllare varie apparecchiature elettriche tramite computer (vedere a pag. 1126)



ATTENZIONE!

Questo strumento è alimentato da una corrente di rete, quindi deve essere usato con grande prudenza.

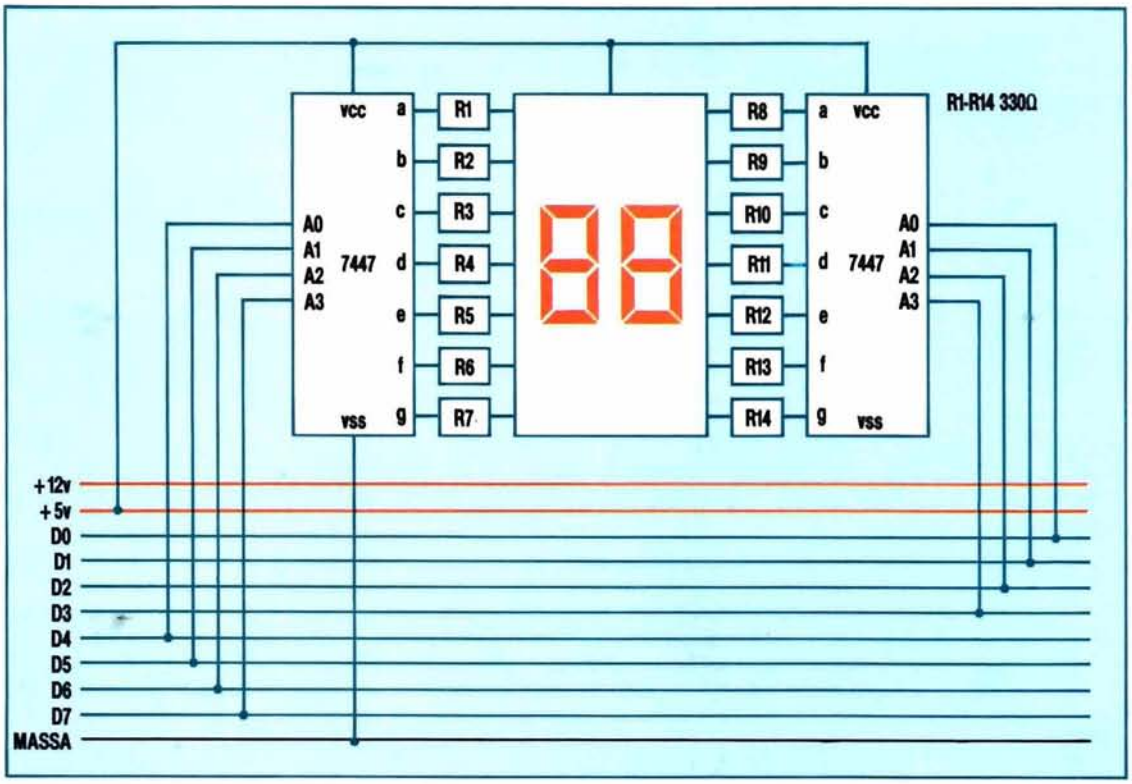
- **Staccare il cavo d'alimentazione prima di qualsiasi intervento.**
- **Seguire alla lettera le istruzioni e gli avvertimenti suggeriti per la realizzazione.**

IL DISPLAY A SETTE ELEMENTI

Un display a sette elementi necessita di quattro ingressi logici per visualizzare i numeri esadecimali. Per controllare due display di questo tipo occorrono dunque tutte e otto le linee disponibili sul bus d'espansione. Lo strumento usa una coppia di cifre esadecimali per visualizzare il contenuto del registro dati collegato alla porta utente e può essere collegato direttamente al buffer box oppure alla presa femmina minicon del box di uscita (vedere pag. 1165)



KEVIN JONES

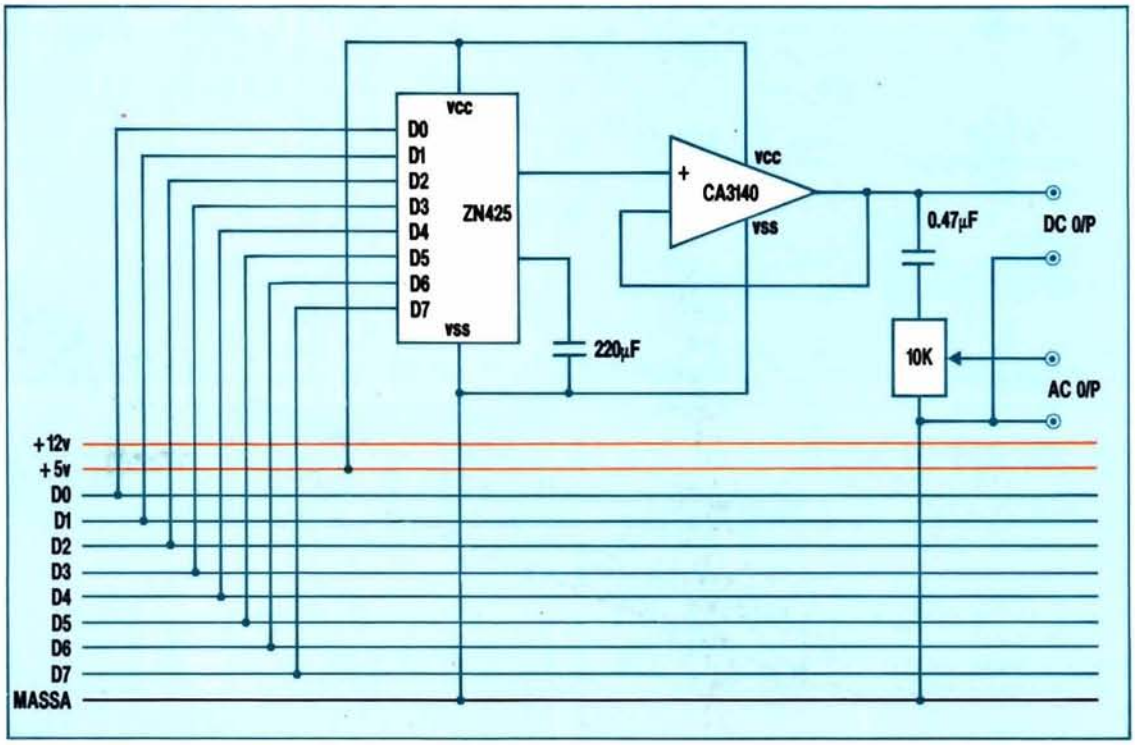


IL CONVERTITORE DIGITALE/ANALOGICO

Questo strumento trasforma in una tensione i valori binari (da 0 a 255) contenuti nel registro dati. Il massimo carico consentito è di soli 600 milliWatt, troppo debole per accendere una lampadina o per azionare un motore. Il segnale può essere amplificato per applicazioni di questo tipo. In alternativa, il convertitore D/A può essere usato per generare suoni attraverso cuffia o amplificatore audio (vedere a pag. 1194)



IAN McKINNELL



Simulare un robot

Se non si possiede un robot, possiamo usare il computer per simularne il funzionamento



STEVE CROSS

Lo sviluppo tecnologico dei computer ha portato ad un crescente impiego dei programmi di simulazione: è possibile progettare "modelli" per computer capaci di imitare alla perfezione quanto accade nella realtà. Tra i primi simulatori realizzati su scala commerciale ci sono quelli di volo, alcuni dei quali permettono l'addestramento "a terra" dei piloti, ossia senza bisogno di salire su un vero aereo. La simulazione computerizzata ha come oggetto anche molte altre applicazioni, quali la previsione degli sviluppi di mercato, la progettazione ingegneristica e lo studio dei fenomeni fisici. In alcuni casi, i modelli consentono di eseguire esperimenti che, nella realtà, sarebbero troppo pericolosi da realizzarsi. Sapere cosa accade in una centrale nucleare quando si verifica una fuoriuscita del liquido di raffreddamento del reattore, ad esempio, è di capitale importanza, ma poiché è praticamente impossibile eseguire questo esperimento all'interno di una vera centrale, si ricorre ad una simulazione computerizzata; se il modello è sufficientemente dettagliato, si possono riprodurre con precisione addirittura tutti gli eventi che seguono la fuoriuscita del liquido.

Nel campo della robotica si ricorre spesso alla simulazione, in particolare durante le fasi di progetto dei robot. Ovviamente nulla vieta di procedere per tentativi reali e costruire un robot, verificarne il funzionamento ed apportare le necessarie modifiche, ma questa tecnica è notevolmente lenta e dispendiosa. Al contrario, un buon programma di simulazione permette di progettare un robot e di controllarne il funzionamento ancor prima di

Passo di danza

Quando più bracci robotizzati operano in vicinanza, occorre curare anche l'aspetto coreografico dei movimenti. È assolutamente necessario evitare eventuali interferenze tra i vari bracci. In questo ipotetico laboratorio, un braccio deve raccogliere e reggere un giocattolo, mentre un altro raccoglie e fissa sul giocattolo un piccolo tamburo. Terminato ciò il primo braccio depone il giocattolo nella scatola. Se il movimento dei due bracci è ben studiato, i nastri trasportatori possono essere posizionati in modo razionale ed occupare meno spazio di quello occorrente ad un uomo

acquistarne i componenti, evitando così di doverlo smontare e rimontare più volte se qualcosa non funziona a dovere.
Supponiamo di voler progettare un gruppo di robot adatti ad una catena di montaggio per automobili. Il software di controllo dovrà provvedere affinché i robot svolgano il loro compito in modo corretto. La fase di programmazione richiede un certo tempo, ma una qualunque interruzione, nel funzionamento della catena di montaggio, causa notevoli perdite di tempo e di denaro.
Per sviluppare i nuovi programmi si può ricorrere ad un duplicato sperimentale della linea di assemblaggio corredata da alcuni robot simili a quelli originali, ma questo procedimento è molto costoso e non consente una corretta analisi della disposizione reale ed ottimale dei robot. Ovviamente, ciascun robot deve operare senza ostacolare gli altri: anche un lieve urto tra due grossi robot industriali, capaci di sollevare considerevoli pesi, può causare ingenti danni ad entrambe le macchine. Ma anche l'errata programmazione di un robot, che si metta a saldare le serrature delle portiere, può creare irritanti conseguenze!
La soluzione meno costosa e, tutto sommato, più sicura è quella di un modello che simuli i movimenti dei robot all'interno della catena di montaggio e permetta di controllare le reciproche interazioni. Soltanto dopo aver verificato la completa affidabilità della simulazione, il programma può essere trasferito nei robot della linea d'assemblaggio reale.
Per meglio chiarire i principi della simulazione di un robot, esaminiamo il programma *Braccio Robot*, che simula un meccanismo del tipo "prendi e metti", dotato di due gradi di libertà. In questa simulazione, il braccio (privo di sensori) viene guidato da un operatore che controlla le articolazioni della spalla e del gomito, oltre al meccanismo di presa. L'obiettivo è quello di afferrare un oggetto e di posarlo in un altro punto del piano di lavoro. Può essere utile rivedere anche il programma presentato a pag. 1202, che permette ad un robot di trovare il percorso d'uscita in un labirinto.
Tale programma è in realtà una simulazione, che imita le azioni di un robot unicamente dotato di sensori di contatto e posto all'interno di un labirinto.
Se imbocca un vicolo cieco, il robot ripercorre all'indietro il tragitto fino all'ultima biforcazione, scegliendo una nuova direzione. Questo modello è molto semplice, ma mostra ugualmente come sia possibile usare un programma di simulazione: il robot obbedisce ad una serie di regole codificate dal software e, progressivamente, si costruisce una mappa dell'ambiente in cui si muove. Se il programma consentisse al robot di "osservare" preventivamente la configurazione del labirinto, la macchina potrebbe raggiungerne direttamente il centro senza commettere errori. Poiché il nostro programma non prevede questa possibilità, il robot è costretto a procedere per tentativi.
In modo analogo, il programma *Braccio Robot* simula il comportamento di un robot privo di sensori: crea un modello del robot e dell'ambiente circostante e riproduce le interazioni che legano i due modelli nel mondo reale. Quindi il braccio non può afferrare un oggetto se non viene posizionato in modo corretto né può raggiungerlo se questo si trova sotto il piano di lavoro. Inoltre, quando il robot lascia la presa, l'oggetto deve cadere verso il basso come se fosse realmente attratto dalla forza di gravità. Questi particolari possono apparire superflui, ma sono essenziali per ottenere una simulazione efficace.



Per concessione della FORD

Coordinazione di movimenti
Questa immagine, che mostra una catena di montaggio della Ford per la Sierra, offre un chiaro esempio di quanto sia indispensabile progettare con perfetto sincronismo i complessi movimenti di tutta una serie di robot

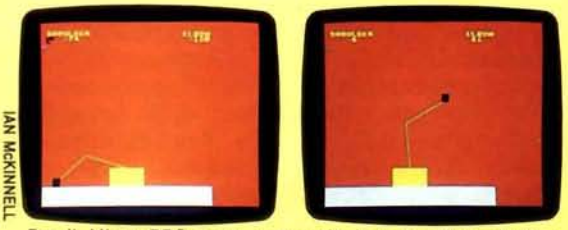
Particolari realistici

I programmi di simulazione sono soggetti a ben poche limitazioni; in effetti, più la simulazione è complessa e particolareggiata, maggiore è il suo fascino. Spesso la simulazione di un robot sullo schermo si dimostra più stimolante del robot reale, perché un modello consente di progettare macchine ed ambienti personalizzati e non condizionati dalle leggi del mondo fisico. Ma torniamo al nostro *Braccio Robot*; abbiamo visto che, quando il braccio lascia andare un oggetto, questo cade verso il basso. Per rendere il modello più realistico, possiamo modificare il programma in modo che la caduta dell'oggetto venga accelerata dalla forza di gravità oppure farlo rimbalzare quando urta il piano di lavoro. Le possibilità sono innumerevoli e limitate soltanto dalla fantasia o dalla memoria che si ha a disposizione.
Il progetto di simulazione condivide molti aspetti dello sviluppo di programmi di gioco. La principale differenza è che una simulazione deve rappresentare la realtà nel modo più accurato possibile. Spesso questo risultato è difficile da raggiungere, ma le simulazioni si dimostrano molto più appaganti di un banale gioco per computer.

PRENDI E METTI

Questo programma simula un braccio robot capace di estendersi per raggiungere un oggetto, sollevarlo e posarlo in una diversa collocazione. Il braccio è programmato per usare coordinate di rivoluzione con due gradi di libertà: un'articolazione è la "spalla" e una il "gomito". La spalla può ruotare di 180°, il gomito di 360°. Il programma è controllato dai seguenti tasti: S dirige i movimenti della spalla, E quelli del gomito. K e H indicano se lo snodo di base deve ruotare in senso orario o antiorario. Ad ogni "battuta" del tasto corrispondente, l'articolazione della spalla ruota di 6°, quella del gomito di 12°. U ordina al braccio di afferrare la sfera: questa manovra ha successo solo se il braccio è posizionato in modo corretto. F ordina al robot di posare la sfera.

```
   4 REM ********** SPECTRUM **********
   5 REM * SIMULAZIONE BRACCIO ROBOT *
   6 REM ********** SPECTRUM **********
  10 CLS: PRINT "BRACCIO ROBOT": PRINT: PRINT "I
     TASTI DI CONTROLLO PER IL BRACCIO ROBOT SONO:"
  15 PRINT "S - ROTAZIONE SPALLA": PRINT "E -
     ROTAZIONE GOMITO":PRINT "K - RUOTA LA BASE IN
     SENSO ORARIO"
  20 PRINT "H - RUOTA LA BASE IN SENSO ANTIORARIO":
     PRINT "U - AFFERRA L'OGGETTO": PRINT "F - POSA
     L'OGGETTO"
  25 PRINT AT 20,11; FLASH 1; "batti un tasto":
     PAUSE 0:RANDOMIZE
1000 GO SUB 9500: REM inizializza
1100 GO SUB 5000: REM immissione tasti di comando
1200 GO SUB 6000: REM caduta
1400 STOP
1500 REM * aggiorna coordinate x y snodo *
1550 LET ex=l1*SIN hd1: LET ey=l1*COS hd1
1560 LET hx=sx+ex: LET hy=sy+ey
1650 LET wx=l2*SIN hd2: LET wy=l2*COS hd2
1660 LET hx=hx+wx: LET hy=hy+wy
1690 RETURN
2000 REM * disegna il braccio robot *
2020 INK acol
2050 PLOT INVERSE cancella; sx,sy
2100 DRAW INVERSE cancella;  ex,  ey: DRAW INVERSE
     cancella; wx,wy:  IF blup THEN LET br=FN r(hy):
     LET bc=FN c(hx): GO SUB 2500
2490 RETURN
2500 REM * disegna l'oggetto *
2600 INK bcol
2650 PRINT AT br,bc; "■ "(cancella+1)
2740 RETURN
2750 REM * posa l'oggetto *
2800 LET cancella=1: GO SUB 2000: LET cancella=0
2820 LET k=INT (xh*RND):IF k>xs THEN IF k<=xs+wd
     THEN GO TO 2820
2850 LET br=FN r(y0+4):  LET bc=FN c(k):
     LET blup=0: GO SUB 2000: GO SUB 2500
2990 RETURN
3000 REM * rotazione *
3100 LET cancella=1: GO SUB 2000
3120 LET t1=dirn*sr*a1: LET t2=t1+dirn*er*a2
3150 LET hd1=hd1+t1: LET hd2=hd2+t2
3200 GO SUB 1500
3300 IF ABS hd1>p2 THEN LET ok=0
3320 LET pt=POINT (hx,hy)
3340 IF pt<>0 THEN LET ok=0:  IF br=FN r(hy) and
     bc=FN c(hx) THEN LET ok=2
3400 LET cancella=0: GO SUB 2000
3450 INK  bacol:  PRINT  AT  21,0;s$;AT  21,2;FN
     d(hd1);AT 21,26;FN d(hd2)
3490 RETURN
5000 REM * elabora i tasti di controllo *
5100 IF INKEY$<>"" THEN GO TO 5100
5120 FOR l=0 TO 1 STEP 0
5150 LET a$=INKEY$:  IF a$>="A" AND a$<="Z" THEN
     LET a$=CHR$ (CODE a$+32)
5200 IF a$="s" THEN  LET sr=1: LET er=0
5220 IF a$="e" THEN  LET er=1: LET sr=0
5250 IF a$="k" THEN  LET dirn=1: GO SUB 3000
5270 IF a$="h" THEN  LET dirn=-1: GO SUB 3000
5300 IF a$="u" THEN  IF ok=2 THEN LET blup=1
5320 IF a$="f" THEN  IF blup THEN GO SUB 2750
5400 IF NOT ok THEN LET l=2
5450 NEXT l
5490 RETURN
6000 REM * caduta *
6100 PRINT AT 8,12;  FLASH 1;  "!!crash!!":
     BEEP .5,-5: BEEP 1,-16: RETURN
9000 REM * disegna la base *
9050 PAPER pacol: CLS
9100 INK grcol
9120 FOR k=0 TO y0: PLOT 0,k: DRAW xh,0: NEXT k
9200 INK bacol: LET xs=(xh-wd)/2
9220 FOR k=y0+1 TO y0+ht
9240 PLOT xs,k: DRAW wd,0
9260 NEXT k
9300 INK bcol: GO SUB 2500: INK acol
9400 PRINT AT 20,1; "SPALLA"; AT 20,26; "GOMITO"
9490 RETURN
9500 REM * inizializza *
9550 DEF FN d(x)=INT (x*180/PI)
9560 DEF FN r(x)=21-INT (x/8)
9570 DEF FN c(x)=INT (x/8)
9600 DIM s$(32): LET x1=0: LET y1=0: LET xh=254:
     LET yh=174
9620 LET y0=23: LET wd=60: LET ht=23
9630 LET blup=0: LET bc=2: LET br=FN r(y0+4)
9640 LET grcol=3:  LET bacol=2:  LET acol=1: LET
     bcol=6: LET pacol=7
9650 LET sx=xh/2:  LET sy=y0+ht+2: LET l1=(yh-ht-
     y0-2)/2: IF l1>xh/4 THEN LET  l1=xh/4
9660 LET l2=l1: LET hx=0: LET hy=0
9670 LET p2=PI/2: LET a1=PI/32: LET a2=2*a1
```



IAN McKINNELL

Movimenti della spalla

Questo programma di simulazione consente di muovere un braccio robot su due dimensioni e di afferrare un oggetto. Quando il braccio posa l'oggetto, il programma lo riposiziona a caso sul piano. Le figure mostrano due diverse angolazioni assunte dal braccio durante l'esecuzione

Per il Micro BBC sono necessarie le seguenti aggiunte e modifiche:

```
   4 REM ************ BBC ************
   5 REM * SIMULAZIONE BRACCIO ROBOT *
   6 REM ************ BBC ************
   7 MODE 1:COLOUR 130:COLOUR 1:CLS
  25 PRINT TAB(15,20)"PREMI UN TASTO":A$=GET$
1000 GOSUB 9600
2020 CGOL 0,acol
2050 MOVE sx,sy
2100 PLOT cancella,ex,ey:PLOT cancella,wx,wy:IF
     blup THEN br=hy:bc=hx:GOSUB 2500
2190 RETURN
2200 REM ****** AFFERRA L'OGGETTO *****
2250 blup=1:cancella=3:GOSUB 2500
2300 cancella=1:GOSUB 2000
2600 GCOL 0,BCOL:MOVE bc,br
2650 PLOT 0,0,bsz:PLOT 80+cancella,bsz,0
2700 PLOT 0,0,-bsz:PLOT 80+cancella,-bsz,0
2800 cancella=3:GOSUB 2000:cancella=1
2820 k=INT(xh*RND(1)):IF k>=xs THEN IF k<=xs+wd
     THEN GOTO 2820
2850 br=y0+5:bc=k:blup=0:GOSUB 2000:GOSUB 2500
3100 cancella=3:GOSUB 2000
3340 IF pt<>pacol-128 THEN ok=0:IF pt=bcol THEN ok=2
3400 cancella=1:GOSUB 2000
3450 COLOUR bacol:PRINT TAB(0,3)s$;TAB(4,3); FNd(hd1);
     TAB(27,3);FNd(hd2)
5100 IF INKEY$(0)<>"" THEN GOTO 5100
5150 a$=INKEY$(0): IF a$>="A" AND a$<="Z" THENa$=CHR$
     (ASC(a$)+32)
5300 IF a$="u" AND ok=2 THEN GOSUB 2200
5400 IF ok=0 THEN l=2
6100 PRINT TAB (12,3) "!!CRASH!!": SOUND 1, -15, 48, 10:
     SOUND 1, -15, 4, 20: RETURN
9050 GCOL 0,pacol: COLOUR pacol: CLS
9100 GCOL 0,grcol
9120 FOR k=0 TO y0: MOVE 0,k: DRAW xh,k: NEXT k
9200 GCOL 0,bacol: xs=(xh-wd)/2
9240 MOVE xs,k: DRAW xs+wd,k
9300 MOVE bc,br: GOSUB 2500: COLOUR acol
9400 PRINT TAB (1,2) "SPALLA"; TAB (26,2) "GOMITO"
9600 s$="": x1=0: y1=0: xh=1000: yh=1000
9620 y0=100: wd=200: ht=100
9630 blup=0: bsz=wd/5: bc=40: br=y0+5
9640 grcol=3: bacol=2: acol=2: bcol=0: pacol=129
9650 sw=xh/2:sy=y0+ht+2:l1=(yh-ht-y0-2)/2:IF l1)
     xh/4 THEN l1=xh/4
9690 sr=1:er=0:dirn=1:cancella=1:ok=1
```

Il TK!Solver

Dai creatori di "VisiCalc", uno spreadsheet che elabora equazioni

I programmi di lavagna elettronica per microcomputer si dimostrano molto utili in numerose applicazioni di carattere matematico, facendo risparmiare tempo ed energie soprattutto a chi deve lavorare su tabulati di grandi dimensioni. Tuttavia, gli spreadsheet presentano limiti non trascurabili; la loro articolazione in linee e colonne, che si adatta perfettamente alla gestione della contabilità o alla creazione di altri modelli finanziari, spesso si dimostra poco funzionale (talvolta del tutto inadeguata) per applicazioni matematiche e scientifiche di più alto livello; inoltre possiedono una flessibilità assai limitata nel calcolo delle equazioni.

La Software Arts, la compagnia statunitense creatrice del *VisiCalc*, ha sviluppato *TK!Solver*, un programma che rinnova enormemente la struttura e le funzioni delle lavagne elettroniche. "TK!" sta per ToolKit mentre "Solver" (letteralmente: risolutore) si riferisce a quella parte del programma che esegue il calcolo vero e proprio delle equazioni. Oltre a suddividere lo schermo in modo diverso rispetto ad altri spreadsheet, il *TK!* possiede le seguenti caratteristiche innovative:

Soluzione di equazioni: mentre le comuni lavagne elettroniche risolvono esclusivamente equazioni a una sola variabile, il *TK!* può risolverle a più variabili (purché, ovviamente, la quantità di dati sia sufficiente).

Soluzioni per approssimazione: un dato omesso o non conosciuto ma necessario per risolvere un'equazione può essere sostituito con un valore ipotetico. Il *TK!* lo utilizza come dato di partenza e risolve l'equazione per successive approssimazioni.

Conversione di unità di misura: il *TK!* può convertire istantaneamente le unità di misura (ad esempio, da piedi a metri, oppure da dollari a sterline ecc.) ricorrendo ad una serie di tabelle di conversione.

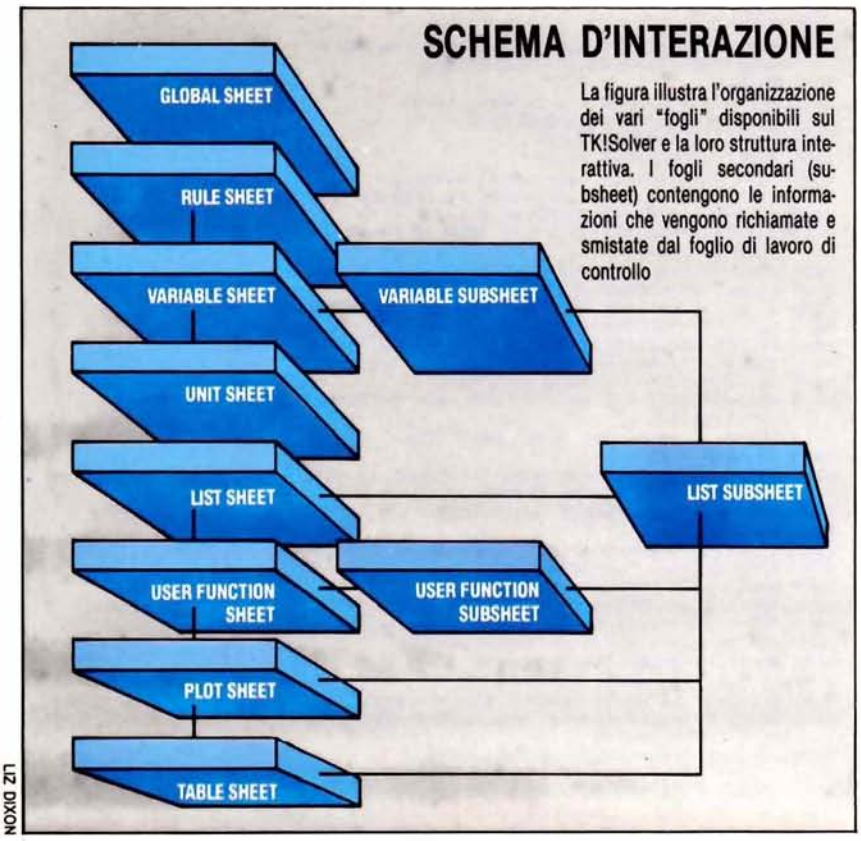
Funzioni matematiche: il TK! incorpora numerose funzioni di questo tipo.

Il *TK!Solver* usa tre diversi "fogli di lavoro" tra loro collegati, ciascuno con una funzione propria. Nelle colonne del foglio delle variabili (Variable sheet), oltre a tutti i nomi delle variabili definite, si trovano i valori immessi dall'utente e quelli calcolati dal computer, un'area riservata a indicazioni sull'unità di misura e un'area per immettere commenti relativi alle singole variabili. Il foglio di lavoro "Variable" viene visualizzato all'inizio del programma nella parte alta dello schermo; inoltre, ogni variabile viene descritta in dettaglio su un altro foglio. Nel foglio delle regole (Rule sheet) si immettono le equazioni da risolvere, ognuna delle quali può contenere fino a 200 caratteri e deve conformarsi alle comuni convenzioni matematiche. Questo foglio occupa la parte inferiore dello schermo di apertura del *TK!*. Il foglio delle unità (Unit sheet) immagazzina i dati occorrenti per convertire le unità di misura collegate alle diverse variabili di un modello.

Il *TK!Solver* utilizza questi tre fogli di lavoro per eseguire la maggior parte delle applicazioni, ma ne esistono altri: "Global", che consente all'utente di personalizzare alcune delle procedure di calcolo del *TK!*; "List", che accoglie una matrice di valori destinati alle variabili; "User Function", per le procedure definite dall'utente ed altri fogli di lavoro per disegnare e stampare serie di punti ricavati dai dati o tabelle di valori.

Vediamo adesso come creare un semplice modello, secondo l'impostazione fornita dal manuale del *TK!Solver*. Si tratta di calcolare in base alla distanza percorsa (in chilometri) la velocità media di un'automobile e il suo consumo di carburante durante un viaggio, convertendo poi i risultati in unità di misura anglosassoni. Nell' immagine d'apertura, il *TK!* posiziona il cursore sull'ultima riga dello schermo, all'interno del foglio delle regole. Innanzi tutto vanno definite le variabili per le equazioni del modello, così:

dist/tempo = vel

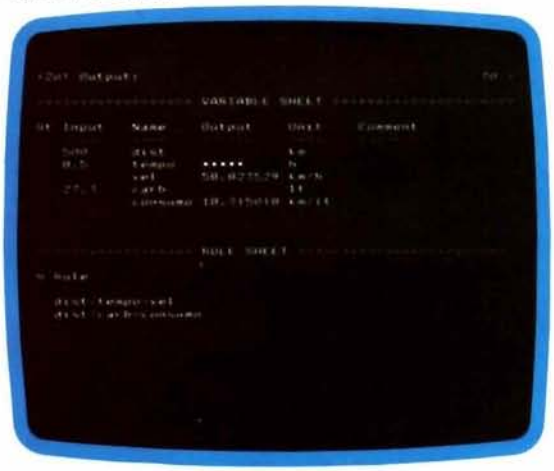


Poi si preme Return. Per prima cosa il *TK!* copia i nomi delle variabili direttamente nella colonna "Name" del foglio delle variabili, in base all'ordine in cui essi compaiono nell'equazione. Quindi visualizza un asterisco nella colonna "Status": ciò indica che l'equazione è incompleta, perché mancano ancora i valori per le variabili. In modo analogo si immette la seconda equazione:

dist/carb = consumo

A questo punto, tutte e cinque le variabili compaiono nella colonna dei nomi del foglio "Variable" (vedere figura).

Equazioni e Variabili



Premendo il tasto del "punto e virgola"(;), il cursore passa dal foglio delle regole a quello delle variabili, di cui si devono inserire i valori. Il cursore si posiziona sulla colonna "Input" a sinistra della prima variabile, dist. L'immissione dei valori, a fianco dei rispettivi nomi, non presenta difficoltà e ogni immissione termina premendo il tasto Return o quello per spostare il cursore in basso.

INPUT	NAME	OUTPUT
500	dist	
8,5	tempo	
	vel	
27.3	carb	
	consumo	

I valori di vel (velocità) e di consumo sono quelli da far calcolare al *TK!* e quindi non vanno immessi. Per far comparire i risultati (nella colonna "Output"), ossia per calcolare l'equazione, si preme il tasto del punto esclamativo (!) chiamato anche tasto "d'azione". Fatto ciò, il *TK!* visualizza al di sopra del foglio "Variable" il messaggio Direct Solver, per indicare che il programma possiede adesso tutti i dati necessari per eseguire direttamente i calcoli. Dopo un breve intervallo, i risultati vengono riportati nella colonna "Output", in corrispondenza delle variabili vel e consumo. A questo punto, si possono variare a piacimento i valori inseriti in precedenza, per ricalcolare le equazioni in base ai nuovi parametri.

I valori, inseriti sino ad ora nel modello, mancano delle corrispondenti unità di misura. Non basta scrivere chilometri o litri nella colonna "Unit", perché le unità di misura non sono state ancora definite. Riportiamo il cursore sul foglio "Rule" (con il tasto ;) e battiamo =U. *TK!* sostituisce "Rule" con il foglio di lavoro "Unit", che presenta 4 colonne:

From To Multiply by Add Offset

Il cursore è posizionato sotto la parola From e l'utente può inserire le unità di misura e i parametri di conversione necessari al programma (vedere figura).

Soluzione con parametri diversi



Terminata questa fase, si digita =R, per tornare al foglio "Rule", seguito da ; per riportare il cursore sul foglio "Variable". Adesso si possono inserire i nomi delle unità di misura nella colonna "Unit": km per distanza, h per tempo km/h per velocità, lt per carburante e km/lt per consumo. A questo punto si immettono nuovi parametri di calcolo (ad esempio: 1.274 per distanza, 60 per velocità e 18.3 per consumo) cancellando i precedenti. Infine si preme ! per risolvere l'equazione: la colonna "Output" visualizza i risultati ; nella colonna delle unità di misura, si posiziona il cursore su km e lo si sostituisce con m (miglia). Premendo Return, il *TK!* trasforma in miglia il valore della distanza.

Tabella Unità di Misura



IAN McKINNELL

Questo esempio illustra soltanto alcune delle facilitazioni offerte dal *TK!*. In seguito viene esaminata la creazione di un modello più sofisticato, che sfrutta le capacità grafiche e le funzioni complesse del *TK!*.

Lo Spectrum +

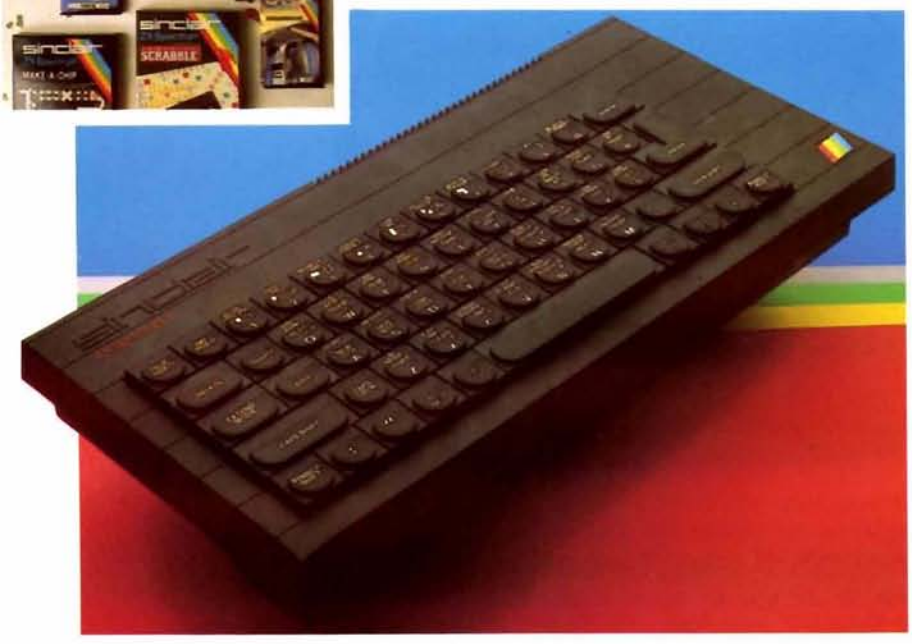
Anatomia di un microcomputer a diffusione europea: tecnologie a confronto

Quando venne presentato al pubblico, nella primavera dell'82, lo Spectrum aveva come unici concorrenti il Vic-20, dotato di appena 3,5 Kbyte di memoria per l'utente, e il TI99 4A della Texas, che costava circa il doppio. Grazie al prezzo molto accessibile, lo Spectrum riscosse un immediato successo e divenne una scelta quasi obbligata per le migliaia di appassionati di micro che desideravano magari un apparecchio più potente dello ZX80 o dello ZX81. Il nuovo micro offriva una memoria di ben 48 Kbyte, un discreto BASIC, grafica a 8 colori ed alcune funzioni audio basilari. Inoltre, la tastiera era stata notevolmente modificata e migliorata rispetto a quella a lamina plastica dello ZX81.
Disponibile in un primo momento solo su ordinazione postale, lo Spectrum, nel volgere di pochi mesi, divenne il computer più venduto in Gran Bretagna.
Nel giro di due anni comparve sul mercato una serie di apparecchi, prodotti dalle ditte concorrenti della Sinclair, che dovevano contendere allo Spectrum il primato delle vendite. Il suo più agguerrito concorrente è stato il Commodore 64 che, pur dotato di un BASIC di livello decisamente inferiore, aveva però una memoria di maggiori dimensioni (le cui potenzialità venivano esaltate dal codice macchina), ottime funzioni audio ed una tastiera di tipo professionale. Un altro concorrente è stato il Micro BBC, che offriva caratteristiche superiori, ma che l'alto prezzo rendeva poco competitivo.

Il "Six Pack" Spectrum

La Sinclair, con lo Spectrum + e lo Spectrum 48K, ha seguito la tendenza di fornire insieme ai microcomputer un package di programmi. I due apparecchi sono infatti corredati da sei ottimi programmi: un word processor, uno spreadsheet, due giochi e due programmi per la grafica

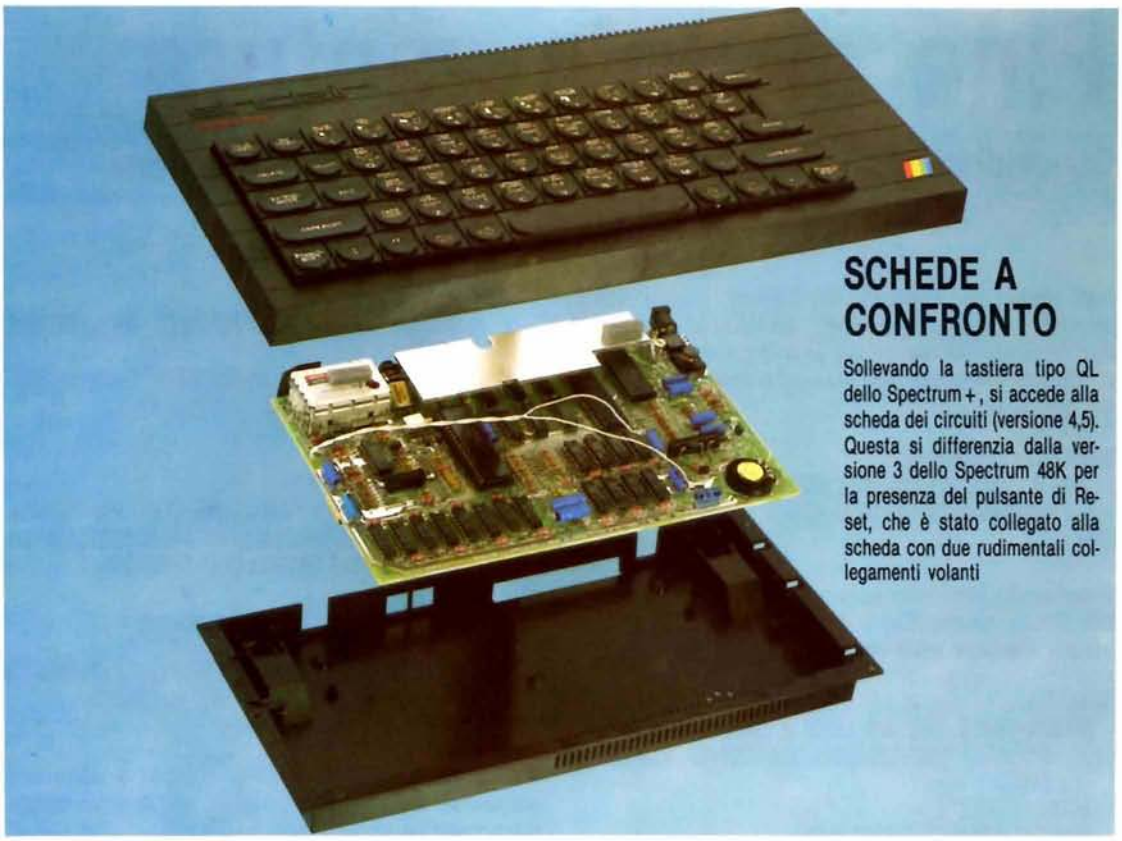




La Sinclair e la Commodore ingaggiarono una dura battaglia commerciale ribassando, alternativamente, i prezzi dei propri apparecchi, fino a giungere ad un prezzo di listino quasi identico.
Da questo momento in poi, la tastiera "gommosa" dello Spectrum è diventata un elemento da ritenersi decisamente a sfavore dell'apparecchio. La Sinclair tentò di correre ai ripari, sottolineando la vastità del software disponibile per lo Spectrum e dotando l'apparecchio di una serie di pacchetti "seri", tra i quali un word processor.
Purtroppo il risultato fu quello di evidenziare ancor più i limiti della tastiera. Molti utenti decisero allora di acquistare tastiere aggiuntive, prodotte da altre Case, e questa tendenza si consolidò quando, con molto ritardo sul previsto, la Sinclair lanciò l'Interfaccia 1 e il Microdrive. Tuttavia, le discutibili scelte tecniche ed i ritardi nella loro attuazione hanno in qualche modo offuscato l'immagine della Sinclair.

Un "design" rivoluzionato

La Sinclair Research ha quindi deciso di voltare pagina modificando radicalmente l'estetica dello Spectrum. Pur conservando quasi inalterate le caratteristiche tecniche del suo predecessore, il nuovo Spectrum+ presenta una versione "ridotta" della tastiera del QL. E' dotata, infatti, di qualche tasto in più, di un interruttore di Reset e di una coppia di piedini retrattili. L'apparecchio è compatibile con tutte le periferiche prodotte per il vecchio modello, ma la Sinclair avrebbe potuto renderlo più competitivo o migliorandone le funzioni relative alla generazione di suoni, che restano mediocri, oppure inserendo una presa per monitor o incorporando l'Interfaccia 1. I due piedini regolabili sollevano la base dal piano di appoggio e così accrescono il volume dei suoni emessi; questa modifica si è però dimostrata svantaggiosa perché il maggior volume amplifica anche i rumori durante il caricamento e il salvataggio dei programmi.
La nuova tastiera ha comunque svariati vantaggi: possiede tasti in più per i modi "grafico" ed "Extended" e per il video normale e inverso, i tasti Break e Delete ed un gruppo di tasti singoli per i caratteri d'interpunzione più usati (punto e virgola, virgolette, virgola e punto), configurazione questa che mira a facilitare la programmazione. Inoltre è stato aggiunto il tasto Simbol Shift, mentre quelli per il



CHRIS STEVENS

SCHEDE A CONFRONTO

Sollevando la tastiera tipo QL dello Spectrum+, si accede alla scheda dei circuiti (versione 4,5). Questa si differenzia dalla versione 3 dello Spectrum 48K per la presenza del pulsante di Reset, che è stato collegato alla scheda con due rudimentali collegamenti volanti

SPECTRUM +

DIMENSIONI

mm. 319 x 149 x 38

CPU, MEMORIA E INTERFACCE

Le stesse dello Spectrum 48K. Il BASIC è "resistente" e la compatibilità software è totale

TASTIERA

58 tasti "scolpiti", inclusa una barra spaziatrice (al posto di un semplice tasto). I tasti sono di tipo a membrana (vedere a pag. 983)

DOCUMENTAZIONE

Il manuale, riccamente colorato ma non del tutto esauriente, è accompagnato da una cassetta dimostrativa che facilita la conoscenza del sistema

PRO

Indiscutibili attrattive sono: enorme quantità di software disponibile, esistenza di numerosi "User group" e innumerevoli pubblicazioni di supporto

CONTRO

Nonostante le modifiche e le aggiunte apportate, la tastiera rimane sempre il punto debole dell'apparecchio

cursore si trovano ora ai due lati della piccola barra spaziatrice.
Le precedenti combinazioni dei tasti non sono state modificate, tuttavia la nuova tastiera può creare qualche difficoltà ai "veterani" dello Spectrum. In particolare, la battuta accidentale del nuovo tasto Edit (posto sotto il tasto A) durante l'inserimento di una linea di programma ne provoca la completa cancellazione.
Lo Spectrum+ è corredato da un package di sei programmi: *Psion Chess, Make-a-Chip, Scrabble, Chequered Flag, Vu-3D* e *Tastword Two*, un eccellente word processor. Tutti i programmi sono di ottima qualità. Sfortunatamente, non si può dire la stessa cosa della documentazione che, presentata con molta cura, è però priva della completezza del vecchio manuale Spectrum. Per questo motivo sono previsti manuali più esaurienti, da acquistare a parte.
L'ottima rispondenza dello Spectrum+ ai programmi di gioco ha indotto molti appassionati del genere a comprare un joystick ed un'interfaccia. L'apparecchio è infatti compatibile con entrambe le interfacce Kempston e Fuller ma, come sul vecchio modello, l'uso del Fuller Soundbox può impedire il funzionamento di alcuni programmi. L'interfaccia Centronics della Kempston e la memoria di massa Wafadrive funzionano invece perfettamente.
Lo Spectrum+ è senza dubbio un micro più avanzato del suo predecessore, ma la nuova configurazione della tastiera è stata oggetto di molte critiche. Sebbene vada apprezzata la decisione della Sinclair di ricorrere alla tecnologia del QL per migliorare le prestazioni dello Spectrum, i vantaggi che ne derivano non sembrano giustificare il forte aumento di prezzo. L'estetica dell'apparecchio è indubbiamente migliorata, ma i tasti "scolpiti", che dovrebbero facilitare la battitura, si dimostrano in realtà poco sensibili e troppo compattati.
Nasce quindi il sospetto che la Sinclair abbia cinicamente lanciato lo Spectrum+ con l'unico intento di aumentare i prezzi, progettando di cessare entro breve tempo la produzione dello Spectrum.
Una scelta più ragionevole sarebbe stata quella di ritoccare il prezzo del vecchio modello dopo aver introdotto le modifiche veramente necessarie: migliori funzioni di suono, una tastiera tipo macchina da scrivere, una presa per monitor e, eventualmente, un Microdrive interno.



NUOVI TASTI

Il giudizio sulla nuova tastiera dello Spectrum+ dipende soprattutto dalle abitudini, dalle esigenze individuali. L'introduzione di tasti indipendenti per le funzioni più importanti offre indubbiamente concreti vantaggi, ma restano valide le combinazioni di tasti dello Spectrum (ad esempio [SYM SHIFT] + [0] per il punto e virgola) che duplicano le nuove funzioni

IAN McKINNELL

Fine di un'avventura

Concludiamo lo sviluppo del gioco definendo le ultime routine

Nel gioco *Il Tempio di Zoltoth* sono stati inseriti solo due "pericoli". In LUOGO.4 compare davanti al giocatore un grosso serpente ed il programma salta ad una speciale procedura di "pericolo":

```
TO ATTACCO.SERPENTE
   PRINTL [[C'È UN ENORME SERPENTE] [CHE STRISCIA
      LENTAMENTE VERSO DI TE!]]
END
```

Il secondo pericolo non comporta una minaccia fisica immediata per il giocatore, ma a lungo andare può causare grossi problemi:

```
TO PORTA
   PRINTL [[ALLE TUE SPALLE SI È CHIUSA UNA PORTA
      DORATA] [IMPEDENDOTI L'ACCESSO ALL'USCITA
      SUD]]
   MAKE "PERICOLI []
   MAKE "LISTA.USCITE [[N7] [E8]]
END
```

In altre fasi del programma si devono fare considerazioni di diverso tipo. La procedura RACCOGLI deve essere modificata in modo da impedire al giocatore di prendere l'anello se già possiede la spada.

```
TO PRENDI :ITEM
   IF :ITEM = "ANELLO THEN RACCOGLI.ANELLO STOP
   AGG.INVENT :ITEM
   RIMUOVI.DA.LUOGO :ITEM
END

TO RACCOGLI.ANELLO
   IF MEMBER? "SPADA :INVENTARIO THEN PRINT [NON
      PUOI RACCOGLIERE L'ANELLO] STOP
   AGG.INVENT :ITEM
   RIMUOVI.DA.LUOGO :ITEM
END
```

Questo è l'unico caso in cui si impedisce al giocatore di raccogliere un oggetto. Le seguenti procedure consentono al giocatore di esaminare ciò che tiene in mano.

```
TO CONTROLLA :OGGETTO
   IF :OGGETTO = "ANELLO THEN DESCRIZ.ANELLO STOP
   IF :OGGETTO = "SCRIGNO THEN DESCRIZ.SCRIGNO STOP
   IF :OGGETTO = "SPADA THEN DESCRIZ.SPADA STOP
   PRINT [NON VEDI NULLA DI PARTICOLARE]
END

TO DESCRIZ.ANELLO
   IF HERE? "ANELLO THEN PRINTL [[SULL'ANELLO È INCISA
      UNA PAROLA ILLEGGIBILE:] [R—— —E]] ELSE PRINT
      [NON VEDO ALCUN ANELLO]
   END
```



```
TO HERE? :OGGETTO
   IF MEMBER? :OGGETTO :CONTENUTO THEN OUTPUT
      "TRUE
   IF MEMBER? :OGGETTO :INVENTARIO THEN OUTPUT
      "TRUE OUTPUT "FALSE
   END

TO DESCRIZ.SCRIGNO
   PRINTL [[È DI OTTIMA FATTURA] [E CERTAMENTE VALE
      UNA PICCOLA FORTUNA] [IN UN ANGOLO DEL COPER-
      CHIO] [È INCISO UN MINUSCOLO TESCHIO]]
END

TO DESCRIZ.SPADA
   IF HERE? "SPADA THEN PRINT [È DI ACCIAIO TEMPRATO]
      ELSE PRINT [NON VEDO ALCUNA SPADA]
END
```

Al giocatore occorre la spada per uccidere il serpente: se non la possiede sarà il serpente ad avere la meglio.

```
TO UCCIDE :CHI
   IF :CHI = "SERPENTE THEN UCCIDI.SERPENTE STOP
   PRINT [NON PUOI FARLO!]
END

TO UCCIDI.SERPENTE
   IF NOT MEMBER? "ATTACCHI.SERPENTE :PERICOLI THEN
      PRINT [NON VEDO ALCUN SERPENTE] STOP
   IF MEMBER? "SPADA :INVENTARIO THEN MORTE.
      SERPENTE ELSE SERPENTE.UCCIDE
END

TO MORTE.SERPENTE
   PRINT [IL SERPENTE AGONIZZA E MUORE]
   MAKE "PERICOLI []
END

TO SERPENTE.UCCIDE
   PRINTL [[NON POSSIEDI ALCUNA ARMA] [CON CUI UCCI-
      DERLO] [MA SEI PARALIZZATO DALLA PAURA] [TI HA
      MORSO! IMPALLIDISCI] [E CADI MORENTE AL SUOLO]]
   MORTE
END
```

La procedura MORTE disillude quei giocatori che credono nella reincarnazione. Se dopo il decesso viene inserita una parola diversa da INIZIO, il computer ricorda al giocatore che la sua vita è terminata!

```
TO MORTE
   PRINT [SEI MORTO!]
   PRINT1 "?
   MAKE "INPUT COMANDO
   IF (:INPUT = "INIZIO) THEN INIZIO STOP
   PRINT [BATTI "INIZIO" PER GIOCARE DI NUOVO]
   MORTE
END
```

```
TO COMANDO
   MAKE "INP REQUEST
   IF :INP = [] THEN PRINT1 "? OUTPUT COMANDO
   OUTPUT FIRST :INP
END
```

Strofinando l'anello compare un "genio":

```
TO STROFINA :OGGETTO
   IF :OGGETTO = "ANELLO THEN STROFINA.ANELLO STOP
   PRINT [ORA BRILLA MOLTO PIÙ DI PRIMA]
END

TO STROFINA.ANELLO
   IF HERE! "ANELLO THEN GENIO ELSE PRINT [NON VEDO
      ALCUN ANELLO]
END
```

Il genio propone al giocatore di trasportarlo fuori dal sotterraneo. Se l'invito viene rifiutato si alza un forte vento che trascina a caso il giocatore in una stanza posta nel settore est del sotterraneo:

```
TO GENIO
   PRINTL [[APPARE UN GENIO CHE TI DOMANDA:] ["VUOI
      TORNARE FUORI DAL SOTTERRANEO?"]]
   PRINT1 "?
   MAKE "RISPOSTA FIRST COMANDO
   IF ANYOF :RISPOSTA = "SI :RISPOSTA = "S THEN FUORI
      ELSE VENTO
END

TO FUORI
   PRINT [FINALMENTE ALL'ARIA APERTA]
   IF MEMBER? "SCETTRO :INVENTARIO THEN PRINT [CON-
      GRATULAZIONI, HAI TROVATO LO SCETTRO!] ELSE
      PRINTL [[ALMENO HAI SALVATO] [LA VITA]]
END

TO VENTO
   PRINT [SI ALZA UN FORTE VENTO]
   PRINT "
   SPOSTAMENTO1 (6 + (RANDOM 5))
END
```

Il solo oggetto che il giocatore può aprire è lo scrigno, che contiene un ragno velenoso. Il teschio sul coperchio sconsiglia di aprirlo, ma molti giocatori ignorano l'avvertimento!

```
TO APRI :OGGETTO
   IF :OGGETTO = "SCRIGNO THEN APRI.SCRIGNO ELSE
      PRINT [NON C'È NULLA DA APRIRE]
END

TO APRI.SCRIGNO
   PRINTL [[C'È UN RAGNO VELENOSO] [NELLO SCRIGNO] [TI
      HA MORSO!]]
   MORTE
END
```

Questa infine è una lista di tutti i nomi usati nel gioco:

```
TO SPADA
   OUTPUT "SPADA
END

TO SCRIGNO
   OUTPUT "SCRIGNO
END

TO SCETTRO
   OUTPUT "SCETTRO
END

TO ANELLO
   OUTPUT "ANELLO
END

TO SERPENTE
   OUTPUT "SERPENTE
END
```

Il comando SAVE "ADVENTURE permette di salvare il gioco (assieme all'area di lavoro) in qualsiasi momento: per riprenderne l'esecuzione si digita READ "ADVENTURE. In molti casi il LOGO si dimostra un linguaggio ideale per la programmazione di giochi di tipo adventure. Purtroppo, attualmente, l'area di lavoro a disposizione dei programmi scritti dall'utente è piuttosto limitata. Il gioco *Il Tempio di Zoltoth* viene a stento contenuto nell'area di memoria LOGO del Commodore 64. Per questo motivo bisogna stare molto attenti se si desidera cambiare le parole o le frasi presenti nel gioco.

Logo & Logo

Molte versioni del MIT LOGO non possiedono EMPTY?, ITEM, COUNT oppure MEMBER?. Le definizioni per queste primitive sono già state date (vedere alle pagg. 1234 e 1255). In tutte le versioni LCSI si usi:

EMPTYP per EMPTY?
LISTP per LIST?
MEMBERP per MEMBER?
TYPE per PRINT1
AND per ALLOF
OR per ANYOF

Esiste anche una primitiva, EQUALP, che verifica l'eguaglianza tra due termini. Si usi questa, al posto del segno uguale, per confrontare liste o parole (anche se in alcune versioni LCSI il segno uguale funziona con le liste).
Nel LOGO della LCSI, la sintassi completa per l'espressione IF è illustrata da questo esempio:

IF EMPTYP :CONTENUTO [PRINT [NIENTE DI SPECIALE]] [PRINT :CONTENUTO]

La prima lista, dopo la condizione, viene eseguita se la condizione è vera, la seconda se la condizione è falsa.
Il LOGO ATARI usa SE per SENTENCE ed RL per REQUEST. La primitiva ITEM è invece assente. La versione del gioco qui presentata "gira" sul Commodore 64: altri apparecchi possono mancare di spazio sufficiente per eseguire il gioco nell'attuale versione. In questi casi è necessario ridurre le dimensioni del gioco, eliminando una parte delle frasi descrittive

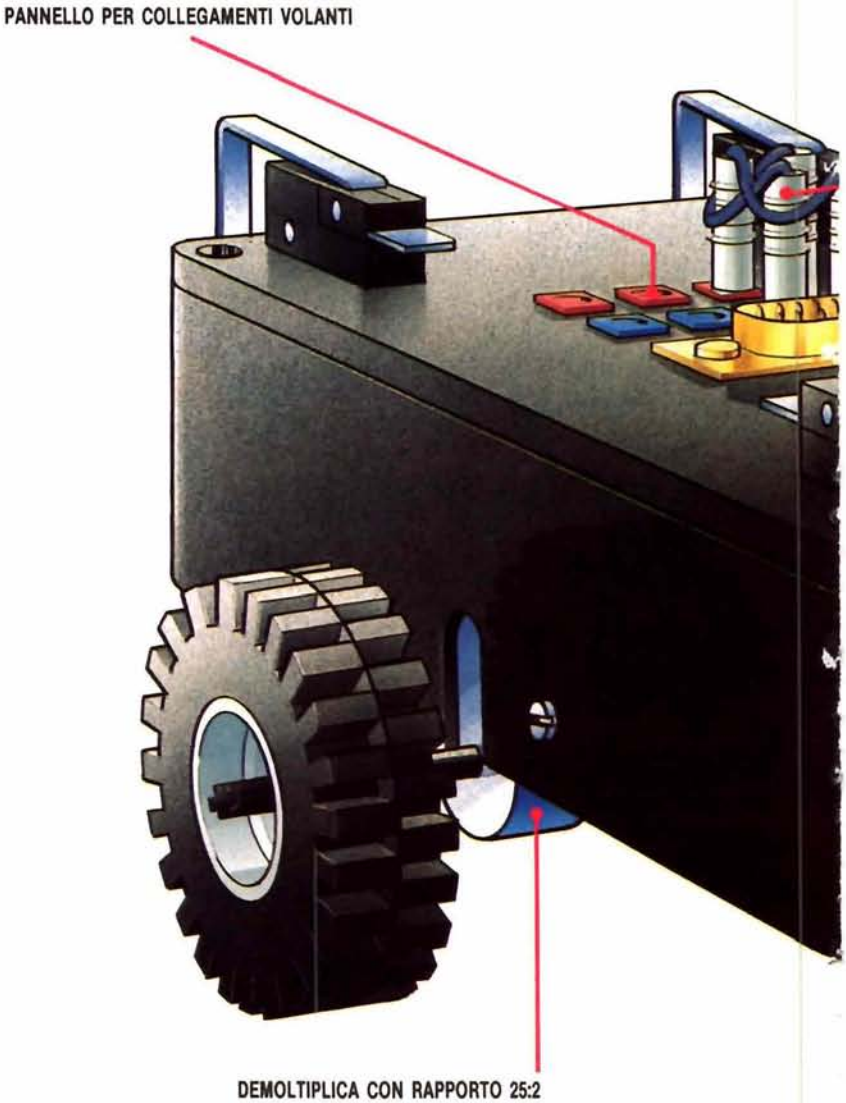
LIZ DIXON

Un semplice robot

La prima parte del progetto di un robot da pavimento dotato di sensori

Questo nuovo progetto è dedicato alla costruzione di un robot da pavimento ed alla realizzazione del relativo software di controllo. Il robot viene mosso da due motori passo passo, che azionano le due ruote per mezzo di un sistema di ingranaggi. Ad ogni passo i due motori ruotano di 7,5°. Applicando al loro alberino una demoltiplica con un rapporto di 25:2, le ruote del veicolo eseguono una rotazione di 0,6° ad ogni passo. Poiché i motori passo passo ruotano di un dato angolo ogni volta che ricevono un impulso, si adattano perfettamente ad un controllo di tipo digitale. In questo progetto i segnali di controllo provengono dalla porta utente del computer e vengono generati da un semplice programma scritto appositamente per il robot. Oltre a possedere una coppia di motori passo passo, il robot è dotato di 4 sensori di contatto e di due fotosensori, che consentono di seguire una linea tracciata sul pavimento. Poiché quattro linee per i dati della porta utente vengono usate per controllare i motori del veicolo, ne restano soltanto quattro per la trasmissione dei segnali provenienti dai sensori. Per dare maggiore flessibilità al sistema, il robot è stato dotato di un piccolo pannello per connessioni "volanti", composto da una serie di prese e spinotti di raccordo che consente di selezionare diverse combinazioni di sensori, collegandoli di volta in volta alle quattro linee disponibili. A seconda delle applicazioni a cui il robot è destinato, si possono attivare tutti e quattro i sensori di contatto, oppure due sensori di contatto e due sensori luminosi.

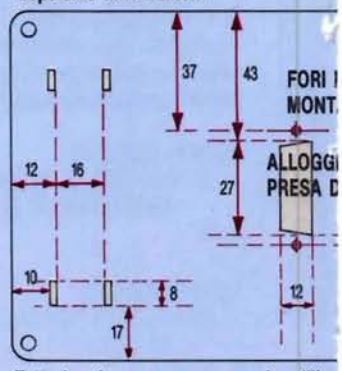
I sensori permettono una discreta precisione nel controllo del veicolo, quindi il software deve essere abbastanza sofisticato da consentire la creazione di una mappa interna delle immediate vicinanze del robot. In seguito vengono affrontati i problemi che riguardano il controllo dei movimenti e gli algoritmi per la loro esecuzione. Per adesso ci occupiamo della costruzione vera e propria del robot. Dobbiamo innanzi tutto forare ed intagliare la scatola in materiale plastico che forma il rivestimento e lo "chassis". Il fissaggio degli ingranaggi e delle viti che bloccano il connettore D richiede la massima precisione, mentre quello dei piedini di bilanciamento non ha bisogno di altrettanta attenzione.



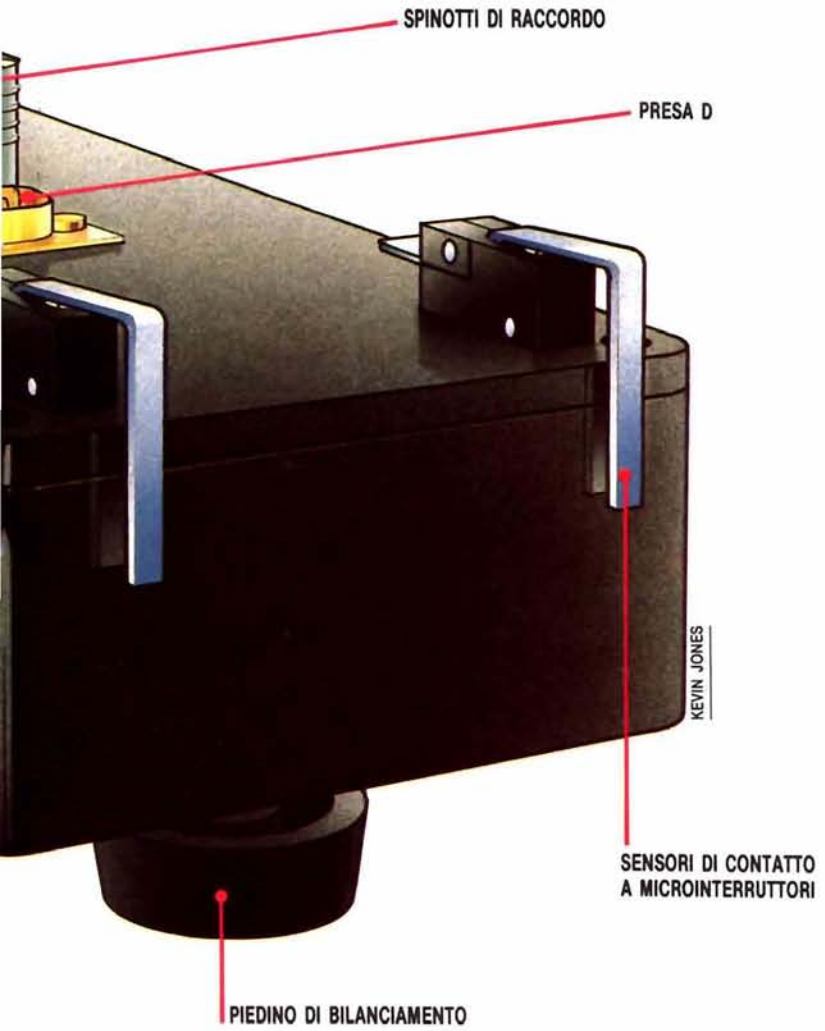
I FASE

Per prima cosa occorre praticare le aperture nella base e sui lati della scatola che forma lo chassis del robot (lo schema a destra ne indica la posizione e le dimensioni). Le aperture laterali accolgono i mozzi delle ruote. I fori di fissaggio per i motori passo passo e le demoltipliche devono trovarsi sullo stesso asse. Ai due fori circolari sul fondo dello chassis vengono fissati i due piedini che equilibrano il robot. Nell'apertura sul coperchio viene montata la presa D per il connettore di collegamento con il computer. Per praticare le aperture per i mozzi e le scatole di trasmissione, si può rimuovere la maggior parte della plastica con un saldatore a punta sottile, quindi si usa una piccola lima per portare i fori alle dimensioni richieste

Coperchio della scatola

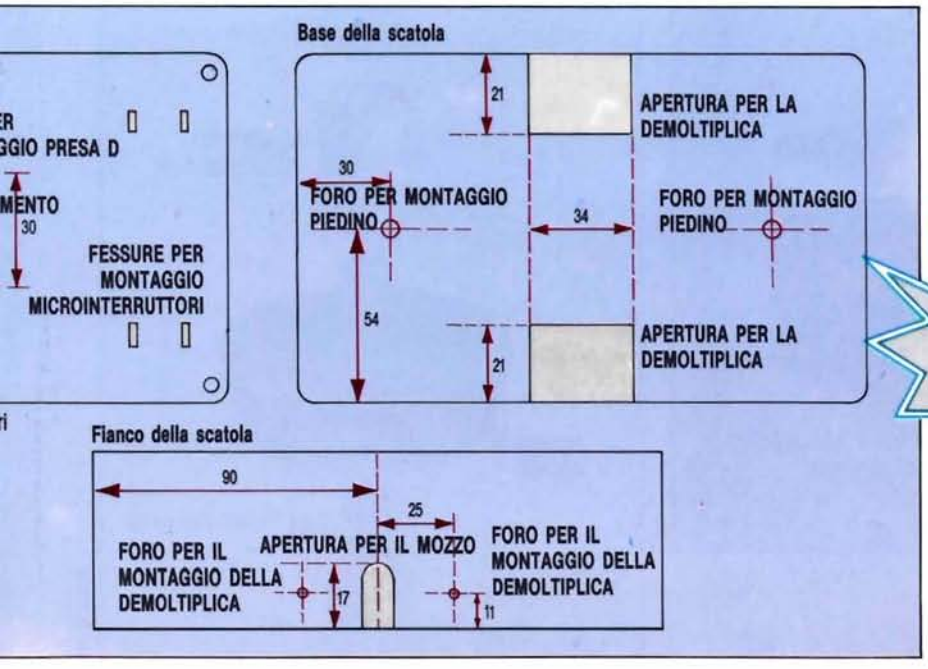


Tutte le misure sono espresse in millim



LISTA DEI COMPONENTI

Quantità	Descrizione
2	Circuiti guida per motori a passo SAA 1027
2	Motori passo passo
2	Demoltipliche sincrone, rapporto 25:2
1	Circuito integrato 40109
3	Prese DIL a 16 piedini
2	Resistenze da 100 ohm
2	Resistenze da 270 ohm (1/2 Watt)
2	Condensatori da 1 microF
1	Condensatore da 1000 microF, 25 VL
1	Veroboard a 50 fori x 24 piste
1	Rocchetto di filo stagnato
1	Connettore D a 15 canali
1	Presa D a 15 canali
1	Calottina di protezione D a 15 canali
1	Presa di alimentazione da 2,1 mm
1	Presa IDC a 20 canali
1	Connettore "a pettine" con 24 canali
1	Scatola, dimensioni 180 x 110 x 55 mm
1	Cuscinetto auto-adesivo
2	Piedini di gomma
1	Scatola di dadi 2BA
1	Scatola di bulloncini 6BA
1	Scatola di dadi 6BA
1	Scatola di bulloncini M5 lunghi 25 mm
1	Scatola di dadi M5
2	Ruote "Lego", diametro 62 mm
1	Scatola di barrette assiali Lego
4 m	Cavo a nastro a 12 canali
1 m	Cavo a nastro a 20 canali
1	Alimentatore a corrente continua da 12 Volt, 1 Ampere
2	Bulloncini 2BA x 4 cm



ATTENZIONE!

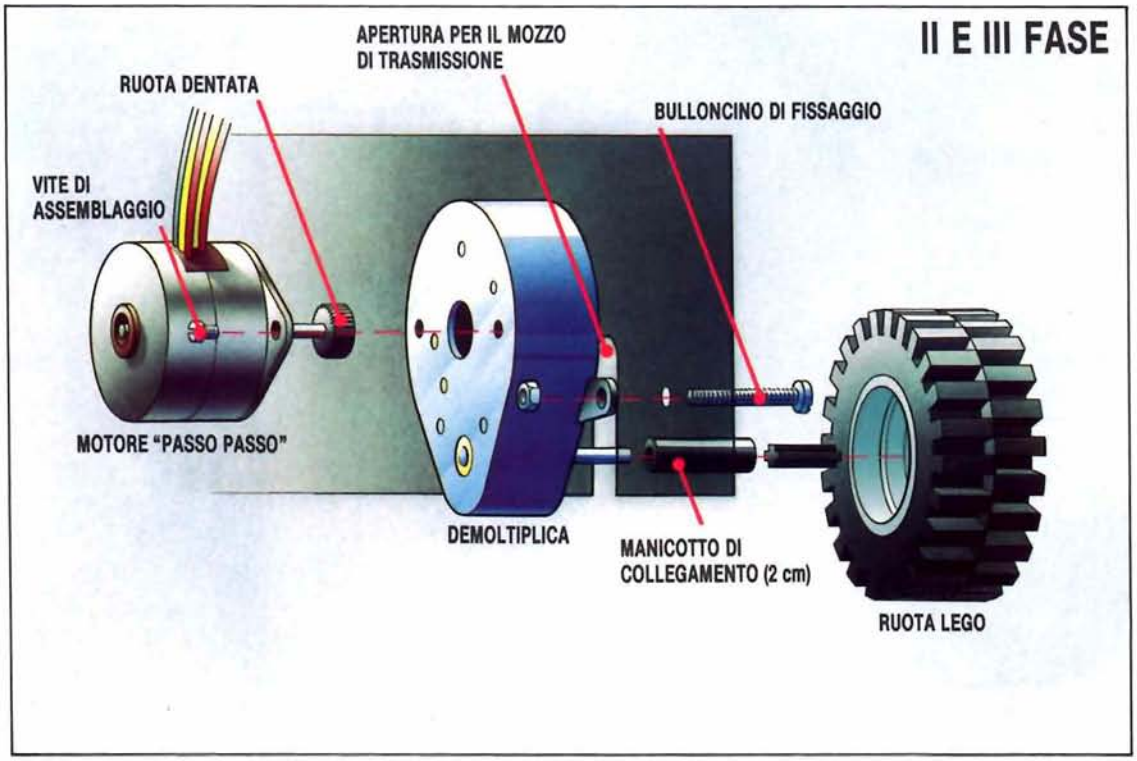
Il robot consuma una notevole quantità di corrente: se l'alimentatore fornisce corrente anche al "buffer box", il veicolo non riesce a muoversi. Pertanto, il robot va collegato direttamente alla porta utente del computer. Se non si è in grado di seguire con precisione le istruzioni, è preferibile abbandonare il progetto: errori di costruzione, anche banali, possono danneggiare irrimediabilmente il computer

IAN McKINNELL

II E III FASE

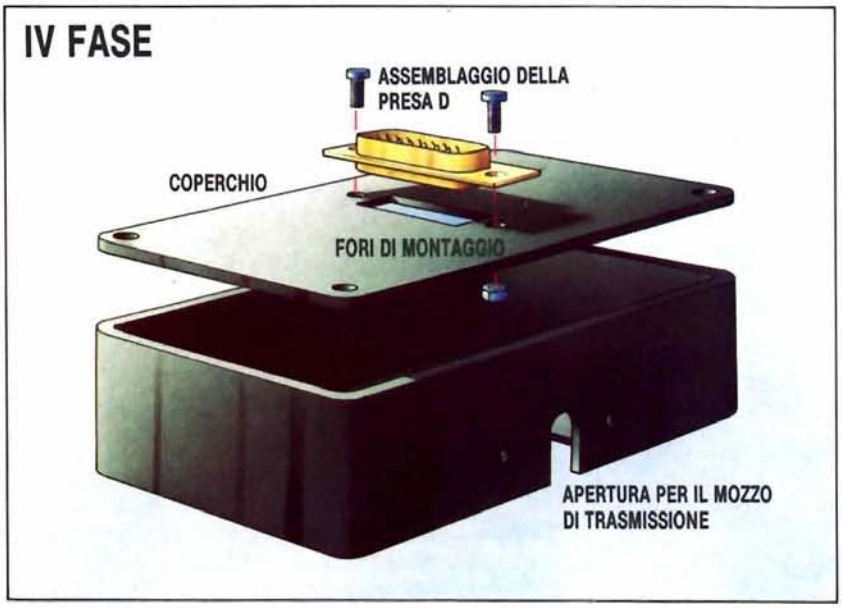
I motori e le demoltipliche vengono acquistati separatamente e quindi assemblati. Il "kit" della demoltiplica contiene un piccolo ingranaggio metallico forato al centro ed un cilindretto spaziatore in materiale plastico. L'ingranaggio deve essere innestato ed incollato all'alberino che sporge dal motore con la parte smussata rivolta dalla parte opposta rispetto al motore. Servirsi di un apposito collante per metalli. Usare una piccola sezione del cilindretto spaziatore per separare nel modo corretto l'ingranaggio dal corpo del motore. Attendere che l'adesivo si secchi completamente

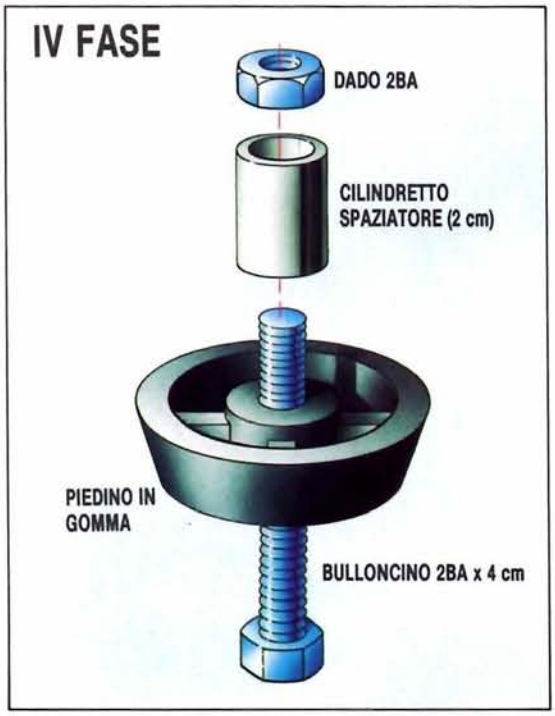
Fissare il motore alla scatola di trasmissione con le apposite viti, facendo in modo che i cavetti di collegamento siano rivolti verso l'alto. Al momento dell'assemblaggio, assicurarsi che l'ingranaggio applicato all'alberino del motore si innesti correttamente negli ingranaggi interni della scatola di trasmissione. Usare i bulloncini M5 per fissare la scatola di trasmissione allo chassis del veicolo. Questa non viene fermata sul rivestimento plastico, ma sul bulloncino che a sua volta viene fissato allo chassis. Quindi la posizione del blocco motore può essere facilmente modificata. Le ruote vengono innestate su due sbarrette Lego con sezione a x. Inserire sull'alberino della scatola di trasmissione un cilindretto lungo 2 cm e di adeguato diametro, come manicotto di collegamento, e fissarlo con la colla. Quindi incollare alla sua estremità libera il mozzo Lego sul quale viene fissata la ruota



IV FASE

Fissare la presa D sul coperchio con i piedini rivolti verso l'alto. Infine montare i due piedini di bilanciamento bloccando internamente i bulloncini 2BA con gli appositi dadi. I piedini vengono separati dalla base della scatola per mezzo di un cilindretto spaziatore lungo 2 cm. oppure usando 2 dadi





Digitaya: gli ambienti

Come descrivere i luoghi visitati e come muoversi dall'uno all'altro

La descrizione dei vari ambienti è contenuta nella matrice LN$() (vedere a pag. 1247) e può essere richiamata usando semplicemente il numero dell'ambiente visitato. Nel gioco *Foresta Incantata*, la posizione occupata dal giocatore in un dato momento viene registrata nella variabile P, cosicché la sua descrizione si trova in LN$(P). Quando si progetta la descrizione di un ambiente, va tenuto conto del contesto della frase di cui andrà a fare parte, che inizia sempre con: «Ti trovi...», seguito dalla descrizione dell'ambiente; la frase viene quindi inviata alla routine di utilità descritta in precedenza (linea 5500). La formattazione e la visualizzazione delle frasi avvengono, in *Foresta Incantata*, a partire dalla linea 2010.

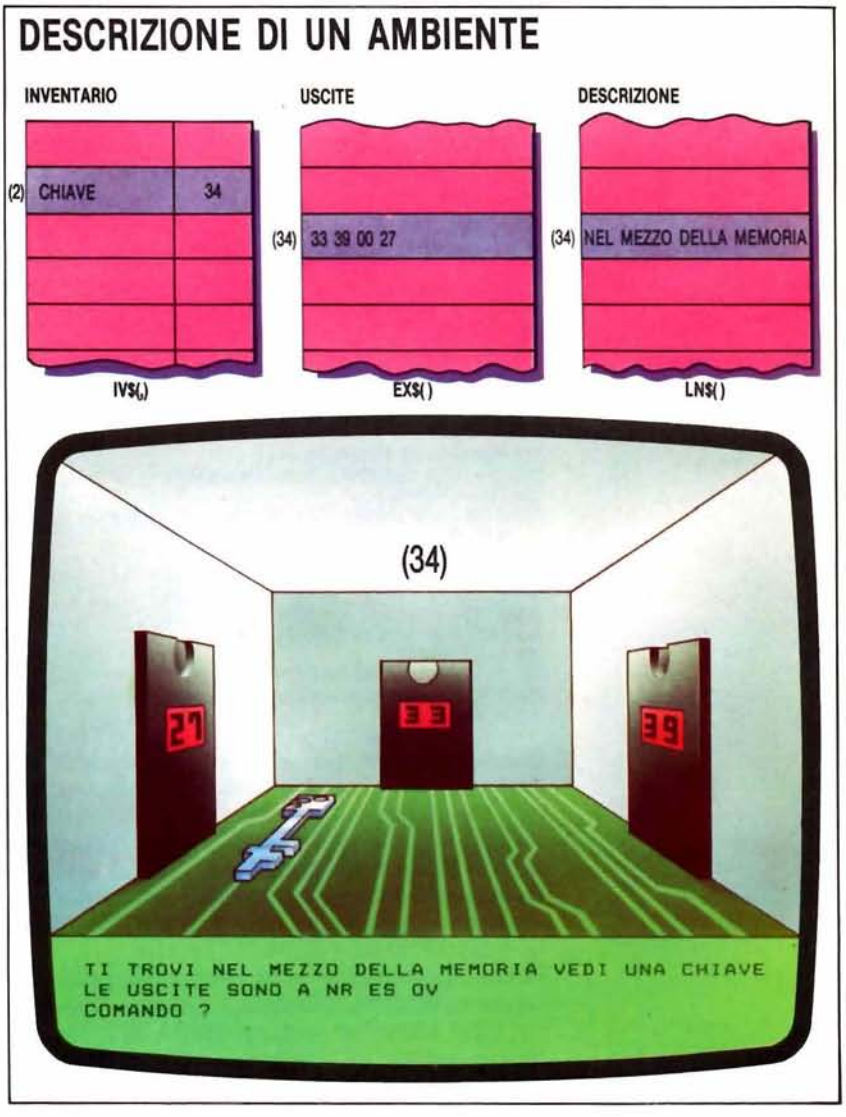
Oltre alla descrizione dell'ambiente, il giocatore va informato sugli oggetti che esso contiene. Gli oggetti usati durante il gioco vengono immagazzinati, assieme alle loro posizioni iniziali all'interno dell'inventario, nella matrice a due dimensioni IV$(,). L'elemento IV$(N,1), ad esempio, contiene la descrizione dell'oggetto "ennesimo" dell'inventario ed IV$(N,2) la sua posizione. Volendo sapere se un dato ambiente contiene un particolare oggetto, basta scorrere l'inventario confrontando la posizione di ciascun oggetto con il numero della locazione nella quale ci si trova. Dato che i giochi *Foresta Incantata* e *Digitaya* usano rispettivamente tre ed otto oggetti, questa operazione può essere eseguita con una semplice procedura di ricerca lineare, basata su un ciclo FOR...NEXT.

Le linee comprese tra la 2040 e la 2080 contengono il ciclo di ricerca usato dal programma *Foresta Incantata*. La seconda colonna della matrice dell'inventario viene confrontata con la posizione corrente P. Se i due valori coincidono, la frase che descrive l'oggetto viene aggiunta alla descrizione dell'ambiente. Poiché un ambiente può contenere più oggetti, vanno previste frasi che contengano una lista di oggetti, separati tra loro da una virgola. Per inserire le virgole in modo corretto si può usare la variabile SP$, nella quale viene depositata, all'occorrenza, una stringa "nulla" o una virgola. Ogni volta che si verifica un confronto positivo durante la fase di ricerca, un flag viene posto ad uno. Se alla fine della ricerca il flag è rimasto a zero, ciò significa che l'ambiente esaminato non contiene alcun oggetto. Questa informazione viene comunicata al giocatore dalla linea 2090 del gioco *Foresta Incantata*.

```
2000 REM •••• DESCRIVE UN AMBIENTE ••••
2010 SN$ = "TI TROVI " + LN$(P): GOSUB 550 N$ = "VEDI "
2020 SN$ = "VEDI "
2030 REM •• VERIFICA INVENTARIO OGGETTI ••
2040 F = 0:SP$ = ""
2050 FOR I = 1 TO3
2060 IF VAL(IV$(I,2)) < > P THEN 2080
2070 SN$ = SN$ + SP$ + "A " + IV$(I,1):F = 1:SP$ = ", "
2080 NEXT I
2090 IF F = 0 THEN SN$ = SN$ + "NESSUN OGGETTO"
2100 GOSUB 5500:REM FORMATA LA FRASE PER LO SCHERMO
2110 RETURN
```

La matrice EX$() contiene le informazioni riguardanti le possibili uscite da ciascun ambiente. Ogni stringa numerica contiene otto cifre che, suddivise in gruppi di due, forniscono (da sinistra a destra) il numero delle locazioni poste a nord, sud, est ed ovest rispetto alla locazione corrente. Per individuare un ambiente all'altro

Le informazioni riguardanti gli ambienti del gioco sono registrate in tre matrici a stringa, che contengono il nome e la collocazione degli oggetti (V$), le descrizioni degli ambienti (LN$) e le uscite possibili (EX$). Ad esempio, EX$(34) può contenere il numero 33390027, a indicare che l'ambiente 34 è collegato agli ambienti 33, 39 e 27, dalle rispettive uscite nord, est ed ovest. LN$(34) contiene la frase "nel mezzo della Memoria", come descrizione dell'ambiente 34. IV$(2,2) contiene il numero 34, per cui IV$(2,1), ossia "una chiave", si trova nell'ambiente 34. In base al corrente numero d'ambiente, il programma raggruppa le varie informazioni che compongono la descrizione.



duare le uscite consentite, il programma suddivide la stringa ad otto cifre in quattro numeri che indicano l'ambiente abbinato ad ogni direzione.

```
2300 REM **** S/R INDICA LE USCITE ****
2310 EX$ = EX$(P)
2320 NR = VAL(LEFT$(EX$,2))
2330 ES = VAL(MID$(EX$,3,2))
2340 SD = VAL(MID$(EX$,5,2))
2350 OV = VAL(RIGHT$(EX$,2))
```

Se un'uscita non è prevista, la corrispondente variabile vale 0: questa indicazione è molto utile per descrivere le uscite da un ambiente. Quindi, prima di scrivere la frase «Le uscite possibili sono a» va eseguito un controllo per verificare che esista almeno un'uscita. Ciò si ottiene ponendo in OR tutte e quattro le variabili di direzione: si otterrà un risultato 0 se e solo se tutte le variabili di direzione valgono 0, ossia se tutte le uscite sono precluse. In caso contrario, la routine prosegue e verifica in successione ciascuna variabile di direzione. Se la variabile è diversa da zero, l'uscita corrispondente è consentita e la routine la inserisce nella frase.

```
2355 IF (NR OR ES OR SD OR OV)=0 THEN RETURN
2360 PRINT: SN$ = "LE USCITE POSSIBILI SONO A "
2370 IF NR <>0 THEN SN$ = SN$ + "NORD "
2380 IF ES <>0 THEN SN$ = SN$ + "EST "
2390 IF SD <>0 THEN SN$ = SN$ + "SUD "
2400 IF OV <>0 THEN SN$ = SN$ + "OVEST "
2410 GOSUB 5500: REM FORMATTA LA FRASE
2415 PRINT
2420 RETURN
```

Dopo aver sviluppato le routine che descrivono i diversi ambienti, passiamo alle procedure che permettono al giocatore di muoversi al loro interno. Più avanti vengono esaminati in dettaglio gli algoritmi che analizzano le istruzioni. Per il momento vediamo come interpretare i comandi di movimento impartiti dal giocatore, ossia comandi del tipo "NORD", "SUD", ecc.. Queste istruzioni vengono passate alla seguente subroutine di movimento tramite la variabile NN$:

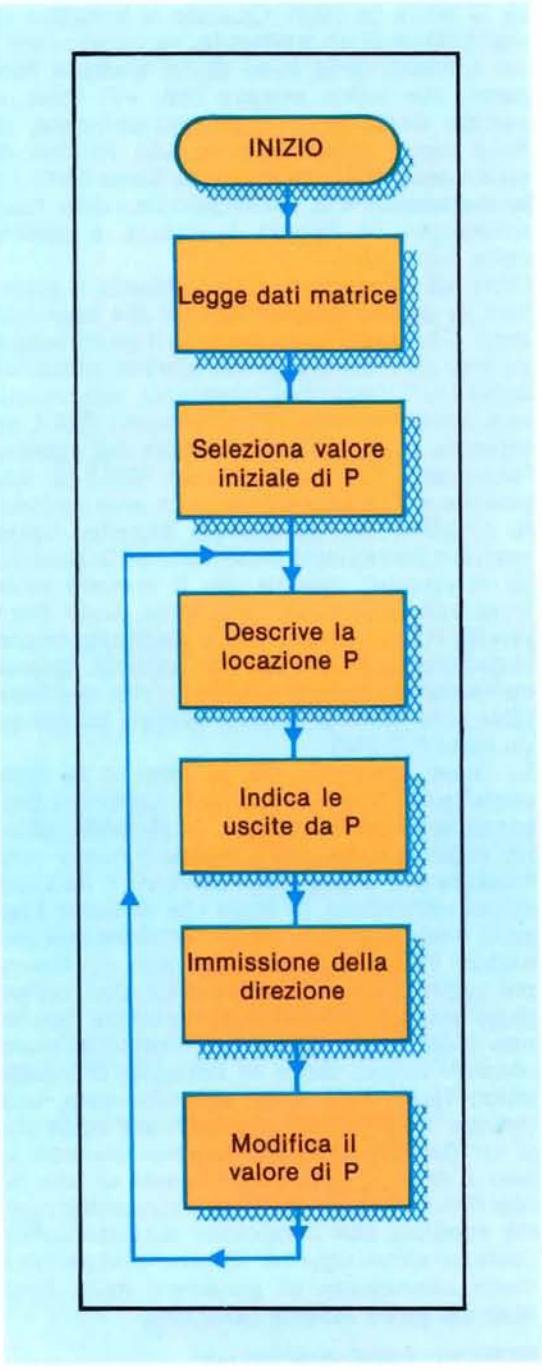
```
3500 REM **** S/R MOVIMENTO ****
3510 MF = 1: REM SELEZIONA IL FLAG DI MOVIMENTO
3520 DR$ = LEFT$(NN$,1)
3530 IF DR$ <>"N" AND DR$ <>"E" AND DR$ <>"S" AND DR$ <>"O" THEN
     GOTO 3590
3540 IF DR$ = "N" AND NR <>0 THEN P = NR: RETURN
3550 IF DR$ = "E" AND ES <>0 THEN P = ES: RETURN
3560 IF DR$ = "S" AND SD <>0 THEN P = SD: RETURN
3570 IF DR$ = "O" AND OV <>0 THEN P = OV: RETURN
3580 PRINT:PRINT "NON PUOI ANDARE A ";IS$
3585 MF = 0: RETURN
3590 REM ** COMANDO DI DIREZIONE ERRATO **
3600 PRINT "CHE SIGNIFICA ";NN$;" ?"
3610 MF = 0: RETURN
```

Questa routine usa solo la prima lettera del comando di direzione. Innanzi tutto verifica che l'istruzione ricevuta sia effettivamente un comando di direzione. Quindi verifica che quest'ultima corrisponda ad un'uscita consentita e, infine, modifica P (la variabile che registra la posizione corrente del giocatore) in base al valore di NR, SD, ES o OV.
Prima di poter usare le routine appena sviluppate, occorre collegarle in modo da creare un ciclo iterativo. Il diagramma di flusso in figura mostra la struttura logica di questo ciclo di chiamata. Sebbene questo non corrisponda alla struttura finale del ciclo principale del programma, consente tuttavia di evidenziare le caratteristiche del programma sin qui sviluppato. La sua esecuzione richiede l'immissione delle seguenti linee nel programma principale.

```
200 GOSUB 6000: REM LETTURA DATI NELLA MATRICE
210 P = INT(RND(TI)*10 + 1): REM PUNTO INIZIALE
230 REM **** INIZIO DEL CICLO PRINCIPALE ****
240 MF = 0: INIZIALIZZA IL FLAG
245 PRINT
250 GOSUB 2000: REM DESCRIVE UN AMBIENTE
255 GOSUB 2300: REM INDICA LE USCITE
260 PRINT:INPUT "COMANDO";IS$
```

Infine vanno inserite le linee seguenti nel ciclo principale di chiamata:

```
270 NN$ = IS$: GOSUB 3500: REM S/R MOVIMENTO
280 GOTO 230: REM RILANCIA IL CICLO PRINCIPALE
```



Modifiche per lo Spectrum

Poiché nello Spectrum ogni elemento di una matrice stringa ha una lunghezza fissa, possono sorgere problemi quando si desidera visualizzare un elemento di una matrice stringa all'interno di una frase. Sullo Spectrum, l'ultimo valore nell'istruzione DIM indica la lunghezza dei suoi elementi. Per esempio: DIM a$(3,2,20) dimensiona una matrice composta da 3x2 elementi, dove ciascuno di essi ha una lunghezza fissa di 20 caratteri. Se un elemento di tale matrice è composto da meno di 20 caratteri, gli spazi finali non occupati vengono riempiti con spazi e ciò comporta un notevole spreco di memoria. Inoltre, prima di inserire una variabile stringa dello Spectrum in una frase, gli eventuali spazi finali vanno rimossi.
Gli utenti Spectrum devono quindi introdurre nel listato di *Foresta Incantata* la seguente routine di cancellazione:

```
7000 REM **** CANCELLAZIONE PER SPECTRUM ****
7010 FOR I=LEN(A$) TO 1 STEP -1
7020 IF A$(I TO I)<>" " THEN LET N=I: LET I=1
7030 NEXT I
7040 LET S$=S$+A$(TO N)
7050 RETURN
```

La stessa cosa si deve fare per il programma *Digitaya*, usando però numeri di linea compresi tra 8500 e 8550.
Questa routine cancella gli spazi superflui di A$ prima di assegnarne il contenuto a S$. Si rammenti che S$ è la variabile stringa usata dal programma per formattare le frasi e che, prima di richiamare la routine, la stringa da trattare (quella estratta dalla matrice) va copiata nella variabile A$.
Pertanto, le modifiche da apportare alle versioni Spectrum dei programmi *Foresta Incantata* e *Digitaya* sono le seguenti:

Foresta Incantata:

```
2010 LET S$="TI TROVI ": A$=L$(P): GOSUB 7000: GOSUB 5500
2070 LET S$=S$+P$+"A": A$=V$(I,1): GOSUB 7000: LET F=1: LET P$=","
```

Digitaya:

```
1450 LET S$="TI TROVI ": A$=L$(P): GOSUB 8500: GOSUB 5880
1500 IF VAL(V$(I,2))=P THEN LET S$=S$+P$+"A":
     A$=V$(I,1): GOSUB 8500: LET F=1: LET P$=" "
```

Basic & Basic

Spectrum:

In entrambi i listati, sostituire EX$() con E$(), EX$ con X$, SN$ con S$, IS$ con T$, LN$() con L$(), NN$ con R$, SP$ con P$, DR$ con D$. Modificare il listato Digitaya con le seguenti linee:

```
1580 LET NR=VAL(X$(TO 2))
1590 LET ES=VAL(X$(3 TO 4))
1600 LET SD=VAL(X$(5 TO 6))
1610 LET OV=VAL(X$(7 TO))
2020 LET D$=R$(TO 1)
```

Nel listato di Foresta Incantata modificare le seguenti linee:

```
 210 RANDOMIZE: P=INT(RND(1)*10+1)
2320 LET NR=VAL(X$(TO 2))
2330 LET ES=VAL(X$(3 TO 4))
2340 LET SD=VAL(X$(5 TO 6))
2350 LET OV=VAL(X$(7 TO))
3520 LET D$=R$(TO 1)
```

Micro BBC:

Modificare il listato Foresta Incantata inserendo la linea:

```
210 P=RND(10)
```

LISTATO DEL PROGRAMMA DIGITAYA

La struttura di Digitaya è simile a quella di Foresta Incantata. Aggiungere le seguenti linee al listato dato in precedenza:

```
1100 GOSUB 6090: REM LEGGE I DATI DELLA MATRICE
1110 PRINT:INPUT "COMANDO";IS$
1120 P=47: REM PUNTO INIZIALE
1130 :
1140 REM **** INIZIO DEL CICLO PRINCIPALE ****
1150 :
1160 MF=0: PRINT
1170 GOSUB 1440: REM DESCRIVE UN AMBIENTE
1180 GOSUB 1560: REM ELENCA LE USCITE POSSIBILI
```

Inserire ancora le linee:

```
1220 NN$=IS$: GOSUB 2000: REM MOVIMENTO
1230 GOTO 1140: REM RILANCIA IL CICLO PRINCIPALE
```

DESCRIVE UN AMBIENTE E LE SUE USCITE

```
1440 REM **** DESCRIVE UN AMBIENTE ****
1450 SN$="TI TROVI "+LN$(P): GOSUB 5880
1460 SN$="VEDI "
1470 REM ** RICERCA DELL'OGGETTO **
1480 F=0:SP$=""
1490 FOR I=1TO8
1500 IF VAL(IV$(I,2))=P THEN
     SN$=SN$+SP$+"A"+IV$(I,1):F=1:SP$=" "
1510 NEXTI
1520 IF F=0 THEN SN$=SN$+"NESSUN OGGETTO"
1530 GOSUB 5880: REM FORMATTAZIONE
1540 RETURN
1550 :
1560 REM **** ELENCA LE USCITE POSSIBILI ****
1570 EX$=EX$(P)
1580 NR=VAL(LEFT$(EX$,2))
1590 ES=VAL(MID$(EX$,3,2))
1600 SD=VAL(MID$(EX$,5,2))
1610 OV=VAL(RIGHT$(EX$,2))
1620 IF (NR OR ES OR SD OR OV)=0 THEN RETURN
1630 PRINT:SN$="LE USCITE SONO A "
1640 IF NR <>0 THEN SN$=SN$+"NORD "
1650 IF ES <>0 THEN SN$=SN$+"EST "
1660 IF SD <>0 THEN SN$=SN$+"SUD "
1670 IF OV <>0 THEN SN$=SN$+"OVEST "
1675 GOSUB 5880: REM FORMATTAZIONE
1680 PRINT:RETURN
```

SUBROUTINE DI MOVIMENTO

```
2000 REM **** S/R MOVIMENTO ****
2010 MF=1: REM SELEZIONA IL FLAG DI MOVIMENTO
2020 DR$=LEFT$(NN$,1)
2030 IF DR$ <>"N" AND DR$ <>"E" AND DR$ <>"S"
     AND DR$ <>"O" THEN 2100
2040 IF DR$="N" AND NR <>0 THEN P=NR: RETURN
2050 IF DR$="S" AND SD <>0 THEN P=SD: RETURN
2060 IF DR$="E" AND ES <>0 THEN P=ES: RETURN
2070 IF DR$="O" AND OV <>0 THEN P=OV: RETURN
2080 PRINT "NON PUOI ANDARE A ";IS$
2090 MF=0: RETURN
2100 REM COMANDO DI DIREZIONE ERRATO
2110 PRINT "CHE SIGNIFICA ";NN$;" ?"
2120 MF=0: RETURN
```

DATABASE

Seconda scheda delle istruzioni per il 6502

CODICE MNEMONICO	DESCRIZIONE	R1 R2	IMPLIC.	IMMED.	PAGINA ZERO	PAG. Z X	PAG. Z Y	X INDIR.	Y INDIR.	INDIRIZZO RELATIVO	INDIRIZZO ASSOL.	X ASSOL.	Y ASSOL.	INDIRETTO	REGISTRO P						
															N	V	•	B	D	I	Z	C
JSR	SALTO ALLA SUBROUTINE	PC/S									20											
LDA	CARICA L'ACCUMULATORE	A		A9	A5	B5		A1	B1		AD	BD	B9		X						X	
LDX	CARICA IL REGISTRO X	X		A2	A6		B6				AE		BE		X						X	
LDY	CARICA IL REGISTRO Y	Y		A0	A4	B4					AC	BC			X						X	
LSR	SCORRIMENTO LOGICO A DESTRA	A/M	4A(A)		46	56					4E	5E			0						X	X
NOP	NESSUNA OPERAZIONE		EA																			
ORA	OR INCLUSIVO SULL'ACCUMULATORE	A		09	05	15		01	11		0D	1D	19		X						X	
PHA	DEPOSITA A SULLO STACK	S A	48																			
PHP	DEPOSITA IL REGISTRO P SULLO STACK	S P	08												X						X	
PLA	ESTRAE L'ACCUMULATORE DALLO STACK	A	68																			
PLP	ESTRAE IL REGISTRO P DALLO STACK	P	28												X	X	X	X	X	X	X	X
ROL	ROTAZIONE A SINISTRA DI UN BIT	A/M	2A(A)		26	36																
ROR	ROTAZIONE A DESTRA DI UN BIT	A/M	6A(A)		66	76					2E	3E			X						X	X
RTI	RITORNO DA INTERRUPT	P/	40								6E	7E			X						X	X
		PC/S													X	X	X	X	X	X	X	X
RTS	RITORNO DA SUBROUTINE	PC/S	60																			
SBC	SOTTRAZIONE CON CARRY	A		E9	E5	F5		E1	F1		ED	FD	F9		X	X					X	X
SEC	PONE IL FLAG DI CARRY A 1	P	38																			1
SED	PONE IL FLAG DECIMALE A 1	P	F8																1			
SEI	PONE LA MASCHERA PER INTERRUPT A 1 (DISABILITAZIONE)	P	78																	1		
STA	DEPOSITA L'ACCUMULATORE	M A			85	95		81	91		8D	9D	99									
STX	DEPOSITA IL REGISTRO X	M X			86		96				8E											
STY	DEPOSITA IL REGISTRO Y	M Y			84	94					8C											
TAX	TRASFERIMENTO A —> X	X A	AA												X						X	
TAY	TRASFERIMENTO A —> Y	Y A	A8												X						X	
TSX	TRASFERIMENTO S —> X	X S	BA												X						X	
TXA	TRASFERIMENTO X —> A	A X	8A												X						X	
TXS	TRASFERIMENTO X —> S	S X	9A																			
TYA	TRASFERIMENTO Y —> A	A Y	98												X						X	

Per concessione della Oric Products

Ultimi comandi

Aggiungiamo altri tre comandi al debugger

Per poter introdurre le breakpoint nei punti desiderati, le istruzioni del programma originale vengono sostituite con un'istruzione SWI (SoftWare Interrupt). Come le altre interrupt del 6809, la SWI viene vettorizzata mediante una specifica locazione di memoria, ossia $FFFA. Al momento della sua esecuzione, i registri vengono appoggiati sullo stack ed il processore carica nel contatore di programma (PC) l'indirizzo a 16 bit contenuto in $FFFA e $FFFB. Quindi l'esecuzione riprende a partire da quell'indirizzo.
Il nostro obiettivo è modificare questo vettore in modo che indichi il punto d'ingresso del programma di debugging. Il problema è che i vettori d'interruzione si trovano quasi sempre nella ROM, quindi i loro indirizzi non sono modificabili. Per questo motivo il sistema operativo deve seguire altre strade per vettorizzare le interrupt.
La tecnica più usata prevede l'impiego di una tabella di salti (vedere a pag. 1119), contenuta in un'area di lavoro della RAM, ossia della memoria abitualmente a disposizione dei programmi ma che, in particolari casi, viene riservata al sistema operativo. L'indirizzo puntato dal vettore contiene un'istruzione JMP seguita da un indirizzo che di solito punta nuovamente al sistema operativo. Questo indirizzo può tuttavia essere modificato in modo che la prima istruzione eseguita dopo l'interruzione del software sia un comando JMP per l'indirizzo iniziale del programma di debugging. Dobbiamo infine fare attenzione a reinserire il contenuto originale della tabella di salto, prima di uscire dal debugger, perché è sempre possibile che, in un momento successivo, il sistema operativo esegua un op-code SWI successivo. È bene ricordare che il 6809 possiede tre tipi di interrupt e che nulla vieta l'uso di SWI2 (op-code 10 3F e vettore alla locazione $FFF4) oppure di SWI3 (op-code 11 3F e vettore all'indirizzo $FFF2). L'op-code di queste istruzioni, però, occupa 2 byte, quindi implica alcune modifiche nel programma di debugging.
Un ulteriore problema è che questo programma può trovarsi in una qualunque area di memoria lasciata libera dal programma sottoposto a debugging, quindi deve essere "rilocabile". Si sarà notato che tutti i riferimenti alle locazioni di memoria del programma sono stati fatti usando la tecnica dell'indirizzamento relativo rispetto al contatore di programma. Tuttavia, in alcune circostanze, occorre conoscere l'indirizzo assoluto del punto d'ingresso del programma, in modo da poterlo inserire nella tabella di salto. Il calcolo di questo indirizzo deve avvenire durante l'esecuzione, dato che il programma assemblatore non può calcolarlo a priori.
Il nostro primo obiettivo è dunque quello di calcolare l'indirizzo ed inserirlo nella tabella di salto. Si rammenti che l'indirizzo del punto d'ingresso per la SWI è diverso dall'indirizzo iniziale del programma di debugging, perché la routine posta a questo indirizzo deve eseguire la procedura di inizializzazione, operazione non necessaria quando il programma viene reinserito tramite l'op-code SWI.
Di conseguenza, l'intera inizializzazione verrà controllata dall'interno di una subroutine; il punto d'ingresso corrisponde quindi all'indirizzo che contiene l'istruzione successiva alla chiamata BSR per la subroutine. Poiché questo indirizzo è esattamente quello appoggiato sullo stack dalla chiamata BSR, possiamo recuperarlo direttamente dallo stack ed inserirlo nella locazione di destinazione all'interno della tabella di salto.
Il secondo compito della procedura di inizializzazione è quello di fornire l'indirizzo iniziale del programma da sottoporre a debugging.

Procedura di inizializzazione

Dati:
- **Indirizzo-Vettore:** è l'indirizzo trovato alla locazione $FFFA, da porre in X
- **Opcode-JMP:** è l'op-code, in A, per l'istruzione JMP
- **Indirizzo-Ingresso:** è l'indirizzo del punto d'ingresso, in Y
- **Indirizzo-Iniziale:** è l'indirizzo iniziale del programma da sottoporre a debugging, in D

Procedura:
Preleva Indirizzo-vettore
Registra Opcode-JMP in Indirizzo-Vettore
Preleva Indirizzo-Ingresso
Lo deposita in (Indirizzo-Vettore + 1)
Accetta Indirizzo-Iniziale dalla tastiera
Salva Indirizzo-Iniziale

Passiamo adesso alla codifica degli ultimi tre comandi. Il comando R visualizza il contenuto dei registri. Ovviamente, non ci interessano i valori correnti presenti durante l'esecuzione del debugger, ma quelli del programma, contenuti nei registri quando si verifica una breakpoint. Quindi conviene visualizzare i valori appoggiati sullo stack dall'istruzione SWI.
Purtroppo, nel frattempo, sullo stack ne sono stati appoggiati altri: per recuperare i valori dei registri che ci interessano, possiamo calcolare e sottrarre il numero dei byte introdotti successivamente. Un'alternativa meno complicata è salvare il valore del puntatore

B (Breakpoint)	Inserimento breakpoint
U (Un-insert)	Eliminazione breakpoint
D (Display)	Visualizza le breakpoint attive
S (Start)	Avvia il programma sotto test
G (Go)	Riprende l'esecuzione dal punto d'interruzione
R (Registri)	Visualizza il contenuto dei registri
M (Memoria)	Esamina e corregge una locazione di memoria
Q (Quit)	Esce da programma di debugging

allo stack come prima operazione, dopo il verificarsi di una interrupt ed usarlo come valore di riferimento.
Il comando R presuppone che questa operazione sia già stata eseguita e che sia quindi possibile ritrovare il contenuto dei registri. La struttura della routine è molto semplice: recupera in sequenza i diversi valori (senza estrarli realmente dallo stack) e li visualizza con le appropriate etichette. Unica eccezione è il valore di S, ossia quello immediatamente precedente all'interrupt, ottenibile sommando il giusto valore al contenuto di S, precedentemente salvato, che la routine usa per calcolare i valori dei registri appoggiati sullo stack.

Comando R

Dati:
- **Puntatore-allo-Stack:** in X; è valore posto all'inizio dello stack dopo l'interrupt
- **Valore-ad-un-byte:** in B; contiene i valori dei registri ad un byte
- **Valore-a-due-byte:** in D; contiene i valori dei registri a due byte
- **Etichette:** contengono i nomi dei nove registri

Procedura:

Preleva il Puntatore-allo-Stack
Carica CC in Valore-ad-un-byte
Visualizza etichetta(1) e Valore-ad-un-byte
Ripete l'operazione precedente per A, B e DP
Carica X in Valore-a-due-byte
Visualizza etichetta(5) e Valore-a-due-byte
Ripete l'operazione precedente per Y, U e PC
Somma 12 al valore originale del Puntatore-allo-Stack
Visualizza etichetta(9) e Puntatore-allo-Stack

Restano ancora due comandi: Q, il comando di uscita dal programma, al quale non occorre una particolare routine e G, che permette di riprendere l'esecuzione del programma dopo una breakpoint. A questo punto dobbiamo sostituire l'istruzione SWI, che provoca l'interruzione, con l'op-code originale e trasferire il controllo a questa istruzione. Infine, possiamo ripristinare i valori originali dei registri con l'istruzione RTI, che li recupera tutti dallo stack.
Dobbiamo tuttavia ricordare che il contenuto recuperato di PC diviene il valore per l'istruzione successiva: poiché questo valore risulta maggiore di uno rispetto a quello richiesto, è necessario decrementarlo, prima di proseguire oltre.

Comando G

Dati:
- **Tabella-breakpoint:** è una tabella di indirizzi a 16 bit contenente le breakpoint
- **Valori-Rimossi:** è una tabella degli op-code sostituiti dalle istruzioni SWI
- **Breakpoint-Successiva:** è un numero compreso tra 1 e 16
- **Puntatore-allo-Stack:** è il valore del puntatore allo stack salvato dopo SWI

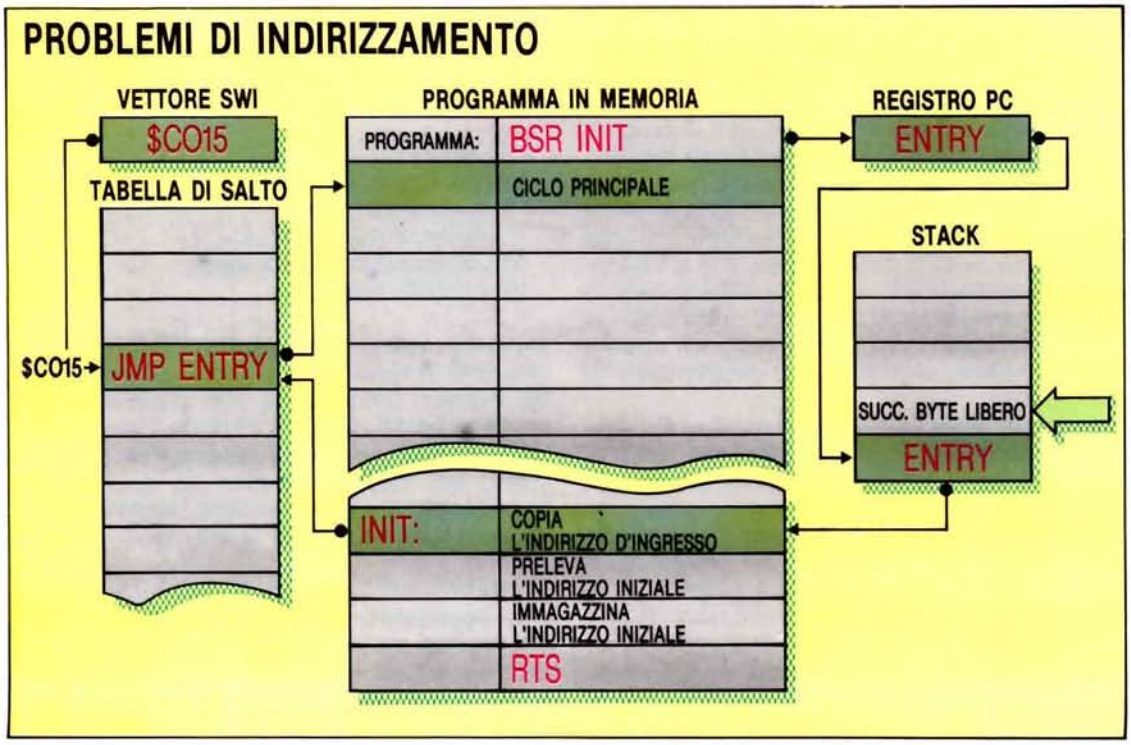
Procedura:

Se (IF) Breackpoint-Succ >0 e < = 16, allora (THEN)
  Accetta l'op-code proveniente da Valori-Rimossi (Numero-breakpoint)
  Lo immagazzina all'indirizzo in Tabella- Breakpoint (Breakpoint-Successiva)
  Carica in S il contenuto di Puntatore-allo-Stack
  Decrementa il valore di PC all'interno dello stack
  Incrementa Breakpoint-Successiva
  Ritorna dalla interrupt
altrimenti (ELSE)
  Ritorna dalla subroutine

Per completare il debugger (e la serie dedicata al codice macchina del 6809) mancano la descrizione per la codifica del modulo principale del debugger e quella del funzionamento globale del programma.

Frugare nello stack

Il programma di debugging comincia con una chiamata BSR alla routine d'inizializzazione, seguita dal codice del ciclo principale. L'inizializzazione, tra gli altri obiettivi, ha quello di calcolare l'indirizzo iniziale assoluto del ciclo principale e di copiarlo nella tabella di salto. In questo modo, dopo l'esecuzione dell'op-code-SWI, il controllo ritorna all'indirizzo iniziale del ciclo, passando attraverso la tabella di salto. Questo indirizzo non può essere calcolato in precedenza, perché il programma deve essere completamente "rilocabile" (ossia collocabile a qualsiasi indirizzo, in base alla lunghezza del programma sotto test). Fortunatamente, l'indirizzo di ritorno appoggiato sullo stack dall'istruzione BSR è proprio l'indirizzo assoluto cercato, quindi alla routine di inizializzazione non resta che copiarlo nella tabella di salto



PROCEDURA DI INIZIALIZZAZIONE

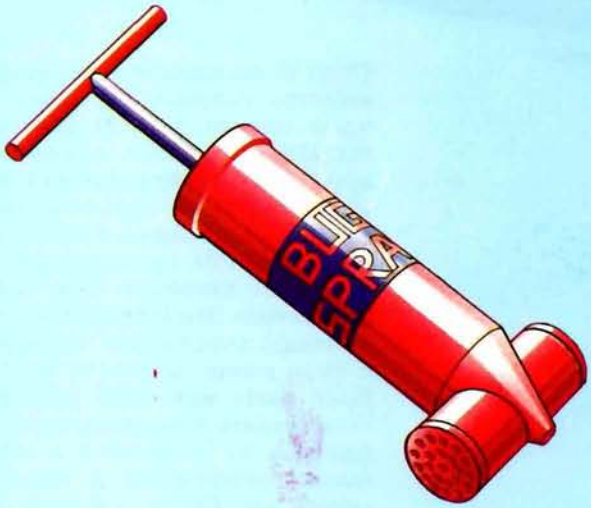
```
START   RMB   2              Per salvare l'indirizzo iniziale
OPJMP   FCB   $0E            Opcode-JMP
INIT    LDX   $FFFA          Preleva Indirizzo-Vettore
        LDA   OPJMP,PCR      Preleva opcode-JMP e lo deposita in
        STA   ,X+            Indirizzo-Vettore
        LDY   1,S            Estrae Indirizzo-Ingresso dallo stack
        STY   ,X             Lo salva in Indirizzo-Vettore + 1
        BSR   GETADD         Riceve Indirizzo-Iniziale dalla tastiera
        STD   START,PCR      Lo salva
        RTS                  Restituisce il controllo al ciclo princi-
                             pale
```

COMANDO R

```
STACKP  RMB   2              Puntatore allo Stack
LABELS  FCC   'CC A BDP X Y
              UPC S'
SPACE   FCB   32             Codice ASCII per "spazio"
CMDR    PSHS  A,B,X,Y        Salva i registri utilizzati
        LDX   STACKP, PCR    Preleva il Puntatore allo Stack
        LEAY  LABELS, PCR    Usa Y per puntare alle etichette
        LDA   #4             Numero dei registri ad un byte
FOR01   BSR   CMDR1          Visualizza il registro successivo
        DECA                 Ripete per quattro volte
        BGT   FOR01
        LDA   #4
FOR02   BSR   CMDR2          Numero dei registri a due byte
        DECA                 Visualizza il registro successivo
        BGT   FOR02          Ripete per quattro volte
        LDA   ,Y+            Primo carattere dell'etichetta
        BSR   OUTCH          Lo visualizza
        LDA   ,Y+            Secondo carattere dell'etichetta
        BSR   OUTCH          Lo visualizza
        LDA   SPACE,PCR      Inserisce uno spazio
        BSR   OUTCH
        TFR   X,D            X adesso contiene il valore di S ne-
                             cessario
        BSR   DSPADD         Visualizza S
        PULS  A,B,X,Y,PC     Ripristina i registri e ritorna
        *S/R per visualizzare registro ad un byte
CMDR1   PSHS  A              Salva A
        LDA   ,Y+            Primo carattere dell'etichetta
        BSR   OUTCH          Lo visualizza
        LDA   ,Y+            Secondo carattere dell'etichetta
        BSR   OUTCH          Lo visualizza
        LDA   SPACE,PCR      Inserisce uno spazio
        BSR   OUTCH
        LDB   ,X+            Copia il registro successivo in Valore-
                             ad-un-byte
        BSR   DSPVAL         Visualizza Valore-ad-un-byte
        PULS  A,PC           Ripristina A e ritorna
        *S/R per visualizzare registro a due byte
CMDR2   PSHS  A              Salva A
        LDA   ,Y+            Primo carattere dell'etichetta
        BSR   OUTCH          Lo visualizza
        LDA   ,Y+            Secondo carattere dell'etichetta
        BSR   OUTCH          Lo visualizza
        LDA   SPACE,P-CR     Inserisce uno spazio
        BSR   OUTCH
        LDD   ,X++           Copia il registro successivo in Valore-
                             a-due-byte
        BSR   DSPADD         Visualizza Valore-a-due-byte
        PULS  A,PC           Ripristina A e ritorna
```

COMANDO G

```
BPTAB   RMB   32             Tabella-Breakpoint
REMTAB  RMB   16             Valori-Rimossi
NEXTBP  RMB   1              Breakpoint Successiva
CMDG    PSHS  A              Salva A in caso di "ritorno" normale
        LDA   NEXTBP,PCR     Breakpoint Successiva
IF04    BLE   ENDF04         Se Breakpoint-Succ >0
        CMPA  MAXBP,PCR      e <=16
        BGT   ENDF04         (numero max. di breakpoint)
        DECA                 Converte A per usarlo come offset
                             nella tabella
        LEAX  BPTAB,PCR      Indirizzo di Tabella-Breakpoint
        LEAY  REMTAB,PCR     Indirizzo di Valori-Rimossi
        LDB   A,Y            Estrae Valori-Rimossi
        LSLA                 Converte A per usarlo come offset
                             nella tabella a 16 bit
        STB   [A,X]          Lo deposita all'indirizzo in Tabella-
                             Breakpoint
        LDS   STACKP,PCR     Copia Puntatore-allo-Stack in S
        DEC   10,S           Decrementa il valore di PC nello stack
        INC   NEXTBP,PCR     Incrementa Breakpoint-Successiva
        RTI                  Ritorna dopo l'interrupt
ENDF04  PULS  A,PC           Ripristina e ritorna
```



Starfinder

Il Micro BBC è l'ideale per la didattica: ecco un programma... astronomico!

Osservando il cielo ...

Lo Starfinder usa mappe celesti rettangolari o circolari a seconda che l'angolo di osservazione sia orizzontale o verticale rispetto all'orizzonte. Inoltre consente di rappresentare qualsiasi sezione del cielo visibile dalla Terra in qualunque giorno del XX Secolo. Permette infine di individuare corpi celesti di vario tipo, come stelle o pianeti, o anche la Cometa di Halley



Visione verticale



Visione orizzontale

Un problema comune a tutti gli astronomi, professionisti o dilettanti, è quello della trasparenza atmosferica. Spesso gli studiosi si preparano per settimane ad un particolare fenomeno astronomico ma, quando questo si verifica... il cielo è coperto di nubi. Per questa ragione la maggior parte degli osservatori si trova in località situate a rilevante altitudine, dove la presenza di nubi è meno frequente, o addirittura anche nello spazio.

La Century Software ha prodotto un package su cassetta, lo *Starfinder,* che permette di visualizzare su un piccolo schermo ogni settore della volta celeste visibile da qualunque punto d'osservazione della Terra fin dall'inizio del XX Secolo.

Dopo il caricamento del programma, sullo schermo compare un menu composto da varie opzioni: una di queste consente di visualizzare la volta celeste corrente (quale appare in un dato momento), mentre l'immagine selezionata automaticamente dal programma rappresenta il cielo notturno visibile da Londra alla mezzanotte (ora di Greenwich) del 21 novembre 1984. Detto per inciso, questa data non corrisponde ad un particolare evento astronomico, bensì al giorno in cui è stato messo in vendita il programma.

Sulla parte alta dello schermo compaiono l'ora, la data e la posizione dell'osservatore, espressa in longitudine e latitudine. Sotto queste informazioni è riprodotta la volta celeste visibile da sud-ovest a sud-est, con un'altezza "azimutale" di 60o. Il programma impiega alcuni secondi per collocare tutti i corpi celesti perché deve eseguire numerosi calcoli per elaborare la posizione di ciascun astro.

Le stelle sono rappresentate da piccoli quadrati bianchi (il programma usa il MODE 4 del Micro BBC), le cui dimensioni variano a seconda della loro brillantezza. È un peccato che i modelli Acorn non possiedano un comando di "brillantezza", che avrebbe reso le immagini molto più realistiche. I pianeti sono rappresentati da quadratini rossi, il sole da un grosso quadrato giallo e la luna da un punto dello stesso colore. Un piccolo puntatore rosso a forma di croce, controllato dai tasti per il cursore, permette di indicare e di identificare i diversi corpi celesti. Quando lo si posiziona su una particolare stella, il programma visualizza al di sopra della mappa stellare il suo nome comune e quello scientifico, indicandone le coordinate come altezza e "azimut" (l'altezza viene espressa con valori positivi o negativi rispetto all'orizzonte, che vale 0. Il valore azimutale indica il numero di gradi verso est o verso ovest rispetto al nord). La posizione dell'osservatore può essere cambiata a piacimento, agendo sulla tastiera.

Ritornato al menu principale, l'utente può cambiare la sezione di cielo visualizzata con quella visibile in un qualsiasi momento da un qualunque punto della Terra. Inoltre, può individuare uno specifico corpo celeste, sia esso una stella, un pianeta oppure la Cometa di Halley.

Pur essendo un programma astronomico abbastanza completo, lo *Starfinder* ha alcune limitazioni. Il suo creatore, Robert Alpiar, ha preso in considerazione solo le stelle con magnitudo pari o inferiore a 4 (la magnitudo è un parametro che indica la luminosità dei corpi celesti ed è inversamente proporzionale alla brillantezza delle stelle). Ad occhio nudo si riescono a vedere stelle di magnitudo 6, quindi la mappa celeste riprodotta sullo schermo non contiene tutte le stelle osservabili in una notte limpida.

Il manuale abbinato alla cassetta, oltre a spiegare il funzionamento del programma, offre agli astronomi dilettanti una serie di interessanti informazioni che riguardano i diversi tipi di telescopi oggi in funzione ed i migliori periodi per l'osservazione di stelle e pianeti.

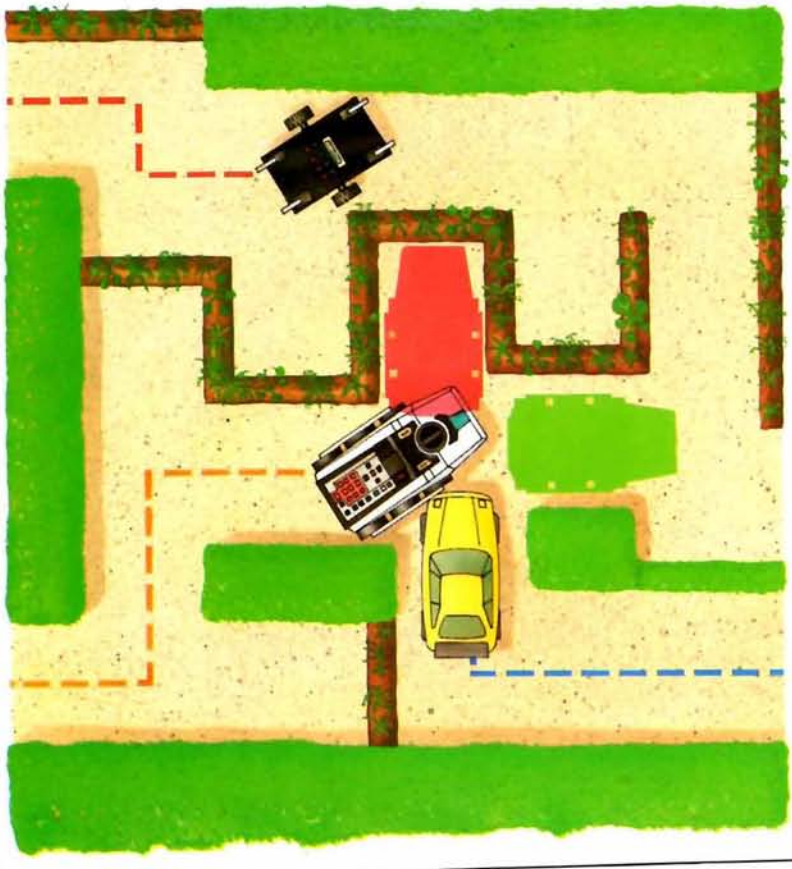
Starfinder Per il Micro BBC e l'Electron
Editori: Century Software, (GB)
Autori: Il manuale è di Heather Couper, il programma di Ronald Alpiar
Joystick: Non necessario
Formato: Su cassetta

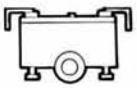
Piccoli robot

Cosa ha offerto, inizialmente, il mercato dei minuscoli apparecchi robotizzati

Gli argomenti fin qui trattati riguardano alcune considerazioni teoriche sulla progettazione e sul funzionamento dei robot. Purtroppo la realizzazione pratica di molti dei progetti esposti, a livello amatoriale, è spesso troppo complessa o eccessivamente costosa; inoltre gli odierni robot, sia quelli impiegati nell'industria sia quelli per usi domestici, mostrano capacità notevolmente inferiori rispetto a quelle ipotizzate in passato. Esistono sensori che permettono di udire suoni, vedere oggetti e toccarli, ma queste percezioni in realtà non hanno alcun significato intrinseco per i robot né possono essere sintetizzate in modo da sviluppare in essi un comportamento autonomo e non programmato. Robbie e altre macchine che popolano la letteratura fantascientifica fanno ancora parte di un mondo lontano dalla realtà!
Ciò nonostante, il nome "robot" viene spesso usato per molti prodotti in commercio: dai piccoli modelli di automobili giocattolo ai costosissimi robot industriali. Dopo aver descritto gli elementi costitutivi di un robot ed i principi base di numerose applicazioni, tentiamo di tracciare con precisione la "carta d'identità" di un vero robot. La prima considerazione, che permette di scartare molti dei prodotti a basso prezzo impropriamente etichettati come robot, riguarda il movimento. Non possiamo certo chiedere ad un robot di autoprogrammarsi o di eseguire una serie di movimenti senza intervento umano; tuttavia, una volta ricevute le necessarie istruzioni, il robot deve essere in grado di agire in totale autonomia. In mancanza di ciò, anche l'oggetto più sofisticato non può essere chiamato robot.
Dopo aver superato il test di movimento, il nostro ipotetico robot deve essere esaminato in base alle caratteristiche del movimento. Per muoversi in linea retta, un piccolo modello d'automobile può sfruttare un motore a pile. Speciali paraurti possono consentirgli di cambiare direzione quando incontra un ostacolo oppure grosse ruote di gomma, abbinate a un ingegnoso posizionamento del centro di

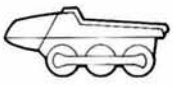
QUAL È IL VERO ROBOT?





Robot

Il robot progettato e realizzato nella sezione "Laboratorio", è alimentato e controllato da un computer, oltre a possedere sensori luminosi e di contatto



Big Trak

Questo piccolo veicolo è provvisto di una tastiera che consente di programmare il processore interno con istruzioni di tipo LOGO sulla distanza e sulla direzione



Bumper Car

È un modellino d'automobile a pile, che si muove solo in linea retta. Quando urta un ostacolo, ruota di 90° e riprende la corsa

Campo di gara

I tre apparecchi cercano di attraversare un labirinto: l'automobile a pile si muove "alla cieca", di ostacolo in ostacolo; il Big Trak segue invece le istruzioni programmate dall'operatore umano; il robot esplora il labirinto integrando i segnali dei sensori con il proprio software di controllo. Non c'è dubbio che il robot vincerà la gara, indipendentemente da ciò che accade all'interno del labirinto, mentre il Big Trak riuscirà ad uscire solo se le istruzioni dell'operatore sono esatte. L'automobile può "risolvere" solo labirinti con incroci ad angolo retto ed anche questa eventualità è piuttosto remota.
Quando l'automobile urta il Big Trak, non subisce alcuna conseguenza dato che i suoi movimenti sono casuali. Al contrario il Big Trak viene ruotato di 90° rispetto al percorso previsto (sagoma in verde) ma continua a muoversi secondo il programma e termina così in un vicolo cieco (sagoma in rosso). Entrambi gli apparecchi rispondono in modo "non intelligente" all'evento fortuito, mentre il robot è in grado di adeguarsi rapidamente alla nuova situazione

STEVE CROSS

gravità, gli permettono di "scalare" gli ostacoli o di ribaltarsi completamente, tornando poi sulle proprie ruote. Un simile modellino è capace di muoversi in modo autonomo e indipendente dal controllo umano, ma non può essere considerato un robot, per i motivi che ora cercheremo di approfondire.

Come abbiamo visto, il movimento di un robot può essere di due tipi (entrambi dipendenti dalla programmazione impiegata): controllato da computer o "intelligente". Il modellino appena descritto non rientra nella categoria "robot" in quanto il suo movimento non può essere programmato: esso segue una serie di spostamenti predeterminati meccanicamente, ma non è in grado di ricevere comandi umani che ne modifichino la direzione. Un'interessante variante è costituita da quei modelli a motore (autocarri o altro), programmati inserendo una piccola scheda perforata in un apposito lettore incorporato. La scheda viene letta meccanicamente e la direzione del movimento cambia in risposta alle istruzioni introdotte. In base alla nostra definizione, queste macchine possono già essere considerate robot, poiché vengono controllate da un "programma". Il concetto di movimento programmabile esclude quindi molti apparecchi spacciati come robot.

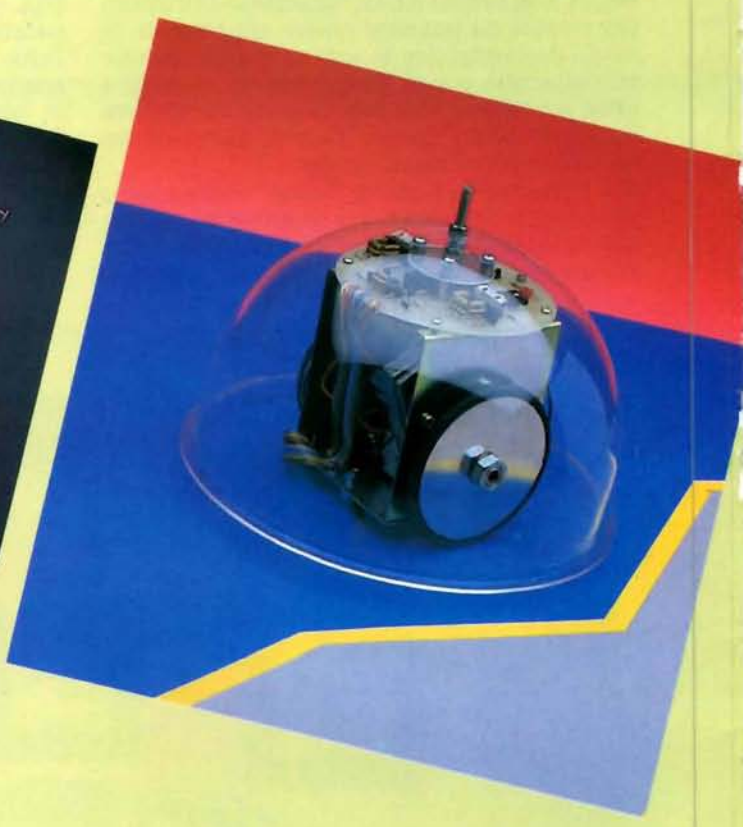
La discriminazione potrebbe proseguire con considerazioni d'altro genere quali, ad esempio: la presenza di sensori, che permettono alla macchina di ricevere informazioni dal mondo esterno; la capacità di esplorare l'ambiente circostante e di riprodurlo in un proprio modello interno modificabile; la possibilità, infine, di "ragionare" in un gioco quale quello degli scacchi. Tuttavia, questi ed altri criteri hanno un'importanza secondaria: la caratteristica determinante è sempre il movimento programmabile.

Qui di seguito vengono esaminate le caratteristiche di alcuni dei primi e più popolari robot sul mercato.



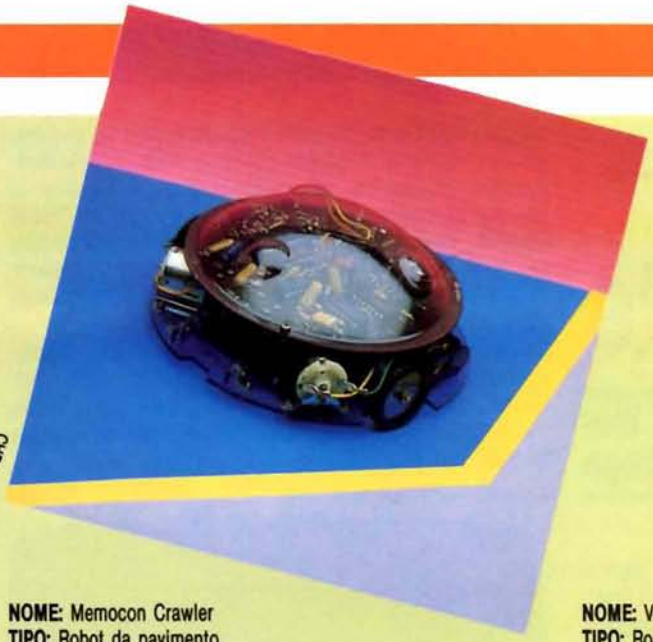
NOME: Beasty
TIPO: Braccio robot
PROGRAMMABILE: Sì
CORREDATO DI SOFTWARE: Sì
SENSORI DI FEEDBACK: Sensori di posizionamento

Il Beasty riceve i comandi tramite la porta utente del Micro BBC e la corrente dalla presa ausiliaria del BBC. Viene venduto in scatola di montaggio e possiede tre servomotori. È corredato di un opuscolo di assemblaggio e di un manuale di funzionamento. Il Beasty possiede un proprio sistema operativo, il Robol, che permette di controllare i servomotori in modo indipendente (per informazioni più particolareggiate vedere a pag. 1250).



NOME: Hebot II
TIPO: Robot da pavimento
PROGRAMMABILE: Sì
CORREDATO DI SOFTWARE: Sì
SENSORI DI FEEDBACK: Sensori di contatto

Hebot II è una tartaruga-robot azionata da motori a corrente continua, collegati a due ruote. È interfacciata per l'uso con lo ZX81, anche se i suoi fabbricanti affermano che, con piccole modifiche, può essere collegata a qualsiasi microcomputer. L'Hebot II viene venduta in scatola di montaggio assieme ad un manuale di assemblaggio. Possiede una penna retrattile e quattro sensori di contatto per l'individuazione degli ostacoli. La corrente di alimentazione proviene direttamente dal computer



CHRIS STEVENS

NOME: Memocon Crawler
TIPO: Robot da pavimento
PROGRAMMABILE: Sì
CORREDATO DI SOFTWARE: No
SENSORI DI FEEDBACK: No

Il Memocon Crawler è senza dubbio il robot più sofisticato della gamma Movits della Prism. Questa tartaruga viene alimentata da pile a 1,5 volt che azionano due motori elettrici a corrente continua. La tastierina di controllo, collegata al robot da un cavo a nastro, possiede cinque tasti, uno per ogni comando. Il Memocon Crawler incorpora anche una serie di LED e un "cicalino" acustico, il tutto controllabile con la tastierina



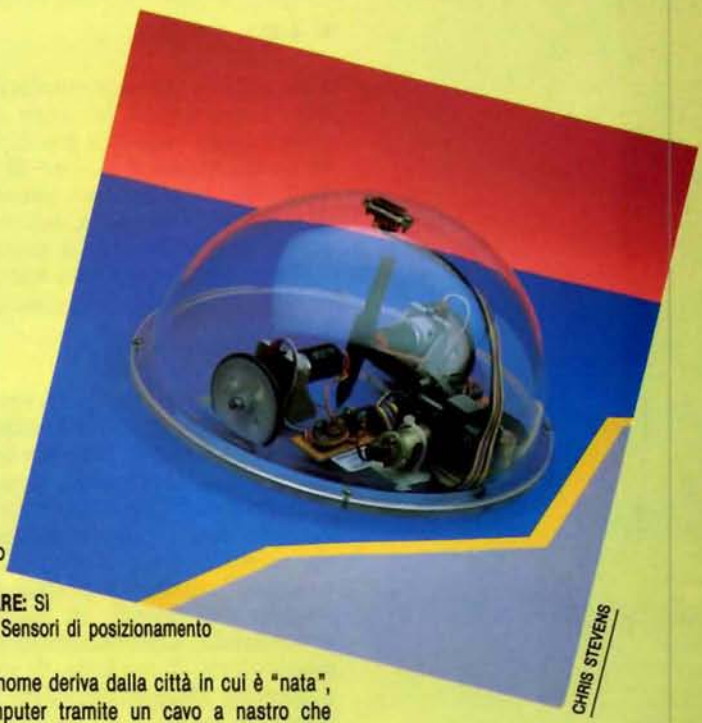
NOME: Valiant Turtle
TIPO: Robot da pavimento
PROGRAMMABILE: Sì
CORREDATO DI SOFTWARE: Sì
SENSORI DI FEEDBACK: Sensori di posizionamento

La Valiant Turtle ha l'aspetto di una vera e propria tartaruga. Viene mossa da una coppia di motori elettrici, ciascuno dei quali aziona una ruota ed è controllata dal computer per mezzo di un fascio luminoso a raggi infrarossi. Il software di corredo è stato progettato per lavorare con il LOGO, ma può funzionare anche in assenza del linguaggio di comando. È alimentata da una batteria interna ricaricabile. Possiede infine una penna retrattile per tracciare linee durante il movimento (vedere a pag. 1012)



NOME: Buggy BBC
TIPO: Robot da pavimento
PROGRAMMABILE: Sì
CORREDATO DI SOFTWARE: Sì
SENSORI DI FEEDBACK: Sensori luminosi, cuscinetti di contatto

La Buggy BBC è una tartaruga dotata di un proprio software ed è usata come "sussidio didattico" in molte scuole inglesi. Abitualmente venduta in scatola di montaggio, si collega alla porta utente del Micro BBC e riceve corrente dalla presa ausiliaria di alimentazione del computer; usa una coppia di motori passo passo, che consentono una grande precisione di movimento e, grazie ai sensori luminosi, può individuare e rispondere ad un segnale luminoso. Inoltre, può azionare una penna e controllare un lettore di codici a barre



CHRIS STEVENS

NOME: Edinburgh Turtle
TIPO: Robot da pavimento
PROGRAMMABILE: Sì
CORREDATO DI SOFTWARE: Sì
SENSORI DI FEEDBACK: Sensori di posizionamento

Questa tartaruga il cui nome deriva dalla città in cui è "nata", viene collegata al computer tramite un cavo a nastro che fornisce anche la corrente d'alimentazione. Le due ruote sono azionate da una coppia di motori elettrici a corrente continua. Possiede inoltre una penna di tracciamento ed un piccolo altoparlante interno. La ditta produttrice, la Jessop (GB), ha realizzato anche un altro modello della tartaruga, dotato di controllo a distanza (vedere a pag. 1012)

Indovinare ragionevolmente

Concludiamo l'esame del "TK!Solver" illustrando altre utili prestazioni del package

Come già accennato nella precedente parte dedicata al *TK!Solver* (vedere a pag. 1284), questo package spinge il concetto di lavagna elettronica ai più alti limiti della matematica e dell'ingegneria. Abbiamo visto come esso permetta di definire le variabili associando loro nomi e come si usano in complesse equazioni. In questa seconda parte esaminiamo un'altra interessante caratteristica del *TK!Solver*: l'abilità di eseguire iterazioni. In sostanza, si tratta di un metodo per valutare la soluzione di un'equazione per successive approssimazioni, ossia tirando a indovinare, ragionevolmente, quale possa essere il risultato. Normalmente, lavorando sulle equazioni, i valori delle variabili vengono determinati tramite le informazioni fornite, che devono essere sufficienti. Per esempio, l'equazione:

$A^2 + B^2 = 2 \cos Y$

è risolvibile per una qualsiasi delle tre variabili, conoscendo le altre due. Fornendo i valori delle variabili A e B, il *TK!Solver* trova in pochi attimi il valore di Y.
Tuttavia esistono occasioni nelle quali la determinazione della soluzione non è altrettanto immediata. È il caso delle equazioni "ridondanti", in cui una variabile è definita in base a se stessa. Supponiamo di avere:

$D = (A+B)/(2*D)$

e di conoscere soltanto il valore di A: azzardare una soluzione non è affatto facile! Problemi simili possono sorgere quando si elabora un modello incompleto oppure composto da innumerevoli variabili interdipendenti.
Poiché il metodo iterativo non è tra i più facili da comprendere, conviene ricorrere ad un esempio pratico. Riprendiamo in esame l'esempio del viaggio in automobile, usato in precedenza, arricchendolo con qualche altro particolare. Il modello, si rammenterà, comprendeva cinque variabili: distanza, tempo, velocità, carburante e consumo. Conoscendo la velocità e il tempo potevamo calcolare, ad esempio, il carburante usato. Oppure la distanza, conoscendo la velocità ed il tempo e via dicendo. Ma come dovremmo comportarci per determinare a quale velocità guidare per completare un viaggio entro un ben preciso limite di spesa?
Tanto per cominciare, al modello originale vanno aggiunti diversi fattori, quali la potenza del motore, l'attrito interno (che ai fini del nostro esempio consideriamo costante) e quello dovuto alla resistenza dell'aria. Tutti questi elementi influiscono sul rendimento globale del veicolo. Inoltre, dobbiamo fissare il "tetto" della spesa per il viaggio e conoscere il costo del carburante.
Iniziamo dalla costruzione del modello, immettendo le equazioni occorrenti, una per linea, nel foglio delle regole (Rule sheet). Le variabili vengono automaticamente riprodotte nel Variable sheet:

Definizione delle equazioni

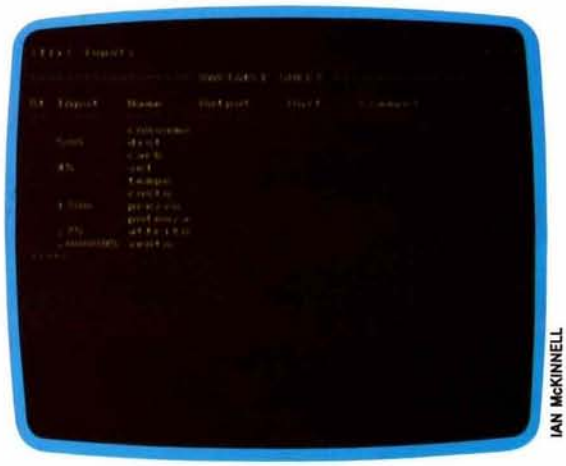


Poiché le variabili sono numerose, lo schermo non riesce a contenerle tutte. Volendo esaminarle in modo globale, si preme il "punto e virgola" (;) (per portare il cursore sul foglio delle variabili) e si digita W1: al foglio delle variabili viene così dedicato l'intero schermo. Adesso possiamo immettere i valori per ciascuna variabile.

Soluzione diretta

Il modello può essere immediatamente adoperato per trovare alcune soluzioni, purché immettiamo dati sufficienti nella colonna "Input". Ecco un esempio:

Immissione dei valori



IAN McKINNELL

L'operazione di calcolo si avvia premendo il tasto "!". Come al solito compare la scritta

«Direct solver» e, poco dopo, i valori calcolati (nella colonna "Output")

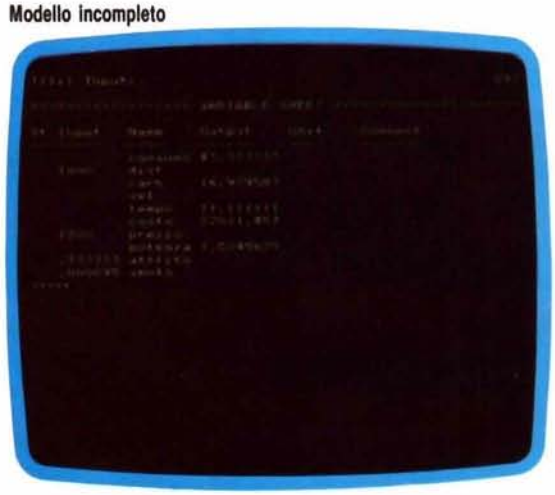
Risultati del calcolo



Ciò fornisce un'utile suddivisione di tutti gli elementi e delle varie soluzioni, ma come fare se si parte con una quantità di dati meno completa? Supponiamo di avere un tetto di 30.000 lire da spendere per un viaggio di 1.000 chilometri: conoscendo il costo della benzina è facile calcolare quanto possiamo spendere al chilometro, mentre è molto più difficile stabilire la velocità di crociera che consente di concludere il viaggio pur restando entro il tetto stabilito.

Anzitutto bisogna azzerare i valori immessi in precedenza: la sequenza di comandi è RVY (per Reset Variables Yes). Poi immettiamo i dati noti: 1.000 per distanza, 30.000 per costo e 1.300 per prezzo. Usiamo il valore frazionario 1/3 (che *TK!Solver* converte in 0,333333) per l'attrito interno e 0,0000095 per la resistenza dell'aria. Adesso si preme ! per avviare il calcolo. Dopo poco sullo schermo appare:

Modello incompleto



Si noti che non sono stati generati valori per la velocità, né per il tempo, né per la potenza, mentre la velocità è proprio l'incognita cercata. Se ci spostiamo sul foglio delle regole, si vede immediatamente che tre delle equazioni rimangono insoddisfatte (ciò viene segnalato da un asterisco nella colonna dello stato, "St").

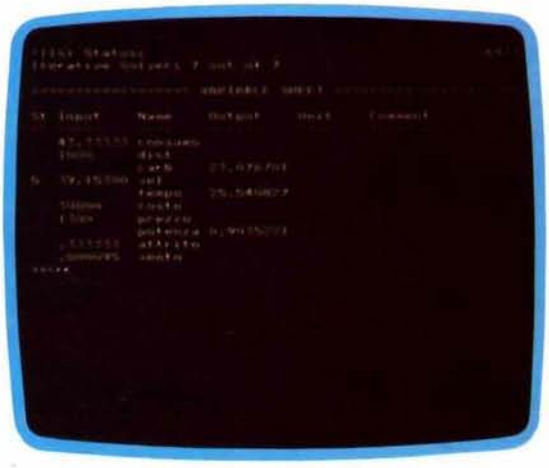
Equazioni non soddisfatte



Soluzione per iterazione

L'impiego del Direct solver, non conduce al risultato voluto: va invece adoperata l'opzione Iterative Solver. In questo caso viene immesso un valore di partenza approssimativo e il programma tenta di risolvere il modello in base ad esso. Se il valore non risulta esatto, il *TK!Solver* ricorre ad una serie di approssimazioni, fino alla soluzione. Occorre spostare il valore del consumo, calcolato precedentemente, dalla colonna "Output" a quella "Input": ciò fornisce un elemento in più per il calcolo. Si digita quindi una I nella colonna dello stato ("St"), a fianco del consumo, nella pagina delle variabili. Poi, nella colonna "Input", a fianco della velocità, va immesso il valore approssimativo, ad esempio: 50 km/h. Nella corrispondente colonna dello stato si digita una G (iniziale di "Guess", ossia "indovina") e, infine, si preme ! per avviare il calcolo. Stavolta sullo schermo compare la scritta Iterative Solver ed un conteggio delle iterazioni eseguite, che dipende dalla precisione del valore approssimativo immesso. Infine compaiono i valori per la velocità, il tempo e la potenza:

Calcolo per iterazione



Il *TK!Solver* è per: l'Apple II, il PC della IBM (e compatibili) e l'Apricot della ACT.

Accettare ordini

Il gioco d'avventura deve saper interpretare i comandi impartiti dal giocatore

Di solito i giochi del genere adventure sono concepiti in modo che il giocatore possa spostarsi da un ambiente all'altro, raccogliendo e depositando oggetti di vario genere lungo il percorso. A questo scopo occorrono appositi comandi, che nel gioco qui sviluppato sono i seguenti:

VAI (direzione)	Per muoversi tra gli ambienti
PRENDI (oggetto)	Per raccogliere un oggetto
LASCIA (oggetto)	Per depositare un oggetto
ELENCO	Per elencare gli oggetti trasportati
GUARDA	Per visualizzare di nuovo la descrizione dell'ambiente attuale
FINE	Per uscire dal gioco

Ovviamente si possono anche aggiungere variazioni del tipo MUOVI per VAI oppure RACCOGLI per PRENDI. Le modifiche necessarie sono talmente semplici che il repertorio di comandi è facilmente ampliabile. Un comando utile in un'avventura può essere NUOTA, ma occorre che il programma visualizzi un messaggio del tipo «Non puoi nuotare qui», se l'ambiente visitato è privo di fiumi o laghi. In alternativa, un tale comando può essere inserito per fuorviare il giocatore, facendogli credere che in qualche ambiente vi sia un corso d'acqua. Il numero di comandi accettati dal programma dipende dalla complessità del gioco e dall'abilità del programmatore nel prevedere ogni possibile eventualità. È importante, in ogni caso, accertarsi che l'immissione di un comando non previsto non faccia bloccare l'intero programma. Talvolta basta una piccola subroutine che, collocata al posto giusto, visualizzi il messaggio «Non capisco il comando». Comunque, va prevista una certa elasticità nell'immissione dei comandi: sarebbe molto fastidioso, infatti, che il programma rispondesse «Non capisco» se, anziché impartire il comando PRENDI LAMPADA, si scrive PRENDI LA LAMPADA. Più avanti viene spiegato come rendere più flessibile il gioco: prima conviene sviluppare una routine in grado di suddividere e interpretare facilmente i comandi impartiti dal giocatore.

Suddivisione dei comandi

Qualunque sia il comando immesso, è molto probabile che la frase sia di tipo imperativo, ossia del tipo VAI A SUD VERSO IL FIUME oppure CATTURA UN ALIENO. Il vantaggio di una simile struttura è che, di solito, il verbo è sempre nella prima posizione, l'oggetto subito dopo e, infine, ci può essere una descrizione per qualificare l'azione da intraprendere. Il primo passo consiste dunque nel separare il verbo dal resto. Questa operazione è semplice se si usa la funzione MID$ (o una equivalente), con la quale si cerca un carattere spazio di separazione tra le parole. Individuato il primo spazio, è ragionevole supporre che la parola alla sua sinistra sia il verbo, che viene assegnato alla variabile VB$. La restante parte della frase viene invece assegnata alla variabile NN$. Questa è la routine di *Foresta Incantata* che si occupa della suddivisione della frase, contenuta nella variabile IS$:

```
2500 REM *** S/R DI SUDDIVISIONE COMANDO ***
2510 IF IS$="ELENCA" OR IS$="FINE" THEN VB$ = IS$:F =
     1:RETURN
2515 IF IS$="GUARDA" THEN VB$=IS$:F=1:RETURN
2520 F=0
2530 LS=LEN(IS$)
2540 FOR C=1 TO LS
2550 A$=MID$(IS$,C,1)
2560 IF A$<>" " THEN 2590
2570 VB$=LEFT$(IS$,C—1):F=1
2580 NN$=RIGHT$(IS$,LS—C):C=LS
2590 NEXT C
2600 :
2610 IF F=1 THEN RETURN
2620 PRINT:PRINT "OCCORRONO ALMENO DUE PAROLE"
2630 RETURN
```

Prima di tentare la suddivisione, la routine verifica se il comando è uno tra quelli composti da una singola parola (nel nostro caso: ELENCA, FINE o GUARDA). In tal caso il comando viene direttamente assegnato a VB$ e la routine termina, altrimenti inizia un ciclo FOR...NEXT che serve per individuare il primo spazio nella frase. Qui vengono adoperate due tecniche che meritano un po' d'attenzione. Entrambe concernono il fatto che saltar fuori da un ciclo, senza passare per la NEXT associata a FOR, è una pessima abitudine di programmazione. Per segnalare che si è verificato un certo evento (in questo caso l'individuazione di uno spazio) viene impiegata, come flag, la variabile F. In secondo luogo, continuare la scansione della frase una volta trovato il primo spazio, rappresenta un'inutile perdita di tempo: il ciclo può essere terminato molto elegantemente assegnando al contatore del ciclo, C, il valore massimo previsto, ossia LC. Ciò implica che il programma esce tranquillamente (e in modo corretto) dal ciclo quando raggiunge la NEXT. A questo punto si può

verificare il valore di F. Se F vale 1 vuol dire che la frase è composta da più di una parola e si può tornare al programma chiamante. Se invece F vale 0 la frase è composta da una sola parola, che non corrisponde però a nessuno dei comandi verificati all'inizio, cosicché viene visualizzato un avviso in tal senso, prima di eseguire RETURN.

Comandi normali

Per quanto riguarda la parte principale dell'avventura, il giocatore si muoverà semplicemente da un luogo all'altro, raccogliendo o depositando oggetti. Per la maggior parte degli ambienti, dunque, i comandi VAI, PRENDI, LASCIA, ELENCA, GUARDA e FINE (ed eventuali loro varianti) sono più che sufficienti. La struttura del programma può quindi essere concepita in modo da controllare soltanto i suddetti comandi, salvo il caso in cui si entri in un ambiente marcato come "speciale". In tal caso si può prevedere l'impiego di routine "ad hoc" per l'interpretazione di comandi particolari. Il ciclo principale di chiamata dovrebbe quindi avere la seguente struttura:

1) Descrivere l'ambiente ed elencare le uscite.
2) Determinare se l'ambiente è speciale.
3) Accettare un comando e, se il luogo non è speciale, verificare solo i comandi normali.

Inoltre, il ciclo principale deve essere capace di distinguere quali comandi implicano un movimento verso un nuovo ambiente e quali no. Nel primo caso occorre tornare indietro per descrivere il nuovo ambiente (e verificare se questo è speciale), nel secondo è sufficiente ripetere la richiesta per un nuovo comando. Questo problema si risolve semplicemente: basta usare una variabile come flag di movimento (MF nel nostro caso). Questa variabile, che è posta a 1 se il comando implica un movimento (altrimenti vale zero), viene esaminata al termine del ciclo principale. Ecco la sezione di programma da aggiungere a *Foresta Incantata:*

```
270 GOSUB 2500:REM SEPARAZIONE COMANDI
275 IF F=0 THEN 260:REM COMANDO NON VALIDO
280 GOSUB 3000:REM COMANDI NORMALI
290 IF VF=0 THEN PRINT:PRINT"NON HO CAPITO"
300 IF MF=1 THEN 240:REM NUOVO AMBIENTE
310 IF MF=0 THEN 260:REM NUOVO COMANDO

3000 REM *** S/R COMANDI NORMALI ***
3010 VF=0:REM FLAG DEL VERBO
3020 IF VB$="VAI" OR VB$="MUOVI" THEN VF=1 :GO-
     SUB3500
3030 IF VB$="PRENDI" OR VB$="RACCOGLI" THEN VF=1
     :GOSUB3700
3040 IF VB$="LASCIA" OR VB$="DEPOSITA" THEN VF=1
     :GOSUB3900
3050 IF VB$="ELENCA" OR VB$="INVENTARIO" THEN VF=1
     :GOSUB 4100
3055 IF VB$="GUARDA" THEN VF=1:MF=1:RETURN
3060 IF VB$="FINE" OR VB$="BASTA" THEN VF=1 :GOSUB
     4170
3070 RETURN
```

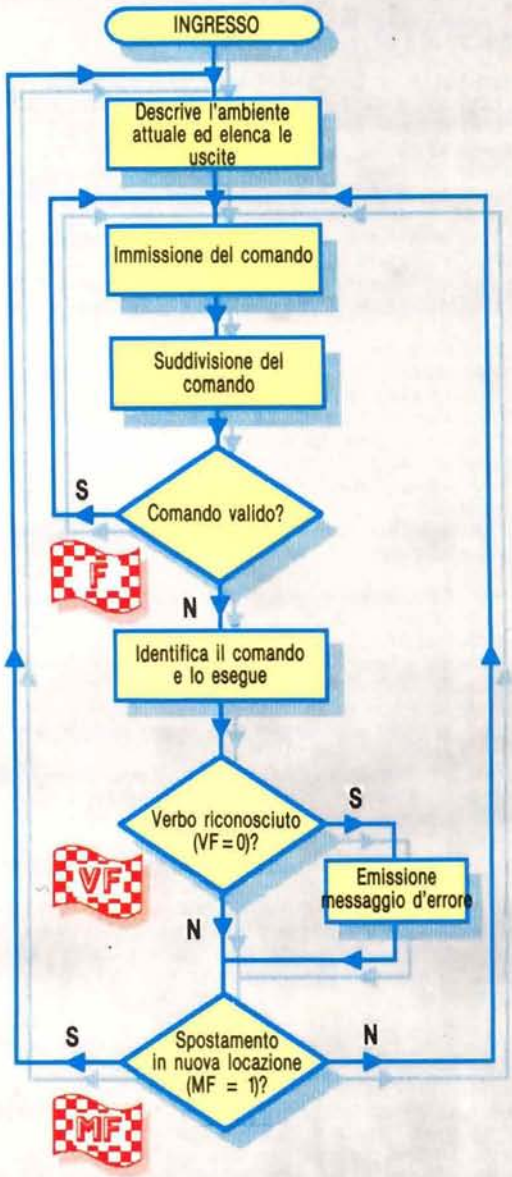
Nella prima routine VF viene usata come flag per indicare se il comando è stato accettato e compreso: soltanto quando il verbo è stato "isolato" VF viene posta a 1. Nel ciclo principale basta verificare se il valore di VF è zero, per decidere se visualizzare il messaggio «Non ho capito».

Le routine per raccogliere e depositare oggetti sono descritte più avanti: per adesso aggiungiamo la breve subroutine che risponde al comando FINE:

```
4170 REM *** S/R DI FINE GIOCO ***
4180 PRINT :PRINT"SEI SICURO (S/N) ?"
4190 GET A$:IF A$<>"S" AND A$ <> "N" THEN 4190
4200 IF A$ ="N" THEN RETURN
4210 END
```

BANDIERINE SUL PERCORSO

I flag (letteralmente: bandiere) sono ampiamente adoperati nei programmi con struttura modulare, poiché consentono di segnalare il verificarsi di un particolare evento in una subroutine, per poi eseguire una decisione, in base al valore dei flag, soltanto in un secondo momento, nel corpo principale del programma chiamante.
Il diagramma di flusso di "Foresta Incantata", che si riferisce alla parte di programma fin qui sviluppata, evidenzia l'impiego delle varie bandierine. F indica se un comando ha un formato valido e viene modificata nella subroutine che isola le parole della frase. La subroutine che identifica ed esegue i comandi adopera due flag. VF serve per segnalare la presenza di un verbo riconosciuto nel comando, mentre MF indica se il comando impartito dal giocatore implica lo spostamento ad un nuovo ambiente. Il ciclo principale del programma controlla il valore di MF e, se questo vale 1 (il giocatore si è spostato), l'esecuzione torna all'inizio del ciclo, per descrivere il nuovo ambiente



IAN McKINNELL

Il comando GUARDA è altrettanto elementare: per ridescrivere l'ambiente attuale basta porre il flag di movimento, MF, a 1 e tornare al ciclo principale, che viene così ripetuto dall'inizio, visualizzando la descrizione e le possibili uscite dell'ambiente in questione, in quanto la variabile di posizione, P, non viene modificata dal comando GUARDA. Questo comando è utile quando, dopo aver impartito una serie di comandi, la descrizione è "scivolata" fuori dallo schermo.

Aggiungere elasticità

Per agevolare il gioco, conviene che uno stesso comando sia riconosciuto anche se impartito in forme diverse: VAI NORD, MUOVI A NORD e VAI VERSO IL NORD, ad esempio, significano tutti la stessa cosa. Anche se non è indispensabile che il programma riconosca tutti questi formati, il gioco diventa tuttavia più scorrevole. I tre comandi di movimento appena citati hanno una struttura comune: iniziano con un verbo e la direzione è indicata per esteso. Realizzare una routine che cerchi la direzione, dopo un verbo di movimento, è abbastanza facile: occorre individuare i vari spazi di separazione ed isolare le varie parole, confrontandole, una alla volta, con le direzioni riconosciute. Ecco la routine:

```
3630 REM *** S/R DI RICERCA DIREZIONE ***
3640 NN$=NN$+" ":LN=LEN(NN$):C=1
3645 FOR I=1 TO LN
3650 IF MID$(NN$,I,1)<>" " THEN NEXT I:RETURN
3655 W$=MID$(NN$,C,I-C):C=I+1
3660 IF W$="NORD" OR W$="EST" THEN NN$=W$:I=LN
3665 IF W$="SUD" OR W$="OVEST" THEN NN$=W$:I=LN
3670 NEXT I
3675 RETURN
```

Adesso però occorre inserire una chiamata a questa routine in quella del movimento, sviluppata in precedenza. Basta aggiungere questa linea:

```
3505 GOSUB 3630:REM RICERCA DIREZIONE
```

Nulla vieta di modificare il programma per aggiungere anche altri sinonimi, ad esempio: Oriente per Est, Meridione per Sud e via dicendo. Così facendo, però, si allunga notevolmente il tempo necessario alla verifica del comando.

LISTATO DIGITAYA

```
1220 GOSUB 1700:REM ANALISI COMANDI
1225 IF F=0 THEN 1210:REM COMANDO NON VALIDO
1230 GOSUB 1900:REM COMANDI NORMALI
1240 IF VF=0 THEN PRINT "NON CAPISCO"
1250 IF MF=1 THEN 1160:REM NUOVA POSIZIONE
1260 IF MF=0 THEN 1210:REM NUOVO COMANDO

1700 REM **** S/R ANALISI DEI COMANDI ****
1705 F=0:REM FLAG DI ZERO
1710 IF IS$="FINE" OR IS$="ELENCO" THEN VB$=IS$:F=1:RETURN
1720 IF IS$="GUARDA" THEN VB$=IS$:F=1:RETURN
1730 :
1740 REM ** SEPARA LE PAROLE **
1750 VB$=" ":NN$=" ":REM AZZERA VERBO E SOSTANTIVO
1770 LS=LEN(IS$)
1780 FOR C=1 TO LS
1790 A$=MID$(IS$,C,1)
1800 IF A$=" " THEN VB$=LEFT$(IS$,C-1):NN$=RIGHT$(IS$,LS-C)
     :F=1:C=LS
1810 NEXT
1830 IF F=0 THEN PRINT"OCCORRONO ALMENO DUE PAROLE"
1840 RETURN
1850 :
1900 REM **** S/R AZIONI NORMALI ****
1910 VF=0
1920 PRINT
1930 IF VB$="VAI" OR VB$="MUOVI" THEN VF=1:GOSUB 2000
1940 IF VB$="PRENDI" OR VB$="RACCOGLI" THEN VF=1:GOSUB 2140
1950 IF VB$="LASCIA" OR VB$="DEPOSITA" THEN VF=1:GOSUB 2360
1960 IF VB$="ELENCA" OR VB$="INVENTARIO" THEN VF=1
     :GOSUB 2540
1965 IF VB$="GUARDA" THEN VF=1: MF=1: RETURN
1970 IF VB$="FINE" OR VB$="SMETTI" THEN VF=1:GOSUB 2610
1980 RETURN

2015 GOSUB 8600: REM RICERCA DELLA DIREZIONE

2610 REM ** S/R FINE GIOCO **
2620 PRINT:PRINT"SEI SICURO (S/N) ?"
2630 GETA$:IFA$<>"S"AND A$<>"N"THEN2630
2640 IFA$="N"THEN RETURN
2650 END

8600 REM ** S/R RICERCA DELLA DIREZIONE **
8610 NN$=NN$+" ":LN=LEN(NN$):C=1
8620 FOR I=1 TO LN
8630 IF MID$(NN$,I,1)<>" " THEN NEXT I:RETURN
8640 W$=MID$(NN$,C,I-C):C=I+1:
8650 IF W$="NORD" OR W$="EST" THEN NN$=W$:I=LN
8660 IF W$="SUD" OR W$="OVEST" THEN NN$=W$:I=LN
8670 NEXT I
8680 RETURN
```

BASIC & BASIC

Spectrum:

In entrambi i programmi si usi I$ al posto di IS$, B$ al posto di VB$ e R$ al posto di NN$.
Cambiare le seguenti linee di Digitaya:

```
1790 LET A$=I$(C TO C)
1800 IF A$=" "THEN LET B$=I$(TO C—1):LET R$=I$
     (LEN(I$)—LS+C+1 TO):LET F=1:LET C=LS
2630 LET A$=INKEY$:IF A$<>"S" AND A$<>"N" THEN
     GOTO 2630
8630 IFR$(I TO I)<>" "THEN NEXT I:RETURN
8640 LET W$=R$(C TO I—1):LET C=I+1
```

Cambiare le seguenti linee di Foresta Incantata:

```
2250 LET A$=I$(C TO C)
2570 LET B$=I$(TO C—1):LET F=1
2580 LETR$=I$(LEN(I$)—LS+C+1 TO):LET C=LS
3650 IF R$(I TO I)<>" "THEN NEXT I:RETURN
3655 LET W$=R$(C TO I—1):LET C=I+1
4190 LET A$=INKEY$:IF A$<>"S" AND A$<>"N" THEN
     GOTO 4190
```

Micro BBC:

In Digitaya sostituire:

```
2630 REPEAT:A$=GET$:UNTIL A$="S" OR A$="N"
```

e in Foresta Incantata:

```
4190 REPEAT:A$=GET$:UNTIL A$="S" OR A$="N"
```

DATABASE

Set di caratteri sostitutivi nazionali

Queste tabelle mostrano i 10 set di caratteri sostitutivi nazionali (NRCS - National Replacement Character Sets) disponibili su alcune stampanti ed usati talvolta nelle comunicazioni tra computer o emulazioni di terminali (come accade, ad esempio, nel package Symphony della Lotus).

Nella prima colonna, sotto la sigla LICS appaiono i caratteri internazionali equivalenti a quelli sostituiti negli standard delle varie nazioni, riportati nella seconda colonna sotto NRCS.

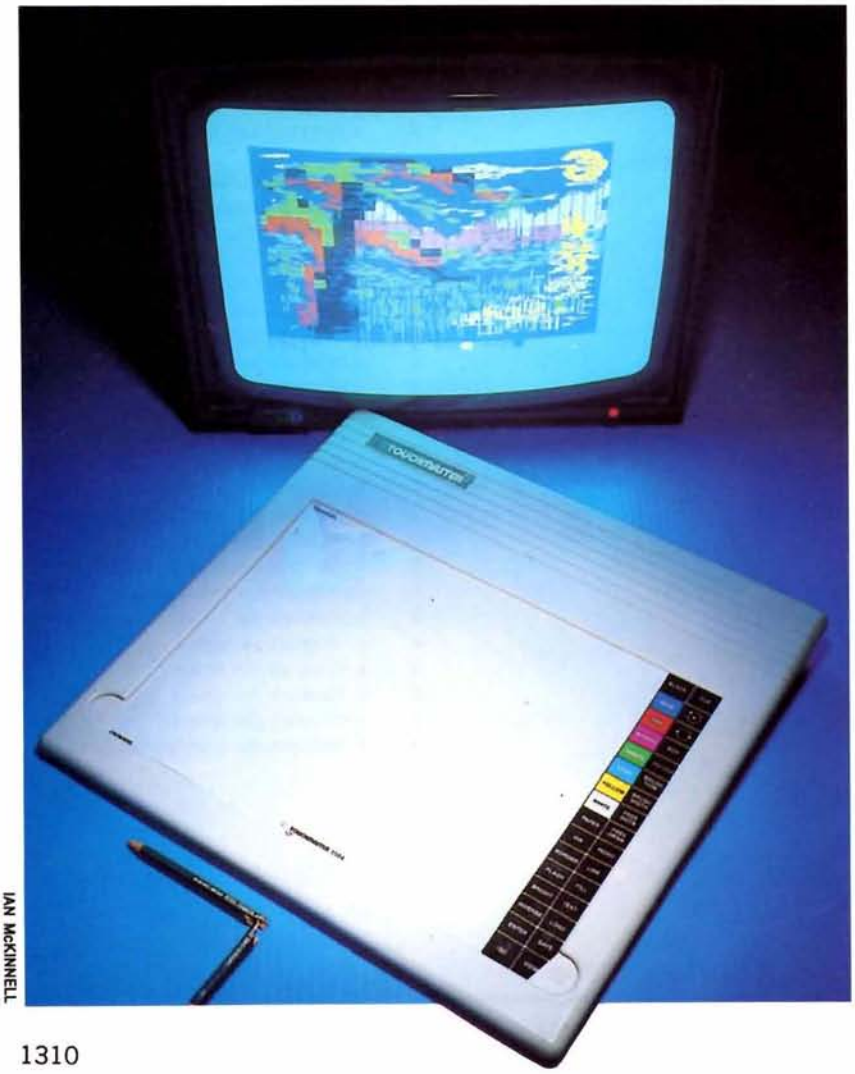
Nazione	Carattere LICS	Carattere NRCS	Descrizione
Spagna			
	#	£	Simbolo Lira Sterlina
	@	§	Simbolo di paragrafo
	[	¡	Punto esclamativo invertito
	\	Ñ	N maiuscola con tilde
	]	¿	Punto interrogativo invertito
	{	°	Simbolo di grado
	¦	ñ	n minuscola con tilde
	}	ç	c minuscola con cediglia
Gran Bretagna			
	#	£	Simbolo Lira Sterlina
Canada Francese			
	@	à	a minuscola con accento grave
	[	â	a minuscola con accento circonflesso
	\	ç	c minuscola con cediglia
	]	ê	e minuscola con accento circonflesso
	^	î	i minuscola con accento circonflesso
	`	ô	o minuscola con accento circonflesso
	{	é	e minuscola con accento acuto
	¦	ù	u minuscola con accento grave
	}	è	e minuscola con accento grave
	~	û	u minuscola con accento circonflesso
Danimarca/Norvegia			
	@	Ä	A maiuscola con umlaut
	[	Æ	Legatura AE maiuscola
	\	Ø	O maiuscola con taglio
	]	Å	A maiuscola con tondino
	^	Ü	U maiuscola con umlaut
	`	ä	a minuscola con umlaut
	{	æ	Legatura ae minuscola
	¦	ø	o minuscola con taglio
	}	å	a minuscola con tondino
	~	ü	u minuscola con umlaut
Finlandia			
	[	Ä	A maiuscola con umlaut
	\	Ö	O maiuscola con umlaut
	]	Å	A maiuscola con tondino
	^	Ü	U maiuscola con umlaut
	`	é	e minuscola con accento acuto
	{	ä	a minuscola con umlaut
	¦	ö	o minuscola con umlaut
	}	å	a minuscola con tondino
	~	ü	u minuscola con umlaut
Francia			
	#	£	Simbolo Lira Sterlina
	@	à	a minuscola con accento grave
	[	°	Simbolo di grado
	\	ç	c minuscola con cediglia
	]	§	Simbolo di paragrafo
	{	é	e minuscola con accento acuto
	¦	ù	u minuscola con accento grave
	}	è	e minuscola con accento grave
	~	¨	Umlaut
Germania			
	@	§	Simbolo di paragrafo
	[	Ä	A maiuscola con umlaut
		Ö	O maiuscola con umlaut
	]	Ü	U maiuscola con umlaut
	{	ä	a minuscola con umlaut
	¦	ö	o minuscola con umlaut
	}	ü	u minuscola con umlaut
	~	ß	Doppia s tedesca
Svezia			
	@	É	E maiuscola con accento acuto
	[	Ä	A maiuscola con umlaut
		Ö	O maiuscola con umlaut
	]	Å	A maiuscola con tondino
		Ü	U maiuscola con umlaut
	`	é	e minuscola con accento acuto
	{	ä	a minuscola con umlaut
	¦	ö	o minuscola con umlaut
	}	å	a minuscola con tondino
	~	ü	u minuscola con umlaut
Italia			
	#	£	Simbolo di Lira Sterlina
	@	§	Simbolo di paragrafo
	[	°	Simbolo di grado
		ç	c minuscola con cediglia
	]	é	e minuscola con accento grave
	`	ù	u minuscola con accento grave
	{	à	a minuscola con accento grave
	¦	ò	o minuscola con accento grave
	}	è	e minuscola con accento grave
	~	ì	i minuscola con accento grave
Svizzera			
	#	ù	u minuscola con accento grave
	@	à	a minuscola con accento grave
	[	é	e minuscola con accento acuto
		ç	c minuscola con cediglia
	]	ê	e minuscola con accento circonflesso
	^	î	i minuscola con accento circonflesso
	`	ô	o minuscola con accento circonflesso
	~	è	e minuscola con accento grave
	{	ä	a minuscola con umlaut
	¦	ö	o minuscola con umlaut
	}	ü	u minuscola con umlaut
	~	û	u minuscola con accento circonflesso

La Touchmaster

Una tavoletta grafica adatta a quasi tutti gli home computer

Molti dei microcomputer più diffusi possiedono capacità grafiche in alta risoluzione.
Tuttavia, se non si possiedono programmi specializzati, la creazione di videate grafiche implica un notevole impegno e moltissimo tempo di lavoro, anche senza sfruttare al massimo le prestazioni dell'apparecchio.
Inoltre, non basta uno strumento per disegnare: occorre la possibilità di memorizzare e di richiamare le immagini create, per usarle in più occasioni.
Esistono in commercio ottime tavolette grafiche, ma spesso sono progettate per essere usate assieme a computer specifici, quali lo Spectrum o il BBC. La tavoletta Touchmaster, invece, è adatta ad una più vasta gamma di apparecchi (dotati, in alcuni casi, di un'adeguata interfaccia). La ditta produttrice della Touchmaster afferma che l'apparecchio può essere usato anche in sostituzione della normale tastiera, almeno per quanto riguarda semplici operazioni di scelta da un menu riportato sullo schermo o comandi in programmi di gioco.
L'involucro della Touchmaster è in plastica grigia con il fondo leggermente inclinato in modo da offrire un piano di lavoro adeguatamente angolato.
L'alimentazione è fornita da un piccolo trasformatore da presa (come quelli adoperati per molti tipi di calcolatrici).
Non esiste un interruttore di accensione, ma un LED rosso segnala che l'apparecchio è in funzione. Per consentire un impiego su più tipi di computer, la tavoletta è fornita di interfacce, sia parallela sia seriale, situate nella parte posteriore del ripiano. Esiste anche un connettore per un comando a pedale, di cui il manuale, però, non fa alcuna menzione.
In effetti la documentazione non è tra le migliori: il manuale per l'hardware spiega come collegare l'apparecchio e fornisce alcuni esempi di programmi in BASIC, ma il tutto con grande parsimonia di dettagli.
Il funzionamento della tavoletta si basa sulla tecnologia a membrana (quella adoperata nella tastiera dello ZX81 e dello Spectrum), con una risoluzione di 256 x 256 pixel. Una leggera pressione sul piano di lavoro provoca il contatto tra i due elementi conduttori della membrana, disposti a reticolo. Un microprocessore incorporato effettua rapidissime scansioni in senso verticale e orizzontale, inviando le coordinate del punto di contatto alle due porte di uscita. In genere la porta seriale serve per apparecchi quali il micro BBC, mentre il Commodore 64, il Vic-20, lo Spectrum e il Dragon vengono collegati alla porta parallela. La risoluzione della tavoletta, però, non è eccessivamente alta, cosicché i possessori di BBC, ad esempio, non riescono ad intervenire su singoli pixel nel MODE grafico 0.

Una tavoletta per disegnare
La mascherina fornita a corredo del software va semplicemente alloggiata sul ripiano di lavoro. A sinistra i vari riquadri corrispondono alle opzioni disponibili. In basso sullo schermo, in un'apposita linea di stato, compaiono le varie scelte fatte



IAN McKINNELL

Pittura policroma

La Touchmaster è accompagnata da un programma grafico chiamato *Multipaint*. Questo programma è ottimo come strumento dimostrativo, ma insufficiente se lo si intende come vero e proprio strumento di lavoro. Una mascherina di plastica, da sovrapporre lateralmente sul ripiano, riassume le opzioni disponibili: quella selezionata viene riprodotta nell'apposita linea di stato, in basso sullo schermo. Sono previsti cinque tipi di "pennellate", la larghezza di ognuna delle quali può



variare tra 2 e 32 pixel, in passi di due pixel. La linea di stato sullo schermo indica anche il "modo" di disegno selezionato (Dots, Points o Freehand) e i colori scelti per lo sfondo e il primo piano. La selezione dei colori avviene premendo un'apposita area sulla mascherina della tavoletta.

Definiti i colori e i "pennelli" da usare, si può passare ad altre opzioni che servono a creare riquadri, cerchi, poligoni e anche a tracciare linee con la tecnica "a elastico". Per il disegno si può usare l'apposito stilo fornito a corredo, ma volendo può bastare anche un semplice dito.

Infatti, la Touchmaster non ha le stesse limitazioni di altre tavolette (da esempio la Koalapad descritta a pag. 1109), quindi la precisione del disegno è soddisfacente anche usando un dito.

Come accennato in precedenza, però, il *Multipaint* non offre che prestazioni rudimentali. Esiste un comando FILL (riportato sia sulla mascherina delle opzioni sia nel manuale) ma, almeno sullo Spectrum, non sembra operare nel modo che ci si aspetta. Non esistono funzioni di ingrandimento né di correzione (non si possono cambiare i colori nel disegno terminato).

Sullo Spectrum, dove spesso è più comodo tracciare il disegno in bianco e nero per colorarlo solo in seguito, questo è un grosso svantaggio.

Dal punto di vista hardware, comunque, la Touchmaster ha molti punti a proprio vantaggio rispetto alle concorrenti: è molto solida, offre un'area di lavoro corrispondente al formato A-4 e può essere collegata ad un gran numero di computer.

TOUCHMASTER

DIMENSIONI

mm. 350 x 330 x 35

INTERFACCE

Sono disponibili due porte, una seriale ed una parallela, che rendono la Touchmaster adatta ad una vasta gamma di computer

DOCUMENTAZIONE

Sebbene i manuali contengano tutte le informazioni necessarie al funzionamento, mancano descrizioni dettagliate su molte delle prestazioni ottenibili

PRO

La possibilità di collegare l'apparecchio a più tipi di computer, grazie alle due interfacce incorporate ed agli appositi programmi di corredo, rende la Touchmaster una periferica molto adattabile

CONTRO

Il software di corredo potrebbe essere migliore ed evidenziare così al massimo le capacità dell'apparecchio. Anche i manuali sono piuttosto limitati

UNA SCENA... TOCCANTE!

La Touchmaster, abbinata al programma Multipaint, può essere usata come digitizer per creare immagini grafiche. Ecco le varie fasi nella realizzazione di un disegno



ALTRE POSSIBILITÀ

Alcune ditte indipendenti hanno realizzato programmi applicativi o di gioco di vario genere, adatti alla Touchmaster. In genere il package comprende una mascherina in plastica da appoggiare sul piano di lavoro. La figura mostra alcune di queste mascherine

Chi è stato?

Un gioco di "caccia all'assassino" mostra come realizzare un database in LOGO

È stato commesso un terribile assassinio in una piccola comunità tra i monti Ozark: Zaccaria è stato aggredito e ucciso con un'ascia. I fatti noti sono che: Matteo e Giona possiedono entrambi un'ascia, Giorgio e Marco hanno una pistola e la cugina Elisa possiede un coltello. Inoltre, Matteo e Giorgio avevano macchie di sangue sulle mani, quando sono stati interrogati dallo sceriffo.
Il nostro database in LOGO sull'efferato delitto consiste in una lista di fatti, ognuno dei quali è composto da una relazione, assieme a uno o più sostantivi. Un fatto quale «Matteo possiede un'ascia» è tradotto in LOGO con [POSSIEDE MATTEO ASCIA]. Per rappresentare il fatto che Giorgio aveva le mani insanguinate si usa [INSANGUINATO GIORGIO].
Le indagini partono con un database vuoto:

```
TO INIZIO
   MAKE "DATABASE [ ]
END
```

I fatti vengono via via aggiunti nel database tramite la procedura METTI, ad esempio: METTI [POSSIEDE ELISA COLTELLO]. Ecco la procedura:

```
TO METTI :FATTO
   IF NOT MEMBER? :FATTO :DATABASE THEN MAKE "DATA-
      BASE FPUT :FATTO :DATABASE
END
```

Il database andrà via via riempiendosi con:

```
[[INSANGUINATO MATTEO][INSANGUINATO GIORGIO][UCCISO
   ZACCARIA ASCIA][POSSIEDE MATTEO ASCIA][POSSIEDE
   GIONA ASCIA][POSSIEDE GIORGIO PISTOLA][POSSIEDE
   MARCO PISTOLA][POSSIEDE ELISA COLTELLO]]
```

Per visualizzare il database si usa la procedura VISUAL, che può essere seguita da "TUTTO (per esaminare l'intero contenuto del database) oppure dal nome di una relazione, nel qual caso vengono visualizzati soltanto i fatti pertinenti. Ad esempio VISUAL "POSSIEDE mostra chi possiede che cosa.

```
TO VISUAL :S
   IF :S = "TUTTO THEN ELENCA. TUTTO :DATABASE
   ELENCA.REL :S :DATABASE
END

TO ELENCA.TUTTO :LISTA
   IF EMPTY? :LISTA THEN STOP
   PRINT FIRST :LISTA
   ELENCA.TUTTO BUTFIRST :LISTA
END

TO ELENCA.REL :S :LISTA
   IF EMPTY? :LISTA THEN STOP
   IF :S = FIRST FIRST :LISTA THEN PRINT FIRST :LISTA
   ELENCA.REL :S BUTFIRST :LISTA
END
```

Adesso serve un sistema per interrogare il database. Un metodo molto semplice consiste nel chiedere se un fatto è vero oppure no. La risposta ad una domanda formulata come VERO [POSSIEDE ELISA COLTELLO], ad esempio, deve produrre la risposta SI.

```
TO VERO :FATTO
   IF MEMBER? :FATTO :DATABASE PRINT "SI ELSE PRINT
      "NO
END
```

Tuttavia sarebbe molto più comodo formulare i quesiti nella forma «Chi possiede un'ascia?». Ciò si può ottenere adoperando delle variabili: nel nostro caso consideriamo tutte le parole che iniziano con un punto interrogativo come variabili. Possiamo così riformulare le domande in questo modo:

```
QUALE [POSSIEDE ?QUALCUNO ASCIA]
```

La risposta al quesito sarà un elenco di tutti i possibili valori, estratti dal database, che si possono abbinare a ?QUALCUNO:

```
[?QUALCUNO MATTEO]
[?QUALCUNO GIONA]
FINE RISPOSTA
```

Inoltre si possono impiegare variabili multiple. Per esempio:

```
CHI [UCCISO ?UOMO ?ARMA]
```

fornisce come risposta:

```
[?UOMO ZACCARIA][?ARMA ASCIA]
FINE RISPOSTA
```

Vediamo, una per una, le procedure che consentono una tale analisi del database. CHI trasferisce l'esecuzione a TROVA, indicando DATABASE come fonte dei fatti:

```
TO CHI :DOMANDA
   TROVA :DOMANDA :DATABASE
   PRINT [FINE RISPOSTA]
END
```

TROVA definisce due variabili globali: VARS e RISP.

La prima è usata per contenere ogni possibile insieme di valori, che vengono poi raccolti in RISP, per la variabile in questione

```
TO TROVA :DOMANDE :DATI
   MAKE "VARS [ ]
   MAKE "RISP [ ]
   CONFRONTA :DOMANDA :DATI
   PRINTL :RISP
END
```

La procedura CONFRONTA esamina, uno per volta, tutti i fatti contenuti nel database, paragonandolo alla :DOMANDA. Se c'è corrispondenza, allora l'insieme di valori in VARS è aggiunto a RISP, prima di azzerare la lista VARS. L'operazione prosegue alla ricerca di altre possibili corrispondenze.

```
TO CONFRONTA :DOMANDA :DATI
   IF EMPTY? :DATI THEN STOP
   IF UGUALE? :DOMANDA FIRST :DATI THEN MAKE"RISP
      FPUT :VARS :RISP
   MAKE "VARS [ ]
   CONFRONTA :DOMANDA BUTFIRST :DATA
END
```

Per comprendere come funzioni UGUALE? consideriamo il caso in cui i due elementi in ingresso siano [POSSIEDE ?QUALCUNO ASCIA] e [POSSIEDE GIONA ASCIA]. La procedura UGUALE? fornisce la condizione TRUE (vero) ed assegna [?QUALCUNO GIONA] a VARS. Viceversa, se gli elementi in ingresso sono [POSSIEDE ?QUALCUNO ASCIA] e [UCCISO ZACCARIA ASCIA] la procedura fornisce la condizione FALSE. Le difficoltà sorgono se nel confronto è coinvolta più di una variabile. In tal caso occorre una nuova procedura, VALORE?, per determinare se alla variabile è già stato assegnato un valore relativo ad un fatto contenuto nel database.
Qui viene impiegata una forma alternativa per le frasi condizionali. TEST valuta una condizione e, se questa risulta vera viene eseguito tutto ciò che segue IFTRUE, altrimenti tutto ciò che segue IFFALSE:

```
TO UGUALE? :DOMANDA :FATTO
   IF ALLOF EMPTY? :DOMANDA EMPTY? :FATTO THEN OUT-
      PUT "TRUE
   TEST FIRST FIRST :DOMANDA = "?
   IFTRUE IF NOT VALORE? FIRST :DOMANDA FIRST:FATTO
      :VARS THEN OUTPUT "FALSE
   IFFALSE IF NOT (FIRST :DOMANDA = FIRST :FATTO)THEN
      OUTPUT "FALSE
   OUTPUT UGUALE? BUTFIRST :DOMANDA BUTFIRST :FATTO
END
```

Per capire come funzioni VALORE? supponiamo che gli elementi in ingresso siano ?ARMA, ASCIA e [?UOMO ZACCARIA]. VALORE? tenta di stabilire se la variabile ?ARMA può contenere il valore ASCIA. Esistono tre possibilità: ?ARMA contiene un valore, ma questo non è ASCIA (VALORE? genera la condizione FALSE), ?ARMA contiene il valore ASCIA (VALORE? genera la condizione TRUE) e, infine, ?ARMA non contiene alcun valore. In quest'ultimo caso alla variabile viene assegnato il valore ASCIA, questa informazione è aggiunta a VARS e viene generata la condizione TRUE.

```
TO VALORE? :NOME :VALORE :VLISTA
   IF EMPTY? :VLISTA THEN MAKE "VARS LPUT :LISTA :NOME
      :VALORE :VARS OUTPUT "TRUE
   TEST :NOME = FIRST FIRST :VLISTA
   IFTRUE IF :VALORE = LAST FIRST :VLISTA THEN OUTPUT
      "TRUE ELSE OUTPUT "FALSE
   OUTPUT VALORE? :NOME :VALORE BUTFITST :VLISTA
END
```

PRINTL si occupa semplicemente di visualizzare i vari elementi di RISP uno sotto l'altro.

```
TO PRINTL :LISTA
   IF EMPTY? :LISTA STOP
   PRINT FIRST :LISTA
   PRINTL BUTFIRST :LISTA
END
```

RICHIESTE PIÙ COMPLESSE

Le indagini non procederanno molto, però, se non si possono formulare quesiti più complessi, del tipo «Quale arma ha ucciso Zaccaria e chi possiede tale arma?» Tradotto in LOGO:

```
CHI [[UCCISO ZACCARIA ?ARMA][POSSIEDE ?SOSPETTO
   ?ARMA]]
```

La procedura CHI va modificata per accettare in ingresso una lista di quesiti. I valori da fornire in uscita sono quelli che soddisfano tutte le condizioni. Adesso, volendo porre un solo quesito al database, andrà formulato:

```
CHI [[POSSIEDE ?QUALSIASI COLTELLO]]
```

Ecco le modifiche da apportare a CHI e a TROVA:

```
TO CHI :DOMANDE
   TROVA :DOMANDE :DATABASE
   PRINT [FINE RISPOSTA]
END

TO TROVA :DOMANDE :DATI
   MAKE "VARS [ ]
   MAKE "RISP [ ]
   CONFRONTA :DOMANDE :DATI
   PRINTL :RISP
END
```

Adesso la procedura CONFRONTA ha un compito piuttosto difficile. Supponiamo che in ingresso essa riceva [[UCCISO ZACCARIA ?ARMA][POSSIEDE ?SOSPETTO ?ARMA]]: i vari fatti nel database vengono esaminati uno per uno, per trovare una corrispondenza per il primo quesito. Il risultato sarà un abbinamento di ASCIA a ?ARMA. La routine passa poi al secondo quesito (ossia [POSSIEDE ?SOSPETTO ?ARMA]), ricominciando il processo da capo. La corrispondenza viene trovata, abbinando ASCIA ad ?ARMA e MATTEO a ?SOSPETTO. Non essendovi altre domande, questa rappresenta una possibile soluzione.
Ma possono esistere anche altri valori che soddisfano alla seconda condizione (fermo restando il primo abbinamento ?ARMA — ASCIA), quindi il processo TROVA riprende da dove è stata trovata l'ultima corrispondenza. In effetti viene trovata una seconda soluzione con ?SOSPETTO abbinato a GIONA. Di nuovo il proces-



DAVID HIGHAM



MATRICE DEL DELITTO

Il signor LOGO è stato trovato morto, pugnalato 30 volte con uno scalpello, in un furgoncino. La polizia ha individuato quattro sospetti: un muratore, un macellaio, una giardiniera e un designer. Tutti avevano accesso a strumenti affilati: il coltello del macellaio, le cesoie della giardiniera, il coltellino del designer e lo scalpello del muratore. Chiunque possa essere in qualche modo collegato al furgoncino è l'assassino. I fatti accertati sono i seguenti: **1)** Nessuno dei sospetti era in possesso del proprio strumento di lavoro nella notte del fattaccio; **2)** Il macellaio venne trovato nella serra, mentre apriva delle buste con il coltellino; **3)** Un testimone oculare afferma di aver visto il muratore all'angolo di una strada, proprio dove la giardiniera trova in seguito le proprie cesoie; **4)** La giardiniera era a letto che si preparava un panino usando il coltello del macellaio. Chi è stato?

so riprende, ma stavolta le informazioni nel database non forniscono ulteriori soluzioni.
Tuttavia, possono esistere altre soluzioni che soddisfano la prima condizione (a parte la combinazione ?ARMA – ASCIA), cosicché la procedura TROVA riparte, ma stavolta da dove aveva trovato l'ultima soluzione alla prima condizione. Questo procedimento è noto col termine "backtracking". Nel nostro caso, però, non porta ad ulteriori soluzioni.
Per non perdere il filo di tutti questi percorsi in avanti e indietro, CONFRONTA appoggia i valori correnti su di uno stack, prima di usare UGUALE?, che potrebbe modificarli. I valori vengono in seguito recuperati. Ecco la procedura completa:

```
TO CONFRONTA :DOMANDE :DATI
   IF EMPTY? :DOMANDE THEN MAKE "RISP FPUT :VARS
      :RISP STOP
   IF EMPTY? :DATI THEN STOP
   PUSH :VARS
   TEST UGUALE? FIRST :DOMANDE FIRST :DATI
   IFTRUE CONFRONTA BUTFIRST :DOMANDE :DATABASE
   PULL "VARS
   CONFRONTA :DOMANDE BUTFIRST :DATI
END
```

In CONFRONTA viene usato lo stack, anziché una semplice variabile provvisoria, perché la procedura può richiamare se stessa ed il valore di una variabile temporanea andrebbe perduto.
PUSH deposita un valore in cima allo stack, creando (se non esiste già) una variabile STACK.

```
TO PUSH :DATI
   IF NOT THING? :STACK THEN MAKE "STACK[ ]
   MAKE "STACK FPUT :DATI :STACK
END
```

PULL estrae un elemento dallo stack e lo assegna ad una variabile:

```
TO PULL :NOME
   MAKE :NOME FIRST :STACK
   MAKE "STACK BUTFIRST :STACK
END
```

Ciò che abbiamo realizzato è un rudimentale linguaggio di "programmazione logica", in cui viene creato un archivio di fatti e relazioni alle quali accedere con quesiti formulati in base a descrizioni logiche dei dati da estrarre. Uno dei migliori esempi di linguaggio di programmazione logica è il PROLOG.

LOGO & LOGO

Alcune versioni del MIT LOGO sono prive di EMPTY? e MEMBER?. Le relative definizioni sono state fornite in precedenza (vedere alle pagg. 1244 e 1256).
In tutte le versioni LCSI si usi EMPTYP al posto di EMPTY? e MEMBERP al posto di MEMBER?. Esiste una primitiva, EQUALP, che confronta l'eguaglianza tra due termini. La si usi per confrontare liste e parole al posto del segno uguale (che in alcune versioni non funziona sulle liste).
La sintassi delle costruzioni IF del LOGO LCSI è la seguente:

IF EMPTYP :CONTENUTO [PRINT [NIENTE DI SPECIALE]][PRINT :CONTENUTO]

La prima lista viene eseguita se la condizione è vera, la seconda se la condizione è falsa.
Il LOGO della LCSI è dotato della struttura condizionale TEST, IFTRUE, IFFALSE

Motore fuoribordo

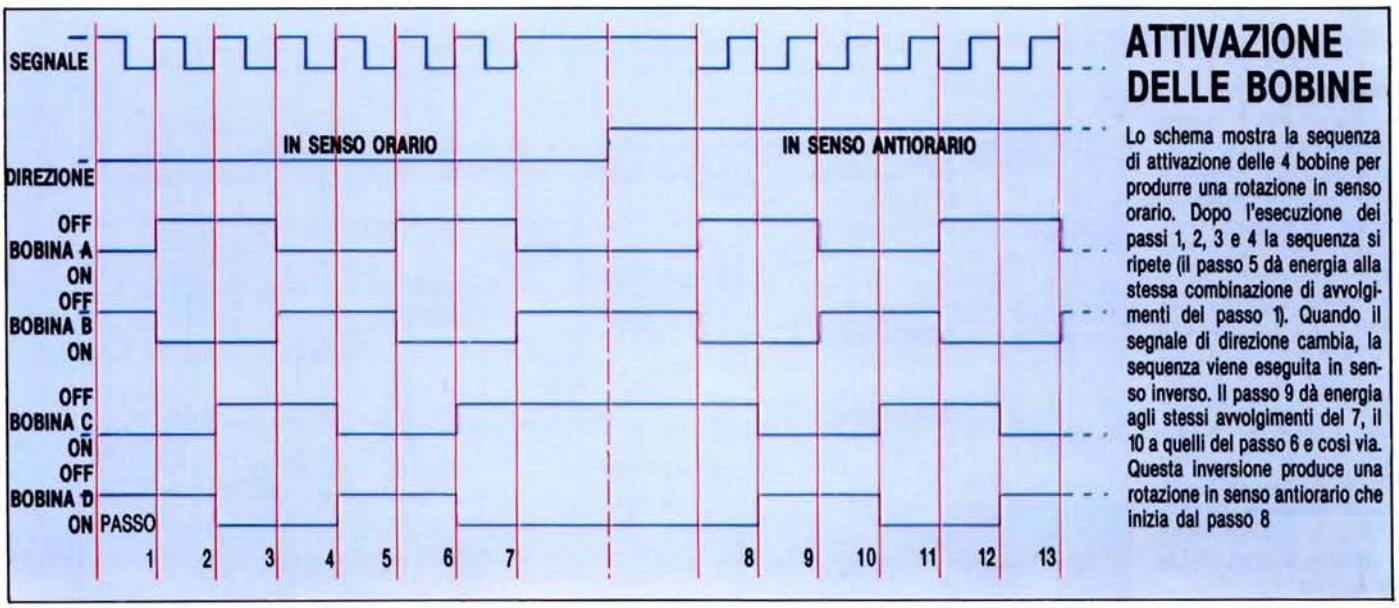
Per il nostro apparecchio viene impiegato un motore "passo passo". Vediamone le caratteristiche

La struttura di un motore passo passo è notevolmente diversa da quella di un comune motore elettrico. Consideriamo il funzionamento di un motore passo passo con una struttura semplificata. Nel nostro esempio (che si riferisce alla figura "Un passo alla volta" alla pagina seguente), ci sono due avvolgimenti o bobine sullo statore e due coppie di poli elettromagnetici sui rotori (i motori usati sul nostro veicolo robot possiedono un numero maggiore di bobine e di poli).
L'unico difetto di questa particolare configurazione elettrica è che il motore consuma più corrente quando è fermo di quando è in movimento. Inoltre non può girare ad alta velocità, perché le varie bobine non possono essere attivate e disattivate con sufficiente rapidità.
Tali problemi non influiscono però sul buon funzionamento del robot.
Questo motore a struttura semplificata ruota con passi di 45°, ma non è possibile controllare la direzione della rotazione. Al contrario, i motori che azionano il robot possiedono quattro serie di avvolgimenti che vengono attivati a coppie e, inoltre, il numero di avvolgimenti sui rotori è superiore. Ciò permette di controllare la direzione della rotazione e di ridurla di ogni passo a 7,5°. Per ottenere una rotazione ad un passo così ridotto, le quattro serie di bobine devono ricevere corrente secondo una particolare e complessa sequenza:

Tabella di attivazione delle bobine dello statore

Passo	Bobina A	Bobina B	Bobina C	Bobina D	
1	acceso	spento	acceso	spento	
2	spento	acceso	acceso	spento	
3	spento	acceso	spento	acceso	
4	acceso	spento	spento	acceso	
5	acceso	spento	acceso	spento	etc

La sequenza di attivazione potrebbe essere generata dal software di sistema, usando quattro bit della porta utente per controllare le quattro bobine. Tuttavia, la sua programmazione è piuttosto complessa e, inoltre, il BASIC non è assolutamente in grado di produrre sequenze sufficientemente veloci.
Un sistema più semplice è quello di usare il chip SAA 1027, appositamente progettato per il controllo dei motori passo passo, che contiene i dispositivi di uscita richiesti e tutti i circuiti logici necessari per attivare le bobine nel corretto ordine.
Un singolo segnale proveniente dalla porta utente fa muovere il motore di un passo, ma è necessaria una seconda linea di trasmissione per determinare la direzione della rotazione. Il circuito del chip contiene particolari stadi d'immissione per individuare i cambiamenti nei tre segnali in ingresso: un segnale per far ruotare il motore di un passo, un segnale di reset e uno di direzione, che inverte la sequenza di attivazione delle bobine. Questi segnali vengono inviati ad un contatore bidirezionale che genera la corretta sequenza di impulsi per gli avvolgimenti.



ATTIVAZIONE DELLE BOBINE

Lo schema mostra la sequenza di attivazione delle 4 bobine per produrre una rotazione in senso orario. Dopo l'esecuzione dei passi 1, 2, 3 e 4 la sequenza si ripete (il passo 5 dà energia alla stessa combinazione di avvolgimenti del passo 1). Quando il segnale di direzione cambia, la sequenza viene eseguita in senso inverso. Il passo 9 dà energia agli stessi avvolgimenti del 7, il 10 a quelli del passo 6 e così via. Questa inversione produce una rotazione in senso antiorario che inizia dal passo 8

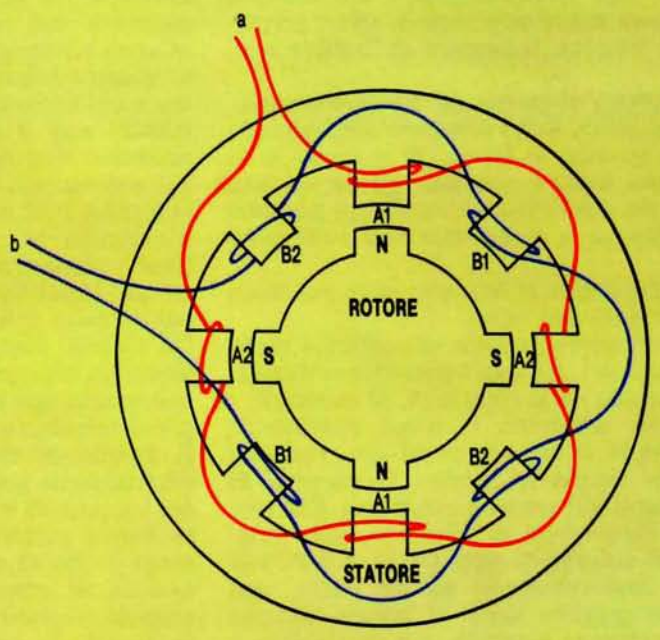
Il chip possiede anche uno stadio pilota di potenza, in grado di erogare fino a 500 mW. Questo circuito permette di collegare il motore direttamente al chip, senza dover usare altri circuiti aggiuntivi. Il circuito integrato per il controllo del motore consente di semplificare la realizzazione del robot. Naturalmente, occorre un chip di controllo per ogni motore usato. Sfortunatamente però, i chip di controllo usano tensioni di 12 volt, mentre le porte utente dei computer funzionano a 5 volt. Cioè un livello logico 0 corrisponde all'incirca a 0 volt e un livello logico 1 a 5 volt. Viceversa, gli ingressi del chip interpretano tensioni di 0 volt come livello logico 0 e tensioni comprese tra 7,5 e 12 volt come livello logico 1. Per collegare la porta utente ai chip di controllo occorre un apposito circuito integrato (buffer) a due tensioni, quale il 40109, dotato di ingressi e uscite che lavorano a tensioni diverse.

LISTA DEI COMPONENTI

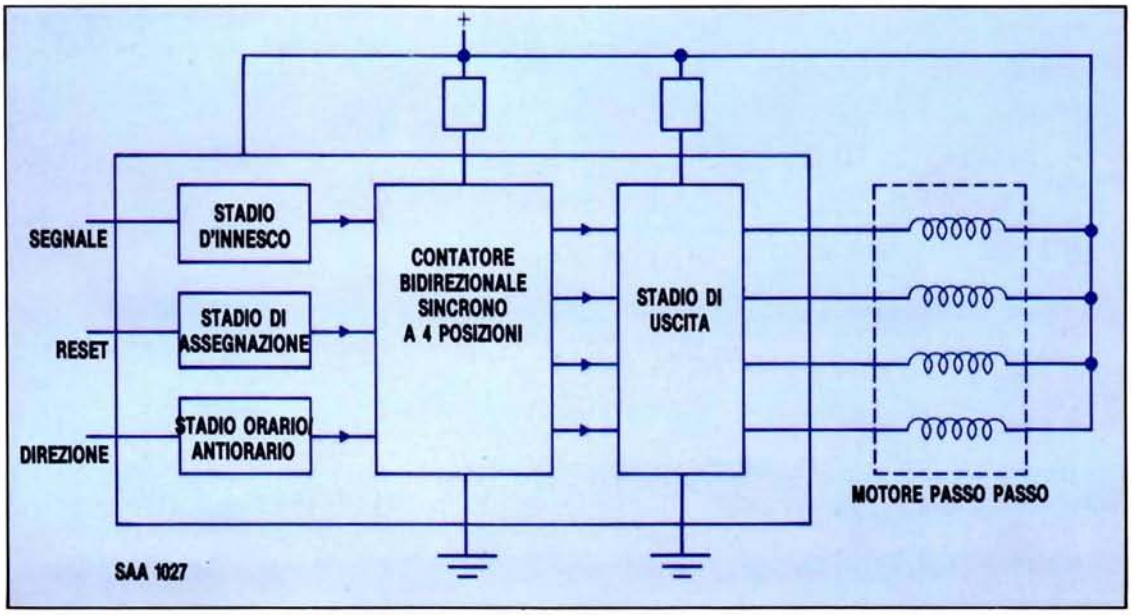
Quantità	Descrizione
1	Chip "buffer" 40109
2	Chip SAA 1027 per motori passo passo
3	Prese DIL a 16 piedini
2	Resistenze da 100 ohm
2	Resistenze da 270 ohm 1/2 watt
2	Condensatori da 0,1 micro-F
1	Condensatore da 1000 micro-F a 25 volt
1	Veroboard da 50 fori x 24 piste
-	Filo elettrico isolato per le connessioni

UN PASSO ALLA VOLTA

Questo schema di motore passo passo semplificato mostra il rotore ed i due circuiti "a" e "b" dello statore. Attivando le bobine dei due circuiti secondo una particolare sequenza il rotore gira in senso orario. Si noti che le coppie di bobine A1 e A2, come pure B1 e B2, sono avvolte in direzioni opposte. Quando il circuito "a" riceve corrente, induce un campo elettromagnetico negativo (polo sud) sulla coppia A1 ed uno positivo (polo nord) sulla coppia A2. La stessa cosa vale per "b". In questo motore, l'unità di passo è di 45°. I motori del veicolo robot possiedono un maggior numero di bobine sullo statore e di poli sul rotore: questa diversa configurazione consente di effettuare passi di 7,5°



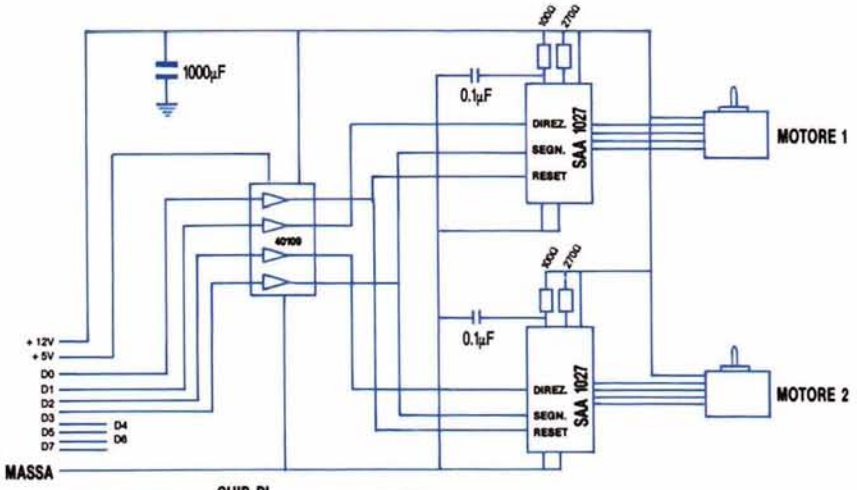
KEVIN JONES

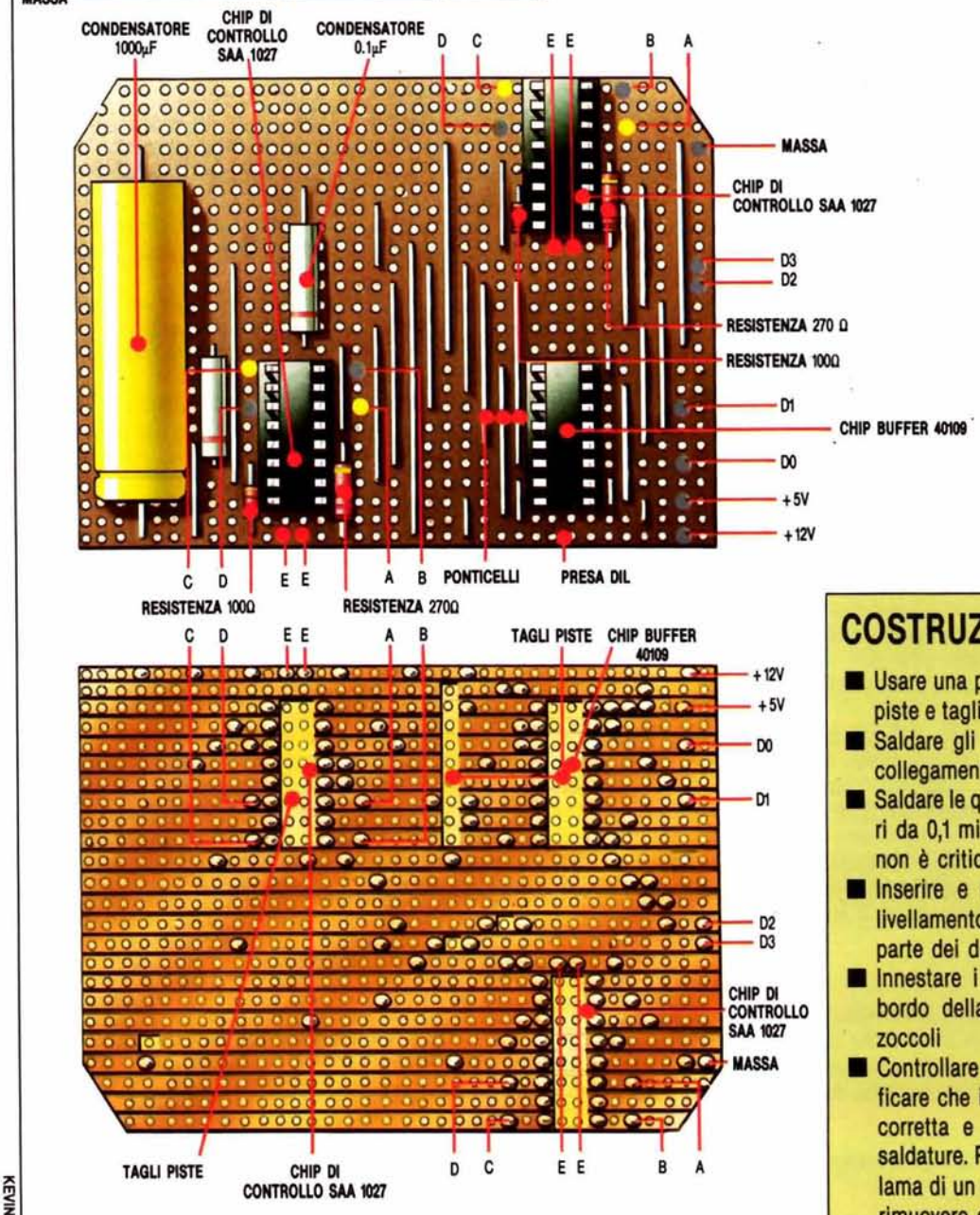


LIZ DIXON

Il chip pilota 1027

La logica del chip di controllo del motore passo passo è complessa, ma il suo principio di funzionamento è abbastanza semplice. Per far girare il rotore, alle bobine dello statore viene fornita energia secondo una precisa sequenza. Questa viene trasmessa al motore, un passo alla volta, da un contatore bidirezionale attivato da un segnale. Quando cambia il segnale di direzione, la sequenza viene eseguita in senso inverso ed il rotore cambia la direzione della rotazione. Se necessario, un terzo segnale (di reset) riporta il rotore alla posizione che aveva all'inizio della sequenza





KEVIN JONES

Schema del circuito

Lo schema del circuito che controlla i due motori passo passo è semplice: i due chip di controllo SAA 1027 vengono usati per trasmettere la corretta sequenza di attivazione alle bobine dei motori. Dato che i chip funzionano a 12 volt, mentre i segnali inviati dal computer attraverso la porta per l'utente sono a 5 volt, sulla scheda viene inserito un apposito chip (buffer), che converte le basse tensioni provenienti dalla porta utente in segnali di tensione più elevata, adatta ai chip di controllo. In seguito viene descritto come collegare la scheda ai due motori, al connettore D ed alla porta utente

COSTRUZIONE DELLA SCHEDA

- Usare una piastrina di veroboard da 35 fori x 24 piste e tagliare le piste come mostrato in figura.
- Saldare gli zoccoli dei tre chip, poi eseguire i collegamenti coi ponticelli.
- Saldare le quattro resistenze e i due condensatori da 0,1 micro-F. La loro orientazione (polarità) non è critica.
- Inserire e saldare il grosso condensatore di livellamento, orientando il piedino positivo dalla parte dei due chip di controllo.
- Innestare i chip, con la tacca rivolta verso il bordo della scheda più vicino ai chip, sugli zoccoli
- Controllare con attenzione l'intera scheda. Verificare che i vari componenti siano in posizione corretta e che non vi siano sbavature nelle saldature. Per maggior sicurezza fare scorrere la lama di un coltellino tra una traccia e l'altra per rimuovere eventuali frammenti di stagno.

Ritocchi finali

Nella parte conclusiva dell'Assembly 6809 aggiungiamo le ultime routine al debugger

Il primo compito del modulo principale è quello di predisporre il meccanismo per le interruzioni, che consente di inserire breakpoint nel programma sotto verifica. Queste trasferiscono il controllo al debugger permettendo di esaminare il contenuto dei registri del processore o delle locazioni di memoria. Poi occorre immettere l'indirizzo iniziale del programma da eseguire tramite il comando Il resto del modulo principale concerne l'acquisizione e l'esecuzione dei comandi. Il controllo è trasferito al programma sotto test mediante i comandi S e G e ritorna tramite le istruzioni SWI inserite nelle breakpoint. Due stadi di inizializzazione sono stati codificati in precedenza (vedere a pag. 1297). Il punto d'ingresso per le breakpoint si trova immediatamente dopo la chiamata alla subroutine. La prima istruzione salva il contenuto del puntatore allo stack, S, in modo da poter intervenire sui valori dei registri salvati sullo stack dalla SWI. Il passo successivo è l'interpretazione dei comandi. Le varie routine sono già state sviluppate: occorre soltanto selezionare quella giusta in base al comando impartito.
Sarebbe possibile realizzare questa parte di programma usando una serie nidificata di strutture IF, ma qui viene usata la routine Accetta-Comando (che fornisce l'offset nella tabella dei comandi) associandone l'uso ad una tabella contenente gli indirizzi delle varie subroutine. Nel nostro caso può sembrare un metodo complicato più del necessario, ma vale la pena di applicarlo come esempio per altre occasioni.
L'istruzione JMP, diversamente da quelle di branch (diramazione), può sfruttare qualsiasi normale metodo d'indirizzamento, compresi quello indiretto e quello indicizzato. Se il registro X viene caricato con l'indirizzo di base della tabella e il registro B con l'offset da usare (raddoppiato in quanto la tabella è composta da elementi a 2 byte), allora il comando

```
JMP [B,X]
```

trasferirà il controllo alla routine appropriata. Poiché la tabella degli indirizzi va costruita in fase di esecuzione del debugger, occorre un'ulteriore routine di inizializzazione.



Preparazione della Jump-Table

Dati:
    Jump-Table è una tabella di indirizzi a 16 bit
    CMDB, CMDU, ecc. sono gli indirizzi iniziali delle subroutine dei comandi

Procedura:
    Per ciascuna routine
        Estrae l'indirizzo iniziale
        Lo deposita nella tabella
    Fine ciclo

Adesso va deciso cosa fare, anche se si tratta di ben poco, quando si termina il programma con il comando Q. La cosa più logica è quella di lasciare intatti sia il debugger che il programma sotto verifica, in modo da poterli riadoperare se necessario.
Al momento dell'uscita, il puntatore allo stack deve essere identico a quando si avvia il debugger.
Una soluzione potrebbe essere quella di usare un diverso stack per il nostro programma, assegnando a S un nuovo valore per poi ripristinare il vecchio in uscita.
Questa soluzione è adatta a molti casi ma non qui, poiché la presenza del debugger e del programma in prova lascia poca memoria libera.
Un'altra soluzione potrebbe essere quella di incrementare adeguatamente S, in modo da perdere qualsiasi cosa vi sia stata lasciata durante l'uso del debugger, ma questa è un'operazione alquanto difficile, perché non sappiamo se, nel frattempo, vi sono state delle interruzioni da parte del sistema. La soluzione più semplice consiste nel salvare il puntatore allo stack quando si avvia il debugger e recuperarlo quando si esce.
Un'altra operazione da effettuare è il ripristino dei tre byte, nel vettore delle SWI a $FFFA, modificati dalla procedura di inizializzazione. Occorre quindi un'ulteriore fase di inizializzazione in cui detti valori sono conservati per esser poi recuperati dalla ruotine di Quit.

Procedura Salva-valori

Dati:
    Saved sono cinque byte ove conservare i valori
    Stack-Pointer è il valore di S aumentato di 2
    SWI-Vector si trova a $FFFA

Procedura:
    Copia Stack-Pointer in Saved
    Preleva SWI-Vector
    Salva i tre byte a SWI-Vector in Saved

La routine di uscita (comando Q) non deve far

altro che ripristinare i valori originali e riaffidare il controllo al sistema operativo. Lo scopo può essere raggiunto in diversi modi: può essere usata la stessa istruzione SWI, dopo aver ripristinato i tre byte, oppure si può eseguire un salto ad un punto d'ingresso del sistema operativo (purché se ne conosca uno!). Un salto tramite il vettore di reset che si trova a $FFFE è il metodo migliore, benché possa produrre un "cold start", ossia l'intera reinizializzazione del sistema.

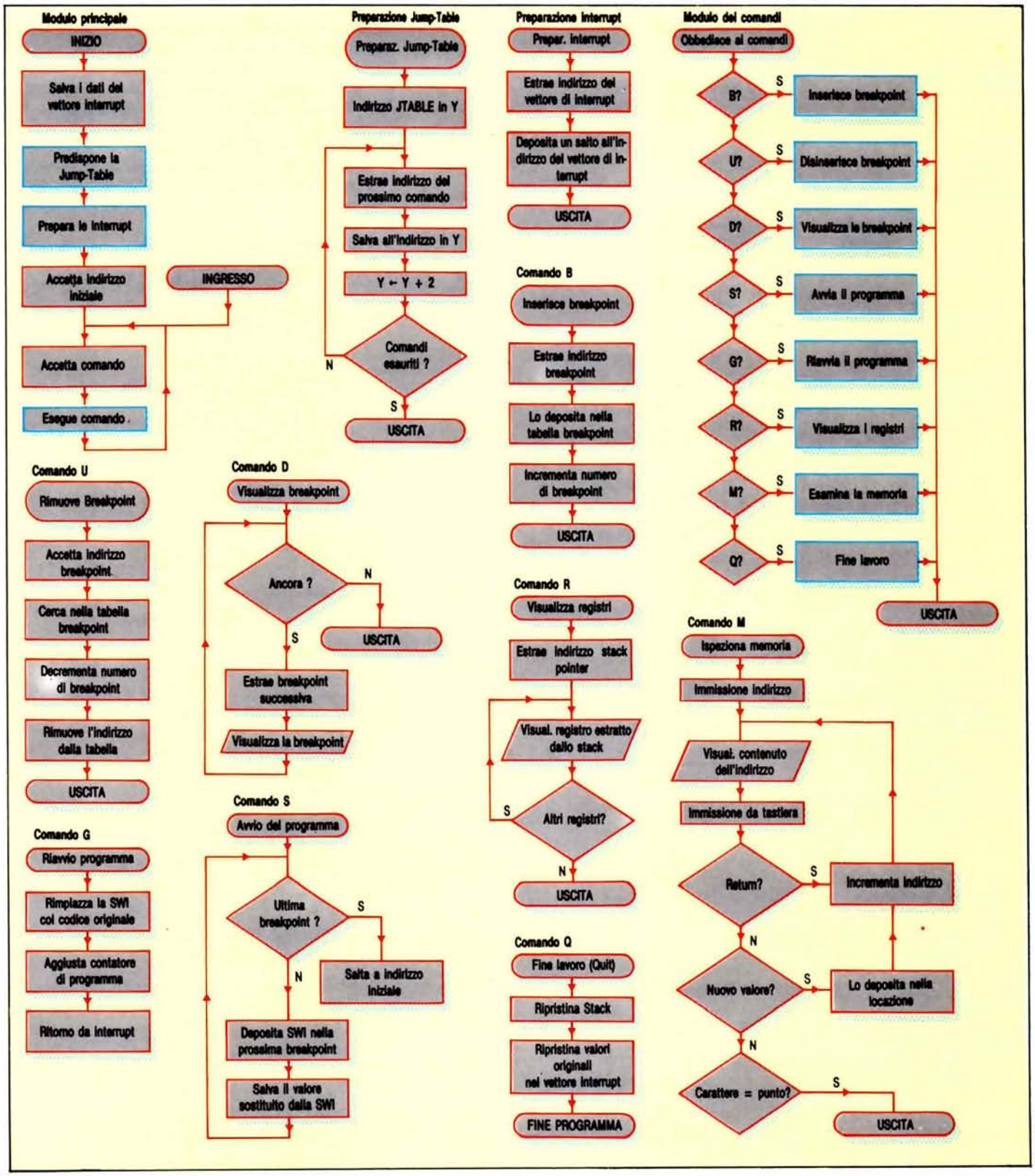
Procedura di uscita

Dati:

Saved sono 5 byte ove si trovano i valori salvati
Stack-Pointer è il valore di S aumentato di 2
SWI-Vector si trova a $FFFA
Reset-Vector si trova a $FFFE

Flusso del programma

Questi diagrammi di flusso corrispondono ai moduli del programma debugger. I riquadri in blu indicano l'esecuzione delle relative routine



Procedura:
Ripristina in SWI-Vector i 3 byte situati in Saved
Ripristina lo Stack-Pointer
Salta al sistema operativo

Adesso abbiamo tutti gli elementi per codificare il modulo principale. La struttura è leggermente diversa da quella concepita inizialmente, ma la sostanza non è variata.

Il modulo principale

Dati:
Prompt è il simbolo ASCII '>'
Command-Offset è lo scarto nella tabella dei caratteri di comando e in quella degli indirizzi



Procedura:
Salva i valori
Prepara la Jump-Table
Predispone le interrupt
Accetta l'indirizzo iniziale
Ripete
Visualizza prompt
Accetta comando
Esegue comando
All'infinito

Con ciò, assieme alla sezione dedicata all'Assembly del 6809, si conclude anche lo sviluppo del nostro debugger. Quest'ultimo può apparire piuttosto frammentario, ma ciò è dovuto al fatto che è stato sviluppato in maniera modulare.
Volendo, i più esperti possono cimentarsi nell'ottimizzare il codice e, magari, nel riunire tutte le zone usate come deposito temporaneo di dati all'inizio o alla fine del programma (questo però a scapito della modularità del progetto).
Il debugger va caricato in una qualsiasi zona di memoria che non venga occupata né usata dal programma sotto verifica, annotandosi l'indirizzo che corrisponde all'etichetta DEBUG.
Per avviare il debugger, infatti, occorre eseguire un salto a tale locazione.

PREPARAZIONE JUMP-TABLE

```
JTABLE  RMB   16              Spazio per 8 indirizzi a 16 bit
SETUPJ  LEAY  JTABLE,PCR      Indirizzo della tabella in Y
        LEAX  CMDB,PCR        Indirizzo della subroutine CMDB
        STX   ,Y++            Lo deposita nella tabella
        LEAX  CMDU,PCR        Indirizzo della subroutine CMDU
        STX   ,Y++            Lo deposita nella tabella
        LEAX  CMDD,PCR        Indirizzo della subroutine CMDD
        STX   ,Y++            Lo deposita nella tabella
        LEAX  CMDS,PCR        Indirizzo della subroutine CMDS
        STX   ,Y++            Lo deposita nella tabella
        LEAX  CMDG,PCR        Indirizzo della subroutine CMDG
        STX   ,Y++            Lo deposita nella tabella
        LEAX  CMDR,PCR        Indirizzo della subroutine CMDR
        STX   ,Y++            Lo deposita nella tabella
        LEAX  CMDM,PCR        Indirizzo della subroutine CMDM
        STX   ,Y++            Lo deposita nella tabella
        LEAX  CMDQ,PCR        Indirizzo della subroutine CMDQ
        STX   ,Y++            Lo deposita nella tabella
```

Questo è l'effettivo salto alla subroutine. X deve contenere l'indirizzo di JTABLE e B l'offset da applicare:

```
DOCMD   JMP   [B,X]
```

SALVA VALORI

```
SAVED   RMB   5               Spazio per i 5 byte da salvare
SAVEIT  LEAX  SAVED,PCR       Preleva l'indirizzo dove salvare i byte
        TFR   S,D             Trasferisce S in D
        ADDD  #2              Vi aggiunge 2 per tener conto dell'indirizzo di ritorno
        STD   ,X++            Lo salva
        LDY   $FFFA           Preleva l'indirizzo del vettore interrupt
        LDA   ,Y+             Estrae il primo byte da salvare
        STA   ,X+             Lo salva
        LDD   ,Y              Estrae gli altri 2 byte
        STD   ,X              Li salva
        RTS
```

COMANDO Q

```
CMDQ    LEAX  SAVED,PCR       Indirizzo del deposito
        LDY   $FFFA           Vettore SWI
        LDA   2,X             Primo dei tre byte
        STA   ,Y+             Deposita il valore
        LDD   3,X             Gli altri 2 byte
        STD   ,Y              Deposita valori originali
        LDS   ,X              Recupera stack originale
        JMP   [$FFFE]         Salto indiretto tramite il vettore reset
```

MODULO PRINCIPALE

```
PROMPT  FCB   ,>
STACKP  RMB   2               Deposito per lo stack pointer
DEBUG   BSR   SAVEIT          Salva valori originali
        BSR   SETUPJ          Prepara la Jump-Table
        BSR   INIT            Prepara le interrupt e accetta indirizzo iniziale
ENTRY   STS   STACKP,PCR      Salva lo stack pointer
        LEAX  JTABLE,PCR
REPT02  LDA   PROMPT,PCR      Preleva il prompt e lo visualizza
        BSR   OUTCH
        BSR   GETCOM          Accetta un comando
        LSLB                  Raddoppia l'offset per la tabella a 16 bit
        BSR   DOCMD           Esegue il comando
        BRA   REPT02          Comando successivo
```

Bracci robotizzati

Panoramica sui prodotti principalmente orientati all'impiego didattico

La prima serie di bracci robotizzati esaminata comprende apparecchi di costo relativamente alto, ma non sufficientemente sofisticati per appartenere alla categoria dei robot industriali. Si tratta dunque di apparecchi adatti, prevalentemente, alla didattica e all'impiego domestico: rispetto agli apparecchi industriali, le dimensioni sia dei bracci sia degli oggetti manipolabili sono minori. La distinzione tra robot domestici e robot industriali, comunque, non ha limiti ben precisi, poiché molto dipende dalla particolare applicazione.

Se si tratta di sollevare e spostare oggetti molto pesanti (come accade in una catena di montaggio) il braccio robot deve essere certamente di tipo industriale, ma per piccoli lavori di precisione (quali il montaggio di circuiti elettronici) può talvolta bastare un braccio del tipo qui esaminato.

Il secondo gruppo di apparecchi comprende invece i robot che rappresentano l'attuale "optimum" nel settore dell'automazione e che sfruttano molti dei sensori descritti in precedenza.

Bracci per insegnare

Un braccio robot che presenta un discreto rapporto prestazioni/prezzo è il Mentor, prodotto dalla Cybernetic Applications: venduto in scatola di montaggio, possiede sei gradi di libertà (vita, spalla, gomito e tre assi di rotazione al polso) e funziona con motori elettrici. Può essere collegato al Micro BBC, al Commodore Vic-20 o allo Spectrum.

La stessa ditta produce il Neptune 1 e il Neptune 2; sono apparecchi più costosi, sebbene sempre in scatola di montaggio, ma che consentono di sollevare pesi fino a 2,5 chilogrammi. La trasmissione avviene con sistemi idraulici funzionanti ad acqua, anziché con i normali fluidi impiegati in questo genere di apparecchi.

Entrambi i modelli funzionano con la stessa gamma di microcomputer. Il Neptune 2, però, possiede una pregevole caratteristica in più: ha due velocità di movimento.

Un altro braccio robot in scatola di montaggio è il Genesis P101, prodotto dalla Powertran Cybernetics: anch'esso è dotato di sei



IAN McKINNELL

Invito per un thè...
Il robot Hero è capace di muoversi e quindi di trasportare oggetti (come una tazza di thè) da un luogo a un altro (purché non ci siano scale di mezzo!). In questa illustrazione Hero serve una tazza di thè al suo collega Mentor



CHRIS STEVENS

IMPARARE... GOMITO A GOMITO

Le due maggiori aree di ricerca, nella microelettronica, sono quelle che riguardano il progetto di nuovi chip e di robot sempre più sofisticati. Da anni gli appassionati di elettronica maneggiano circuiti integrati di vario genere, ma adesso si va diffondendo anche una non indifferente "cultura robotica", nonostante i costi relativamente alti. Scopo principale degli apparecchi qui esaminati è quello di diffondere la conoscenza della robotica: si tratta quindi di robot essenzialmente didattici, da collegare ad un microcomputer. Solo attraverso la pratica si possono mettere a fuoco e superare i limiti che simili apparecchi possiedono attualmente



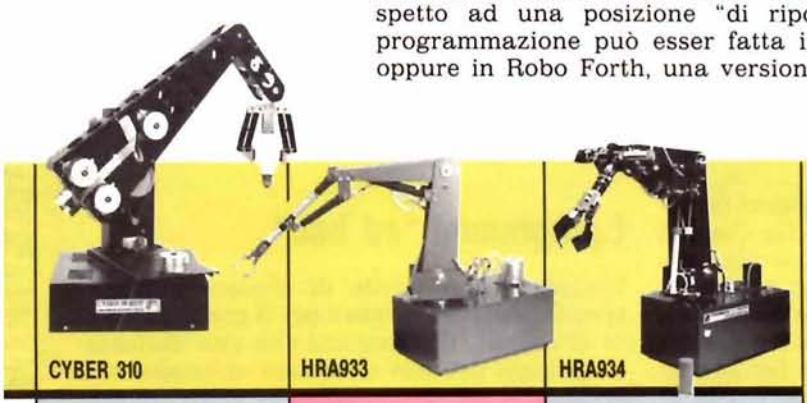






NOME	NEPTUNE 1 / NEPTUNE 2	MENTOR	GENESIS P101	HERO
TIPO	Braccio	Braccio	Braccio	Robot da pavimento
SCOPO PRIMARIO	Didattico	Didattico	Didattico	Didattico/sperimentale
SENSORI	Alcuni potenziometri rilevano la posizione. Il meccanismo di presa può regolare la forza applicata	Alcuni potenziometri rilevano la posizione. Il meccanismo di presa può valutare la forza applicata	Alcuni potenziometri rilevano la posizione. Il meccanismo di presa può regolare la forza applicata	Feedback di posizione	Sensori di movimento a ultrasuoni; i sensori sono in grado di rilevare 256 livelli luminosi o acustici. Sensori tattili sul meccanismo di presa
GRADI DI LIBERTÀ	6: vita, spalla, gomito, elevazione e rotazione del polso, meccanismo di presa	7: vita, spalla, gomito; elevazione, rotazione e stabilizzazione del polso, meccanismo di presa	6: vita, spalla, gomito, elevazione e rotazione del polso, meccanismo di presa	6: vita, spalla, gomito, elevazione e rotazione del polso, meccanismo di presa	4: spalla, gomito, polso e meccanismo di presa
FUNZIONAMENTO	Idraulico: pompe ad acqua azionate dalla corrente di rete	Idraulico: pompe ad acqua azionate dalla corrente di rete	Corrente di rete	Idraulico: pompe ad acqua azionate dalla corrente di rete	Alimentazione a batterie
MICROCOMPUTER COLLEGABILI	Micro BBC, Spectrum e Vic-20	Micro BBC, Spectrum e Vic-20	Micro BBC, Spectrum e Vic-20	Micro BBC, Spectrum, Vic-20 ed altri: il circuito di controllo possiede un'interfaccia seriale RS232	Un'interfaccia seriale consente di collegare qualsiasi micro dotato di software adatto
PRODUTTORE/ DISTRIBUTORE	Cybernetic Applications Ltd. (GB)	Cybernetic Applications Ltd. (GB)	Cybernetic Applications Ltd. (GB)	Powertran Cybernetics (GB)	Zenith (USA)

gradi di libertà e funziona con un sistema idraulico. A corredo è fornita una scatola di controllo per programmare il robot, con un'interfaccia seriale RS232. Ciò significa che il braccio è collegabile alla maggior parte dei microcomputer. Questo apparecchio è venduto anche già montato e collaudato, ma ad un prezzo notevolmente superiore al kit.
Il Cyber 310, prodotto dalla Cyber Robotics e adatto al Micro BBC, allo Jupiter Ace, all'Apple II, ai PET 3000/4000 e 8000 della Commodore, è un altro braccio robot dalle interessanti caratteristiche. Pur non potendo sollevare più di 250 grammi, questo braccio possiede cinque gradi di libertà, motori passo passo e la possibilità di variare l'accelerazione e la decelerazione dei movimenti. Ciò significa che esso è in grado di simulare con più accuratezza il movimento di un braccio umano, ivi compreso l'effetto dell'inerzia. Tutte le articolazioni sono azionabili simultaneamente e le posizioni possono essere specificate sia in modo relativo (rispetto alla posizione precedente) sia in modo assoluto (rispetto ad una posizione "di riposo"). La programmazione può esser fatta in BASIC oppure in Robo Forth, una versione del linguaggio FORTH, sviluppata dalla stessa Cyber Robotics.
Ad un prezzo notevolmente superiore troviamo gli HRA933 e HRA934, della Feedback Instruments. Entrambi funzionano con sistemi idraulici, vengono venduti già montati, possiedono cinque gradi di libertà e sono capaci di sollevare fino a 1,35 kg di peso, con una precisione 3 millimetri nei movimenti. Oltre a sensori di posizionamento nelle articolazioni, questi bracci sono dotati di sensori "tattili" sul meccanismo di presa. Questi ultimi comunicano al computer di controllo se un oggetto è stato afferrato o meno e permettono di graduare la quantità di forza applicata nella presa. Il controllo avviene tramite una porta seriale RS232 e vengono fornite precise indicazioni sull'impiego coi computer Apple II, Tandy TRS-80, Commodore PET, AIM 65 e MAT385.



	CYBER 310	HRA933	HRA934
	Braccio	Braccio	Braccio
	Didattico	Didattico	Didattico
	Nessuno	Capacità di determinare la propria posizione, il meccanismo di presa è in grado di regolare la forza applicata	Capacità di determinare la propria posizione, il meccanismo di presa è in grado di valutare la forza applicata
	6: rotazione della base, della spalla (180°) e del gomito, elevazione e rotazione del polso, meccanismo di presa	5: rotazione della base e del gomito, elevazione e stabilizzazione del polso, meccanismo di presa	5: rotazione della base e del gomito, elevazione e stabilizzazione del polso, meccanismo di presa
	Corrente di rete	Corrente di rete	Corrente di rete
	Micro BBC, Apple II, Jupiter Ace, serie PET della Commodore e Hector HRX	Apple II, Tandy TRS-80, serie PET della Commodore, AIM 65, Micro BBC, MAT 385	Apple II, Tandy TRS-80, serie PET della Commodore, AIM 65, Micro BBC, MAT 385
	Cyber Robotics (GB)	Feedback Systems Ltd. (GB)	Feedback Systems Ltd. (GB)

Lo stato dell'arte

Un robot molto interessante, che forse rappresenta l'attuale "state of art" (ossia il massimo livello qualitativo) di questo genere di prodotti è l'Hero-1, prodotto dalla Zenith Data Systems. È un apparecchio semovente, dotato di un braccio telescopico, con cinque gradi di libertà, che si muove in base a coordinate polari (vedere a pag. 1141). Hero è provvisto di una vasta gamma di sensori per rilevare suoni e fonti di luce e anche di un sistema di rilievo a raggi ultrasonici per evitare ostacoli lungo il cammino. Un sintetizzatore vocale fornisce all'apparecchio un vocabolario definibile dallo stesso utente. La costruzione di Hero richiede una certa abilità ma, considerato che il prezzo della versione già montata e collaudata è incredibilmente superiore, c'è una forte tendenza ad acquistare Hero in scatola di montaggio.
Tutto sommato, quanto offre il mercato è più orientato verso la didattica, che non ad un reale impiego: questi robot sono ottimi strumenti per quanto riguarda l'apprendimento della robotica e della cibernetica, ma in un ambiente casalingo non sono che dei giocattoli costosi. Sempre più spesso vengono impiegati a scopo pubblicitario, in ambiti fieristici o commerciali, per distribuire depliant ed accogliere i visitatori.
Non si creda, tuttavia, che la tecnologia impiegata appartenga ad un livello inferiore: pur non essendo ancora dotati di una vera e propria "vista", questi robot possiedono strumentazioni molto sofisticate e sono in grado di muoversi e di rispondere a comandi in modo decisamente molto "intelligente" (per intelligenza si intende ovviamente quella simulabile su macchine programmate e controllate da computer).
Prevedere che cosa riservi il futuro della robotica, a questi livelli, è molto arduo. Tuttavia, se il mercato si svilupperà con la stessa rapidità di quello dei microcomputer, tra non molto ognuno sarà in grado di possedere un proprio maggiordomo meccanico.

Il software verticale

Un termine gergale che indica programmi orientati verso applicazioni specifiche

Esistono innumerevoli impieghi dei personal computer, talvolta non del tutto intuibili ad un primo esame sommario. Si tratta, in sostanza, di applicazioni molto particolari (per settori professionali specifici, ad esempio) per le quali il software viene redatto appositamente. Nel gergo informatico-commerciale questa parte di utenza è indicata con l'espressione "mercato verticale". Questo termine mette in evidenza la divergenza esistente tra i programmi specifici impiegati e quelli di uso generico. Questa è la prima di una serie di sezioni dedicate agli aspetti che contraddistinguono il software verticale.
Ecco alcuni esempi che, sebbene ricavati da esperienze compiute all'estero, possono aiutare a comprendere la problematica di tale settore.
Un'associazione britannica di genitori impiega il programma *BrainStorm* (della Caxton) per progettare una campagna di sensibilizzazione nei confronti dei propri figli eroinomani, diretta a medici e autorità assistenziali locali. Il proprietario di un noto ristorante impiega lo spreadsheet *Practicalc* per analizzare le ordinazioni dei propri clienti, con lo scopo di ottimizzare le proprie forniture. Una ditta di floricoltura adopera il medesimo programma per elaborare tutti i dati relativi alla propria attività: dalle esportazioni al progetto del risparmio energetico. Un medico chirurgo londinese adopera il programma *Superbase* della Precision Software come strumento di analisi per le ricerche sulle cause del cancro.
Un'applicazione alquanto insolita è quella di un designer inglese, Alan Batton: assieme ad alcuni amici appassionati di biliardo ha realizzato, su di un Micro BBC, un programma in grado di "giostrare" le varie suppellettili all'interno degli ambienti da lui progettati. Il tutto avviene comodamente sullo schermo del BBC.
Questi sono soltanto alcuni esempi di come gli stessi utenti abbiano ideato nuovi usi per i microcomputer. Nonostante ciò, alcune stime affermano che nella maggior parte dei casi l'effettiva potenzialità di un computer è sfruttata appena al 10 per cento. Ne consegue che l'investimento relativo all'acquisto di un computer diviene relativamente improduttivo, se non si tenta di sfruttare maggiormente la versatilità di questi apparecchi.
Le vie da seguire sono due. Una consiste nel ricercare nuove applicazioni per il software generico offerto dal mercato. Esistono già numerosi fotografi che adoperano programmi di archiviazione generici, anche su computer di modeste capacità, per gestire una sorta di banca dati sulle negative e le diapositive scattate nel corso degli anni. Un'altra applicazione abbastanza inconsueta di programmi di archiviazione è quella usata da alcune agenzie matrimoniali: le coppie vengono selezionate in base a criteri di ricerca incrociata e, automaticamente, vengono stampati elenchi pronti ad essere inviati ai vari nominativi. Talvolta ciò risulta più economico che non l'affidarsi a programmi specializzati, quali il *Body Matching and Marketing package*, appositamente creato per casi del genere, ma notevolmente più costoso di un semplice database.

I programmi "ad hoc"

L'alternativa è quella di cercare package specificamente realizzati per il proprio settore di attività. Supponiamo che uno studente di teologia desideri effettuare un'analisi sul testo della Bibbia protestante: esiste un package della Bible Research Systems, chiamato *The Word*, che contiene un'intera versione della Bibbia oltre ad alcuni programmi di ricerca incrociata sul testo memorizzato (il costo del package, che funziona soltanto su apparecchi compatibili con l'IBM , non è però alla portata di tutti). Simili applicazioni esistono, in italiano, per la ricerca su testi danteschi.
Per fare un altro esempio di programmi altamente specializzati, esiste un package chiamato *Miduss*, per la progettazione di impianti idraulici (scarichi, fogne, tubature e via dicendo). Il programma è composto anche da un modulo grafico, che consente di visualizzare sullo schermo i vari progetti. Anche in tal caso il costo del programma è notevole. I sistemi idraulici di scarico sembrano aver fortemente stimolato la produttività dei programmatori anglosassoni, tant'è vero che in Gran Bretagna un progetto computerizzato di rilevazione, basato su minuscoli microcomputer "da campo", è stato esteso a livello nazionale: il tecnico o l'ispettore, dopo aver immesso i dati nell'elaboratore portatile, li trasmette via telefono all'unità centrale. Ciò consente la stampa selettiva di tabulati riguardanti, ad esempio, tutte le fognature in mattone costruite durante un arco di anni specificato.

Proseguendo nel settore dei portatili, si è visto un proliferare di programmi orientati verso utenti che, per ragioni di lavoro, sono costretti a frequenti spostamenti: oltre ai soliti programmi di elaborazione di testi (word processor) esiste tutta una schiera di package (*Travelling Project Manager, Travelling Appointment Manager, Travelling Sales Manager* e via dicendo) indirizzata ai rappresentanti di commercio.

Anche i commercianti possono avvalersi di programmi altamente specializzati, da usare per migliorare e finalizzare meglio le proprie tattiche di vendita. Un esempio è il modulo *Sales Edge*, che fa parte del package *Human Edge* (disponibile su IBM e Apricot). *Sales Edge* suggerisce una strategia di vendita da adottare per ciascun cliente, in base ai dati caratteristici della ditta venditrice. Di nuovo, il prezzo di simili programmi può sconcertare, poiché può raggiungere quello dello stesso computer. D'altra parte, si tratta di un investimento che, se ben gestito, può essere ammortizzato in breve tempo.

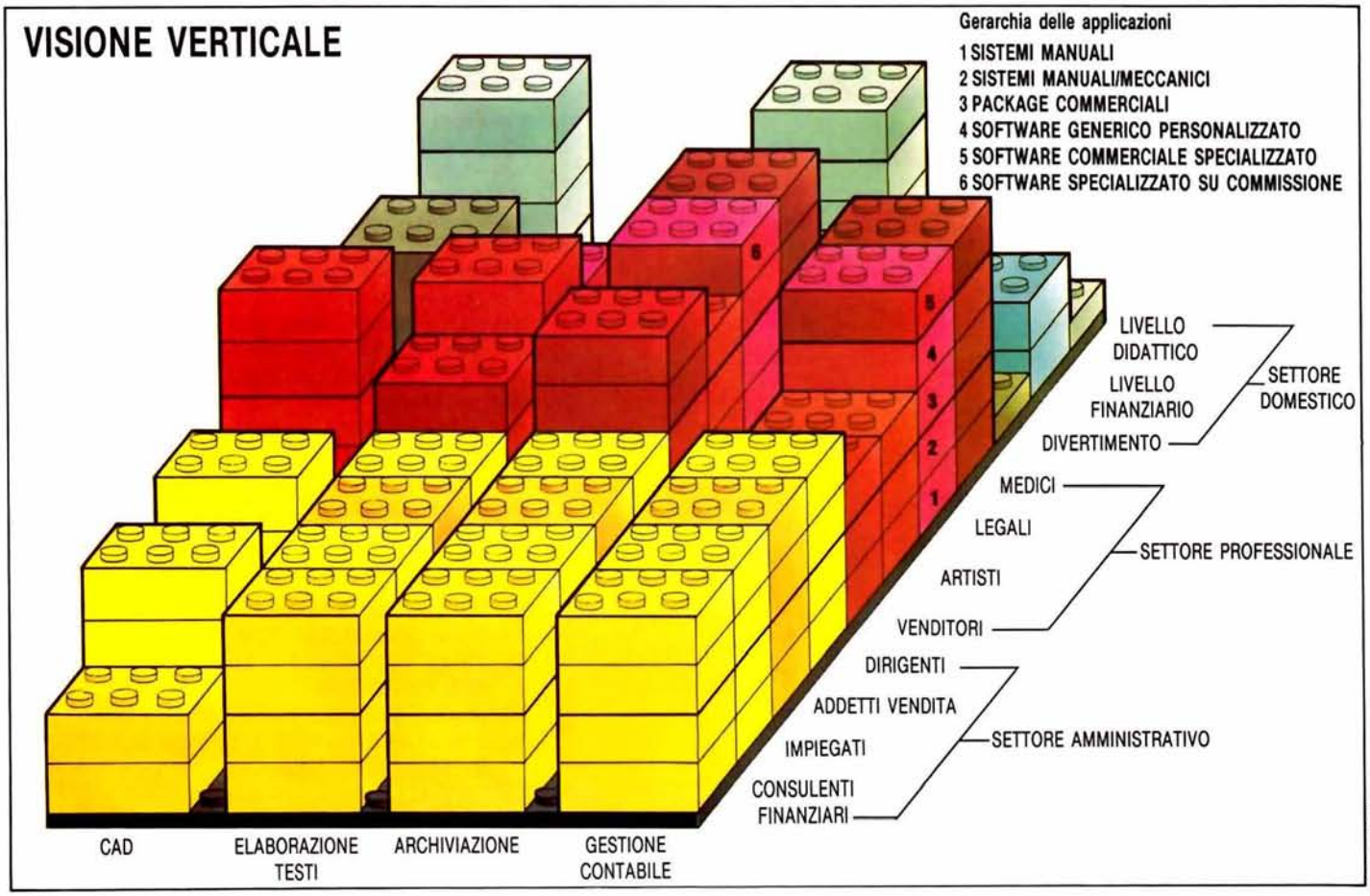
Per chi è interessato agli investimenti in valuta, esistono programmi molto specifici, in grado di fornire "storia" e modelli di confronto sull'andamento del Dollaro, della Sterlina, del Franco svizzero, dello Yen e del Marco tedesco. Il programma produce 37 tabulati comparativi, indici di forza relativi, tassi d'interesse ed altri dati per ciascun giorno a partire dal 1 Ottobre 1983.

L'organizzazione di impegni e di appuntamenti merita infine un capitolo a parte. La maggior parte dei computer più recenti è dotata di orologi interni, benché siano pochi ancora i programmi "intelligenti" (come quello, su ROM, progettato per l'HP-75C della Hewlett-Packard), in grado di interrompere quanto si sta facendo per ricordare un appuntamento o una scadenza.

Nella prossima sezione dedicata al software verticale vengono prese in esame le caratteristiche del package *BrainStorm*, un programma famoso per esser capace di organizzare efficacemente i pensieri così come un word processor elabora e riorganizza le parole di un testo. In seguito, vengono presi in considerazione altri package per applicazioni specifiche (mediche, commerciali, di ricerca e via dicendo).

Profili di mercato

Il software viene prodotto per soddisfare le esigenze di gruppi primari di utenza, in modo da essere commerciabile su vasta scala. I settori più nutriti dunque, comprendono package per l'elaborazione di testi, per la gestione contabile e per l'archiviazione. Un quarto settore, in costante crescita, è quello del CAD (Computer Aided Design), ossia del disegno assistito da computer. Nel diagramma vengono illustrate le varie aree di utilizzazione, suddivise per categorie di utenza. L'asse verticale rappresenta il livello di complessità della particolare applicazione (ad un'altezza minore corrispondono procedure manuali, alla massima i programmi commissionati "ad hoc"). Appare subito evidente che i programmi commerciali "già pronti" soddisfano le esigenze di utenti generici, ma che per applicazioni particolari occorre adattare programmi esistenti o crearne di nuovi.



Raccogliere oggetti

Le routine per prelevare e depositare oggetti lungo il percorso

A pag. 1293 abbiamo visto come suddividere e analizzare i comandi, creando anche un gruppo di comandi "normali". Tra questi abbiamo incluso i comandi PRENDI e LASCIA, assieme alle varianti RACCOGLI e DEPOSITA. Una volta riconosciuto un particolare comando, possiamo realizzare le routine che gli obbediscono. Consideriamo il comando PRENDI.

Per meglio comprendere il funzionamento della routine, è bene ricordare come il programma tenga conto della posizione dei vari oggetti nel mondo dell'avventura. Inizialmente abbiamo definito le frasi DATA che contengono la descrizione di ciascun ambiente, i nomi degli oggetti presenti e informazioni sulle vie d'uscita dall'ambiente. Eseguita la lettura delle DATA, gli elementi della matrice IV$(,), che nel gioco *Foresta Incantata* contiene i dati sugli oggetti, saranno:

N	IV$(N,1)	IV$(N,2)
1	PISTOLA	10
2	TORCIA	9
3	CHIAVE	5

La prima colonna della matrice contiene i nomi degli oggetti, la seconda i numeri corrispondenti alla loro collocazione iniziale nell'avventura. Durante la fase della descrizione, la seconda colonna della matrice viene esaminata per estrarre i nomi degli oggetti presenti nell'ambiente visitato (variabile P). Affinché il giocatore possa prelevare un oggetto e trasportarlo altrove (con un comando del tipo PRENDI LA LAMPADA), occorre prima considerare una serie di fattori:

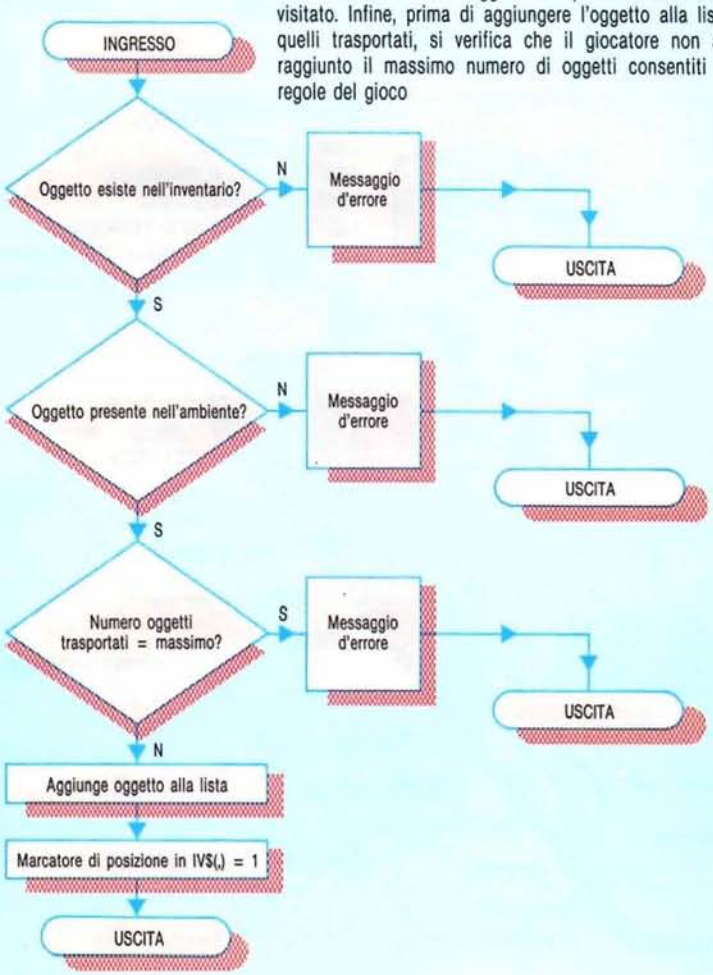
- La validità del nome dell'oggetto citato: va verificato se esso compare in IV$(,), la matrice dell'inventario.
- La presenza dell'oggetto nell'ambiente.
- La capacità di poter raccogliere l'oggetto (il giocatore può aver superato il massimo numero di oggetti trasportabili consentito dalle regole).

Se tutte queste condizioni sono soddisfatte, allora al giocatore è permesso prelevare l'oggetto in questione. Ciò comporta un'aggiunta alla lista degli oggetti trasportati, IC$(), e l'eliminazione del marcatore di posizione nella matrice IV$(,). Si noti che il nome dell'oggetto non va affatto cancellato. Se, quando si raccoglie un oggetto, il relativo marcatore di posizione viene sostituito con il valore – 1, la sua descrizione non viene visualizzata assieme a quella dell'ambiente visitato. Infatti, sarebbe piuttosto illogico se, prelevando una PISTOLA dall'ambiente 10, uscendo e poi rientrando in seguito ritrovassimo tale oggetto ancora al suo posto nell'ambiente 10. La matrice IV$(,) tiene quindi conto della posizione degli oggetti non trasportati dal giocatore. Il diagramma di flusso della routine PRENDI illustra la semplice logica da applicare.

```
3700 REM *** S/R PRENDI ***
3710 GOSUB 5300:REM OGGETTO VALIDO?
3720 IF F=0 THEN SN$=W$ + "NON ESISTE"
     :GOSUB5500:RETURN
3730 OV=F:GOSUB5450:REM VERIFICA INVENTARIO
3740 IF HF=1 THEN SN$=IV$(F,1) + "E' GIA' IN TUO POSSES-
     SO":GOSUB5500:RETURN
3750 :
3755 REM ** L'OGGETTO E' QUI? **
3760 IF VAL(IV$(F,2))<>P THEN SN$=IV$(F,1)+" NON E' QUI":
     GOSUB 5500:RETURN
```

LA ROUTINE PRENDI

Il diagramma di flusso di questa routine illustra le verifiche eseguite sul comando immesso dal giocatore. Il test di validità cerca la presenza dell'oggetto nell'inventario, poi viene controllato che l'oggetto sia presente nell'ambiente visitato. Infine, prima di aggiungere l'oggetto alla lista di quelli trasportati, si verifica che il giocatore non abbia raggiunto il massimo numero di oggetti consentiti dalle regole del gioco



GARY CAPPS-JENNER

```
3770 :
3780 REM ** AGGIUNGE OGGETTO ALL'ELENCO **
3790 A=0
3800 FOR J=1 TO 2
3810 IF IC$(J)="" THEN IC$(J)=IV$(F,1):AF=1:J=2
3820 NEXT J
3830 :
3840 REM ** TROPPI OGGETTI **
3850 IF AF=0 THEN PRINT "HAI GIA' DUE OGGETTI":RETURN
3860 :
3870 SN$="HAI RACCOLTO "+IV$(F,1):GOSUB 5500
3880 IV$(F,2)="-1":REM CANCELLA OGGETTO DA INVENTA-
     RIO
3890 RETURN
```

Esaminiamo, una alla volta, le tre verifiche.

Il test di validità

La verifica più importante (e più difficile) è quella sulla validità dell'oggetto citato nel comando. Nella sua forma più semplice potrebbe consistere in una routine che, usando la seconda parte del comando, la confronti con i nomi contenuti nella matrice IV$(,) dell'inventario. Questo metodo, però, limiterebbe i comandi al formato PRENDI OGGETTO, per cui la semplice variazione PRENDI LA LAMPADA non verrebbe riconosciuta valida. Per aumentare l'elasticità dei comandi occorre dunque ideare una routine più sofisticata.

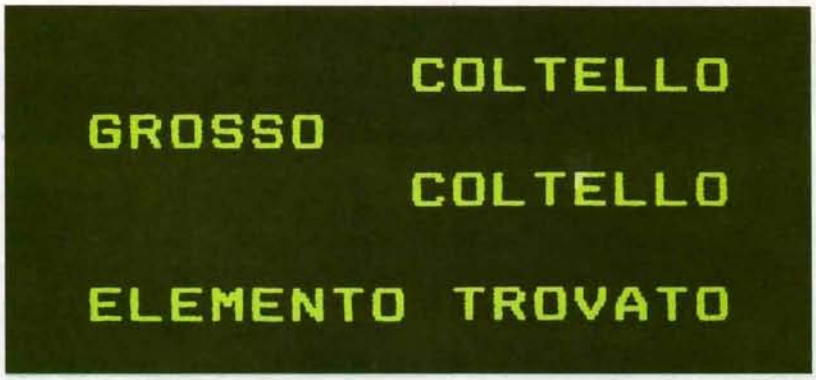
Un sistema può essere quello di suddividere ulteriormente la seconda parte del comando nelle singole componenti, per poi confrontare ciascuna di queste con il contenuto dell'inventario. Benché migliore del precedente, anche questo sistema presenta degli inconvenienti. Se, cioè, si volesse impartire un comando del tipo PRENDI IL GROSSO COLTELLO, la routine andrebbe a cercare nell'inventario, separatamente, le parole IL, GROSSO, e COLTELLO, senza trovare l'elemento GROSSO COLTELLO. Tuttavia, si può aggirare l'ostacolo: invece di tentare un confronto tra le singole parole del comando, la routine può scandire la parola cercata lettera per lettera, finché viene trovata una corrispondenza per intero oppure termina il nome dell'oggetto. A fianco viene fornita una routine dimostrativa di questo metodo; un altro vantaggio è che i nomi degli oggetti, nel comando, possono assumere anche una forma abbreviata. Ad esempio PRENDI IL COLTELLO può essere abbreviato in PRENDI IL COLT (purché non esistano altri nomi di oggetti che con la medesima combinazione di lettere COLT). Una scelta oculata dei nomi per gli oggetti, e della loro collocazione nella matrice dell'inventario, può portare ad una semplificazione nei comandi. Se due parole contengono lo stesso gruppo di lettere (COLT può riferirsi ad una pistola, ma anche formare parte di COLTELLO), basta collocare la parola più breve prima di quella più lunga. Inoltre, conviene evitare descrizioni di oggetti che contengono parole identiche, come GROSSO COLTELLO, quando esiste già COLTELLO.

```
5300 REM *** S/R OGGETTO VALIDO? ***
```

L'OGGETTO ESISTE?








```
  10 REM *** PROGRAMMA DIMOSTRATIVO ***
  40 MODE 5:COLOUR  2:DIM V$(3)
  60 FOR I=1 TO 3:READ V$(I):NEXT I
  90 A$="IL":B$="COLTELLO":C$="COLT"
  95 S$="
 110 M$=A$:GOSUB 1000:REM TENTA CON "IL"
 130 M$=B$:GOSUB 1000:REM TENTA CON "COLTELLO"
 150 M$=C$:GOSUB 1000:REM TENTA CON "COLT"
 160 END
1000 REM *** S/R DI ABBINAMENTO ***
1010 CLS:F=0:LW=LEN(M$)
1030 FOR J=1 TO 3:LI=LEN(V$(J))
1042 X=1:Y=6:GOSUB2000:PRINT S$
1045 X=1:Y=6;:GOSUB2000:PRINT V$(J)
1047 FOR I=1 TO LI-LW+1
1050 X=1:Y=5:GOSUB2000:PRINT S$
1060 X=I:Y=5:GOSUB2000:PRINT M$
1070 IF MID$(V$(J),I,LW)=M$ THEN F=I:I=LI:J=3
1075 FOR D=1 TO 300:NEXT D,I:REM DELAY
1086 IF F<>0 THEN GOSUB 2500
1087 NEXT J:RETURN
2000 REM *** POSIZIONA CURSORE A X,Y ***
2010 PRINT TAB(X,Y);:RETURN
2500 REM *** CORRISPONDENZA TROVATA ***
2502 COLOUR 1:X=F:Y=6:GOSUB2000:PRINT M$
2505 FOR K=1 TO 5:X=1:Y=10:GOSUB2000:PRINT S$
2508 FOR D=1 TO 500:NEXT:REM CICLO DI RITARDO
2510 X=1:Y=10:GOSUB2000:PRINT "ELEMENTO TROVATO"
2512 FOR D=1 TO 500:NEXT D,K:REM CICLO DI RITARDO
2520 A$=GET$:COLOUR 2:RETURN
3000 REM *** DATI PER L'INVENTARIO ***
3005 DATA "PICCOLA FORCHETTA","PORTA ROSSA","GROSSO COLTELLO"
```

La subroutine per il test di validità, da usare all'interno della routine PRENDI, esegue una scansione lettera per lettera del comando immesso per trovare una corrispondenza nell'inventario degli oggetti. Questo breve programma dimostrativo illustra il criterio di confronto adottato. In questa dimostrazione l'inventario contiene tre oggetti e il programma tenta il confronto di tre parole: "IL", "COLTELLO" e "COLT". Quando viene trovata una corrispondenza, il programma attende la pressione di un tasto

```
  10 REM *** VARIAZIONI PER LO SPECTRUM ***
  40 INK 6:DIM V$(3,20)
1070 IF V$(J,I TO I+LW-1)=M$ THEN F=I:I=LI:J=3
2010 PRINT AT (Y,X);:RETURN
2502 INK 2:X=F:Y=6:GOSUB2000:PRINT S$
2520 A$=INKEY$:IF A$="" THEN 2520
2525 INK 6:RETURN
```

```
  10 REM ***  VARIAZIONI PER IL CBM 64  ***
  40 PRINT CHR$(158):DIM V$(3)
  50 DN$=CHR$(17):FOR K=1 TO 5:DN$=DN$+DN$:NEXT:DN$=CHR$(19)+DN$
2010 PRINT LEFT$(DN$,Y)TAB(X);:RETURN
2502 PRINTCHR$(28):X=F:Y=6:GOSUB2000:PRINT S$
2520 GETA$:IF A$="" THEN 2520
2525 PRINT CHR$(158):RETURN
```

```
5310 NN$ = NN$ + " ":LN = LEN(NN$):F = 0:C = 1
5315 FOR K = 1 TO LN
5320 IF MID$(NN$,K,1) < > " "THEN NEXTK:RETURN
5325 W$ = MID$(NN$,C,K—C):C = K + 1
5330 LW = LEN(W$)
5335 FORJ = 1 TO 3
5340 LI = LEN(IV$(J,I)):REM LUNGH. OGGETTO
5350 FOR I = 1TO LI—LW + 1
5360 IF MID$(IV$(J,1),I,LW) = W$ THEN F = J:I = LI:J = 3:K = LN
5370 NEXT I,J,K
5380 RETURN
```

Se trova una corrispondenza, la routine assegna al flag F il valore dell'oggetto in questione, altrimenti il valore di F rimane zero a indicare che l'oggetto non esiste nel gioco.

La presenza dell'oggetto

Verificata l'esistenza dell'oggetto, occorre adesso controllare la sua presenza nell'ambiente visitato, il cui codice è in P. L'oggetto da raccogliere è IV$(F,1) e la sua collocazione è IV$(F,2). La linea 3760 della routine PRENDI, in *Foresta Incantata*, verifica l'identità tra questo valore e la variabile P. Tuttavia, l'apparizione del messaggio «L'oggetto non è qui» può non risultare del tutto giustificata, se prima non si controlla che l'oggetto non sia tra quelli trasportati dal giocatore. In tal caso va visualizzato un diverso messaggio, ad esempio: «L'oggetto è tra quelli trasportati». La seguente routine esegue una tale verifica, ponendo il flag HF a 1 se l'oggetto è già in possesso del giocatore. Questa condizione si rileva dalla presenza di un — 1 nell'elemento della matrice IV$(,) interessato.

```
5450 REM *** S/R OGGETTO TRASPORTATO? ***
5460 HF = 0
5470 IFIV$(OV,2) = "—1"THEN HF = 1
5480 RETURN
```

Gli oggetti trasportati

Un'unica routine controlla il numero degli oggetti trasportati dal giocatore usando la matrice IC$( ) per contenere l'elenco degli oggetti trasportati, basta un ciclo FOR...NEXT per trovare il primo posto vuoto, ove collocare il nuovo oggetto. Le regole di *Foresta Incantata* stabiliscono che il giocatore può portare con sé soltanto due oggetti alla volta, quindi il ciclo FOR...NEXT è usato ed eseguito soltanto due volte. Se c'è un posto vuoto, il nuovo oggetto viene inserito, altrimenti viene visualizzato un messaggio appropriato. L'ultimo compito della routine, se l'oggetto è trasportabile, consiste nel modificare il marcatore di posizione nell'inventario. Ciò si ottiene semplicemente ponendo a — 1 l'elemento IV$(F,2).

Quali oggetti ho con me?

Il giocatore ha ora la facoltà di raccogliere oggetti e possiamo aggiungere un comando per avere un elenco di quelli trasportati. Infatti, spesso capita di essere talmente presi dal gioco da dimenticare quali oggetti si hanno con sé. La routine è semplice: si limita a visualizzare, tramite un ciclo FOR...NEXT, il contenuto della matrice IC$( ):

```
4100 REM *** S/R VISUALIZZA INVENTARIO ***
4110 PRINT "OGGETTI TRASPORTATI:"
4120 FOR I = 1 TO 2
4130 PRINT" ";IC$(I)
4140 NEXT I
4150 RETURN
```

LISTATO DIGITAYA

```
2140 REM *** S/R PRENDI ***
2145 IV$(4,1)="BIGLIETTO PER IL TRI-STATO"
2150 GOSUB5730:REM OGGETTO VALIDO?
2160 IF F=0 THEN PRINT W$;" NON ESISTE":RETURN
2170 REM *** OGGETTO GIA' RACCOLTO? ***
2180 OV=F:GOSUB5830
2190 IFHF=1 THEN SN$="HAI GIA' RACCOLTO "+IV$(F,1):GOSB5880:RETURN
2200 :
2210 REM *** L'OGGETTO E' QUI? ***
2220 IF VAL(IV$(F,2))<>P THENSN$=IV(F,1)+" NON E' QUI":GOSUB5880
2230 :
2240 REM ** AGGIUNGE OGGETTO ALL'ELENCO **
2250 AF=0:FOR J=1TO4
2260 IFIC$(J)=""THENIC$(J)=IV$(F,1):AF=1:J=4
2270 NEXT J
2280 :
2290 REM *** TROPPI OGGETTI ? ***
2300 IF AF=0THENPRINT"HAI GIA' 4 OGGETTI":RETURN
2310 :
2320 SN$="HAI RACCOLTO "+IV$(F,1):GOSUB5880
2330 IV$(F,2)="-1":REM RIMUOVE OGGETTO DALL'ELENCO
2340 RETURN


5730 REM *** S/R OGGETTO VALIDO ***
5740 NN$=NN$+" ":LN=LEN(NN$):F=0:C=1
5745 FOR K=1 TO LN
5750 IF MID$(NN$,K,1)<>" " THEN NEXTK:RETURN
5755 W$=MID$(NN$,C,K-C):C=K+1:LW=LEN(W$)
5760 FORJ=1 TO 8
5770 LI=LEN(IV$(J,I)):REM LUNGH. OGGETTO
5780 FOR I=1TO LI-LW+1
5790 IFMID$(IV$(J,1),I,LW)=W$THENF=J:I=LI:J=8:K=LN
5800 NEXT I,J,K
5810 RETURN
5820 :
5830 REM *** S/R OGGETTO TRASPORTATO? ***
5840 HF=0
5850 IFIV$(OV,2)="-1"THEN HF=1
5860 RETURN


2540 REM *** S/R VISUALIZZA INVENTARIO ***
2550 PRINT "OGGETTI TRASPORTATI:"
2560 FOR I=1TO4
2570 PRINT"  ";IC$(I)
2580 NEXT I
2590 RETURN
```

Basic & Basic

Spectrum:

Modifiche per "Foresta Incantata":

Sostituire SN$ con S$ IV$(,) con V$(,) IC$( ) con I$( )
NN$ con R$.

```
5320 IF R$(K TO K)< > " " THEN NEXT K:RETURN
5325 LET W$ = R$(C TO K—1)
5360 IF V$(I TO I + LW—1) = W$ THEN LET F = J:LET
     I = LI:LET J = 3:LET K = LN
```

Per il listato di "Digitaya", cambiare i nomi delle variabili come descritto sopra e modificare le linee 5750, 5755 e 5790 in modo analogo alle linee riportate qui sopra.

L'Osborne Encore

Anatomia di un microcomputer a diffusione europea: tecnologie a confronto



CHRIS STEVENS

Un compatto portatile

L'Osborne Encore è uno dei primi computer IBM compatibili ad esser dotato di uno schermo a cristalli liquidi. Durante il trasporto la tastiera viene ripiegata sullo schermo formando così una valigetta compatta e robusta. Nella foto è visibile anche la cinghia da usare come tracolla

L'Osborne-1, il primo computer compatto trasportabile, produsse alla sua uscita una vera e propria rivoluzione. Dotato di un monitor, di doppio disk drive e di interfacce di vario tipo (il tutto montato a formare un'elegante valigetta) è stato il primo microcomputer commerciale "tutto compreso" a funzionare sotto il sistema operativo CP/M. Ben presto la concorrenza, vedendo enormi potenzialità in prodotti di questo tipo, si è data da fare creando non pochi problemi alla Osborne, che dovette assorbire il colpo peggiore quando la IBM annunciò la produzione di una propria serie di personal computer. A poco valse il lancio dell'Osborne Executive, un apparecchio con doppio microprocessore e funzionante con i sistemi operativi CP/M e MS–DOS: la scarsità sul mercato del chip 8086 (che avrebbe reso l'Executive compatibile con l'IBM) determinò il crollo commerciale della Osborne, che si pose in liquidazione volontaria nel 1983.

Tuttavia, araba fenice dell'informatica, questa ditta si è di nuovo lanciata, dopo una sostanziale operazione di "cosmesi aziendale", nella produzione dell'Encore.

Benché non altrettanto compatto quanto il PX-8 della Epson, l'Encore riempie il vuoto esistente tra i computer "da viaggio" e quelli "da tavolo".

L'Encore pesa circa sei chilogrammi ed è molto più maneggevole dei suoi predecessori: il tutto ha dimensioni pargonabili a tre elenchi telefonici sovrapposti; l'involucro, dotato di una robusta cinghia che ne agevola il trasporto, è in resistente plastica grigia; la tastiera, che si ribalta a formare il coperchio laterale dell'apparecchio proteggendo lo schermo a cristalli liquidi, ha la classica disposizione QWERTY, ma è dotata di una serie di tasti a membrana per quanto riguarda speciali funzioni e simboli relativi ai programmi incorporati. In basso a destra sono collocati i tasti per il movimento del cursore e sono presenti due tasti Control alle due estremità laterali. Per poter raccogliere in così poco spazio, tutte le funzioni compatibili con l'IBM sono stati necessari alcuni compromessi; le funzioni della tastiera numerica, che sull'IBM è separata dal resto, sono qui ottenute sfruttando una sezione della tastiera principale, contraddistinta da simboli in blu, mentre per il resto dei tasti è stato usato il colore bianco. Per abilitare la tastiera numerica

dell'Encore, si agisce su uno dei tasti iconografici posti in alto rispetto alla tastiera principale. Le icone riportate su questi tasti si riferiscono alle funzioni di calcolatrice, al modem, ai dischi e al calendario incorporato. La presenza di questa serie di tasti a membrana sembra sminuire l'effetto molto professionale dell'apparecchio e ricorda vagamente la tastiera del vecchio ZX81: non si è mai del tutto certi che la pressione sul tasto sia sufficiente a chiudere il contatto.
Lo schermo a cristalli liquidi (cm. 23.7 x8) garantisce un basso consumo, ma pone alcuni problemi riguardo alla compatibilità con lo schermo dell'IBM: non tanto nella colorazione (riproducibile parzialmente per mezzo di toni grigi), quanto per le dimensioni; quello dell'IBM, infatti, prevede una risoluzione di 25 per 80 caratteri, mentre il formato dello schermo dell'Encore è di sole 16 righe di 80 caratteri (ciò pare sia dovuto a difficoltà di realizzazione della ditta giapponese che costruisce i display). Di fatto, quindi, non tutti i programmi scritti per l'IBM sono utilizzabili sull'Encore: finché il display funziona in modo "scroll" tutto scorre bene, ma le difficoltà sorgono se è prevista una visualizzazione a pagine, nella quale i prompt appaiono di solito in fondo allo schermo.
Lateralmente c'è l'alloggio per due unità a dischetti, benché la versione standard preveda una sola unità. Sul lato opposto c'è la presa di alimentazione, l'interruttore d'accensione, un comando per regolare l'intensità dello schermo e l'alloggiamento per le batterie ricaricabili (al nichel-cadmio), che permettono di usare il computer anche lontano da prese di corrente.

Avviamento del sistema

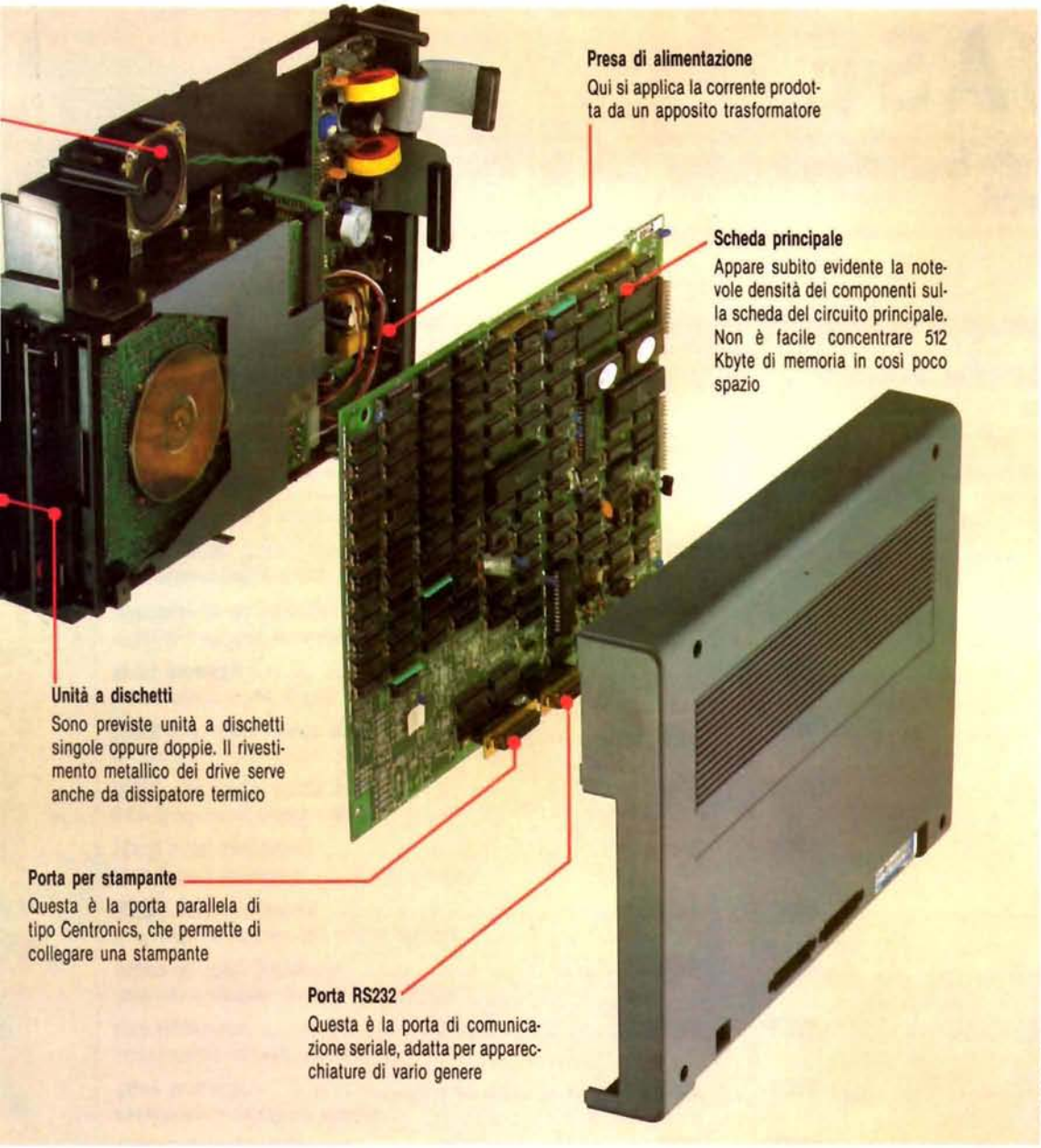
Dietro lo schermo si trovano i connettori di ingresso/uscita, da sinistra a destra: la presa telefonica per il modem incorporato, un'interfaccia parallela per stampante di tipo Centronics e una seriale di tipo RS232.
Per avviare il sistema operativo MS-DOS basta premere, sulla tastiera a membrana, l'icona del disco. I programmi incorporati si possono richiamare in qualsiasi momento, indipendentemente dal programma richiamato da dischetto.
Alcune prove con dischetti per l'IBM non hanno presentato problemi di caricamento, ma in certi casi i programmi (per le ragioni esposte in merito al display) non funzionano correttamente, oppure è necessario immettere comandi alla cieca, dato che il relativo prompt non è visibile. Tuttavia, un programma come il *Lotus 1-2-3*, notoriamente difficile da far funzionare sugli IBM compatibili (per le complesse chiamate alle routine incorporate nell'IBM), funziona perfettamente.
Un ultimo inconveniente, che può influire sul futuro commerciale dell'Encore è la dimensione dei caratteri, molto piccoli rispetto a quelli normalmente impiegati sui display a cristalli liquidi.





FIANCO APERTO ... AI DISCHI

L'idea di fissare lateralmente le unità a dischetti rende tutto l'apparecchio molto compatto. Tuttavia, è scomodo controllare quale drive sia in funzione, poiché le spie non sono visibili dalla normale posizione di lavoro. A seconda della versione, sull'Encore vengono montate una oppure due unità a dischetti da 5 pollici



Presa di alimentazione
Qui si applica la corrente prodotta da un apposito trasformatore

Scheda principale
Appare subito evidente la notevole densità dei componenti sulla scheda del circuito principale. Non è facile concentrare 512 Kbyte di memoria in così poco spazio

Unità a dischetti
Sono previste unità a dischetti singole oppure doppie. Il rivestimento metallico dei drive serve anche da dissipatore termico

Porta per stampante
Questa è la porta parallela di tipo Centronics, che permette di collegare una stampante

Porta RS232
Questa è la porta di comunicazione seriale, adatta per apparecchiature di vario genere

CHRIS STEVENS

OSBORNE ENCORE

DIMENSIONI

mm. 241 x 325 x 141

CPU

8086 a 16 bit

MEMORIA

La versione standard possiede 512 Kbyte di RAM

VIDEO

A cristalli liquidi, con 16 linee di 80 caratteri (480 x 128 pixel). La Osborne ha annunciato anche uno schermo LCD di 25 x 80, facilmente installabile sul modello attuale

INTERFACCE

Interfaccia seriale RS232, connettore per telefono (tipo RJ11) e porta parallela tipo Centronics

LINGUAGGI FORNITI

BASIC della Microsoft fornito su dischetto

TASTIERA

Di tipo QWERTY, con 62 tasti e tastierina supplementare a membrana per 10 funzioni programmabili e 4 predefinite

DOCUMENTAZIONE

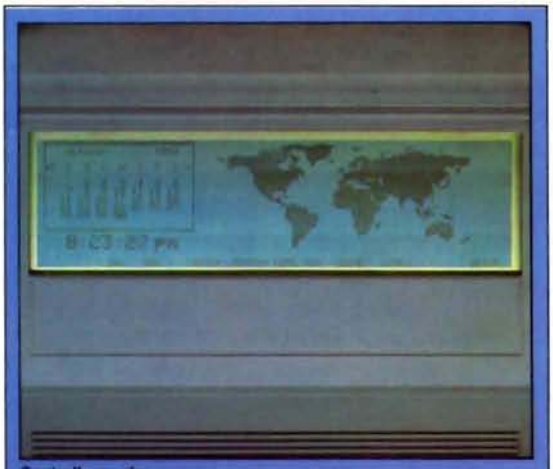
I due manuali forniti in dotazione sono, come di consueto per la Osborne, redatti con molta cura e abbondanza di particolari sul funzionamento dell'apparecchio

PRO

L'Encore è un computer molto compatto, che ben si adatta sia all'uso in postazione fissa che durante spostamenti da un luogo all'altro (l'autonomia è di circa quattro/cinque ore)

CONTRO

Lo schermo, realizzato con cristalli liquidi, non è totalmente compatibile con il formato del PC IBM. Inoltre i caratteri, piuttosto piccoli, sono difficili da leggere



Controllo orario
Il programma agenda/calendario incorporato consente di organizzare appuntamenti o programmare scadenze. Inoltre, è possibile esaminare o modificare l'ora dell'orologio in base ai diversi fusi orari. Si noti la leggera scanalatura sotto il display, che dovrebbe servire per montare uno schermo LCD da 25 linee x 80 caratteri



Tasti a membrana
Sopra la tastiera principale c'è una fila di tasti a membrana (simili a quelli adoperati sullo Spectrum e sullo ZX81): la prima fila serve per le 10 funzioni programmabili, mentre la seconda contiene le icone abbinate al funzionamento dei programmi incorporati

DATABASE

Ecco la prima tabella sulle funzioni delle routine ROM del Micro BBC, assieme ai relativi indirizzi.

La tabella è ordinata alfabeticamente ma, per comodità di consultazione, a fianco è riportata anche la tabella in ordine crescente di indirizzo.
Si noti che gli indirizzi sono espressi in forma esadecimale.

DESCRIZIONE	INDIRIZZO
! binary operator operatore binario !	967F
! unary operator operatore di unità !	95A5
$ unary operator operatore di unità $	95B0
*** operator routine** routine operatore *	9D3C
*** statement routine** routine istruzione *	8B73
+ operator routine routine operatore +	9C4E
— operator routine routine operatore —	9CB5
¯ unary operator operatore di unità ¯	AD8C
/ operator routine routine operatore /	9DE5
< operator routine routine operatore <	9BC0
<= operator routine routine operatore <=	9BD4
<> operator routine routine operatore <>	9BDF
= operator routine routine operatore =	9BAE
= statement routine routine operatore =	8B47
> operator routine routine operatore >	9BE8
>= operator routine routine istruzione >=	9BFA
? binary operator operatore binario ?	967B
? unary operator operatore di unità ?	95A7
ABS function routine routine funzione ABS	AD6A
ACS function routine routine funzione ACS	A8D4
ADVAL function routine routine funzione ADVAL	AB33

INDIRIZZO	DESCRIZIONE
8000	**ROM entry point** indirizzo d'ingresso alla ROM
8005	**ROM header** indirizzo testata della ROM
8023	**Initialisation of language** inizializzazione del linguaggio
8071	**Keyword table** tabella parole chiave (token)
836D	**Keyword action addresses (LSB)** indirizzo tabella azioni (parte bassa)
83EO	**Keyword action addresses (MSB)** indirizzo tabella azioni (parte·alta)
8451	**Mnemonic table (LSB)** indirizzo tabella cod. mnemonici (parte bassa)
848B	**Mnemonic table (MSB)** indirizzo tabella cod. mnemonici (parte alta)
84C5	**Assembler base op-codes** indirizzo iniziale tabella op-code (assembler)
84FD	**Assembler exit** indirizzo d'uscita dell'assembler
8504	**Assembler entry** indirizzo d'ingresso dell'assembler
85A5	**Assembler label definition** indirizzo definizioni etichette (assembler)
85BA	**Assemble a single instruction** assemblaggio singola istruzione
8821	**Evaluate integer expression** calcolo espressioni (intere)
883A	**EQUX in assembler** direttiva EQUX (assembler)
887C	**Delete bytes in buffer** cancella i byte nel buffer
889C	**Convert a line number to binary** conversione in binario del numero di linea
88DB	**Tokenise line numbers** 'tokenizza' i numeri di linea
88F3	**Line number encryption** codifica dei numeri di linea
8926	**Check if A is alphanumeric** verifica se A è numerico
8951	**Tokenise a line of text** 'tokenizza' una linea di testo

Procedure autogeneranti

Come aggiungere nuovi comandi e realizzare procedure che creano nuove procedure

La primitiva RUN del LOGO accetta in ingresso una lista, trattandola poi come una procedura da eseguire. Questa particolarità può servire per creare facilmente nuove strutture di controllo come e quando occorrono. Ecco come definire strutture del tipo WHILE:

```
TO WHILE :CONDIZIONE :AZIONE
   IF NOT (RUN :CONDIZIONE) THEN STOP
   RUN :AZIONE
   WHILE :CONDIZIONE :AZIONE
END
```

Ecco un'applicazione concreta di tale struttura. POTENZA visualizza tutti i quadrati dei numeri immessi minori di 1000:

```
TO POTENZA :X
   MAKE "P :X
   WHILE [:P < 1000][PRINT :P MAKE "P:P * :X]
END
```

Le strutture del tipo WHILE, REPEAT e FOR sono comuni in altri linguaggi, ma non del tutto indispensabili nel LOGO. La procedura POTENZA può anche essere scritta così:

```
TO POTENZA :P
   IF NOT :P < 1000 THEN STOP
   PRINT :P
   POTENZA :P * :P
END
```

La struttura REPEAT è presente in LOGO, anche se raramente usata, in quanto si può definire una procedura REPT così concepita:

```
TO REPT :NUM :LISTA
   IF :NUM = 0 THEN STOP
   RUN :LISTA
   REPT :NUM -1 :LISTA
END
```

La primitiva RUN è molto utile per la programmazione in LOGO: un programma può costruire una lista che viene poi eseguita tramite RUN. Tra breve vedremo un esempio concreto di applicazione.

Separare le procedure

Prima di tutto definiamo, nel modo consueto, una procedura per tracciare triangoli:

```
TO TRI
   FD 50 RT 120 FD 50
   RT 120 FD 50 RT 120
END
```

Adesso si digiti PRINT TEXT "TRI. Il risultato dovrebbe essere:

```
[ ][FD 50 RT 120 FD 50][RT 120 FD 50 RT 120]
```

Il testo della procedura viene dato come lista di liste, dove ogni lista corrisponde ad una riga della procedura.
Per vedere come mai la prima lista sia vuota, si digiti:

```
TO ADD :A :B
   PRINT :A + :B
END
```

Adesso, usando PRINT TEXT "ADD, si ha:

```
[:A :B][PRINT :A + :B]
```

La prima lista contiene, dunque, l'elenco dei parametri in ingresso. TEXT consente di estrarre e visualizzare il contenuto di una procedura.
DEFINE permette invece l'operazione inversa: la definizione di una procedura tramite una lista di liste, senza ricorrere all'editor.
Si digiti:

```
DEFINE "L [[:A][FD :A][RT 90][FD :A/2]]
```

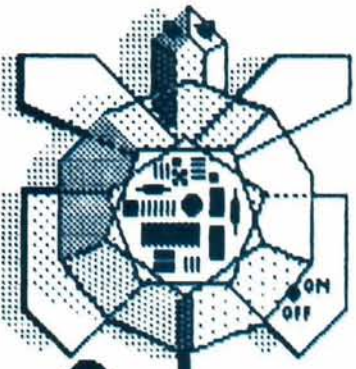
e poi si usi la procedura L (ad esempio: L 30). Usare DEFINE in modo diretto non porta a grandi vantaggi, mentre invece usata all'interno di una procedura consente di definirne di nuove.

Un programma di crescita

Realizziamo adesso un sistema per esaminare la "crescita" di alcune figure. I comandi previsti sono: CHIEDI, per selezionare la figura voluta, e CRESCI, per modificare le dimensioni di detta figura. CHIEDI "QUADRATO, ad esempio, disegna un quadrato sullo schermo.
Usando CRESCI [*10] cancella il quadrato precedente e lo ridisegna usando un lato 10 volte più lungo.
Per non complicare troppo il programma, i comandi sono soggetti ad alcune limitazioni. Per prima cosa le procedure richiamate da CHIEDI non devono contenere strutture REPEAT né chiamate a sottoprocedure. In secondo luogo il programma s'inceppa se viene prodotto un risultato negativo. Tuttavia, con qualche accorgimento, questi ostacoli si possono aggirare.
CHIEDI funziona assegnando il nome della figu-

Disegnare con la tartaruga

Non si può progredire, in LOGO, senza imbattersi nella "ricursione" (ossia la definizione di qualcosa in termini di sé stessa). Si sono visti numerosi esempi di procedure ricursive, che richiamano sé stesse, di liste definite in base alle liste stesse e, in ultimo, di procedure che generano altre procedure. Con un po' di immaginazione si potrebbero addirittura creare disegni nello stile di M.C. Escher, usando una tartaruga che disegna una tartaruga, che disegna...



ra alla variabile globale "CORRENTE, per poi avviare la procedura.
Ciò si ottiene creando una lista di un solo elemento (il nome della procedura) ed usando RUN per eseguirla.

```
TO CHIEDI :OGGETTO
   HIDETURTLE
   MAKE "CORRENTE :OGGETTO
   RUN (LIST :OGGETTO)
END
```

CRESCI, dopo aver cancellato il disegno originale (il LOGO del Commodore impiega a tal scopo PENCOLOR—1), usa DEFINE per definire la procedura corrente come nuova procedura. Il colore della penna è ripristinato e viene tracciata la nuova sagoma.
Si noti che il parametro in ingresso è conservato nella variabile OPLIST, usata anche in seguito.

```
TO CRESCI :OPLIST
   PENCOLOR—1
   RUN (LIST :CORRENTE)
   DEFINE :CORRENTE RISCRIVI.PROC TEXT :CORRENTE
   PENCOLOR 1
   RUN (LIST :CORRENTE)
END
```

La procedura RISCRIVI.PROC suddivide il testo in linee e le passa, una alla volta, a RISCRIVI.LINEA:

```
TO RISCRIVI.PROC :TESTO
   IF EMPTY? :TESTO THEN OUTPUT[ ]
   OUTPUT FPUT RISCRIVI.LINEA FIRST :TESTO RISCRIVI.
      PROC BUTFIRST :TESTO
END
```

RISCRIVI.LINEA effettua una scansione della linea alla ricerca di un FD o FORWARD: se lo trova, passa il resto della linea alla procedura CAMBIA:

```
TO RISCRIVI.LINEA :LINEA
   IF EMPTY? :LINEA THEN OUTPUT[ ]
   IF ANYOF FIRST :LINEA = "FD FIRST :LINEA =
      "FORWARD THEN OUTPUT CAMBIA BUTFIRST:LINEA
   OUTPUT FPUT FIRST :LINEA RISCRIVI.LINEA BUTFIRST
      :LINEA
END
```

CAMBIA costruisce la linea "riscritta". Il primo elemento di LIST (parametro in ingresso di CAMBIA) consiste nel valore relativo a FORWARD nella procedura originale. Supponiamo che esso sia 50 e che OPLIST contenga [*2], allora SENTENCE FIRST :LIST :OPLIST dà per risultato [50 * 2]. CAMBIA usa RUN per calcolare tale risultato (ossia 100).
Infine, viene costruita una lista contenente FD, il valore calcolato e il resto della linea originale.

```
TO CAMBIA :LIST
   OUTPUT (SENTENCE "FD (RUN SENTENCE FIRST :LIST
      :OPLIST) RISCRIVI.LINEA BUTFIRST :LIST)
END
```

IAN McKINNELL

Una procedura... pappagallo!

Talvolta può essere utile poter duplicare una procedura: realizziamo allora una procedura COPYDEF, in modo che COPYDEF "NUOVONOME "VECCHIONOME definisca NUOVONOME come copia di VECCHIONOME (lasciando inalterata VECCHIONOME). La soluzione più ovvia sarebbe:

```
TO COPYDEF :NUOVO :VECCHIO
   DEFINE :NUOVO TEXT :VECCHIO
END
```

Il problema è che, se VECCHIO non esiste, allora la procedura crea un NUOVO completamente vuoto. Sarebbe più utile se la procedura generasse un messaggio per tale eventualità:

```
TO COPYDEF :NUOVO :VECCHIO
   IF NOT PROCEDURA? :VECCHIO THEN (PRINT [NON ESISTE
      LA PROCEDURA] :VECCHIO) STOP
   DEFINE :NUOVO TEXT :VECCHIO
END
```

Qui c'è una chiamata a PROCEDURA?, che produce TRUE (vero) se la procedura citata esiste o FALSE (falso) in caso contrario. PROCEDURA? e la sua controparte PRIMITIVA? sono procedure di verifica molto utili, ma non esistono nel LOGO MIT. Eccone però una versione adatta al LOGO dell'Apple e del Commodore:

```
TO PROCEDURA? :NOME
   IF NUMBER? :NOME THEN OUTPUT "FALSE
   IF LIST? :NOME THEN OUTPUT "FALSE
   TEST WORD? :NOME
   IFTRUE IF WORD? TEXT :NOME THEN OUTPUT "FALSE
      ELSE IF NOT (TEXT :NOME = [ ]) THEN OUTPUT "TRUE
      OUTPUT "FALSE
END

TO PRIMITIVA? :NOME
   IF NUMBER? :NOME THEN OUTPUT "FALSE
   IF LIST? :NOME THEN OUTPUT "FALSE
   TEST WORD? :NOME
   IFTRUE IF WORD? TEXT :NOME THEN OUTPUT "TRUE ELSE
   OUTPUT "FALSE
END
```

Conclusioni

Come si è visto dagli esempi illustrati di volta in volta in questa sezione, il LOGO è un linguaggio estremamente versatile, specialmente per la grafica.
La tipica struttura a blocchi (procedure), più famigliare a chi abbia una certa conoscenza di linguaggi quali il PASCAL, meno consueta per chi usa il BASIC, non rappresenta certo un ostacolo all'apprendimento del LOGO. Inoltre, esistono numerosissime pubblicazioni sul LOGO, utili a chi desideri approfondire le applicazioni del LOGO in particolari settori (specialmente quello della didattica).
Nella prossima sezione dedicata al LOGO viene realizzata una nuova applicazione: un programma per la generazione di grafici a barre.

IL LOGO: LE VIRTÙ...

- È un linguaggio "interpretato" (come il BASIC) quindi i programmi sono facili da usare e da modificare
- È un linguaggio strutturato, con vere procedure, diversamente dalla maggior parte dei BASIC
- È un linguaggio espandibile, come il FORTH, in quanto si possono creare ed aggiungere nuovi comandi che diventano parte del "vocabolario" del linguaggio
- Possiede la capacità di elaborare liste, come il LISP. Ciò consente di esplorare campi quali l'intelligenza artificiale

La programmazione in LOGO avviene secondo un metodo molto ordinato e logico. Per realizzare un programma, infatti, si parte dalla definizione di un modulo principale, per poi aggiungere tutte le sottoprocedure occorrenti. L'estrema flessibilità del linguaggio, anche se il suo apprendimento non è così immediato quanto quello del BASIC, consente di giungere con rapidità alla definizione di algoritmi perfetti. Aggiungere o modificare le procedure, inoltre, è cosa piuttosto facile.

... E I VIZI

- I problemi maggiori riguardano la limitata memoria a disposizione dei programmi e la lentezza nell'esecuzione
- Mancano funzioni che sarebbero molto utili, in particolare: sofisticate routine per la gestione degli errori, delle variabili a matrice e dei file per archivi. Alcune versioni di LOGO possiedono tali caratteristiche, ma esse non fanno parte dello "standard"

Logo & Logo

In tutte le versioni LCSI si usi:
NUMBERP al posto di NUMBER?
LISTP al posto di LIST?
WORDP al posto di WORD?
EMPTYP al posto di EMPTY?

Per la sintassi da usare nelle IF vedere a pag. 1295.

Il LOGO dello Spectrum prevede come primitive sia COPYDEF sia PRIMITIVEP (equivalente alla nostra PRIMITIVA?) e DEFINEDP (equivalente alla nostra PROCEDURA?)

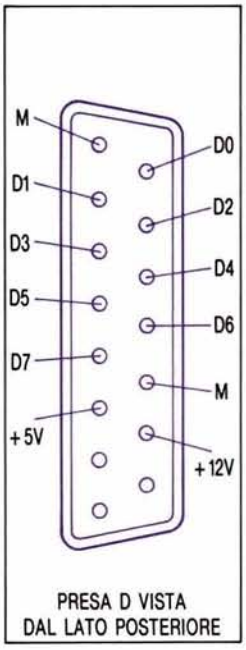
Con il LOGO per Atari si usi PC per PENCOLOR, SE per SENTENCE e HT per HIDETURTLE. DEFINE e TEXT non esistono come primitive, ma il manuale offre un modo per realizzarle.

I colori della penna differiscono da un apparecchio all'altro. Qui, per cancellare, è stato usato PC—1, ma in alcune versioni esiste un'apposita primitiva PE (per Pen Erase)



Riuniamo le parti

Colleghiamo i motori, realizziamo un'interfaccia e collaudiamo l'apparecchio



PRESA D VISTA DAL LATO POSTERIORE

IL CONNETTORE DELL'INTERFACCIA

Una volta montati i motori, non rimane che realizzare il collegamento per la presa D, situata sul coperchio del robot. Il diagramma mostra la disposizione dei contatti sulla presa D, vista dalla parte posteriore. Usando un cavo corto a nastro con 12 vie, si colleghino al circuito le linee dei dati (D0 — D3), quelle dell'alimentazione a + 5 e + 12 Volt e quella di massa (si confronti, a tal scopo, lo schema a pag. 1317). Si presti particolare attenzione al collegamento della linea a 12 Volt, poiché un errore potrebbe danneggiare seriamente il funzionamento del computer. Si noti che, per il momento, le linee dati D4 — D7 non vanno ancora connesse, poiché sono destinate al collegamento dei vari sensori, come descritto più avanti.
Tutti i collegamenti interni al robot sono così completati. Si trovi un'adeguata collocazione del circuito, all'interno dell'involucro, e lo si fissi con qualche cuscinetto autoadesivo. Non resta che fissare il coperchio del robot con le apposite viti di bloccaggio

PROGRAMMA DI PROVA

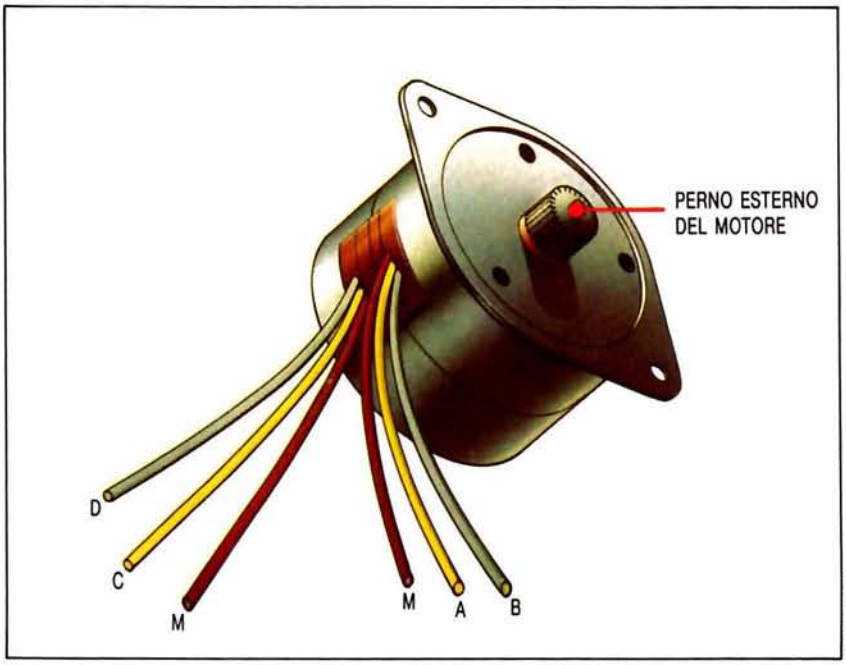
Avendo terminato la fase preliminare del montaggio, possiamo usare un breve programma per collaudare il funzionamento del robot tramite comandi impartiti dalla tastiera. I bit da 0 a 3 del registro dati della porta utente controllano i motori. Il bit 0, normalmente posto a 1, è quello di reset, mentre i bit 1 e 2 controllano le direzioni a destra e a sinistra dei due motori. Il bit 3 serve per inviare ai motori un impulso che li fa muovere di un passo. Il programma usa i tasti T, B, F e H per controllare le direzioni e per inviare un treno d'impulsi ai motori

```
1000 REM *** CONTROLLO ROBOT - BBC ***
1010 DDR=&FE62:REGDAT=&FE60:?DDR=15:REM LINEE 0-3
     IN USCITA
1020 PROCinizial:REPEAT
1030 A$=INKEY$(1):IF A$<>"" THEN PROCtastiera
1040 PROCimpulso(10)
1050 UNTIL A$="X":?REGDAT=0:END
1060 DEF PROCinizial
1070 avanti=4:indietro=2:sinistra=&:destra=0
1080 dir=avanti:?REGDAT=dir+1:ENDPROC
1100 DEF PROCimpulsi(m)
1110 FOR c=1 TO m
1120 ?REGDAT=(?REGDAT OR 8):PROCritardo(2)
1130 ?REGDAT=(?REGDAT AND 247):PROCritardo(2)
1140 NEXT c:ENDPROC
1150 DEF PROCritardo(n)
1160 FOR I=1 TO n:NEXT I
1170 ENDPROC
1180 DEF PROCtastiera
1190 IF A$="T" THEN dir=avanti
1200 IF A$="B" THEN dir=indietro
1210 IF A$="F" THEN dir=sinistra
1220 IF A$="H" THEN dir=destra
1230 ?REGDAT=((?REGDAT AND 249)OR dir)
1240 ENDPROC
```

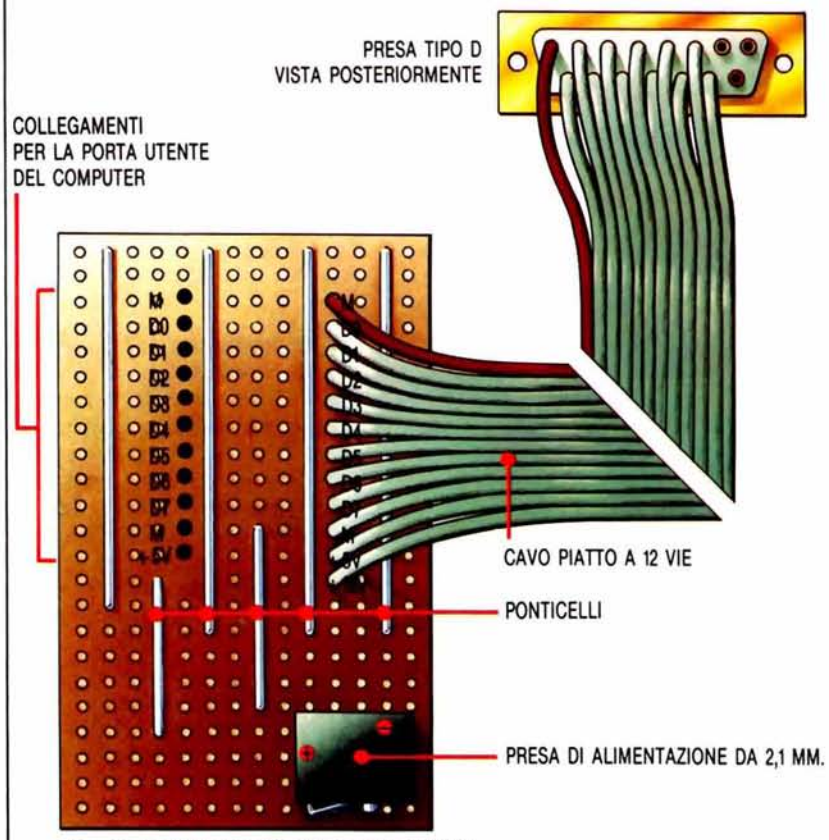
```
10 REM *** CONTROLLO ROBOT - C64 ***
20 DDR=56579:REGDAT=56577:POKEDDR,15
30 GOSUB 1000:REM INIZIALIZZA
40 GETA$:IF A$<>"" THEN GOSUB 3000:REM TASTI
50 M=10:GOSUB 1500:REM IMPULSO
60 IF A$<>"X" THEN 40
70 POKEREGDAT,0:END
1000 REM *** S/R INIZIALIZZAZIONE ***
1010 AV=4:ND=2:SN=&:DS=0
1020 DR=AV:POKEREGDAT,DR+1:RETURN
1500 REM *** S/R IMPULSO ***
1510 FOR C=1 TO M
1520 POKEREGDAT,(PEEK(REGDAT)OR8):GOSUB2000:REM RITARDO
1530 POKEREGDAT,(PEEK(REGDAT)AND247):GOSUB2000:REM
     RITARDO
1540 NEXT C:RETURN
2000 REM *** S/R RITARDO ***
2010 FOR I=1 TO N:NEXT I:RETURN
3000 REM *** S/R TASTI ***
3010 IF A$="T" THEN DR=AV
3020 IF A$="B" THEN DR=ND
3030 IF A$="F" THEN DR=SN
3040 IF A$="H" THEN DR=DS
3050 POKEREGDAT,(PEEK(REGDAT)AND249)OR DR)
3060 RETURN
```

COLLEGAMENTO DEI MOTORI

Per questa fase occorre far riferimento allo schema riportato a pag. 1317, che mostra dove vanno saldati i sei cavetti che fuoriescono dal motore passo passo e la disposizione dei contatti per il cavo di collegamento con la presa tipo D.
I sei cavetti colorati di ciascun motore sono suddivisi in due gruppi. Di solito i cavetti sono colorati, per facilitarne l'individuazione. Facendo riferimento alla figura, i cavetti sono contrassegnati rispettivamente con le lettere A, B, M per il gruppo più vicino al perno esterno del motore e C, D, M per l'altro gruppo. Questa nomenclatura corrisponde a quella riportata sullo schema a pag. 1317.
Se il motore usato dovesse avere caratteristiche fisiche diverse, si faccia attenzione a non confondere la disposizione dei collegamenti, altrimenti il robot non funzionerà



KEVIN JONES



SCHEDA DELL'INTERFACCIA PER PORTA UTENTE

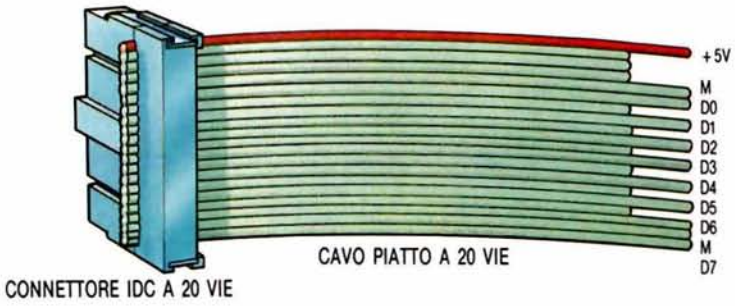
ELENCO COMPONENTI

Quantità	Descrizione
1	Presa D a 15 vie
1	Copripresa per presa D
1	Presa per alimentazione (2.1 mm.)
1	Connettore IDC a 20 vie (per il BBC)
1	Connettore da scheda a 24 vie (per il Commodore 64)
1	Confezione di cuscinetti autoadesivi
4 m	Cavo a nastro a 12 vie
1 m	Cavo a nastro a 20 vie (per il BBC)
1	Alimentatore corrente continua 12 Volt, 1 Ampere

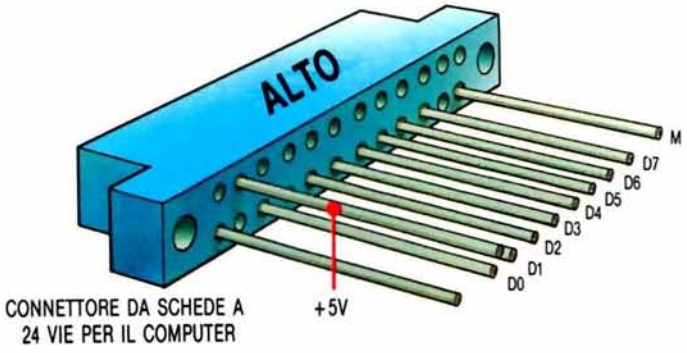
L'INTERFACCIA CON LA PORTA UTENTE

Completati i collegamenti interni del robot, occorre adesso realizzare una semplice interfaccia che permetta di controllare il robot tramite la porta utente del computer e che fornisca la tensione di 12 Volt necessaria al funzionamento dei motori. Per il circuito si usi una scheda prestampata e preforata del tipo Veroboard, con 24 piste per 14 fori. Ad essa vanno saldati i capi del cavo a nastro a 12 vie, come illustrato in figura. Poi, usando come riferimento il cavetto colorato di rosso del cavo a nastro, si saldino i capi dell'altra estremità alla presa tipo D, come illustrato in figura. Non si dimentichi di inserire il copripresa, che serve ad evitare cortocircuiti. Si monti poi la presa di alimentazione da 2,1 mm., facendo attenzione alla disposizione dei piedini. Il polo centrale della presa corrisponde al negativo dell'alimentazione. Il collegamento dell'interfaccia al computer dipende dal computer usato, quindi occorrono due diversi tipi di cavi e di connettori, come spiegato più avanti.

MICRO BBC



COMMODORE 64



I COLLEGAMENTI DEL CONNETTORE

Qui di fianco è riportato il diagramma dei collegamenti tra il circuito di interfaccia e la porta utente del computer. Si osservi attentamente la disposizione dei contatti nel diagramma, che va seguita con estrema cura. I possessori di BBC usino un cavo a nastro da 20 vie ed un connettore IDC (anch'esso a 20 vie).

I possessori di Commodore 64 usino invece un cavo a nastro a 12 vie ed un connettore da scheda a 24 vie. Si presti molta attenzione alla disposizione dei contatti sulla presa tipo D. Inoltre, poiché il connettore per questo computer può essere inserito anche capovolto (non esiste alcuna tacca di riferimento) si provveda a marcare in modo indelebile quale parte del connettore è rivolta verso l'alto.

Una volta realizzata la scheda, si può collegare tutto l'insieme dei circuiti. Alla presa di alimentazione dell'interfaccia va collegato un comune alimentatore da 12 Volt 1 Ampere. Nell'acquistare quest'ultimo componente, si rammenti che il polo più interno della presa corrisponde al negativo dell'alimentazione

Concetti operativi

Inizia una serie dedicata ai sistemi operativi dei più diffusi home computer

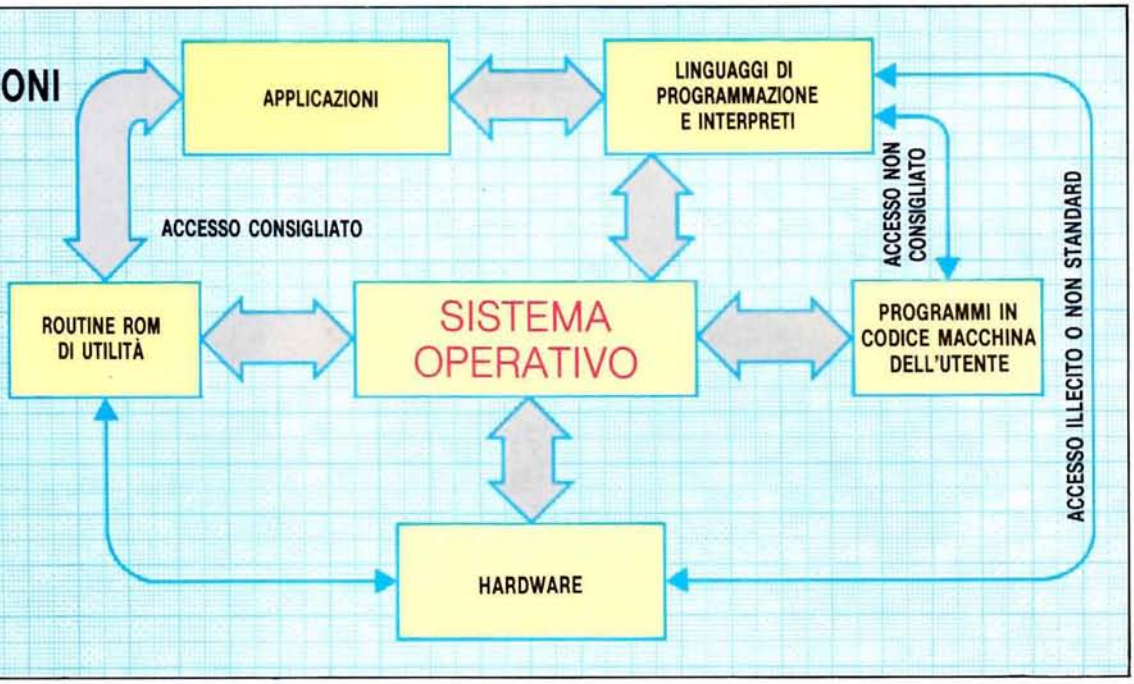
Un sistema operativo è normalmente realizzato in codice macchina: in particolare nel codice macchina adatto al microprocessore impiegato nel computer. Pertanto, il SO del BBC è scritto in codice 6502, mentre quello per lo Spectrum in codice Z80.
Un SO consiste in un gruppo di routine che si occupano di svariate funzioni dell'apparecchio. Per esempio, una routine è dedicata alla scansione della tastiera, per rilevare eventuali pressioni di tasti. Generalmente le routine del SO evitano al programmatore il compito di realizzare di volta in volta routine più banali.
In effetti, la qualità di un SO si giudica proprio in base alla quantità di routine fornite e alla possibilità di accedere facilmente alle operazioni di ingresso/uscita e a tutte le altre funzioni del computer. Una caratteristica da non trascurare, in un SO, è la possibilità di apportare modifiche all'hardware senza dover alterare i programmi scritti dall'utente. Ciò consente di migliorare le versioni del SO, ferma restando la compatibilità dei programmi applicativi.
Il SO del BBC, ad esempio, è molto ben concepito: un programma scritto in BASIC su di un apparecchio in versione normale funziona perfettamente anche quando è innestata la scheda col microprocessore aggiuntivo, benché ciò rappresenti una consistente mutazione dell'hardware. Il SO del BBC è passato attraverso varie fasi di sviluppo prima di giungere a quella definitiva. La versione 0.1, la prima realizzata, era conservata su 4 chip di EPROM, ma mancava di versatilità e, soprattutto, non prevedeva l'uso di dischetti. La versione 1.0 aggiungeva questa importante caratteristica ed era conservata su due chip da 8 Kbyte, montati su una scheda da inserire nel computer.
L'ultima versione realizzata, la 1.2, è quella adottata dalla maggior parte dei BBC in circolazione. Ne esiste una variante (la versione 1.2US) appositamente realizzata per il mercato USA.
Per sapere quale versione è montata sul proprio apparecchio basta digitare il comando *FX0 e premere RETURN: la sigla della versione compare sullo schermo. Anche il comando *HELP fornisce la medesima informazione.

Il funzionamento del SO del BBC

I compiti svolti dal sistema operativo del BBC possono essere suddivisi in quattro gruppi principali:

IL TEATRO DELLE OPERAZIONI

Il miglior sistema operativo è quello la cui presenza... non si nota. All'utilizzatore esso dovrebbe infatti apparire come un'interfaccia totalmente trasparente tra il programma applicativo e l'hardware, occupandosi della gestione dei file e di tutti i dispositivi annessi. In sostanza, il SO operativo deve garantire un corretto espletamento di tutte le funzioni nel modo più semplice. Per non legare un programma applicativo ad una particolare configurazione hardware andrebbero evitati accessi diretti alle routine ROM e ai vari dispositivi, se queste non sono previste tramite il SO



LIZ DIXON

1. *Routine di input:* queste accettano informazioni tramite quello che viene indicato come "attuale flusso di input". Normalmente questo coincide con la tastiera, ma altri flussi di input sono la porta seriale RS423 e il sistema di archiviazione a nastri (o a dischi), al quale si accede tramite il comando *EXEC. La principale routine di questo gruppo è chiamata OSRDCH (che in inglese sta per: Operating System ReaD CHaracter, ossia "lettura di carattere per mezzo del sistema operativo").

2. *Routine di output e di visualizzazione:* queste routine si occupano dei dati in uscita dal computer. Nel BBC esistono vari flussi di uscita selezionabili: lo schermo, la stampante, la porta seriale RS423 e, usando il comando *SPOOL, anche il sistema di archiviazione dei file. Oltre ad occuparsi dell'emissione di caratteri, le routine di questo gruppo sono dedicate anche alla gestione del chip video 6845 e all'uso dei caratteri predefiniti.
   Le chiamate al SO sono: OSWRCH (per la scrittura di caratteri), OSASCII e OSNEWL.

3. *Sistema di archiviazione dei file:* un sistema operativo deve consentire di salvare il contenuto della memoria su di un supporto più duraturo. Questa sezione del SO è dedicata alle operazioni relative al "sistema di archiviazione dei file" correntemente selezionato. Sul BBC la possibilità di scelta comprende: dischi, nastri, Econet, ROM e telesoftware. Le routine di questo gruppo si avvalgono di altre routine del SO per espletare i propri compiti riguardo a particolari sistemi di archiviazione.
   Per consentire un semplice interfacciamento dei supporti magnetici, sono previste numerose routine per la gestione dei file. Queste includono varie routine per la scrittura e la lettura di interi file, oppure di singoli byte all'interno di file. Le chiamate al sistema operativo interessate sono sette, ognuna delle quali adopera un vettore per puntare all'appropriata routine per la gestione dei file. Se viene montata una ROM diversa (ad esempio quella del DFS, per abilitare l'uso dei dischetti), basta che i vettori vengano modificati in modo da puntare alle routine della nuova ROM.

4. *Le interrupt:* una interrupt non è che un segnale generato da software o dall'hardware, che "richiede l'attenzione" della CPU: quest'ultima interrompe quanto sta eseguendo per "servire" (tramite un'apposita ruotine) l'interruzione. Al termine la CPU riprende in modo ordinato il lavoro interrotto. Sul BBC esiste una discreta varietà di funzioni pilotate da interrupt, facilmente raggiungibili tramite chiamate al SO.

Oltre a queste quattro aree principali, esistono due chiamate al SO di capitale importanza. Esse sono: OSBYTE e OSWORD, che vengono usate per controllare una gran quantità di funzioni, tra le quali quelle del chip del suono e del tasto break.

Perché usare chiamate al SO?

Nella maggior parte dei casi, le informazioni relative alla configurazione interna della memoria e dei vari dispositivi vengono rese pubbliche (solo raramente vengono taciute per riservatezza industriale) poco dopo il lancio commerciale del computer. Usare le chiamate al SO può quindi sembrare superfluo: non basta operare direttamente sulla memoria o sui dispositivi?
La risposta è già parzialmente contenuta in quanto detto all'inizio: ricorrere alle chiamate del sistema operativo garantisce la compatibilità dei programmi con successive modifiche all'hardware dell'apparecchio. Analogamente, usare chiamate dirette alle ROM può compromettere l'uso dei programmi qualora si renda disponibile una nuova versione delle ROM stesse.
Facciamo un esempio concreto. La seguente istruzione, su di una normale versione del BBC, invia un valore 200 alla porta utente, il cui indirizzo è &FE60.

```
?&FE60 = 200
```

Ma se viene aggiunta la scheda con il secondo processore, tale valore non è affatto inviato alla porta utente, bensì viene scritto nella memoria del secondo processore. Usando una chiamata appropriata al sistema operativo, si può aggirare questo ostacolo, poiché il SO conosce con esattezza il nuovo indirizzo della porta utente.
La chiamata è questa:

```
*FX 151,96,200
```

Questa chiamata è estremamente utile per accedere ai vari dispositivi del BBC, quali l'adattatore d'interfaccia VIA e il bus a 1 MHz.
In definitiva, quindi, non conviene reinventare la ruota: infatti se nel SO esiste già una routine che svolge una certa funzione, è molto più semplice usare questa che non realizzarne un duplicato, col rischio che funzioni peggio.
Gli unici casi in cui può convenire un accesso diretto alle routine ROM o ai vari dispositivi si verificano quando non esiste una specifica chiamata al SO, oppure la temporizzazione del programma è molto critica e va controllata direttamente. Ad ogni modo, includere simili accessi diretti in un programma significa legarlo ad una particolare configurazione dell'apparecchio, col rischio che non funzioni in circostanze diverse.
Il consiglio di evitare chiamate dirette riguarda particolarmente il BBC, che ha subito notevoli cambiamenti nel corso del tempo, mentre ha meno importanza con apparecchi quali lo Spectrum.

Summer games

Le Olimpiadi del 1984 hanno stimolato la produzione di molti giochi di tipo sportivo

Il programma *Summer Games* è stato prodotto dalla Epyx, una software house statunitense. Il package comprende otto competizioni sportive (dall'atletica al nuoto e al tiro a segno) a ciascuna delle quali possono partecipare fino a otto giocatori. Ogni giocatore può scegliere quale nazione rappresentare, tra le 17 disponibili. Se il giocatore sceglie una nazione la cui bandiera non è prevista dal programma, gli viene assegnato lo stendardo della Epyx.

Appena lanciato il programma, sullo schermo appare la cerimonia inaugurale dei Giochi, con tanto di staffetta che corre accompagnata da una colonna sonora appropriata. I suoni e la grafica, sia della cerimonia inaugurale sia del resto del gioco, sono di buon livello. Lo sfondo è realizzato con molta cura e ricchezza di dettagli, sfruttando la grafica ad alta risoluzione. La scorrevolezza dei movimenti offre un ottimo esempio delle possibilità di animazione con sprite del Commodore 64. L'accompagnamento musicale invece, sebbene di buon livello, non sfrutta appieno le caratteristiche del chip SID del Commodore. Comunque, l'impressione generale è che il computer sia stato usato molto sagacemente e quasi al massimo delle sue capacità.

I giocatori possono scegliere di competere in tutte le gare oppure anche in una sola, di allenarsi e di esaminare la tabella dei record per una specialità particolare. Esiste anche la possibilità di collegare uno o due joystick, per facilitare la competizione tra due antagonisti in gare quali il nuoto, ad esempio.

La prima gara, forse la più difficile fra tutte, è il salto con l'asta. Ogni giocatore partecipa singolarmente, rispondendo ad una serie di domande poste dal computer. Dopo aver fissato l'asta a quattro metri, il programma chiede se l'altezza è quella richiesta, poi si procede all'immissione di altri dati necessari al salto.

Controllare i movimenti e le azioni durante questa gara non è molto facile, anche se si adopera un joystick: la sequenza delle operazioni avviene ad un ritmo molto serrato, quindi occorre una discreta pratica per non mandare a vuoto i propri tentativi.

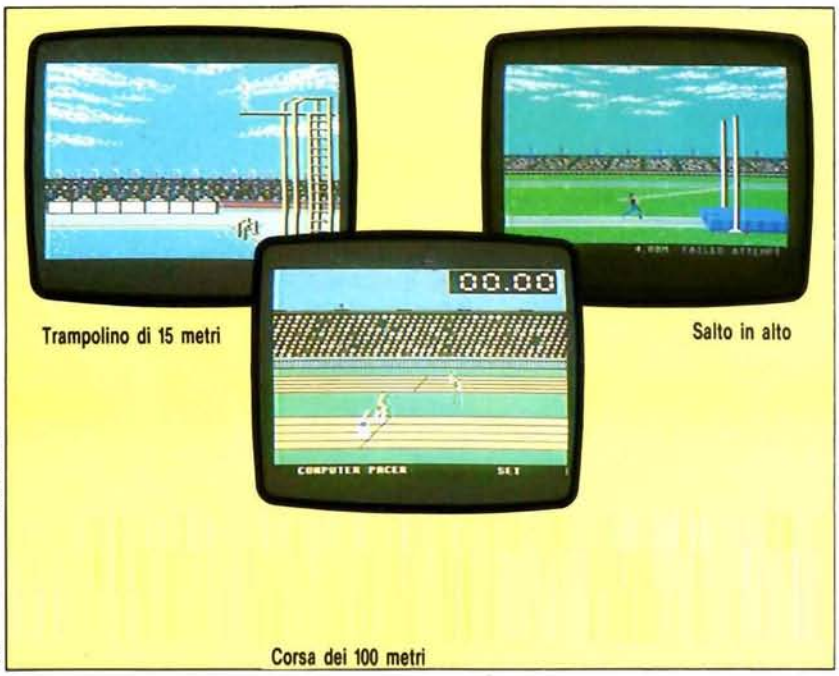
La gara del salto in alto illustra molto bene la complessità delle operazioni di cui deve tener conto il programma: non solo deve esserci un costante controllo dello schermo (per rendere i movimenti scorrevoli), ma il computer deve anche verificare i comandi impartiti e reagire immediatamente ad essi tramite il joystick e il pulsante di fuoco.

Infine, il programma deve anche verificare, tramite opportuni calcoli, se la posizione e lo slancio del concorrente sono tali da rendere il salto valido. Tutto ciò, ovviamente, deve avvenire senza che il giocatore noti la minima interferenza con lo sviluppo del gioco.

Al termine di ogni gara viene visualizzata la tabella dei punteggi e delle medaglie assegnate alle varie nazioni partecipanti: quella vincitrice viene onorata col proprio inno nazionale.

Il fatto che ogni gara venga caricata in memoria separatamente fa intuire la complessità del codice impiegato.

Summer Games è un gioco molto ben realizzato e sicuramente divertente da giocare, specialmente in compagnia.



Trampolino di 15 metri

Salto in alto

Corsa dei 100 metri

IAN McKINN

Summer Games: Per il Commodore 64
Editori: Epyx Software (USA)
Autori: Randy Glover, Stephen Landrum, John Leupp, Brian McGhie, Stephen Mundry, Erin Murphy, Scott Nelson
Joystick: Necessari
Formato: Su cassetta o su dischetto

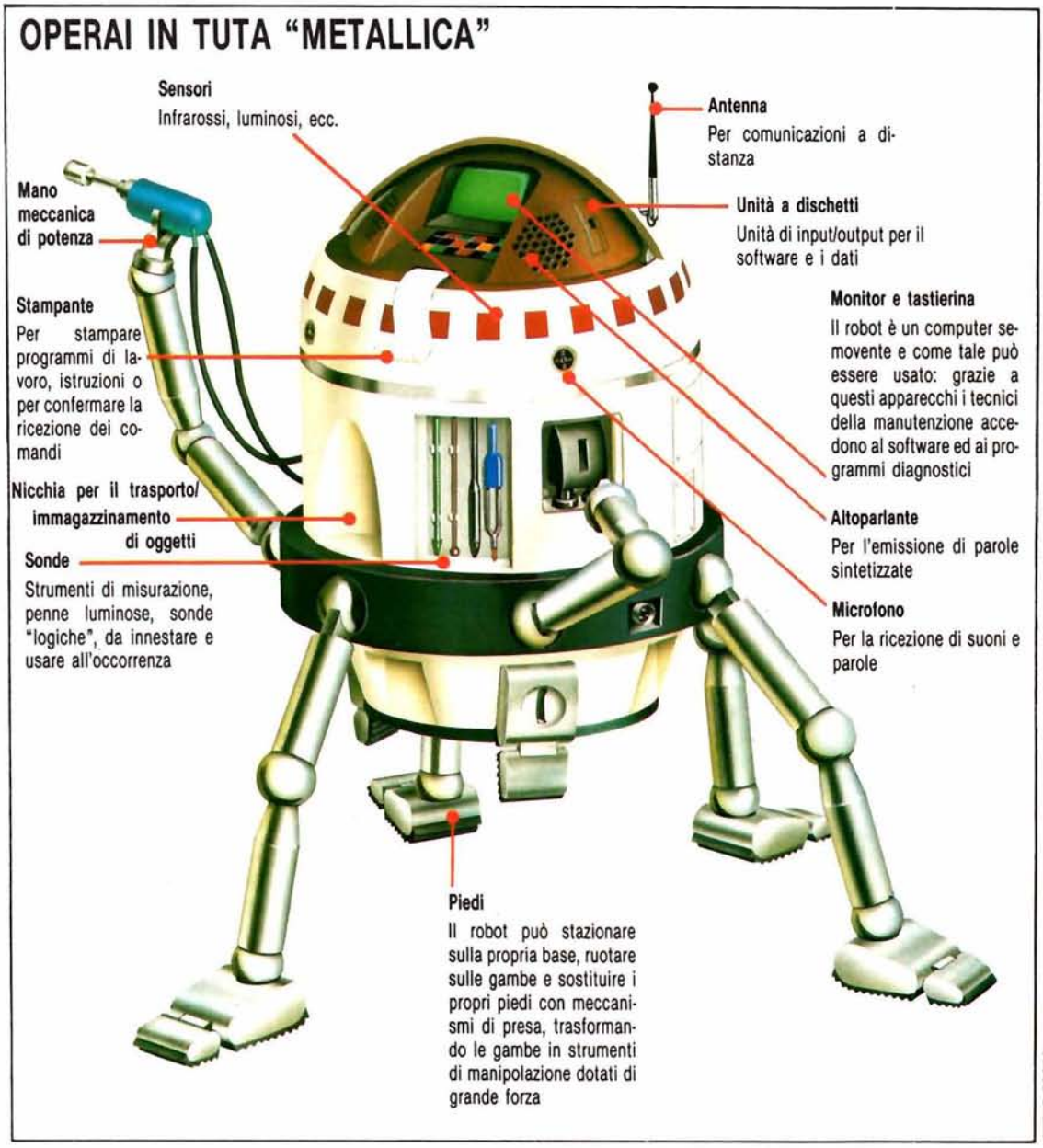
Gare olimpioniche

L'ottima qualità delle immagini di questo programma della Epyx si può notare da queste foto. Ogni gara è caricata in memoria separatamente (da disco o da nastro), per sfruttare al massimo tutta la memoria disponibile per le complesse sezioni di program-

ma è per l'ingombrante grafica ad alta risoluzione

I robot nel futuro

Le attuali limitazioni e i possibili sviluppi



Due passi nel futuro

Se mai sarà possibile costruire versatili robot dall'aspetto umano ad un costo che consenta di sostituirli alla attuale manodopera di media specializzazione, sarà necessario dotarli di un'"intelligenza" altamente sviluppata, coadiuvata da banche dati generiche e tecniche, sensori integrati e software di apprendimento. Simili robot potrebbero avere diverse forme: il modello in figura potrebbe operare come operaio semi-specializzato nell'industria leggera o in quella pesante

La visione che comunemente si ha del mondo dei robot è stata notevolmente influenzata dalla letteratura fantascientifica. Ci si aspetta infatti che un robot possa muoversi in modo libero ed autonomo: vedere, udire e toccare ciò che lo circonda; parlare di problemi scientifici e filosofici o almeno comunicare in modo "intelligente"; in definitiva, ci si aspetta che si comporti come un essere umano! In realtà, osservando il funzionamento degli odierni robot industriali, commerciali e "hobbistici", ci accorgiamo con sorpresa che essi si limitano ad eseguire i compiti loro assegnati e nulla di più.

Basandoci sulle conoscenze attuali dei principi di progettazione e sviluppo dei robot, proviamo ad immaginare quali saranno le principali caratteristiche dei robot del futuro.

Movimento

È estremamente improbabile che i robot in un prossimo futuro possano camminare su

Variazioni d'immagine

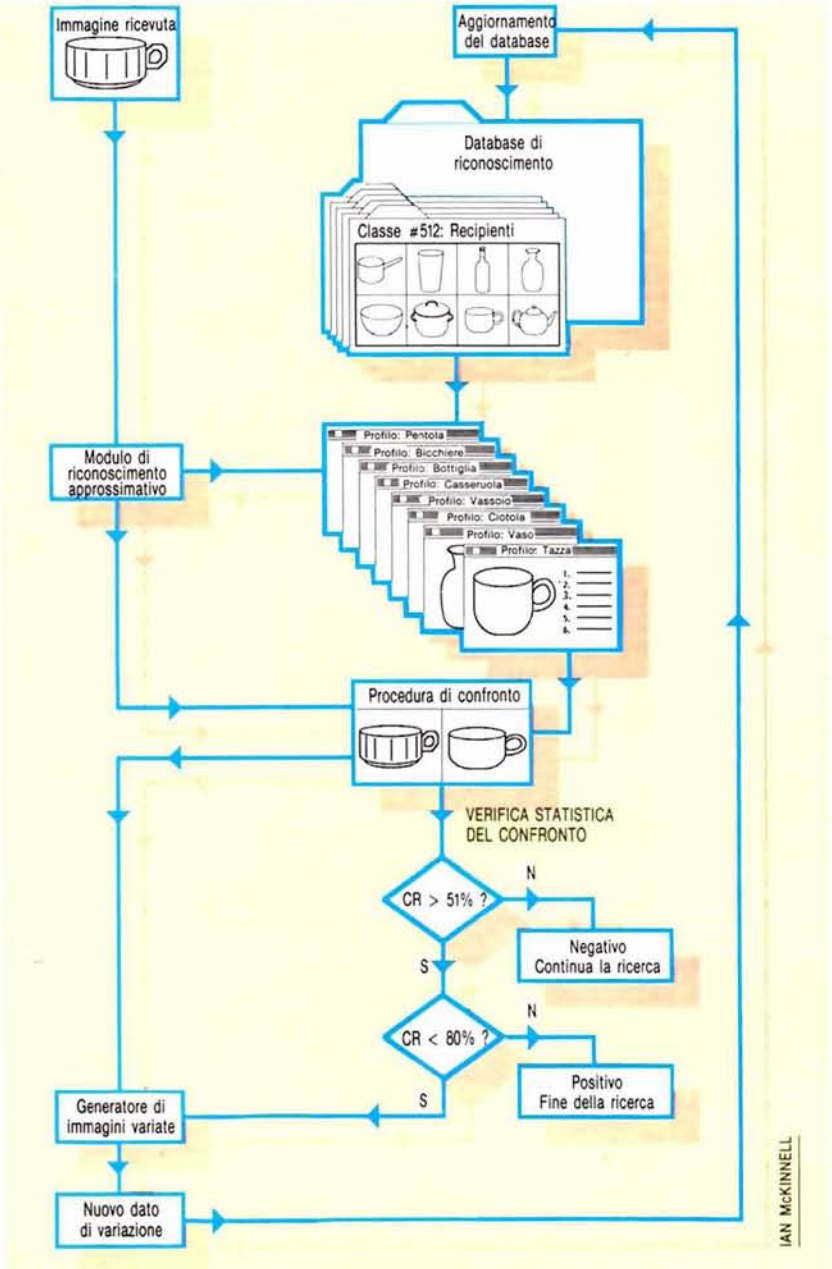
Il riconoscimento di un oggetto si basa sulla memorizzazione di un suo "profilo archetipo", ossia del modello generico abbinato ad una serie di dati di variazione. Questi vengono sovrapposti al profilo per ottenere variazioni le più vicine possibile alla realtà. L'immagine viene confrontata con moduli di riconoscimento che forniscono un'ipotesi sulla classe dell'oggetto osservato. Gli oggetti memorizzati, di quella classe, vengono confrontati nelle loro variazioni con l'immagine ricevuta alla ricerca del nostro oggetto. Con una verifica statistica si determina se la diversità dell'immagine dal profilo di base richiede l'introduzione di altri dati di variazione

gambe simili a quelle umane. Innanzi tutto il mantenimento dell'equilibrio richiede un impiego di memoria eccessivo ed un tempo di elaborazione troppo alto. Inoltre, le articolazioni e la "muscolatura" fornita dai motori elettrici o idraulici manca dell'elasticità e della duttilità tipiche delle strutture anatomiche dell'uomo (muscoli, tendini, cartilagini ecc.). Infine la presenza di due sole gambe limita notevolmente la versatilità di movimento dei robot. Per questo sono stati costruiti e sperimentati robot a quattro o sei gambe, molto simili ad insetti, dimostratisi particolarmente adatti a specifici campi d'applicazione.

D'altra parte, le ruote rappresenteranno anche in futuro il sistema di movimento più funzionale per applicazioni come trasporti militari, esplorazioni di pianeti ed usi più comuni come quelli domestici. Nuove tecnologie renderanno il movimento dei robot più scorrevole, ma probabilmente non gli consentiranno di raggiungere l'eleganza e la grazia di una ballerina o della corsa di un atleta.

Molti robot industriali e bracci robotizzati di medie dimensioni devono essere fissati al suolo: la loro libertà di movimento, dunque, è limitata ad una particolare area di lavoro. Se non intervengono radicali cambiamenti nelle tecniche di assemblaggio industriale, è probabile che anche i robot semoventi conservino una ridotta mobilità, magari limitandosi a seguire tracce sul pavimento, spostarsi su binari oppure oscillare da linee di montaggio sospese al soffitto. Ovviamente i progressi dell'automazione e della robotica introdurranno grosse innovazioni nei metodi produttivi industriali, ma è molto difficile prevederne concretamente gli effetti.

Un elemento chiave del movimento dei robot è l'interazione con l'ambiente circostante: il robot deve possedere un sistema di sensori i cui segnali devono condizionare direttamente i suoi spostamenti.



Sensori

Queste sofisticate apparecchiature consentono ai robot di estendere le loro percezioni a settori che i sensi umani non possono esplorare. Sensori di vicinanza, rivelatori di movimento, sensibili strumenti di posizionamento, rivelatori di rumore ad ampia gamma di frequenze permettono ai robot di raccogliere una quantità di informazioni superiore a quella possibile ad un uomo. I sistemi di "vista" sono divenuti più precisi grazie alla maggior risoluzione delle immagini ricevute. Le tecniche di riconoscimento e sintesi della voce, pur trovandosi ancora in fase di sviluppo, si fanno più sofisticate e giocano un ruolo importante nell'evoluzione della robotica.

I robot di servizio, utilizzati soprattutto in campo industriale, continueranno ad avere solo i sensori necessari all'esecuzione del loro lavoro. I bracci per saldatura, ad esempio, non richiedono sensori vocali o sofisticati sistemi di visione: un ristretto numero di informazioni è più che sufficiente per svolgere con rapidità e precisione i compiti loro assegnati.

I robot polivalenti, progettati per imparare dall'esperienza ed imitare i processi logici dell'uomo, hanno invece bisogno di un gran numero di sensori. Simili robot devono poter esplorare in modo autonomo l'ambiente che li circonda ed elaborare le informazioni accumulate. Nell'uomo la comprensione del linguaggio si basa essenzialmente su un'analisi combinata di segnali visivi e acustici. Tale sintesi viene spesso trascurata quando si esamina il progetto di un robot, ma è di fondamentale importanza perché la macchina possa comunicare con l'uomo e giungere non solo al riconoscimento, ma anche alla comprensione del linguaggio parlato.

Attualmente il numero di informazioni che un robot può immagazzinare dipende dalla



KEVIN JONES

Trasporto robotizzato

La robotica è destinata a trasformare profondamente molte delle attività umane, soprattutto quelle che utilizzano macchinari specializzati, ma mediamente sofisticati, quali: gru, macchine scavatrici, veicoli per la distribuzione e per i trasporti pesanti. La figura mostra una ruspa robot che carica un convoglio di carrelli robot all'interno di un cantiere. La colonna di veicoli viene mossa dai motori dei carrelli ed è pilotata da un operatore umano che si trova sul veicolo di testa. In un prossimo futuro la capacità di decisione e di comando dell'uomo, abbinata alla forza e all'"intelligenza" dei robot, consentirà uno sfruttamento più economico e vantaggioso delle risorse esistenti e delle nuove tecnologie

quantità di memoria necessaria per memorizzare i segnali dei sensori e dalla velocità e potenza d'elaborazione. Un robot può memorizzare l'immagine di un oggetto (ad esempio una mela) e associarla a un nome, ma la quantità di memoria impiegata aumenta al crescere della risoluzione del sistema visivo. Dato che non esistono due mele uguali, il robot deve possedere una memoria capace di contenere una campionatura di immagini della mela oppure un algoritmo che, ruotando l'immagine ricevuta, consenta un'analisi delle variazioni. In questo modo il robot è in grado di riconoscere la mela indipendentemente dall'angolo di osservazione. Ma già nel caso di un livello di risoluzione minimo, pari a 256 pixel per immagine, le variazioni possibili sono migliaia!

Le future esigenze di memoria verranno probabilmente soddisfatte da chip RAM di maggior capacità (chip da 1 Mbit e oltre) e dall'impiego di chip RAM dedicati all'archiviazione di dati "variazionali". Questi possono essere utilizzati dal processore per identificare immagini di diverso tipo quasi come una subroutine all'interno di un programma BASIC.

Oltre a possedere un'ampia memoria, i futuri robot dotati di complessi sistemi di sensori dovranno poter ricevere simultaneamente ed elaborare con rapidità informazioni provenienti da più sorgenti. I processori attuali non sono in grado di gestire tutti i dati in arrivo, che vengono quindi "parcheggiati" in attesa di essere elaborati. Una possibile soluzione potrebbe essere l'impiego di due o più processori ad alta velocità e capacità, collegati in parallelo. La gestione del sistema verrebbe affidata ad un processore di controllo che distribuirebbe alternativamente i compiti al processore momentaneamente libero.

Grazie al lavoro dei ricercatori, molti problemi riguardanti l'hardware sono stati risolti, mentre lo sviluppo della "mente" dei robot, ossia del sofisticato software necessario al robot per elaborare le proprie percezioni, procede con maggior lentezza.

La mente dei robot

Come accennato parlando del movimento e dei sensori, la robotica segue due grandi direttrici. La prima, probabilmente destinata ad essere maggiormente sfruttata a breve termine, è quella delle macchine "intelligenti" o robot di servizio. I bracci industriali ed i sistemi di produzione automatizzata, indipendentemente dalla loro complessità, richiedono un software formulato con grande precisione, ma al tempo stesso ben delimitato. Un braccio robotizzato si limita a ricevere una serie di coordinate e ad eseguire una sequenza di azioni programmate, senza realmente comprendere ciò che fa, e tantomeno conoscere i particolari dell'ambiente in cui opera.

Il risultato del suo lavoro può essere un pannello perfettamente verniciato, posto su una linea d'assemblaggio di automobili, oppure una vettura ben lustrata in una stazione di lavaggio automatizzata.

Al contrario, le istruzioni impartite ai robot sono necessariamente più generiche, in quanto si riferiscono a situazioni talvolta imprevedibili. Un robot che si muove in una stanza il cui contenuto cambia ogni giorno deve essere in grado di raccogliere ed elaborare informazioni, ma anche di usare le nuove conoscenze per "riconoscere" l'ambiente che lo circonda. Deve quindi possedere un software operativo e di controllo, ma anche una memoria tale da consentire il massimo sviluppo operativo.

La creazione di una mente meccanica suscita interessanti interrogativi (molti dei quali ancora senza risposta) sul funzionamento del cervello umano. Forse, in un prossimo futuro, la creazione di robot sempre più "umani" contribuirà ad accrescere le conoscenze che l'uomo ha di se stesso.

Sebbene queste previsioni non siano puramente teoriche ma si basino su concreti risultati sperimentali, resta il fatto che dovrà trascorrere ancora molto tempo prima che sia possibile costruire un robot veramente in grado di pensare.

BrainStorm

Caratteristiche e impieghi di un package molto insolito

Il *BrainStorm*, realizzato dalla Caxton Software (GB), viene spesso definito un "thought processor" (letteralmente: elaboratore di pensieri), così come "word processor" si riferisce all'elaborazione delle parole. Il suo scopo non è quello di sostituirsi al cervello umano ma, più semplicemente, quello di aiutare l'utente ad organizzare le proprie idee. Per comprenderne il funzionamento, conviene confrontarlo con altri sistemi non computerizzati.
Durante la pianificazione di un complicato progetto, può essere utile ricorrere a liste gerarchiche dei vari elementi che lo compongono. Partendo da una lista degli obiettivi principali, vengono sviluppate alcune sottoliste che completano ciascun elemento della lista precedente. Purtroppo questo tipo di pianificazione "manuale" su carta spesso risulta confuso e poco funzionale; qualsiasi modifica (il trasferimento di un elemento dalla lista principale a una sottolista, ad esempio) costringe a cancellazioni e inserimenti che rendono il sistema ingovernabile. Al contrario, l'impiego del *BrainStorm* rende le fasi di correzione e sviluppo semplici come le funzioni "taglia e cuci" dei word processor. Ogni elemento di una lista o sottolista può divenire l'intestazione (o titolo) di una sottolista di livello inferiore: ciò consente di collegare argomenti simili, posti in liste differenti, assegnando loro lo stesso nome (tali elementi vengono detti "omonimi"). In questo caso, l'introduzione di una nuova informazione in una sottolista che ha per titolo un omonimo vale per tutte le liste che hanno lo stesso nome.


Sviluppo di una sottolista

Creazione di una lista generale

Chi già conosce i comandi di editing di un word processor quale il *WordStar* della MicroPro, non ha difficoltà ad adoperare comandi "non mnemonici" per il controllo del cursore. I movimenti del cursore in alto, in basso, a destra e a sinistra si ottengono premendo il tasto Control (CTRL), rispettivamente assieme a uno dei quattro tasti E, X, S e D (la classica disposizione "a diamante"). Comunque, l'utente può anche ridefinire l'abbinamento dei tasti associati ai comandi per il cursore. Meno facile è abituarsi al fatto che, durante l'immissione di testi, è possibile muovere verticalmente il cursore ma, per gli spostamenti laterali, è necessario abilitare il modo "correzione", usando il comando CTRL-A.

Funzionamento del programma

Il menu d'apertura del *BrainStorm* prevede 11 opzioni, selezionabili battendo la loro lettera iniziale: Use, Load, Print, ID Drive (per abilitare un'unità a dischetti diversa da quella corrente), Clear (per cancellare dalla memoria il modello corrente), Save, Write (per le registrazioni su disco), Directory, Xit (per uscire), Merge e Kill.
All'inizio della seduta di lavoro, l'utente batte U (abilitando automaticamente il modo "tastiera"), quindi inserisce più o meno casualmente gli argomenti da sviluppare in una lista (detta "modello"). Ad esempio:

Leggi il manuale
Comincia a scrivere la lista
Scrivi la sottolista
Riguarda la lista

Per visualizzare i comandi di controllo si preme il tasto "?". Posizionando il cursore su un elemento della lista e battendo CTRL-R, si assegna il suo nome ad una sottolista. Analogamente, qualsiasi elemento di una sottolista può essere usato come titolo per una nuova sottolista, e così via fino all'esaurimento della memoria disponibile.
Il ritorno alla lista di provenienza è controllato dal comando CTRL-C.
Per trasferire informazioni da una lista all'altra, anche di differente livello, l'utente deve inserire il carattere é come marcatore del nome e battere i comandi CTRL-G (per prelevare il dato) e CTRL-P (per depositare il dato nella lista di destinazione).
Eseguito lo spostamento, il marcatore @ passa automaticamente all'elemento successivo, che può così essere trasferito battendo semplicemente la sequenza CTRL-G/CTRL-P. Questa procedura consente di scambiare gruppi di informazioni tra liste di diverso livello.
Per inserire nuovi elementi in una lista, si

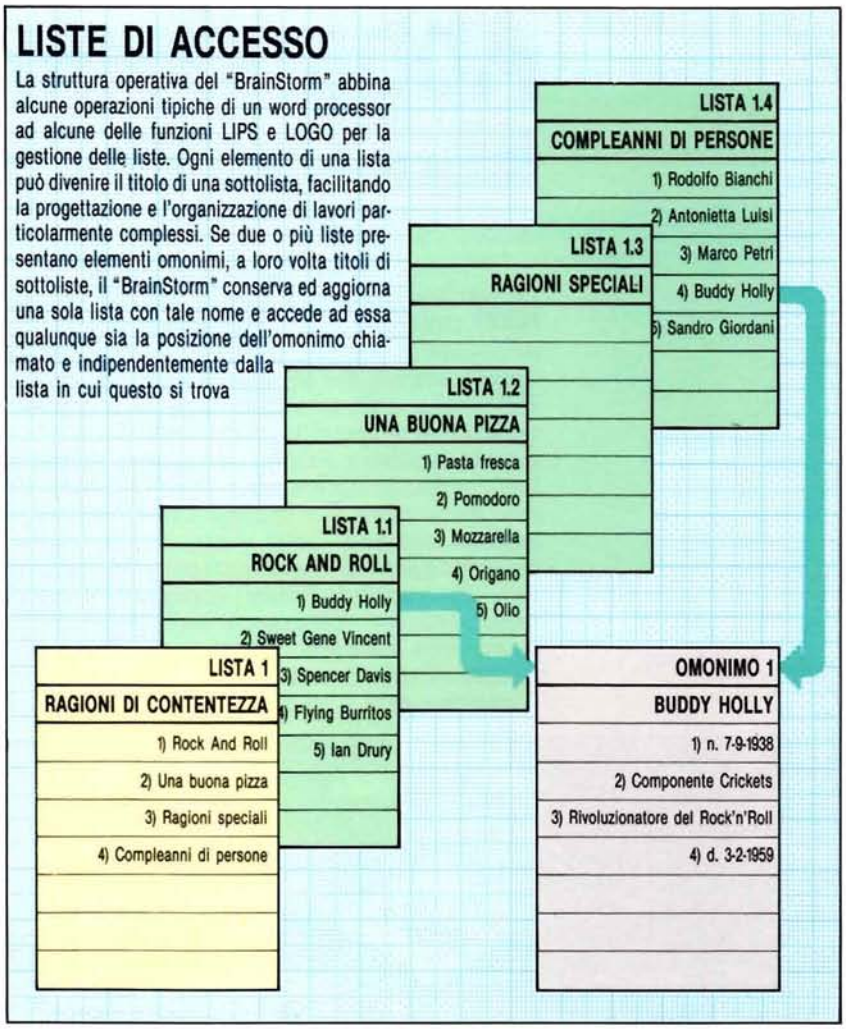
deve posizionare il cursore all'inizio della linea e battere l'informazione, quindi premere Return per spostare in basso di una linea gli elementi successivi della lista.
Il comando CTRL-A consente invece di apportare correzioni.
Il programma costruisce automaticamente un "riferimento incrociato" (detto "cross-reference") per tutti gli elementi omonimi (per esempio, una stessa data che ricorre in più liste).
Ciò permette, ad esempio, di creare facilmente una lista di eventi che capitano tutti lo stesso giorno.
È consentita la creazione di un numero illimitato di omonimi, che è poi possibile esaminare in sequenza, usando CTRL-D e CTRL-S per leggere, rispettivamente, l'elemento precedente o quello successivo.
Giunto all'ultimo elemento, CTRL-S riparte dal primo, quindi la sequenza è del tipo "ad anello".

Stampa delle liste

Naturalmente il *BrainStorm* consente di stampare le diverse liste che, di solito, vengono redatte tabulando opportunamente i vari componenti, a seconda del loro livello gerarchico. Il "rientro" delle tabulazioni è definito dall'utente.
Ad esempio:

```
Leggi il manuale
    Prendi il manuale
        Sfoglia le pagine
            Trova la pagina cercata
        Leggi la pagina
    Posa il manuale
Comincia a scrivere la lista
    Premi CTRL-R per trasformare un elemento della lista in un
        titolo di sottolista
        Scrivi la sottolista
    Premi CTRL-C per tornare alla lista precedente
Riguarda la lista
    Premi CTRL-A per correggere un'informazione
```

Il programma permette, inoltre, di redigere, stampare e salvare anche liste non destinate a *BrainStorm*, purché vengano salvate con il comando Write-to-Disk.
Queste liste hanno un formato tale da poter essere elaborate dal *WordStar* o da word processor simili. È quindi possibile preparare l'impostazione di un libro usando il *BrainStorm*, per poi sfruttare un comune word processor (che sia in grado di accedere al file BrainStorm.Doc) per redigere ed ampliare il testo stesso del libro.
Il programma consente, in sostanza, di pianificare oculatamente gli argomenti trattati, suddivisi per titoli di capitolo, da sviluppare in seguito. Se, durante la fase preliminare, gli argomenti di un capitolo crescono troppo è sempre possibile suddividerlo ulteriormente, creando una nuova sottolista, ossia un nuovo capitolo.
La stessa cosa vale per l'eventuale eliminazione di capitoli ritenuti superflui. Il programma è corredato da un modello dimostrativo (chiamato SAMPLE.BRN), composto da un modulo semplificato di catalogazione, da un file per nome e indirizzi e da una lista esplicativa delle operazioni previste (definita tramite le sottoliste "importante", "urgente", "da ricordare").
In più è presente anche una sezione per le annotazioni, che può essere abbinata anche ai modelli creati dall'utente selezionando l'opzione Merge.
Il manuale di corredo, redatto usando lo stesso *BrainStorm*, è molto chiaro ma solo in rari casi è necessario consultarlo, poiché il programma visualizza in permanenza il menu dei comandi.
Non tutti i comandi sono facili da ricordare, tuttavia è possibile cambiarne il formato e adattare il menu alle nuove denominazioni usando il programma *Installb* completamente controllato da menu.



BrainStorm: Adatto a circa 25 apparecchi CP/M, MS-DOS e PC-DOS, compresi l'IBM, il Sirius e l'Apricot
Editori: Caxton Software Ltd. (GB)
Autori: David Tebbutt e Mike Liardet
Formato: Su disco

Abbandonare oggetti

Ecco le routine per depositare gli oggetti raccolti durante l'avventura

La routine LASCIA somiglia molto alla routine PRENDI, descritta a pag. 1326: è quindi possibile utilizzare le stesse routine di verifica per gli oggetti sviluppate in precedenza. La routine PRENDI esegue tre test su ogni oggetto. Il primo verifica se la seconda parte della frase di comando contiene o meno un oggetto valido. Il test confronta sistematicamente ciascuna parola della frase di comando con i nomi degli oggetti contenuti nella matrice dell'inventario principale IV$(,). Se il confronto risulta positivo, alla variabile F viene assegnato il valore della posizione dell'oggetto individuato all'interno della matrice. Anche la routine LASCIA ricorre al test di validità per determinare se l'oggetto esiste e, in caso affermativo, per individuarne la posizione all'interno della matrice.

Il secondo test usato da entrambe le routine controlla se l'oggetto indicato nel comando è già presente tra quelli "trasportati" dal giocatore, ossia nella matrice IC$(). È ovvio, infatti, che il giocatore non può abbandonare un oggetto di cui non è in possesso! Nella routine PRENDI, il terzo test verifica che l'oggetto da raccogliere si trovi nella locazione visitata dal giocatore, come indicato dalla variabile di posizione P. In questo caso, poiché il giocatore deve possedere l'oggetto da abbandonare, la posizione di quest'ultimo non compare nell'inventario principale, e non occorre che la routine LASCIA esegua l'ultimo test.

Se entrambe le verifiche danno un risultato positivo, l'inventario principale e quello del giocatore vanno così modificati:

1) In F viene specificata la posizione dell'oggetto da abbandonare, mentre la posizione corrente, P, deve essere inserita nell'elemento IV$(F,2) dell'inventario principale.
2) La descrizione dell'oggetto deve essere cancellata dall'inventario personale IC$() del giocatore, scorrendo la matrice fino a trovare l'oggetto cercato e sostituendo la descrizione con una stringa nulla.

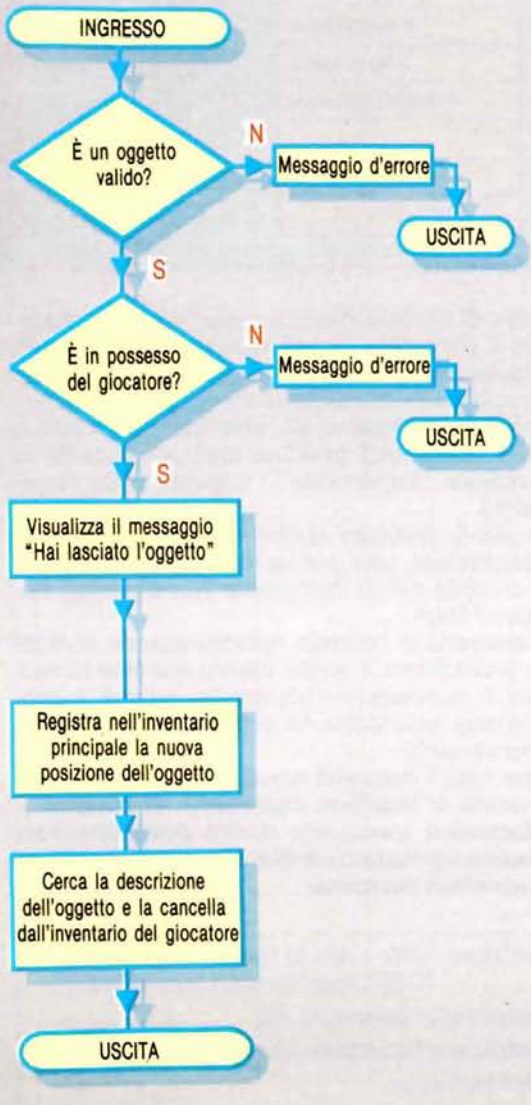
Il diagramma di flusso (vedere illustrazione) chiarisce la logica della routine LASCIA. Il seguente listato è il codice della routine per il gioco *Foresta Incantata*:

```
3900 REM **** S/R LASCIA ****
3910 GOSUB 5300: REM OGGETTO VALIDO?
3920 IF F=0 THEN SN$=W$ + " NON ESISTE":GOSUB
     5500:RETURN
3930 :
3940 REM ** OGGETTO FRA QUELLI TRASPORTATI? **
3950 OV=F: GOSUB 5450
3960 IF HF=0 THEN SN$="NON HAI CON TE ":+IV$(F,1)
     GOSUB 5500: RETURN
3970 :
3980 REM ** ABBANDONA L'OGGETTO **
3990 SN$="HAI ABBANDONATO ":+IV$(F,1)GOSUB 5500
4000 IV$(F,2)=STR$(P): AGGIORNA INVENTARIO PRINCIPALE
4010 :
4020 REM ** RIMUOVE OGGETTO DAI TRASPORTATI **
4030 FOR J=1 TO 2
4040 IF IC$(J)=IV$(F,1) THEN IC$(J)="": J=2
4050 NEXT J
4060 RETURN
```

La possibilità si adoperare per scopi differenti le stesse routine è uno dei principali vantaggi offerti dalla programmazione modulare e qui ne abbiamo un chiaro esempio. Ricorrendo ad

LASCIARE UN OGGETTO

La logica della routine LASCIA è simile quella della routine PRENDI (vedere a pag. 1326). Tuttavia, prima di eseguire il comando LASCIA, la routine verifica solo la validità dell'oggetto e la sua presenza nell'inventario del giocatore. Quindi aggiorna l'inventario principale registrandovi la nuova posizione dell'oggetto e cancella il nome dall'inventario del giocatore



IAN McKINNELL

un uso appropriato dei flag, è possibile utilizzare una serie di brevi subroutine decisionali, il cui effetto si manifesta solo quando il controllo ritorna alla routine chiamante. Un buon esempio di questo tipo di programmazione è il test di validità appena descritto, che viene chiamato da entrambe le routine PRENDI e LASCIA. In ambedue i casi, la subroutine decide sulla validità dell'oggetto indicato nella frase di comando, ma il flusso del programma non viene modificato finché un comando RETURN non restituisce il controllo alle routine. Solo allora viene selezionato il valore del flag F ed imboccata la relativa diramazione. Una giusta critica rivolta a questo metodo è che la stessa condizione viene verificata due volte: quando si assegna il valore del flag e quando lo si verifica. Tuttavia le maggiori flessibilità e facilità di debugging, ottenibili con questa tecnica, fanno passare in secondo piano la minor velocità d'esecuzione della routine.

Locazioni speciali

Con la precedente routine si conclude la programmazione dello "scheletro" del gioco, ossia di quella parte del programma che permette al giocatore di raccogliere gli oggetti e di muoversi all'interno dello scenario di gioco. Passiamo adesso alla progettazione delle locazioni speciali, al cui interno il giocatore adopera gli oggetti in suo possesso, va incontro a svariati pericoli e mette alla prova la propria abilità.
Prima di esaminare in dettaglio la programmazione delle routine per una delle locazioni speciali del gioco *Foresta Incantata*, inseriamo nel ciclo principale del programma le seguenti linee che consentono di individuare le locazioni speciali:

```
257 GOSUB 2700: REM P È UNA LOCAZIONE SPECIALE ?
258 IF SF = 1 THEN 300: REM ISTRUZIONE SUCCESSIVA
```

La linea 257 chiama una subroutine che verifica se la locazione corrente è tra quelle speciali. In caso affermativo il flag SF viene posto a 1 cosicché, quando il controllo ritorna al ciclo principale del programma, è possibile evitare quella parte di esso che tratta le istruzioni. La seguente subroutine è quella che decide se la locazione corrente è speciale:

```
2700 REM **** S/R P LOCAZIONE SPECIALE? ****
2705 SF = 0: REM AZZERA FLAG PER LOCAZ. SPECIALI

2716 REM ** ALTRE LOCAZIONI SPECIALI **
2720 ON P GOSUB 4590, 4690, 4790, 4590
2730 RETURN
```

Si rammenti che durante la formulazione della mappa dello scenario di gioco, per prima cosa si sono numerate le 4 locazioni speciali (pag. 1246). Possiamo quindi semplificare la selezione delle subroutine relative a ciascuna locazione speciale usando il comando ON...GOSUB. Questo comando, alla linea 2720, è seguito da una serie di numeri di linea che vengono selezionati in base al valore di P. Per P uguale a 1, ad esempio, il comando seleziona il primo numero di linea della lista, per P uguale a 2, il secondo e così via.
La lista contiene quattro numeri, uno per ogni locazione speciale di *Foresta Incantata*. Se P è maggiore di 4, il controllo passa alla linea successiva (che qui contiene una RETURN). Ciascuna delle quattro subroutine chiamate dalla linea 2720 pone a 1 il flag SF, indicando così che P è una locazione speciale. Se non viene chiamata alcuna subroutine, il flag SF rimane a 0 indicando così che P corrisponde ad una locazione normale. Il comando ON...GOSUB è una conveniente alternativa alla serie di istruzioni IF...THEN necessaria per verificare il valore di una variabile e per saltare alle relative subroutine.

L'ingresso della galleria

Le locazioni speciali 1 e 4 del gioco *Foresta Incantata* corrispondono ai due ingressi di una galleria. Perché il giocatore possa muoversi nello scenario di gioco che la contiene, occorre creare una routine che elabori i comandi normali e permetta al giocatore di entrare nella galleria o di tornare indietro.

```
4590 **** S/R INGRESSO ALLA GALLERIA ****
4600  SF = 1
4605 SN$ = "SEI GIUNTO ALL'INGRESSO DI UNA LUNGA GAL-
     LERIA": GOSUB 5500
4610 SN$ = "PUOI ENTRARE O TORNARE INDIETRO SUL SEN-
     TIERO": GOSUB 5500
4620 :
4625 PRINT:INPUT "IL TUO COMANDO";IS$
4630 GOSUB 2500: REM DIVIDE I COMANDI
4635 IF F = 0 THEN 4625: REM COMANDO NON VALIDO
4637 GOSUB 3000: REM COMANDO NORMALE
4640 IF MF = 1 THEN RETURN: REM GIOCATORE INDIETREG-
     GIA
4645 IF VF = 1 THEN 4625: REM COMANDO ESEGUITO
4650 REM ** NUOVI COMANDI **
4655 IF VB$ = "ENTRA" THEN GOSUB 4700: RETURN
4660 IF VB$ = "TORNA" AND P = 4 THEN MF = 1: P = 6: RETURN
4665 IF VB$ = "TORNA" AND P = 1 THEN MF = 1: P = 9 RETURN
4667 SN$ = "NON CAPISCO": GOSUB 5500: GOTO 4625
```

Per prima cosa la routine pone SF a 1, per indicare che si è raggiunta una locazione speciale. Poi visualizza un messaggio che descrive l'ingresso della galleria, le scelte consentite e chiede istruzioni al giocatore. Anche in questo caso si può ricorrere alla struttura modulare del programma per analizzare i diversi comandi richiamando anche le subroutine "suddivisione comandi" e "comando normale" usate dalle routine PRENDI e LASCIA. Un attento esame dello stato dei vari flag modificati da queste due subroutine consente di trasferire il controllo alla routine adatta al caso. Esaminiamo separatamente ciascun flag.
Il flag F viene selezionato dalla routine "suddivisione comandi" e indica se l'istruzione immessa è valida. Se questa è un comando a parola singola non riconosciuto dalla routine, il flag assume valore zero: così il giocatore

deve immettere un nuovo comando. Il flag MF viene posto a 1 dalla routine "comando normale", qualora occorra una descrizione dell'ambiente (il giocatore ha immesso il comando VAI oppure GUARDA). Una semplice pressione di RETURN fa sì che venga eseguito il comando di spostamento nel primo caso oppure, nel secondo caso, che venga descritto l'ambiente richiamando poi nuovamente la routine "locazioni speciali".
La routine "comando normale" modifica anche il flag VF: il valore uno indica che il comando è stato riconosciuto ed eseguito, nel qual caso il giocatore può immettere il comando successivo. Se invece VF<>1, il comando immesso non è un comando normale. Dato che la routine tratta solo comandi normali, possiamo introdurre due nuovi comandi: ENTRA (per avventurarsi nella galleria) e TORNA (per allontanarsi di una locazione dal suo ingresso). Poiché il progetto di questa routine prevede due ingressi alla galleria, il comando TORNA deve tener conto dell'entrata a cui il giocatore si riferisce: questa viene indicata da P, che contiene i valori 1 o 4. Quindi, prima di lasciare la routine, a P va assegnato un valore opportuno in modo che tornando al ciclo principale del programma si verifichi il cambiamento di locazione appropriato.
In seguito vengono progettati i pericoli speciali, che minacciano il giocatore all'interno della galleria.

Basic & Basic

Spectrum:

In entrambi i programmi sostituire i seguenti nomi di variabili: S$ per SN$, R$ per NN$, V$(,) per IV$(,), I$( ) per IC$( ), T$ per IS$, B$ per VB$. Sostituire le seguenti linee nel listato di "Foresta Incantata":

```
2720 IF P=1 THEN GOSUB 4590
2722 IF P=2 THEN GOSUB 4690
2724 IF P=3 THEN GOSUB 4790
2726 IF P=4 THEN GOSUB 4590
```

Sostituire le seguenti linee nel listato "Digitaya":

```
2710 IF P=1 THEN GOSUB 2850
2711 IF P=2 THEN GOSUB 2960
2712 IF P=3 THEN GOSUB 3450
2713 IF P=4 THEN GOSUB 3830
2714 IF P=5 THEN GOSUB 4180
2715 IF P=6 THEN GOSUB 4550
2716 IF P=7 THEN GOSUB 5150
3890 LET RD=RND(1)
4090 LET P=INT(RND(1)*40+7)
```

Micro BBC:

Sostituire le seguenti linee nel listato "Digitaya":

```
3890 RD=RND(1)
4090 P=RND(40)+7
```

LISTATO DIGITAYA

```
1190 GOSUB 2670: REM P LOCAZIONE SPECIALE?
1200 IF SF=1 THEN 1250: REM CICLO SUCCESSIVO

2360 REM ** S/R LASCIA **
2370 GOSUB 5730: REM OGGETTO VALIDO?
2380 IF F=0 THEN PRINT W$;" NON ESISTE":RETURN
2390 :
2400 REM ** L'OGGETTO E' PRESENTE ? **
2410 OV=F: GOSUB 5830
2420 IF HF=0 THEN PRINT "NON HAI CON TE ";IV$(F,1): RETURN
2430 :
2440 REM ** ABBANDONA L'OGGETTO **
2450 SN$="HAI ABBANDONATO "+IV$(F,1): GOSUB 5880
2460 IV$(F,2)=STR$(P): REM AGGIORNA POSIZ. OGGETTO
2470 :
2480 REM ** RIMUOVE OGGETTO DA QUELLI TRASPORTATI **
2490 FOR J=1 TO 4
2500 IF IC$(J)=IV$(F,1) THEN IC$(J)="": J=4
2510 NEXT J
2520 RETURN

2670 REM **** S/R LOCAZIONE SPECIALE IN P ****
2680 SF=0: REM AZZERA FLAG PER LOCAZ. SPECIALI

2710 ON P GOSUB 2850, 2960, 3450, 3830, 4180, 4550, 5150
2720 RETURN

2850 REM **** S/R USCITA DA TV ****
2860 SF=1
2870 SN$="HAI SCELTO L'USCITA TV E NON C'E' VIA DI SCAMPO."
2880 SN$=SN$+"SEI CONDANNATO A PARTECIPARE PER SEMPRE AD UNO SHOW
     TELEVISIVO"
2890 GOSUB 5880: REM VISUALIZZA MESSAGGIO
2900 PRINT
2910 PRINT "BENVENUTO NELLO SHOW ....."
2920 FOR J=1 TO 500: NEXT J
2930 GOTO 2910
2940 END

3830 REM **** LOCAZIONE: PORTA DEL JOYSTICK ****
3840 SF=1
3850 SN$="UN UTENTE DAGLI OCCHI ARROSSATI TI  CENTRA PIU' VOLTE COL
     SUO LASER."

3860 GOSUB 5880: REM VISUALIZZA MESSAGGIO
3870 :
3880 REM ** IMMISSIONE COMANDI **
3890 RD=RND(TI): IF RD>.65 THEN 4110: REM COLPITO!
3900 PRINT:INPUT "IL TUO COMANDO";IS$
3910 GOSUB 1700: GOSUB 1900: REM ESAMINA IL COMANDO
3920 IF MF=1 THEN MF=0:  PRINT "NON PUOI MUOVERTI... PER ORA": GOTO 3880
3930 IF VF=1 THEN 3880: REM COMANDO SUCCESSIVO
3940 IF VB$<>"USA" THEN PRINT "NON CAPISCO": GOTO 3880
3950 GOSUB 5730: REM OGGETTO VALIDO
3960 IF F=0 THEN PRINT NN$;" NON ESISTE":GOTO 3880: REM COMANDO
     SUCCESSIVO
3970 :
3980 REM ** L'OGGETTO E' UNO SCUDO ANTI-LASER **
3990 IF F=3 THEN 4020: REM TUTTO OK
4000 SN$="IL TUO "+IV$(F,1)+" NON FUNZIONA": GOSUB 5880: GOTO 3880
4010 :
4020 OV=3: GOSUB 5830: REM OGGETTO E' LO SCUDO ANTI-LASER
4030 IF HF=0 THEN SN$="NON POSSIEDI LO "+IV$(3,1):GOSUB 5880:
     GOTO 3880
4040 :
4050 REM ** SALVO! **
4060 SN$="LO SCUDO TI DIFENDE. COLPI A RAFFICA."
4070 SN$=SN$+"ESCI DALLA PORTA DEL JOYSTICK E RIENTRI NELLA
     MACCHINA."
4080 GOSUB 5880: REM VISUALIZZA MESSAGGIO
4090 P=INT(RND(TI)*40+7): MF=1: RETURN
4100 :
4110 REM ** COLPITO! **
4120 SN$="SEI  STATO DISINTEGRATO DAL LASER E TE  NE  RENDI
     CONFUSAMENTE CONTO"
4130 SN$=SN$+" GLI ATOMI DEL TUO CORPO SI SONO SPARSI"
4140 SN$=SN$+" PER TUTTO L'UNIVERSO"
4150 GOSUB 5880: REM VISUALIZZA MESSAGGIO
4160 END

5150 REM **** S/R INGRESSO IN MEMORIA ****
5160 SF=1
5170 SN$="UN GUARDIANO TI INFORMA CHE NON PUOI ENTRARE "
5180 SN$=SN$+" SE NON DAI UN INDIRIZZO": GOSUB 5880
5190 REM ** IMMISSIONE COMANDO **
5200 PRINT:INPUT "IL TUO COMANDO";IS$
5210 GOSUB 1700: GOSUB 1900: REM ESAMINA IL COMANDO
5220 IF MF=1 THEN RETURN: REM USCITA
5230 IF VF=1 THEN 5200: REM COMANDO SUCCESSIVO
5240 IF VB$<>"DAI" THEN PRINT "NON CAPISCO": GOTO 5200
5250 :
5260 GOSUB 5730: REM OGGETTO VALIDO?
5270 IF F=0 THEN PRINT W$;" NON ESISTE": GOTO 5200: REM COMANDO
     SUCCESSIVO
5280 :
5290 REM ** L'OGGETTO E' UN INDIRIZZO? **
5300 IF F=1 THEN 5330: REM TUTTO OK
5310 PRINT "HA BISOGNO DEL TUO INDIRIZZO": GOTO 5200
5320 :
5330 OV=1: GOSUB 5830: REM INDIRIZZO TRASPORATO?
5340 IF HF=1 THEN 5370
5350 SN$=IV$(1,1)+ " NON E' IN TUO POSSESSO":GOSUB 5880: GOTO 5200
5360 :
5370 REM ** INGRESSO IN MEMORIA CONSENTITO **
5380 SN$="IL  GUARDIANO  ESAMINA IL TUO  INDIRIZZO  E  TI LASCIA
     ENTRARE"
5390 SN$=SN$+" ATTRAVERSO": GOSUB 5880
5400 P=40: MF=1: RETURN
```

Il Compaq Plus

Anatomia di un microcomputer a diffusione europea: tecnologie a confronto

L'introduzione sul mercato statunitense del Personal Computer IBM ha notevolmente accresciuto il rispetto del pubblico verso i microcomputer. In particolare gli uomini d'affari, che in precedenza avevano considerato tali apparecchi poco più che semplici giocattoli, sono attirati dalla fama di questo gigante dell'informatica.
Il notevole successo del PC IBM ha indotto molte software house a sviluppare programmi per questo apparecchio e molti fabbricanti di computer a rendere le proprie macchine compatibili, ossia capaci di utilizzare la grande quantità di software creata per il PC della IBM.
Una delle compagnie più attive in questo settore è stata la Compaq Computer Corporation, fondata appositamente per produrre il Compaq Plus, uno dei primi portatili IBM compatibili. Il modello qui descritto è la versione a due unità a dischetti da 256 Kbyte, ma la Compaq fabbrica anche apparecchi ad un solo dischetto o con disco fisso da 10 Mbyte.

Caratteristiche dell'apparecchio

Il termine "portatile" è forse improprio per un micro che, pesando circa 14 Kg., è trasportabile solo per brevi distanze. La Compaq ha cercato tuttavia di ridurre le difficoltà di trasporto dotando la macchina di una comoda maniglia in materiale plastico.
Come in molti altri portatili, la tastiera viene agganciata al frontale del computer. Poiché la maniglia si trova sul retro del computer, durante il trasporto la tastiera viene a trovarsi in basso e funge da base d'appoggio ma, grazie alla sua robustezza, non sembra risentire del peso che deve sostenere. Nell'insieme l'unità ha le dimensioni di una macchina da cucire comune. Il collegamento tra la tastiera e l'unità centrale, una volta sganciata, è assicurato da un robusto cavo a spirale. La casa costruttrice afferma che questo tipo di collegamento consente di orientare la tastiera nel modo più confortevole ma, in realtà, la limitata elasticità del cavo riduce notevolmente la distanza tra lo schermo e l'utente; un cavo più sottile avrebbe consentito un uso più agevole del sistema.
Intervenendo sui piedini regolabili della tastiera e della stessa unità centrale che incorpora lo schermo, è possibile variare la loro angolazione per ottenere l'orientamento voluto. Come sulla maggior parte dei computer gestionali, i tasti del Compaq Plus sono affidabili e facili da usare. La loro disposizione sulla tastiera è identica a quella del PC IBM, con 10 tasti funzione alla sinistra dei tasti di scrittura ed una tastierina numerica sulla destra. Ovviamente questa configurazione facilita l'uso dell'apparecchio agli utenti abituati alla tasiera IBM, ma ne ripropone gli stessi inconvenienti. Sul PC IBM, il tasto Enter ha le stesse dimensioni degli altri tasti risultando di difficile individuazione; ulteriori problemi di battitura sono dovuti all'insolita disposizione dello Shift, che non si trova sotto il tasto A (al suo posto c'è il tasto "\"). La stessa cosa si può dire per la tastiera del Compaq Plus.
Con l'MS-DOS (la versione Microsoft del PC-DOS), ai 10 tasti sulla sinistra vengono assegnate funzioni di aiuto (richiamabili con help) per la redazione dei testi; tuttavia la funzione dei tasti varia a seconda del programma applicativo usato. In ambiente BASIC, ad esempio, i tasti funzione vengono adoperati per introdurre singole parole chiave come LOAD, SAVE e LIST. Un meccanismo simile è stato adottato per i tasti di funzione delle macchine MSX.
La tastierina numerica sulla destra svolge



CHRIS STEVENS

Assemblaggio "compatto"
Le linee morbide e luminose della custodia del Compaq Plus, riprese dal semplice frontale dell'apparecchio e dalla tastiera, ricordano molto quelle del PC della IBM. I pannelli laterali consentono di accedere alle porte d'interfaccia, alla presa d'alimentazione e ad un connettore d'espansione multi-uso, un'unità di grande importanza spesso sottovalutata!

due funzioni. In condizioni normali può essere usata per muovere il cursore sullo schermo, mentre quando viene premuto il tasto Num Lock si trasforma in una calcolatrice.
Lo schermo è un monitor standard a fosfori verdi di 17,7 x 13,3 mm., con una risoluzione di 80 x 25 caratteri, che consente una confortevole lettura dei testi. La luminosità viene regolata tramite una piccola manopola posta a sinistra. Sulla destra si trovano le due unità a dischetti da 5 1/4 pollici. Poiché la configurazione base del Compaq Plus prevede una sola unità, il disco di sistema dell'MS-DOS fa costante riferimento all'unità A: ciò comporta una certa perdita di tempo soprattutto nelle operazioni di copia perché l'utente è costretto ad alternare più volte i dischetti nello stesso drive.
Tuttavia l'MS-DOS può essere configurato in modo da consentire l'impiego di entrambe le unità il cui funzionamento si dimostra più rapido di quello dei drive IBM.

Collegamenti esterni

Sollevando il pannello mobile sulla destra dell'apparecchio, si accede alle porte d'interfaccia. Il Compaq Plus possiede una porta per stampante parallela tipo Centronics, un'interfaccia RGB ed un'uscita RF. Inoltre incorpora tre connettori d'espansione sui quali possono essere innestate schede IBM compatibili (schede di memoria aggiuntiva, una scheda VDU per il colore oppure un modem, che permette all'apparecchio di comunicare attraverso la rete telefonica). Un secondo pannello sulla sinistra copre la presa d'alimentazione di tipo standard, sopra la quale si trova l'interruttore d'accensione del sistema. Lateralmente è visibile il ventilatore che garantisce il raffreddamento dell'unità centrale. All'interno del computer, i circuiti sono protetti da una custodia metallica (che contribuisce in modo significativo ad aumentare il peso dell'apparecchio); questa non solo protegge la macchina da eventuali urti, ma scherma anche i circuiti da interferenze radio che potrebbero disturbare l'elaborazione. Per di più il metallo favorisce la dispersione di calore.
Il Compaq Plus è corredato dai dischetti di sistema e da tre manuali: una guida al funzionamento e due manuali di consultazione (uno per l'MS-DOS e uno per il BASIC). Dei tre manuali, solo il primo ha un'impostazione di tipo didattico quindi, a chi desidera imparare ad usare organicamente il BASIC o l'MS-DOS, conviene acquistare pubblicazioni complementari.
Come tutti i primi apparecchi IBM compatibili, il Compaq Plus possiede lo stesso processore 8088 Intel del PC IBM e non il più avanzato 8086 o 80186, montato su apparecchi prodotti in seguito, quali l'Olivetti M24. Sebbene l'8088 sia un processore a 16 bit con un bus d'indirizzamento a 20 bit, esso possiede un bus dati ad 8 bit invece del bus a 16 bit presente sull'8086. Questa caratteristica sta a significare che il Compaq Plus, funzionante alla stessa velocità del PC, accede ai dati più lentamente degli apparecchi concorrenti basati sull'8086.
Per quanto riguarda la compatibilità software, il Compaq Plus è in grado di eseguire anche i più complessi programmi prodotti per l'IBM, quali il *Lotus 1-2-3*. Uno dei pochi programmi "proibiti" all'apparecchio è il dischetto diagnostico del PC IBM. Tale limitazione non deve meravigliare dato che questo programma si rivolge direttamente alla routine ROM del Sistema di Input/Output di Base (BIOS). Grazie alla sua pressoché totale compatibilità con il software IBM, il Compaq Plus si è affermato come uno dei più affidabili apparecchi IBM compatibili.
Le uniche note negative riguardano la sua configurazione decisamente superata, evidenziata tra l'altro dall'impiego del vecchio chip 8088. Inoltre il successo degli pseudoportatili (di peso superiore ai 10 Kg.) è seriamente compromesso dallo sviluppo di computer tascabili sempre più sofisticati, anch'essi IBM compatibili.

UN BUON ACQUISTO

Il principale punto di forza del Compaq Plus è la sua totale compatibilità con il PC dell'IBM e, quindi, la possibilità di accedere alla immensa quantità di software disponibile per tale apparecchio. La figura mostra l'inconfondibile disegno del PC IBM, pietra di paragone per il Compaq Plus. In termini di prezzo, quello del Compaq Plus è sensibilmente inferiore di quello del PC dell'IBM



CHRIS STEVENS

Scheda per stampante
Questa scheda dotata di interfaccia per stampante parallela consente di collegare l'apparecchio a stampanti di tipo Centronics

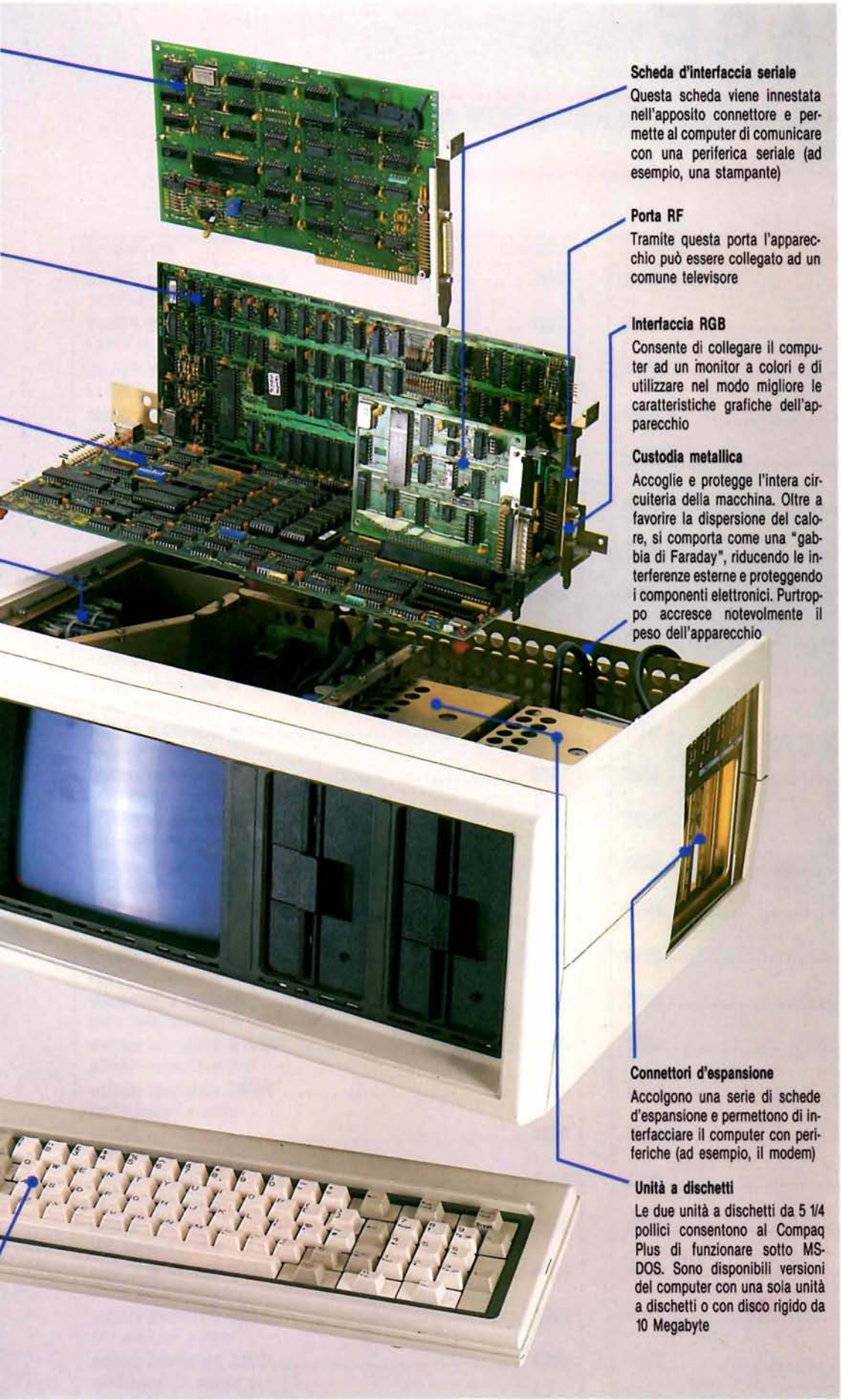
Scheda VDU
Viene innestata in una delle porte d'espansione del Compaq Plus e permette di controllare un televisore esterno o un monitor

Scheda principale
Accoglie una RAM di 256 Kbyte, il processore 8088 e i chip di input/output. Presenta anche zoccoli di riserva per l'introduzione di chip accessori, quali il coprocessore matematico 8087

Alimentazione e ventilatore
Accanto alla presa d'alimentazione si trova un ventilatore che garantisce il raffreddamento interno della macchina



Tastiera tipo IBM
La tastiera è identica a quella dell'IBM anche nella disposizione leggermente inclinata dei tasti. Si notino i tasti funzione sulla sinistra e la tastierina numerica sulla destra



Scheda d'interfaccia seriale
Questa scheda viene innestata nell'apposito connettore e permette al computer di comunicare con una periferica seriale (ad esempio, una stampante)

Porta RF
Tramite questa porta l'apparecchio può essere collegato ad un comune televisore

Interfaccia RGB
Consente di collegare il computer ad un monitor a colori e di utilizzare nel modo migliore le caratteristiche grafiche dell'apparecchio

Custodia metallica
Accoglie e protegge l'intera circuiteria della macchina. Oltre a favorire la dispersione del calore, si comporta come una "gabbia di Faraday", riducendo le interferenze esterne e proteggendo i componenti elettronici. Purtroppo accresce notevolmente il peso dell'apparecchio

Connettori d'espansione
Accolgono una serie di schede d'espansione e permettono di interfacciare il computer con periferiche (ad esempio, il modem)

Unità a dischetti
Le due unità a dischetti da 5 1/4 pollici consentono al Compaq Plus di funzionare sotto MS-DOS. Sono disponibili versioni del computer con una sola unità a dischetti o con disco rigido da 10 Megabyte

CHRIS STEVENS

COMPAQ PLUS

DIMENSIONI

mm. 480 x 400 x 200

CPU

8088 della Intel, frequenza di lavoro 4,7 MHz

MEMORIA

256 Kbyte di RAM, espandibile a 640 Kbyte

VIDEO

Modo testuale: 25 righe di 80 caratteri. Modo grafico: 640 x 200 pixel

INTERFACCE

Porta parallela per stampanti tipo Centronics, uscita RGB e per RF. Sono presenti connettori per schede aggiuntive

LINGUAGGI FORNITI

BASIC, caricato dal dischetto principale di sistema

TASTIERA

Tipo QWERTY, con 47 tasti, più 10 tasti di controllo, 10 tasti di funzione e 10 tasti con funzione di calcolatrice

DOCUMENTAZIONE

I manuali di corredo sono tre: una guida operativa, una guida al BASIC e una al sistema operativo MS-DOS. Tra questi, soltanto la guida operativa è impostata con criteri didattici e risulta molto adatta ai meno esperti. Può rendersi necessario l'acquisto di ulteriori pubblicazioni per meglio apprendere le funzioni del BASIC e dell'MS-DOS

PRO

L'apparecchio è affidabile e compatto. Inoltre la compatibilità con il PC dell'IBM è pressoché totale

CONTRO

Il peso dell'intera unità rientra a malapena nella categoria dei veri portatili. L'impiego di un chip già "superato" limita la flessibilità e la velocità del sistema

DATABASE

Ecco la seconda tabella sulle funzioni delle routine ROM del Micro BBC, assieme ai relativi indirizzi

DESCRIZIONE	INDIRIZZO
AND operator routine routine operatore AND	9B7A
ASC function routine routine funzione ASC	AC9E
ASN function routine routine funzione ASN	A8DA
ATN function routine routine funzione ATN	A907
AUTO command routine routine comando AUTO	90AC
Assemble a single instruction assemblaggio singola istruzione	85BA
Assembler base op-codes indirizzo iniziale op-code (assembler)	84C5
Assembler entry indirizzo d'ingresso dell'assembler	8504
Assembler exit indirizzo d'uscita dell'assembler	84FD
Assembler label definition indirizzo definizioni etichette (assembler)	85A5
Assembler mnemonics table (LSB) indirizzo tabella cod. mnem. (assembler)	8451
Assembler mnemonics table (MSB) indirizzo tabella cod. mnem. (parte alta)	848B
Assign value to numeric variable routine per ass. valori num. alle variabili	B4B1
BASIC error handler routine trattamento di errori BASIC	B433
BGET statement routine routine istruzione BGET	BF6F
BPUT statement routine routine istruzione BPUT	BF58
BRK handling routine routine per la gestione di BRK	B402
CALL statement routine routine istruzione CALL	8ED2
CHAIN statement routine routine istruzione CHAIN	BF2A
CHR$ function routine routine funzione CHR$	B3BD
CLEAR statement routine routine istruzione CLEAR	928D
CLG statement routine routine istruzione CLG	8EBD
CLOSE statement routine routine istruzione CLOSE	BF99
CLS statement routine routine istruzione CLS	8EC4
COLOUR statement routine routine istruzione COLOUR	938E

INDIRIZZO	DESCRIZIONE
8A8C	**Get character from PTR#2** prende un carattere da PTR#2
8A97	**Get character from PTR#1** prende un carattere da PTR#1
8AAE	**Check for a comma at PTR#2** controllo per virgola a PTR#2
8AB6	**OLD command routine** routine comando OLD
8AC8	**END statement routine** routine istruzione END
8AD0	**STOP statement routine** routine istruzione STOP
8ADA	**NEW command routine** routine comando NEW
8ADD	**Cold start** avvio del sistema 'a freddo'
8AF3	**Warm start** avvio del sistema 'a caldo'
8B47	**= statement routine** routine istruzione =
8B73	*** statement routine** routine istruzione *
8B7D	**DATA statement routine** routine istruzione DATA
8B7D	**DEF statement routine** routine istruzione DEF
8B7D	**REM statement routine** routine istruzione REM
8BE4	**LET statement routine** routine istruzione LET
8C1E	**String assignement** assegnazione stringa
8CC1	**Unstack a parameter** estrazione di parametri dallo stack
8D2B	**PRINT# statement routine** routine istruzione PRINT#
8D9A	**PRINT statement routine** routine istruzione PRINT
8E24	**TAB (X** routine funzione TAB (X
8E40	**TAB (X) routine** routine funzione TAB (X)
8E58	**SPC routine** routine funzione SPC
8EDB	**CLG statement routine** routine istruzione CLG
8EC4	**CLS statement routine** routine istruzione CLS
8ED2	**CALL statement routine** routine istruzione CALL

Grafici in Logo

Qualche metodo per rappresentare dati in una forma grafica comprensibile a colpo d'occhio

I grafici a barre sono semplici rappresentazioni grafiche di una serie di dati numerici. La loro funzione è quella di fornire un quadro abbastanza significativo di un gruppo di informazioni. Quasi tutti i package di grafica gestionale oggi disponibili, oltre a permettere la creazione di grafici di vario tipo (istogrammi, grafici "a torta", ecc.), sono spesso compatibili con i programmi spreadsheet e possono quindi visualizzare rapidamente, in forma grafica, i risultati elaborati.
Ecco come produrre un grafico a barre usando il linguaggio LOGO. Il programma è stato scritto per il Commodore 64: per adattarlo ad altri apparecchi occorre consultare le informazioni già fornite in "Logo & Logo". La procedura per la creazione del grafico a barre si compone di tre fasi:

a) immissione dei dati;
b) individuazione del valore massimo (necessario per la definizione di una "scala" che consenta un adeguato dimensionamento di tutte le barre);
c) tracciamento del grafico a barre, in base alla scala calcolata e aggiunta delle scritte esplicative.

La procedura principale, GRAFBARRE, richiama INIZ, che definisce il valore di tutte le costanti usate dal programma (questo raggruppamento facilita eventuali modifiche successive). COLORI contiene la lista dei 15 diversi colori per le 15 barre visualizzate dal programma.

Immissione e calcolo dei valori

Ogni barra del grafico viene definita da due dati immessi da tastiera: il primo è la scritta da abbinare alla barra e il secondo indica il valore numerico associato, ossia l'altezza della barra. La routine di immissione prevede alcune semplici operazioni di convalida sulla correttezza dei dati immessi. La procedura ACCETTA.DATO suddivide le operazioni d'inserimento in due fasi: una per l'immissione dei titoli di ogni colonna, l'altra per l'introduzione del valore numerico delle barre. Per terminare l'immissione dei dati, occorre digitare FINE come ultimo "titolo".
La procedura ACCETTA.TITOLO consente l'impiego di qualsiasi carattere, ma rifiuta un'immissione nulla. ACCETTA.VALORE accetta solo valori numerici: se si prova ad immettere altri caratteri il programma reagisce invitando l'utente a ripetere l'operazione. Quando la procedura ACCETTA.DATI ha ricevuto una corretta coppia titolo/valore, i due nuovi dati vengono aggiunti ad una lista. Le informazioni vanno introdotte nell'ordine in cui devono poi essere visualizzate sullo schermo, ossia da sinistra a destra.
La procedura CALCOLA usa VALORE.MAX per individuare il valore massimo e per stabilire la scala. Abitualmente l'altezza massima delle barre dovrebbe corrispondere all'incirca ai tre quarti della larghezza dell'intero grafico.

Costruzione del grafico

La procedura TRACCIA.GRAFICO calcola la larghezza di ciascuna barra del grafico, traccia un asse verticale sulla sinistra dello schermo, colora le barre e visualizza le etichette. Il valore della larghezza deve essere un multiplo di otto, per evitare alcuni inconvenienti sul Commodore: l'abituale modo grafico del LOGO non consente di inserire due colori in uno stesso blocco di 8 x 8 pixel. TRACCIA.ASSE traccia sullo schermo l'asse verticale e stampa alla sua sommità il valore della più alta barra del grafico: si ha così un parametro di confronto approssimativo delle diverse colonne visualizzate. Il numero che indica il valore massimo deve essere visualizzato ad una certa distanza dall'asse verticale; il LOGO tratta allo stesso modo i valori numerici e le parole, quindi possiamo usare la primitiva COUNT sia per calcolare la lunghezza del numero, sia per allontanarci dall'asse verticale della distanza necessaria.
Per visualizzare una scritta sullo schermo nel modo grafico, occorrono le coordinate x e y della posizione di ciascun carattere e la scritta da visualizzare. SCRIVI usa la primitiva STAMPCHAR, che visualizza un carattere sullo schermo grafico nella posizione occupata dalla tartaruga del LOGO. TRACCIA.GRAF1 traccia, una alla volta, le barre del grafico selezionandone il colore, calcolando l'altezza in scala, ma è la procedura COLORA che si muove lungo la barra colorandola. SCRITTA usa la procedura SCRIVI per visualizzare la parola di commento sullo schermo procedendo dall'alto verso il basso (le etichette troppo lunghe vengono interrotte e riprese a partire dal margine superiore dello schermo).

Grafici a torta

Un altro sistema di rappresentazione grafica è quello dei grafici cosiddetti "a torta". Ecco

alcune considerazioni per scrivere un programma che tracci grafici di questo tipo:

- I dati vengono elaborati con la stessa tecnica usata per il grafico a barre.
- La procedura di calcolo deve sommare i numeri introdotti per poter determinare l'ampiezza della fetta di cerchio (360°) corrispondente a ciascun valore.
- Le procedure di tracciamento e di colorazione della torta sono semplici, ma la loro programmazione sul Commodore 64 può presentare difficoltà (i colori tendono a sovrapporsi gli uni agli altri). Per ovviare a ciò si può ricorrere al modo "double colour", ossia usare la primitiva DOUBLECOLOUR al posto di DRAW nella procedura che disegna la torta. Se questo non basta, si può provare a introdurre un "foro" con diametro di 10 unità al centro della torta.

Logo & Logo

Nelle versioni LOGO prive di un comando equivalente alla primitiva STAMPCHAR, si incontrano grosse difficoltà nel visualizzare caratteri o scritte sullo schermo grafico.
I numeri di colori e le dimensioni del grafico a barre vanno adattati a seconda dell'apparecchio usato. Queste costanti sono state riunite nella procedura INIZ per facilitarne la modifica.

Per tutte le versioni LCSI:

Usare TYPE al posto di PRINT1 ed EMPTYP al posto di EMPTY?
Può essere necessario sostituire la primitiva EQUALP al segno "="
SETXY deve essere seguito da una lista
Il comando IF presenta una diversa sintassi; ad esempio:

IF EMPTYP :LISTADATI [STOP]

GRAFICI A VOLONTÀ

I grafici a barre ed a torta sono ottimi sistemi per rappresentare in forma grafica le informazioni. Il LOGO è un linguaggio piuttosto lento e non è certo lo strumento ideale per questo tipo di applicazioni.

DATI IN INGRESSO

	Londra	New York
GEN	53	94
FEB	40	97
MAR	37	91
APR	38	81
MAG	46	81
GIU	46	84
LUG	56	107
AGO	59	109
SET	50	86
OTT	57	89
NOV	64	76
DIC	48	91

Precipitazioni (in millimetri)

GRAFICO A BARRE - Londra

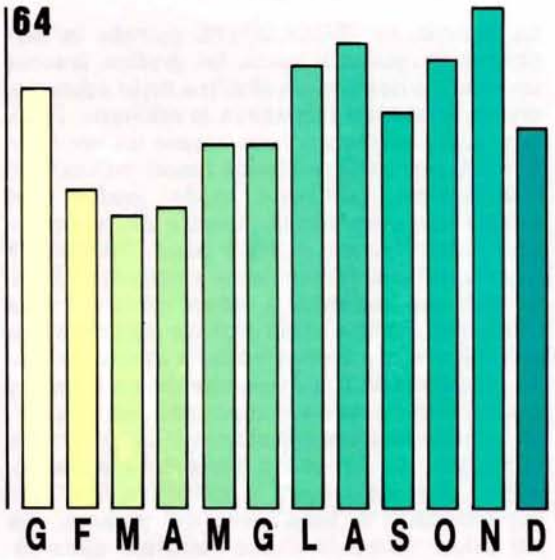


GRAFICO A TORTA - Londra

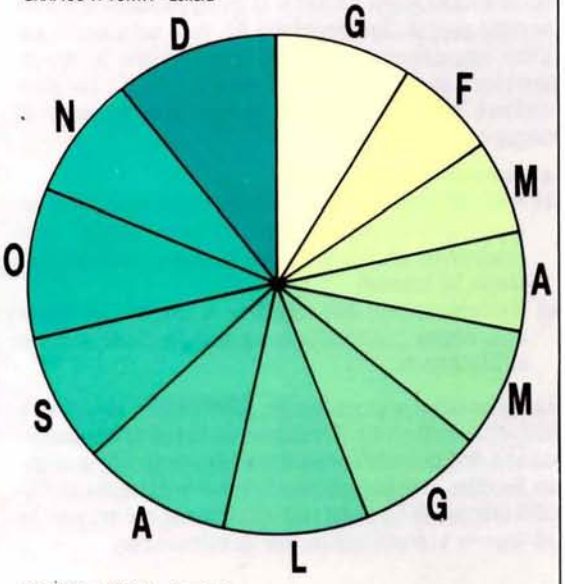


GRAFICO A BARRE - New York

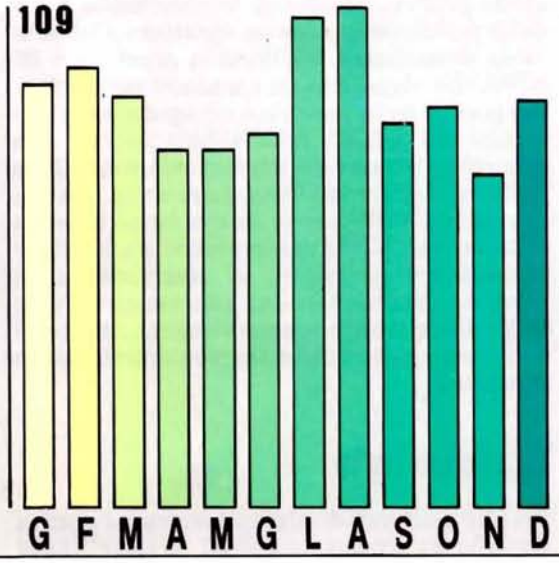
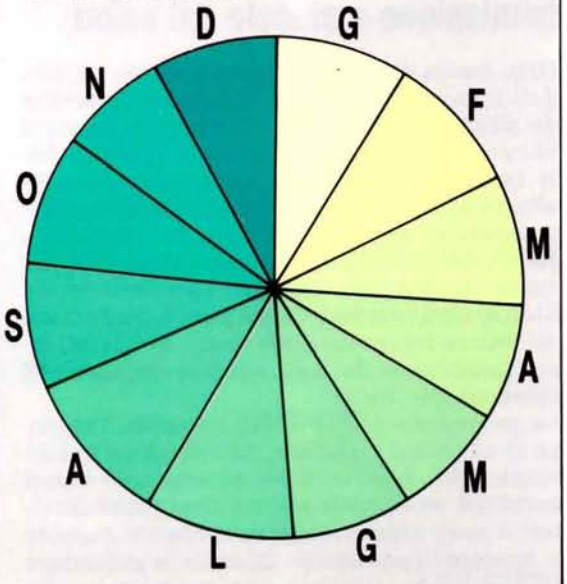


GRAFICO A TORTA - New York



La visualizzazione delle scritte è identica a quella del grafico a barre, ma ci possono essere difficoltà quando si posiziona il cursore (ossia la tartaruga) prima del testo.

Come esercizio, si provi a sviluppare un programma che, usando uno stesso gruppo di dati, tracci un grafico a barre sulla sinistra dello schermo ed uno a torta sulla destra.

PROGRAMMA DEL GRAFICO A BARRE

```
TO GRAFBARRE
   INIZ NODRAW
   ACCETTA.DATO CALCOLA
   TRACCIA.GRAFICO
END

TO INIZ
   MAKE "COLORI [0 1 2 5 6 0 1 2 5 6 0 1 2 5.6]
   MAKE "SFONDO 11
   MAKE "DATO []
   MAKE "ALTEZZA 160
   MAKE "LARGHEZZA 190
   MAKE "XSTART (-104)
   MAKE "YSTART (-40)
END

TO ACCETTA.DATO
   ACCETTA.TITOLO
   IF FIRST :TITOLO = "END THEN STOP
   ACCETTA.VALORE
   MAKE "DATO LPUT SENTENCE :TITOLO :VALORE :DATO
   ACCETTA.DATO
END

TO ACCETTA.TITOLO
   PRINT "
   PRINT [(SCRIVI FINE PER TERMINARE L'INTRODUZIONE
      DEI DATI)]
   (PRINT1 "TITOLO:"' ')
   MAKE "TITOLO REQUEST
   IF EMPTY? :TITOLO THEN ACCETTA.TITOLO
END

TO ACCETTA.VALORE
   (PRINT1 "VALORE:"' ')
   MAKE "VALORE REQUEST
   IF EMPTY? :VALORE THEN ACCETTA.VALORE ELSE IF
      NOT NUMBER?
   FIRST :VALORE THEN ACCETTA.VALORE
END

TO CALCOLA
   MAKE "MAX 0
   ACCETTA.MAX :DATO
   MAKE "SCALA :ALTEZZA/:MAX
END

TO VALORE.MAX :LISTADATI
   IF EMPTY? :LISTADATI THEN STOP
   IF LAST FIRST :LISTADATI>:MAX THEN MAKE "MAX
      LAST FIRST:LISTADATI
   VALORE.MAX BUTFIRST :LISTADATI
END

TO TRACCIA.GRAFICO
   DRAW BACKGROUND :SFONDO
   HIDETURTLE FULLSCREEN
   MAKE "LARG ((ROUND((:LARGHEZZA/COUNT :DATO)/
      8))*8)-8
   PENUP SETXY :XSTART :YSTART
   TRACCIA.ASSE
   TRACCIA.GRAF1 :COLORI :DATO
   SCRITTA :DATO
END

TO TRACCIA.ASSE
   PENDOWN
   SETY YCOR + :ALTEZZA + 10
   SETY YCOR - 10
   SETX XCOR + 4
   SETX XCOR - 8
   PENUP
   SETX XCOR - (8*COUNT :MAX)
   SCRIVI 8 0 :MAX
   SETX XCOR + 4
   SETY :START
END

TO SCRIVI :XINC :YINC :NOME
   IF EMPTY? :NOME THEN STOP
   STAMPCHAR FIRST :NOME
   SETXY XCOR + :XINC YCOR + :YINC
   SCRIVI :XINC :YINC BUTFIRST :NOME
END

TO TRACCIA.GRAF1 :PENCOLORS :LISTADATI
   IF EMPTY? :LISTADATI THEN STOP
   SETX XCOR + 8
   PENCOLOR FIRST :PENCOLORS
   PENDOWN
   COLORA ((LAST FIRST: LISTADATI)*SCALE):LARG
   PENUP
   TRACCIA.GRAF1 BUTFIRST :PENCOLORS BUTFIRST
      :LISTADATI
END

TO COLORA :LUNG :AMP
   IF AMP = 0 THEN STOP
   FORWARD :LUNG RIGHT 90
   FORWARD 1 LEFT 90
   BACK :LUNG RIGHT 90
   FORWARD 1 LEFT 90
   COLORA :LUNG :AMP - 2
END

TO SCRITTA :LISTADATI
   SCRITTA1 (:XSTART + 8 + :LARG/2) (:YSTART-10)
      :LISTADATI
END

TO SCRITTA1 :X :Y :LISTADATI
   IF EMPTY? :LISTADATI THEN STOP
   SETXY :X :Y
   SCRIVI 0 (-10) FIRST FIRST :LISTADATI
   SCRITTA1 (:X + :LARG + 8) :Y BUTFIRST:LISTADATI
END
```



Micro elettronica

Quattro interruttori miniaturizzati usati come sensori nel nostro robot

Il controllo bidirezionale dei motori passo passo impiega quattro delle otto linee per i dati della porta utente. Restano quindi quattro linee, utilizzabili per inviare al computer le informazioni provenienti dai sensori. Per rendere più adattabile il funzionamento del veicolo robot è previsto un piccolo pannello di connessione, che permette di variare le combinazioni dei sensori collegati alle quattro linee d'ingresso disponibili. Per il momento al pannello vengono collegati quattro sensori di contatto a microinterruttore: in seguito al robot vengono aggiunti anche due fotosensori. Per selezionare la configurazione desiderata dei sensori (ad esempio, due di contatto e due fotosensibili), basta collegare ciascun sensore ad una delle prese blu inserite sul coperchio del robot. Le quattro prese impiegate vengono poi collegate alle 4 linee dati, da D4 a D7, del connettore D.
I sensori vengono connessi senza un particolare ordine alle quattro linee dati per mezzo di brevi ponticelli "volanti", innestati tra le prese del sensore e l'appropriata presa del connettore D.
Per verificare il corretto funzionamento dei quattro sensori, possiamo realizzare un semplice programma che esegue una scansione dei quattro bit superiori del registro dati e visualizza i valori decimali dei bit bassi. La "chiusura" di un microinterruttore provoca un cambiamento nel valore visualizzato sullo schermo (ci si rammenti, naturalmente, di collegare i sensori!).

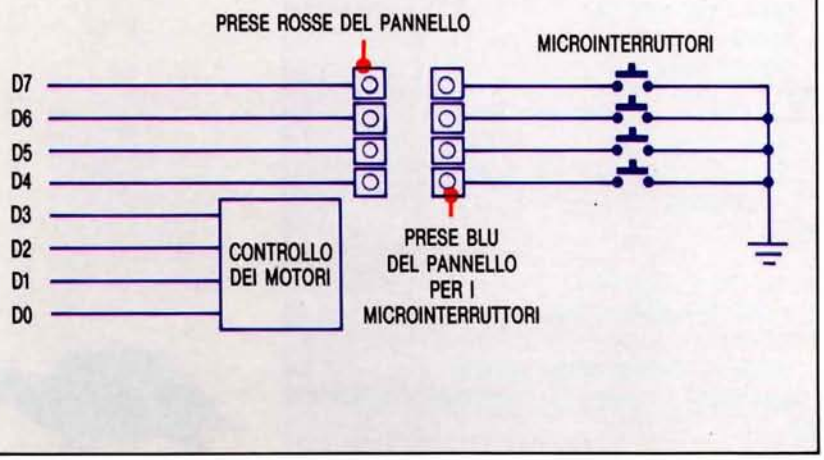
```
10 REM *** PROG. BBC PER CONTROLLO SENSORI ***
20 MODE 7:OP=-1:DDR=&FE62:REGDAT=&FE60:?DDR=15
30 PE=240-(?REGDAT AND 240):IF PE=OP THEN 30
40 CLS:PRINT PE:OP=PE:GOTO 30
```

```
10 REM *** PROG. CBM 64 PER CONTROLLO SENSORI ***
20 OP=-1:DDR=56579:REGDAT=56577:POKE DDR,15
30 PE=240-(PEEK(REGDAT) AND 240):IF PE=OP THEN 30
40 PRINT CHR$(147):PRINT PE:OP=PE:GOTO 30
```

MONTAGGIO DEI MICROINTERRUTTORI

I sensori di contatto (semplici microinterruttori) sono reperibili presso i rivenditori di materiale elettronico. Per prima cosa si eseguano le otto incisioni sul coperchio del robot, che devono corrispondere ai contatti posti sotto la base dei microinterruttori. Quindi vanno inserite, negli appositi fori da 5 mm., le prese per il pannello dei collegamenti volanti: quelle rosse devono trovarsi allineate vicino al connettore D. I microinterruttori richiedono qualche piccola modifica (vedere la figura a pag. 1357). Le lunghe "levette" di contatto vanno piegate ad angolo retto (si usino delle pinzette) mantenendole però ad una certa distanza dalla custodia del microinterruttore. Una volta montati i sensori sul coperchio del robot, si verifichi che le levette facciano scattare regolarmente il microinterruttore (deve bastare una minima pressione). Dal retro del microinterruttore sporgono due piedini di collegamento: quello superiore, detto NC (contatto normalmente chiuso), non viene utilizzato e può essere tagliato; quello inferiore, indicato con NO (contatto normalmente aperto) viene invece usato e va piegato ad angolo retto verso il basso in modo da infilarsi, assieme al contatto COM (contatto comune), nelle due fessure aperte sul coperchio per ciascun microinterruttore. Per fissare saldamente i quattrò microinterruttori al coperchio del veicolo, si usi una colla per materiali plastici. Le levette che azionano i microinterruttori devono trovarsi "sospese" sul davanti e sul retro del veicolo

CIRCUITO DEI MICROINTERRUTTORI

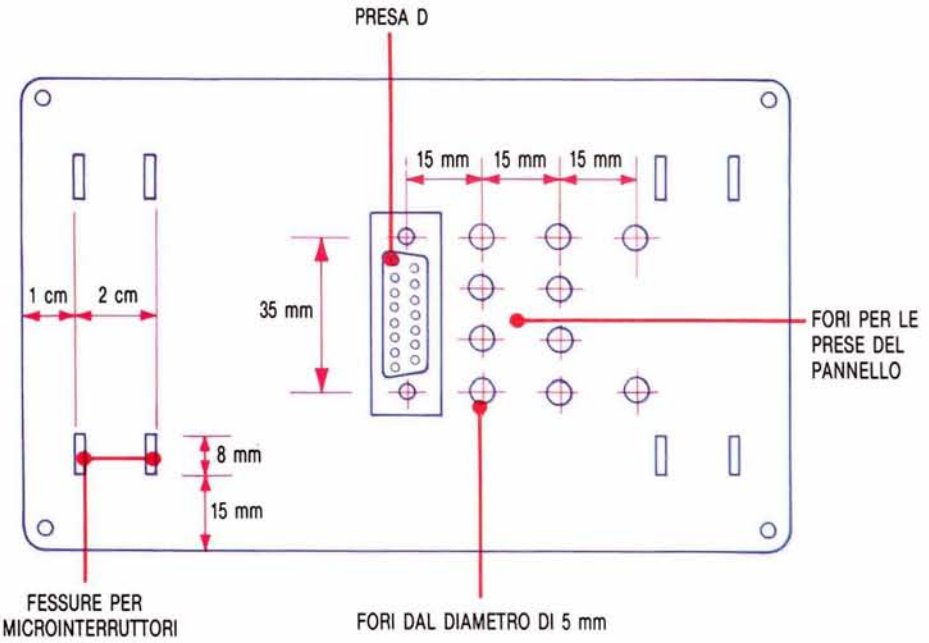


IL CIRCUITO DEI MICROINTERRUTTORI

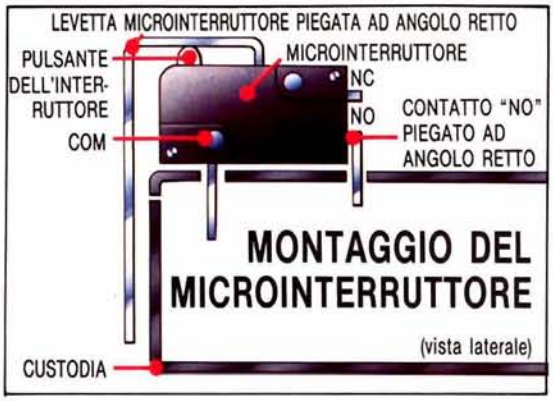
Il circuito che collega i microinterruttori al robot è molto semplice. Le linee per i dati da D4 a D7 vengono collegate alle quattro prese rosse del pannello sul coperchio del robot. Le linee per i dati da D0 a D3 vengono usate per controllare i due motori passo passo. Ciascun microinterruttore è in contatto da un lato con la corrispondente presa blu del pannello, dall'altro con una massa comune. Se la configurazione richiesta è a quattro sensori, questi vanno collegati alle linee dati da quattro ponticelli "volanti". Se i quattro bit superiori del registro dati della porta utente vengono selezionati per l'ingresso, il loro valore è normalmente 1. La chiusura di un microinterruttore collega la corrispondente linea dei dati a massa azzerando il relativo bit del registro dati

FORI DEL COPERCHIO

(vista dal'alto)



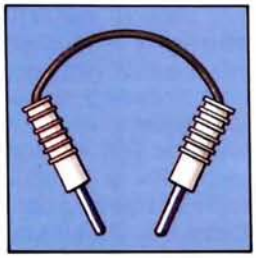
KEVIN JONES



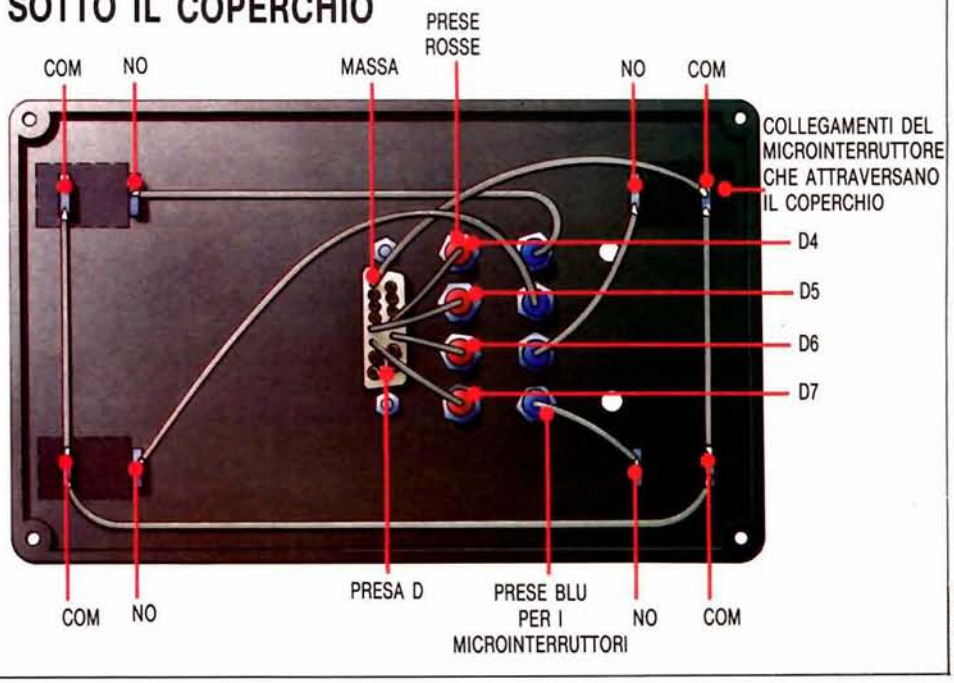
ELENCO DEI COMPONENTI

Quantità	Descrizione
8	spinotti (diam. 2 mm)
4	prese rosse (diam. 2 mm)
6	prese blu (diam. 2 mm)
4	microinterruttori
	cavo a nastro a 4 vie

Dopo aver fissato i microinterruttori si capovolga il coperchio. Saldare quattro brevi cavetti rivestiti alle quattro prese rosse e collegarli ai rispettivi piedini della presa D. Connettere l'uno all'altro i quattro contatti COM e collegarli al piedino di massa della presa D, come indicato. Quindi si colleghino i contatti NO alle corrispondenti prese blu del pannello. Infine fabbricare quattro cavetti "volanti" da innestare sul pannello di collegamento (vedi figura in basso)



SOTTO IL COPERCHIO



KEVIN JONES

Il sistema operativo del BBC

Uso della memoria ed impiego dei vettori

Il Micro BBC usa diverse aree di memoria, molte delle quali svolgono più funzioni. Un tipico esempio è la pagina di memoria &9 (l'area RAM compresa tra le locazioni &900 e &9FF), che viene usata di volta in volta come buffer di uscita per la porta RS232, buffer di uscita per cassetta, spazio di lavoro per la sintesi vocale e spazio di lavoro "esteso" per il suono. Non si può certo accusare la Acorn di non sfruttare al massimo la memoria dei propri computer!

Un ulteriore esempio è il blocco di memoria compreso tra le locazioni &0E00 e &1900. Su di un Micro BBC che usa un registratore a cassetta per l'archiviazione dei file, questo blocco di memoria viene riservato al BASIC. Viceversa, quando il computer adotta come memoria esterna un'unità a dischetti, questa area viene impiegata dal sistema di archiviazione su disco, riducendo di oltre 2,5 Kbyte la memoria disponibile per i programmi BASIC. Ciò comporta qualche limitazione nei MODE grafici da 0 a 2, nei quali la memoria disponibile è ancor più limitata.

Come ultimo esempio del molteplice uso della memoria, si consideri l'area compresa tra &8000 e &BFFF. Se inseriamo nel computer una ROM DFS, un word processor o una ROM di utilità, esse occupano gli indirizzi di quest'area, normalmente riservata all'interprete BASIC (anch'esso su ROM). Questa organizzazione di ROM è chiamata "a pagine " o "impaginata", poiché ciascuna di esse viene selezionata dal SO a seconda delle necessità e idealmente collocata in questa pagina di memoria. In condizioni normali è selezionata la ROM che contiene l'interprete BASIC, il che consente di inserire ed eseguire i programmi BASIC. Tuttavia, quando incontra un comando per i dischetti, il SO "impagina" la ROM DFS, esegue il comando, e poi riseleziona la ROM del BASIC. Durante l'esecuzione dei comandi DFS, la ROM del BASIC viene completamente ignorata. Tuttavia ci sono altre ROM, come il Sistema di Archiviazione Telesoftware o il word processor *Wordwise*, che occupano questa area e vengono selezionate quando è necessario. Questa tecnica consente la stesura di programmi anche relativamente lunghi, dedicando quasi esclusivamente ad essi l'area RAM.

Area di memoria per l'utente

Dopo che il SO ha occupato l'area di RAM che gli compete, la parte rimanente è a disposizione dell'utente. PAGE, una variabile di sistema del BASIC, contiene l'indirizzo iniziale dell'area di programma BASIC, mentre HIMEM punta all'inizio della memoria video. L'area compresa tra i due indirizzi è totalmente a disposizione dei programmi BASIC, delle variabili e dei programmi in codice macchina. La seguente sequenza calcola, in base al valore di questi indirizzi, la memoria disponibile per i programmi BASIC:

```
PRINT"PAGE,"HIMEM"(HIMEM-PAGE)"BYTE DISPONIBILI"
```

La seguente tabella fornisce una breve descrizione delle caratteristiche relative alle varie aree di memoria utilizzabili sul BBC per programmi in codice macchina:

Aree utilizzabili per i programmi in codice macchina	Sistema di Archiviazione	
	Nastro	Disco
&0D00-&0DFF Pagina &D - solo se non ci sono ROM impaginate nella memoria del computer	(✓)	X
&0B00-&0CFF Pagina &B e &C - se il programma non usa caratteri o tasti di funzione definiti dall'utente	(✓)	(✓)
&0A00-&0AFF Pagina &A - se il programma non usa un'interfaccia seriale	(✓)	(✓)
Spazio salvato nell'area di programma BASIC nel comando DIM	✓	✓

✓ = OK (✓) = OK, con riserva X Non usato

Nella maggior parte dei casi la migliore collocazione di brevi routine in codice macchina da usare assieme ai programmi BASIC, è proprio nell'area di programma BASIC libera. Quasi sempre il codice macchina depositato in memoria necessita di un ulteriore spazio di lavoro e nella maggior parte dei casi a questo scopo vengono adoperate alcune locazioni di pagina zero (anche perché certe istruzioni di indirizzamento indicizzato obbligano all'impiego di tali locazioni). Il sistema operativo dell'Acorn occupa già gran parte della pagina zero, ma alcune locazioni sono proprio riservate alla programmazione in codice macchina. Una descrizione dettagliata (byte per byte) delle locazioni adoperate dal SO non è particolarmente utile, soprattutto perché è sconsigliabile accedere direttamente ad esse, anziché per mezzo delle apposite chiamate al SO. Infatti, se un programma applicativo indi-

rizza direttamente tali locazioni, si corre il rischio che esso non funzioni su un apparecchio che adoperi una versione del SO leggermente diversa.

Accesso al SO

Il sistema migliore, per accedere alle routine residenti nel SO del BBC, è sempre quello di usare le routine OSBYTE ed OSWORD:

• OSBYTE permette di intervenire sul funzionamento di molte routine del SO caricando codici di controllo e parametri nei registri A, X ed Y del 6502. In BASIC, si può accedere alle routine OSBYTE con il comando *FX. Questo è seguito da due o tre numeri: il primo è il codice di controllo o di funzione, caricato poi nel registro A, il secondo è il parametro caricato in X ed il terzo quello caricato in Y. Quest'ultimo non è necessario in tutte le chiamate di OSBYTE. In codice macchina, OSBYTE corrisponde all'indirizzo &FFF4. Le seguenti versioni producono lo stesso risultato:

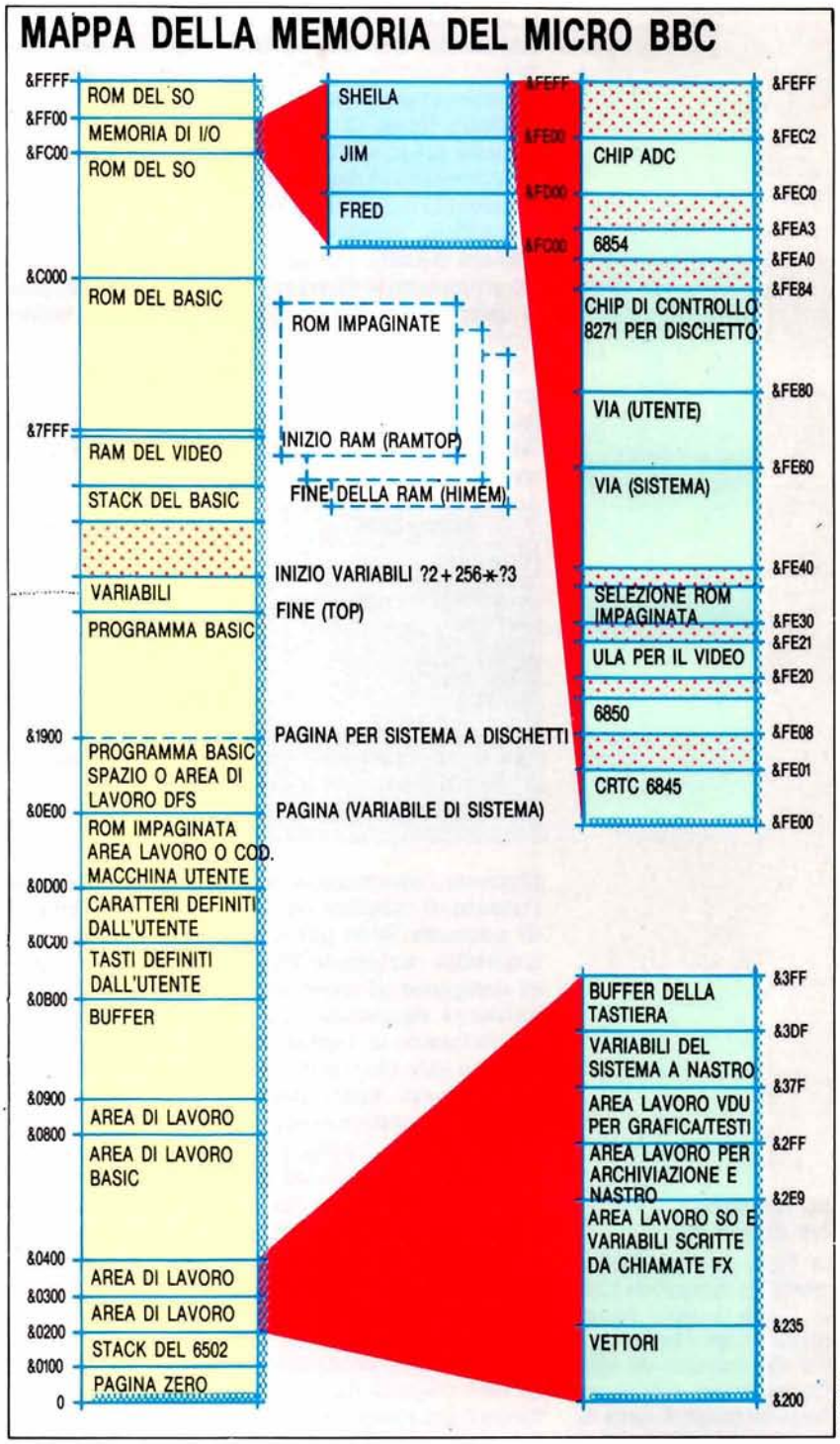
BASIC	Linguaggio Assembly
*FX4,1	LDA #4 LDX #1 JSR &FFF4

Il valore in A definisce esattamente quale chiamata di OSBYTE adoperare: la chiamata mostrata nell'esempio modifica il funzionamento dei tasti per il cursore; il parametro fornito al registro X specifica se i tasti per il cursore devono mantenere la loro normale funzione di editing o se devono semplicemente generare un codice ASCII.

Disgraziatamente non tutte le routine del SO possono essere modificate attraverso OSBYTE. Il suo principale limite è il ridotto numero di parametri che possono essere passati alla routine: a parte il contenuto del registro A (che comunica al sistema operativo quale routine di OSBYTE vogliamo usare), restano soltanto i due parametri nei registri X ed Y. Per fornire un maggior numero di parametri dobbiamo ricorrere alla routine OSWORD.

• OSWORD consente di accedere a varie funzioni quali la generazione di suoni, lettura e scrittura su dichi ecc.

La differenza tra OSBYTE ed OSWORD è che la prima routine interviene sul "come" il SO deve eseguire una particolare operazione, mentre la seconda "fa eseguire" al SO una data operazione. I parametri per la OSWORD« vengono passati tramite un "blocco parametri", puntato dai registri X ed Y del 6502 quando si accede alla routine OSWORD. Le dimensioni e la disposizione di questo blocco di parametri, situato in RAM, dipendono dalla chiamata di OSWORD eseguita. Il parametro A contiene un codice di funzione che comunica al SO quale funzione OSWORD eseguire. Dopo aver predisposto i registri ed il blocco di parametri, si richiama la routine OSWORD che risiede all'indirizzo &FFF1. OSBYTE ed OSWORD rappresentano i principali sistemi di accesso al SO. Data la loro importanza, esamineremo in dettaglio le due routine in un momento successivo.



Con un opportuno impiego del simbolo asterisco, premesso a certi comandi BASIC, è possibile accedere ad altre routine del SO. La presenza dell'asterisco permette al comando di evitare l'interprete BASIC e di raggiungere una routine del SO, chiamata OSCLI, che agisce da "interprete della linea di comando del sistema operativo". Questa decodifica i comandi impartiti e chiama le relative routine del SO. I comandi qui descritti vengono spesso chiamati comandi stella o comandi *. La tabella a pagina 1360 riporta i comandi stella accettati dal BBC: quelli non riconsciuti, scrit-

ti in modo errato o privi del giusto numero di parametri provocano la comparsa del messaggio d'errore Bad Command.
I comandi possono essere passati all'interprete della linea di comando (CLI) direttamente oppure tramite la routine del SO. OSCLI può essere usato in due modi: per comunicare a CLI i comandi mostrati in tabella, usando il codice macchina, oppure per fornire a CLI variabili stringa BASIC. I programmi che seguono usano entrambe le funzioni. Si noti che le variabili intere A%, X% ed Y% caricano i loro valori direttamente nei registri A, X ed Y. X% e Y% (ossia i registri X e Y) puntano all'indirizzo della stringa di caratteri da gestire come un comando *, quindi interpretata ed eseguita dalla routine OSCLI. X contiene il byte basso dell'indirizzo, Y quello alto.

Versione BASIC	Versione BASIC + Assembly
10 DIM C 100 20 OSCLI = &FFF7 30 INPUT "IMMETTI IL COMANDO", A$ 40 $C = A$ 50 X% = C MOD 256 60 Y% = C DIV 256 70 CALL OSCLI 80 GOTO 30	10 DIM C 100 20 OSCLI = &FFF7 30 FOR I% = 0 TO 2 STEP 2 40 P% = &C00 50 [OPT I% 60 .codice LDX #C MOD 256 70 LDY #C DIV 256 80 JSR OSCLI 90 RTS 100];NEXT I% 110 INPUT "IMMETTI IL COMANDO",A$ 120 $C = A$ 130 CALL codice 140 GOTO 110

Durante l'esecuzione, il programma chiede all'utente di inserire un comando * (ricordarsi di premere Return per inviarlo in esecuzione). L'insolita istruzione DIM alla linea 10 comunica al computer di riservare 100 byte di memoria nell'area destinata alle variabili BASIC e di inizializzare la variabile C con l'indirizzo iniziale di tale blocco di memoria. I 100 byte possono essere usati per immagazzinare programmi in codice macchina o, come in questo caso, dati per i programmi in codice macchina. L'istruzione $C=A$ deposita nel blocco di 100 byte, che inizia dal primo byte riservato dal comando DIM, i byte che compongono la stringa di comando A$. Entrambi i programmi inizializzano i registri X ed Y (oppure le variabili X% e Y%) e chiamano la routine OSCLI per eseguire il comando * immesso.
Il precedente programma costituisce la base di una routine da usare nei programmi controllati da menu, durante i quali l'utente può aver bisogno di eseguire operazioni come la catalogazione di dischi o nastri senza uscire dal programma. Il comando immesso viene semplicemente inserito in una variabile stringa e comunicato ad OSCLI per l'esecuzione. L'immissione di comandi inesistenti (*A$, ad esempio) non produce alcun risultato.
Questa tecnica permette, inoltre, di abbinare ad un comando * variabili numeriche, il cui valore però va previamente trasformato in stringhe tramite STR$. CLI, infatti, non è in grado di valutare le variabili espresse per nome, ma si limita a rispondere con il messaggio d'errore Bad Command.
Il SO comunica i comandi * non riconosciuti alle varie ROM impaginate, chiedendo loro se sono in grado di interpretarli. Se la risposta è positiva il comando viene eseguito, altrimenti esso viene considerato errato, ma non prima di un'ultima verifica. Se il computer dispone di un sistema di archiviazione dei file veloce (un'unità a dischetti, per esempio) allora viene prima letta la "directory" del disco per verificare se contiene un file con lo stesso nome del comando (senza l'asterisco). In caso affermativo, il file viene caricato in memoria e trattato come un programma in codice macchina. Ovviamente, se il file non è un programma in codice macchina, sorgono grossi problemi: il computer si blocca ed è necessario l'intervento dell'operatore. Infine, se non esiste alcun file corrispondente al comando, viene visualizzato il messaggio d'errore Bad Command.

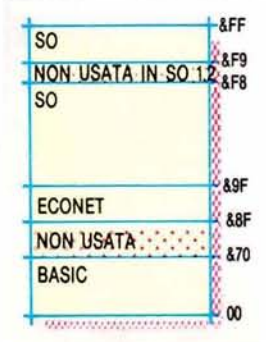
I COMANDI *

Comando	Descrizione e Commento
*HELP	Fornisce la sigla della versione BASIC. Inoltre permette di ottenere informazioni sulle ROM impaginate presenti in memoria; ad esempio: *HELP DFS
*BASIC	Abilita il linguaggio BASIC. Una variazione di questo comando, ad esempio *WORD, seleziona una ROM impaginata (in questo caso View)
*CODE *LINE	Solo nel SO versione 1.2. Permette all'utente di aggiungere nuovi comandi al SO
*KEY	Usato per programmare i tasti di funzione rossi
*MOTOR n	Usato per controllare il relais del motore del registratore: n=0 spegne il relais, n=1 lo accende
*ROM, *TAPE, *DISK, *NET	Usati per inizializzare il relativo sistema di archiviazione dei file. Ad esempio: *TAPE abilita il sistema di archiviazione su nastro (1200 baud), *DISK quello su disco
*FX	Permette al programmatore di modificare i valori delle variabili del SO, quindi di controllare il funzionamento del SO
*RUN, *OPT *LOAD, *. *CAT, *SAVE	Questi sono comandi per l'archiviazione dei file e vengono descritti in seguito
*SPOOL *EXEC	*SPOOL invia l'output per il video allo schermo e ad un file. *EXEC memorizza i dati provenienti da un file come se li ricevesse dalla tastiera
*TV x,y	Interviene sulla disposizione verticale delle immagini video: x=0 non provoca alcun cambiamento; x=1 sposta l'immagine in basso di una linea; x=255 la sposta di una linea in alto; y=0 ed y=1 attivano ed inattivano la scansione. Gli effetti di questo comando si manifestano al primo cambiamento di modo dello schermo e rimangono tali finché non viene battuto un CTRL-BREAK o un altro comando *TV

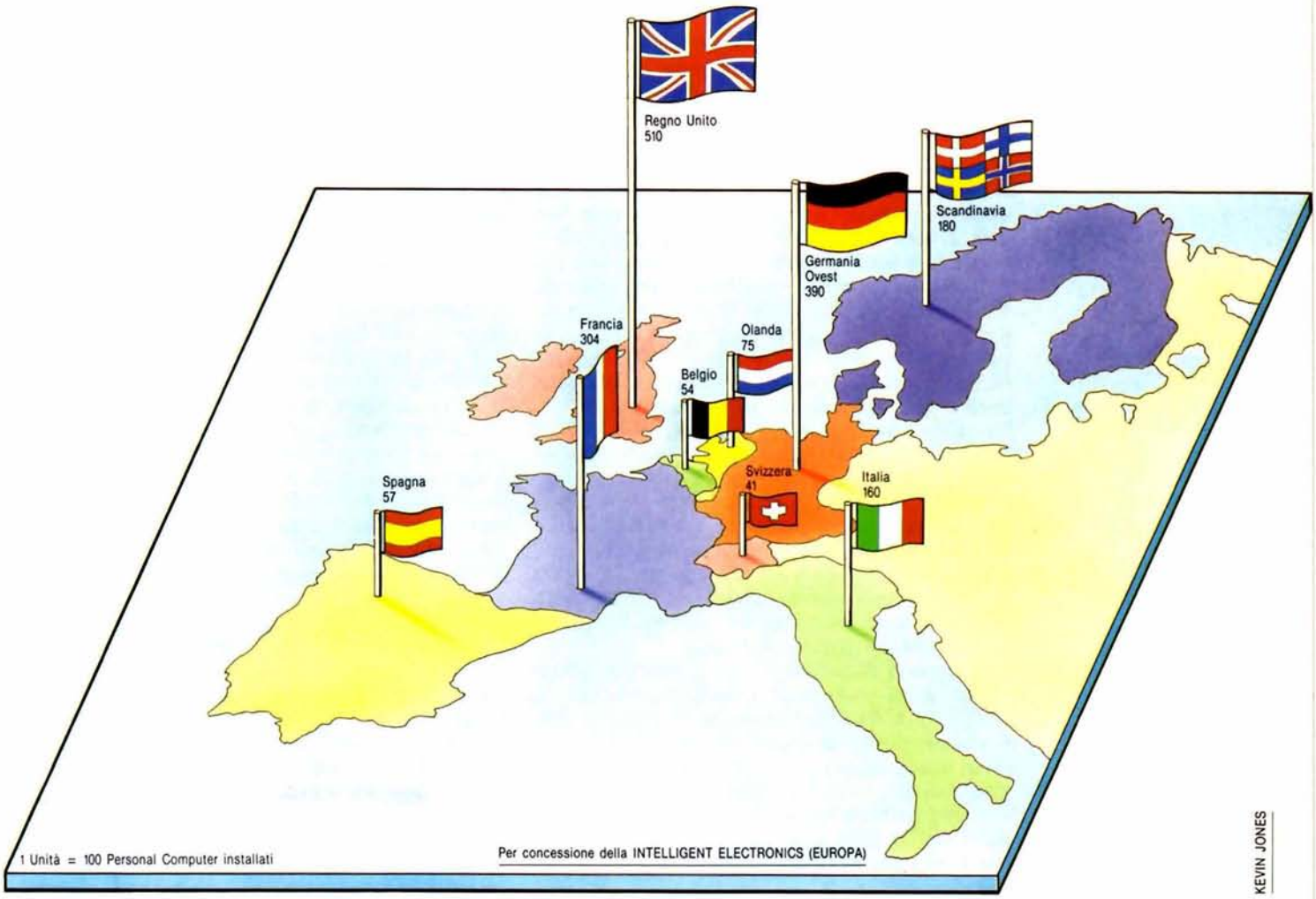
Per ulteriori spiegazioni sull'uso di questi comandi, si consulti il manuale per l'utente del Micro BBC

Uso della Pagina Zero del Micro BBC

La Pagina Zero (da &0000 a &00FF), qui rappresentata come un insieme di registri d'indice ausiliari, integra le limitate capacità d'indirizzamento del 6502. Ciò naturalmente comporta un massiccio impiego di queste locazioni, ma un certo numero di byte è riservato all'utente



Software di taglio europeo



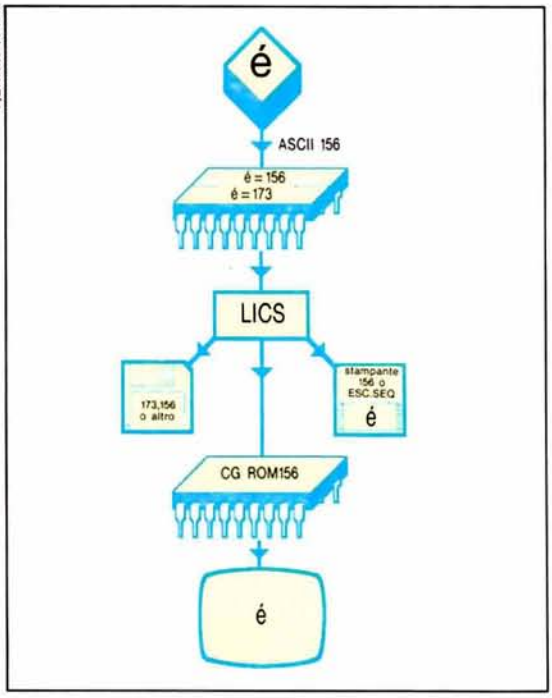
I produttori di software si stanno sempre più orientando verso lingue diverse dall'inglese

Fino a pochi anni fa, quasi tutto il software (e relativa documentazione) era in lingua inglese, considerata la forte presenza statunitense nel settore informatico fin dagli anni '50.
Questa tendenza però, con l'avvento dei microcomputer, ha creato non pochi ostacoli alla diffusione di nuovi apparecchi nei paesi di lingua diversa da quella inglese, ad esempio in tutta l'Europa centrale. Pian piano si è formato un circolo vizioso: infatti, i potenziali acquirenti non inglesi erano poco inclini ad acquistare software difficilmente comprensibile e utilizzabile, mentre le software house, dal canto loro, non se la sentivano di investire nella traduzione e nella conversione di programmi destinati a mercati esteri in cui le vendite erano a livelli molto bassi. Era inevitabile che, in una situazione simile, la Gran Bretagna divenisse il mercato più florido in Europa per i microcomputer.
Negli ultimi tempi, comunque, le cose stanno cambiando. Una delle prime software house ad accorgersi che l'Europa costituisce un vasto mercato potenziale è stata la Lotus Software, una delle più grandi fornitrici di software commerciale per i computer IBM. Il *Lotus 1-2-3* (vedere a pag. 1124) e il *Symphony* sono tra i più venduti package di software integrato. I package che appartengono a questa nuova generazione contengono di solito uno spreadsheet, un database, un word processor più o meno sofisticato e un programma di grafica. I dati possono essere comodamente trasferiti dall'uno all'altro mo-

Mercato personal in Europa
La Gran Bretagna ha tratto notevole profitto dal fatto di avere in comune la lingua con gli Stati Uniti. Trovandosi così avvantaggiata rispetto ad altri paesi europei, il fiorire del mercato dei microcomputer è stato inarrestabile. Tuttavia, le nazioni europee stanno riguadagnando molto terreno e le più recenti stime indicano la Germania Occidentale quale futura nazione leader in Europa, in questo settore

dulo del package (dallo spreadsheet al word processor, ad esempio, per incorporare una tabella statistica in una relazione).
La traduzione di un simile insieme di programmi dall'inglese all'italiano a prima vista può apparire un compito abbastanza semplice: basta che le scritte e i comandi che appaiono sullo schermo siano trasposti nella lingua scelta. Tuttavia, sorgono immediatamente alcuni problemi. In primo luogo, specialmente se la parte testuale del programma è notevole, la sua individuazione e modifica può richiedere molto tempo. In secondo luogo il testo tradotto occupa, nella totalità dei casi, più spazio dell'originale. Ciò significa una crescita, talvolta notevole, del codice finale con tutti i problemi che essa comporta (problemi di spazio, modifiche agli indirizzi, ecc.). Un altro problema, connesso al precedente, è che talvolta l'inglese consente di utilizzare parole o interi spezzoni di frasi per comporre più messaggi di vario genere: una cosa quasi impossibile in italiano.
Un ultimo problema è costituito dalle regole sintattiche della lingua in questione: ad esempio, mentre in inglese è logico far seguire un comando dal nome di un file, in tedesco è vero il contrario. Un altro esempio ancora è il formato delle date: nel sistema americano si indica prima il mese, poi il giorno e infine l'anno e gran parte del software statunitense adotta questa convenzione, che risulta a dir poco irritante per un europeo.
I produttori di software devono inoltre tener conto degli eventuali caratteri speciali in certe lingue (la c francese con la cediglia o la n spagnola con la tilde); anche la collocazione di simili caratteri nell'alfabeto, non essendo costante, complica ulteriormente le cose. Per non parlare dei codici estranei allo standard ASCII, adoperati da molti fabbricanti di hardware per "personalizzare" le proprie tastiere.

IAN McKINNELL



Standard Lotus

Per ridurre i problemi dovuti alla differenza di alcuni caratteri alfabetici nelle varie lingue, la Lotus Software ha realizzato una serie di tabelle di conversione, denominata LICS (da Lotus International Character Set). Ad esempio, digitando una é (come in café) su una tastiera francese, viene emesso il codice 156, che corrisponde al codice LICS 173. Il software del Symphony tiene conto di queste "deviazioni", gestendo automaticamente l'emissione del corretto codice sul video e sulla stampante inserendo, se occorre, anche una sequenza "di escape". Il testo può essere indifferentemente memorizzato in codice ASCII "nazionalizzato", in LICS o nell'ASCII di qualsiasi altra nazione

Il progetto di programmi

Nel tradurre il *Symphony* in alcune lingue europee, la Lotus ha deciso di impostare il proprio software in modo da facilitare al massimo questo tipo di interventi linguistici, perché la traduzione del precedente package, ossia il *Lotus 1-2-3*, aveva provocato problemi a non finire; il *Symphony* invece è stato tradotto in ben 4 lingue europee, compresa quella italiana.
Ovviamente, la soluzione più semplice è quella di sviluppare il programma da tradurre in due moduli ben distinti: uno contiene tutte le routine del programma, l'altro esclusivamente i blocchi di testo. A questa suddivisione è stato già assegnato un nome: "localizzazione".
Organizzare un programma secondo tale criterio risolve due dei problemi maggiori: l'individuazione delle scritte in un programma e i problemi di spazio derivanti dalla diversa lunghezza delle parole tradotte.
In programmi quali il *Symphony* sorgono però altri inconvenienti: nella versione originale i comandi possono essere immessi semplicemente digitando la loro lettera iniziale, che però difficilmente è la stessa nella versione tradotta; oppure bisogna stare attenti ad evitare casi di omonimia (due comandi con la stessa iniziale). Anche il simbolo @, ampiamente adoperato nei comandi del Symphony ha provocato non pochi fastidi, dato che in molte tastiere europee questo segno è sostituito da un altro. Del resto, rinunciare alla semplicità di comandi "a singola pressione" è ormai inconcepibile e quindi, inevitabilmente, vanno studiati ed applicati dei compromessi caso per caso.
La Lotus ha preso talmente a cuore (o dovremmo dire a chip?) il problema della conversione del software, che ha addirittura studiato e messo a punto una serie di tabelle di conversione per i caratteri nazionali (vedere a pagina 1309).
Denominata LICS (da Lotus International Character Set), questa serie di tabelle contiene 250 caratteri: tutte le lettere maggiormente in uso nelle lingue europee.
Il programma *Symphony* ne fa uso per adeguare al codice del programma i vari standard individuali.
Per facilitare la stampa dei caratteri speciali non presenti sulla tastiera, il software prevede un meccanismo tale che è sufficiente la pressione di un apposito tasto (ALT nel PC dell'IBM) e l'immissione sotto forma numerica del codice desiderato.
La traduzione e l'adattamento di un package commerciale implicano investimenti non indifferenti sia di tempo che di denaro. Ci possono volere anche otto/nove mesi per la versione tradotta, con una spesa che può oscillare dalle decine fino alle centinaia di milioni, se il programma è davvero complesso.
Nonostante ciò, i produttori di software non possono ignorare un mercato europeo di 300 milioni di potenziali utenti dei loro prodotti.

Spiriti e fantasmi

Il resto della routine per la galleria e... apparizioni soprannaturali!



Gli ingressi alla galleria, nel gioco, corrispondono a due locazioni speciali. Il giocatore può scegliere di entrare nella galleria oppure di tornare sul sentiero che conduce all'ingresso: nel primo caso, il programma chiama la subroutine alla linea 4655. Questa è stata sviluppata secondo le regole definite da chi ha concepito il gioco: innanzi tutto, il giocatore può entrare nella galleria solo se possiede la lanterna e questa va accesa per illuminare il percorso dal momento che anche durante l'attraversamento della galleria il giocatore deve poter introdurre comandi, la subroutine inizia con una sequenza che accetta un comando e lo suddivide per interpretarlo. Il giocatore può usare comandi normali, quali PRENDI, LASCIA, ELENCA e FINE. È però da notare che, per quanto riguarda il contenuto del puntatore alla locazione P, il giocatore si trova ancora all'ingresso della galleria. In queste condizioni il comando VAI, per scegliere una delle direzioni consentite, rimarrebbe attivo. Occorre così disabilitarne l'impiego, finché ci si trova all'interno della galleria.
Se la subroutine per i comandi normali riconosce il comando VAI, il valore del flag MF viene posto diverso da zero e l'indicatore della posizione P viene anch'esso modificato. Quindi per disabilitare VAI (o simili comandi di movimento) basta ripristinare in P il valore posseduto prima della chiamata alla subroutine dei comandi normali.
Passiamo adesso alla gestione dei comandi speciali per questa fase di gioco. Le istruzioni consentite sono: TORNA, per tornare all'ingresso della galleria e ACCENDI con la variazione USA. Se il giocatore batte un comando diverso, la routine visualizza il messaggio NON CAPISCO e torna a richiedere un nuovo comando. Se il comando immesso è ACCENDI o USA, è necessario eseguire diversi controlli:

1) L'oggetto indicato è valido ?
2) È in possesso del giocatore ?
3) L'oggetto indicato è la lanterna ?

Se la risposta alle tre domande è affermativa, il giocatore può raggiungere l'uscita della galleria perché sono soddisfatte tutte le condizioni necessarie per il suo attraversamento. Poiché questa procedura di verifica degli oggetti è quasi identica a quella adoperata nelle routine PRENDI e LASCIA (vedere a pag. 1326), possiamo utilizzare tali subroutine.

```
4700 REM ** INGRESSO NELLA GALLERIA **
4705 SN$ = "ENTRI NELLA GALLERIA MA TI TROVI NEL BUIO
     PIÙ COMPLETO"
4710 SN$ = SN$ + " E NON VEDI USCITE.": GOSUB 5500
4725 PRINT:INPUT "IL TUO COMANDO";IS$
4730 GOSUB 2500: REM SUDDIVIDE IL COMANDO
4732 :
4735 IF F = 0 THEN 4725: COMANDO NON VALIDO
4740 OP = P:GOSUB3000:REM COMANDI NORMALI
4745 IF MF = 1 THEN SN$ = "È TANTO BUIO CHE RIESCI A
     SCORGERE SOLO":P = OP
4747 IF MF = 1 THEN SN$ = SN$ + "L'INGRESSO DELLA GALLE-
     RIA":GOSUB 5500: MF = 0: GOTO 4725
4750 IF VF = 1 THEN 4725: REM COMANDO ESEGUITO
4755 IF VB$ = "TORNA" AND P = 4 THEN MF = 1: P = 6: RETURN
4760 IF VB$ = "TORNA" AND P = 1 THEN MF = 1: P = 9: RETURN
4762 IF VB$ < > "USA" AND VB$ < > "ACCENDI" THEN SN$ = "
     NON CAPISCO"
4765 IF VB$ < > "USA" AND VB$ < > "ACCENDI" THEN GOSUB
     5500: GOTO 4725
4777 :
4780 REM ** CERCA LA LANTERNA **
4790 GOSUB 5300: REM OGGETTO VALIDO ?
4795 OV = F: GOSUB 5450: REM IN POSSESSO DEL GIOCATO-
     RE?
4797 IF F = 0 THEN SN$ = W$ + " NON ESISTE": GOSUB 5500:
     GOTO 4725
4800 IF HF = O THEN SN$ = "NON POSSIEDI " + IV$(F,1): GO-
     SUB 5500: GOTO 4725
4810 REM ** L'OGGETTO È LA LANTERNA ? **
4815 IF F < > 2 THEN SN$ = "QUI "+IV$(F,1)+" È INUTI-
     LE":GOSUB 5500:GOTO 4725
4835 REM ** SUCCESSO! **
4840 SN$ = "USI LA LANTERNA PER ILLUMINARE LA GALLE-
     RIA"
4845 SN$ = SN$ + " E FINALMENTE RAGGIUNGI L'USCITA.":
     GOSUB 5500
4850 IF P = 1 THEN MF = 1: P = 4: RETURN
4855 IF P = 4 THEN MF = 1: P = 1: RETURN
```

Eventi soprannaturali

Durante il gioco si va incontro a pericoli ed eventi soprannaturali che si verificano casualmente. Finora non abbiamo mai parlato degli spettri, né questi compaiono nella mappa dello scenario di gioco (pag. 1246). Gli spettri appaiono improvvisamente quando il giocatore attraversa la foresta: egli può difendersi solo seguendo un particolare comportamento. Prima di esaminare in dettaglio la routine spettri, vediamo come inserire nel programma principale la routine per la loro comparsa casuale. Il ciclo principale del programma richiama una subroutine, posta alla linea 2700, che verifica se una nuova locazione è speciale oppure no; questo è il momento giusto per inserire le linee che generano gli spettri in modo casuale:

```
2707 REM ** CREAZIONE CASUALE DI SPETTRI **
2710 IF P>4 AND RND (1)<0.1 THEN GOSUB 4290: RETURN
```

Innanzi tutto la linea 2710 verifica che la locazione corrente non sia speciale, in quanto l'apparizione di spettri in locazioni speciali

complicherebbe il programma. Se la locazione è normale, c'è una probabilità su dieci che il programma generi uno spettro. La funzione RND produce numeri "pseudo-casuali" (così chiamati perché la loro sequenza è in parte prevedibile). Per rendere la sequenza imprevedibile, sul Micro BBC e sul Commodore 64 si fornisce un operando negativo alla RND, mentre sullo Spectrum si usa l'istruzione RANDOMISE (vedere "Basic & Basic" a pag. 1365).

```
207 R=RND(-1)
```

Se la routine spettri viene chiamata, si entra in uno scenario speciale in cui si affronta lo spettro. La routine visualizza un messaggio, riceve un'istruzione e separa il verbo dalla frase. I comandi normali vengono gestiti dalla solita routine ma è necessario disabilitare il comando VAI (il giocatore viene informato che, paralizzato dal terrore, non può moversi).

Rete di sicurezza

Questa fase di gioco prevede una serie di nuovi comandi e, come sempre, il risultato finale dipende dall'impegno profuso nella progettazione della routine. Anche in questo caso il programma risponde col messaggio NON CAPISCO all'immissione di istruzioni non contemplate; tuttavia la routine spettri accetta anche altre risposte del giocatore, che non sono però di alcun aiuto. Quando appare uno spettro, l'istinto del giocatore è quello di impartire un comando del tipo LOTTA o UCCIDI (ma è possibile uccidere i fantasmi?). La routine spettri risponde a questi comandi chiamando una subroutine speciale che, invece della scritta NON CAPISCO, visualizza un messaggio che invita il giocatore ad astenersi dall'impartire inutili comandi.

```
4290 REM ** S/R SPETTRO CASUALE **
4295 SF=1: GC=0
4300 SN$="BRIVIDI CORRONO LUNGO LA TUA SCHIENA"
4305 SN$=SN$+" IMPROVVISAMENTE UNA BIANCA APPARIZIONE"
4310 SN$=SN$+" SCIVOLA FUORI DALLA FORESTA E"
4315 SN$=SN$+" SI DIRIGE VERSO DI TE": GOSUB 5500
4320 :
4325 0 SN$="LO SPETTRO SI AVVICINA SEMPRE PIÙ": GOSUB 5500
4330 GC=GC+1: IF GC>4 THEN GOSUB 4455: REM MORTE!
4335 PRINT:INPUT "IL TUO COMANDO";IS$
4340 GOSUB 2500: REM SUDDIVIDE IL COMANDO
4345 IF F=0 THEN 4325: REM COMANDO SUCCESSIVO
4350 OP=P: GOSUB 3000: REM ANALIZZA IL COMANDO
4355 IF MF=1 AND VB$="VAI" THEN GOSUB 4400: GOTO 4325
4357 IF MF=1 AND VB$="GUARDA" THEN GOSUB 2000: GOSUB 2300:GOTO 4325
4360 IF VF=1 THEN 4325: REM COMANDO SUCCESSIVO
4365 REM ** NUOVE PAROLE DI COMANDO **
4370 IF VB$="UCCIDI" OR VB$="LOTTA" THEN GOSUB 4425:GOTO 4325
4375 :
4385 IF VB$="CANTA" THEN GOSUB4500: RETURN
4390 SN$="NON CAPISCO": GOSUB 5500: GOTO 4325
4395 :
4400 REM ** TENTATIVO DI MOVIMENTO **
4405 SN$="SEI PARALIZZATO DAL TERRORE E NON RIESCI"
4410 SN$=SN$+" A MUOVERTI ...": MF=0: GOSUB 5500: P=OP
4415 RETURN
4420 :
4425 REM ** LOTTA O UCCISIONE **
4430 SN$="LO SPETTRO È UN ESSERE SOPRANNATURALE"
4435 SN$=SN$+" E SI BEFFA DEI TUOI DEBOLI TENTATIVI"
4440 SN$=SN$+" DI COLPIRLO": GOSUB 5500
4445 RETURN
4450 :
4455 REM ** MORTE **
4460 SN$="IL DOLORE AL PETTO DIVENTA INSOPPORTABILE"
4465 SN$=SN$+" E CADI PESANTEMENTE AL SUOLO.": GOSUB 5500
4470 SN$="IL TUO SPIRITO SI STACCA DAL CORPO INERTE"
4475 SN$=SN$+" E FLUTTUA VIA, SCOMPARENDO NELLA NEBBIA,"
4480 SN$=SN$+" PER UNIRSI ALLE ALTRE ANIME TORMENTATE"
4485 SN$=SN$+" DELLA FORESTA INCANTATA.": GOSUB 5500
4490 END
```

Attenti al trabocchetto

Se il giocatore inserisce un comando normale o uno dei comandi inutili, la routine lo esegue, se possibile, e passa ad accettare nuove istruzioni. La routine contiene però un piccolo trabocchetto, in quanto viene conteggiato il numero di comandi impartiti dal giocatore durante l'incontro con lo spettro: se è maggiore di quattro, il giocatore muore; l'unica possibilità di sfuggire alla morte è "cantare" una canzone. Infatti, col comando CANTA, può scegliere fra tre canzoni di cui però solo una a caso rabbonisce lo spettro.
Se il giocatore sceglie la canzone sbagliata, viene ucciso ugualmente ed il suo spirito va ad unirsi alla schiera di anime tormentate della Foresta Incantata.

```
4500 ** CANTO DI SALVEZZA **
4505 SN$="CONOSCO SOLO TRE TEMI DI FILM:": GOSUB 5500
4510 SN$="1) GHOSTBUSTERS": GOSUB 5500
4515 SN$="2) IL FANTASMA DELL'OPERA": GOSUB 5500
4520 SN$="3) FANTASMI A ROMA": GOSUB 5500
4525 PRINT:INPUT "QUALE SCEGLI (1/2/3)"; C$
4530 IF VAL(C$)>3 OR VAL(C$)<1 THEN PRINT:PRINT "RIPETI!": GOTO 4525
4535 CR=INT(RND(1)-3)+1
4537 IF CR<>VAL(C$) THEN GOSUB 4542: REM CANZONE SBAGLIATA!
4540 GOSUB 4565: REM CANZONE GIUSTA!
4542 REM ** S/R CANZONE SBAGLIATA **
4545 SN$="LO SPETTRO NUTRE UNA PARTICOLARE AVVERSIONE"
4550 SN$=SN$+" PER QUESTA CANZONE E TI COLPISCE.": GOSUB 5500
4555 GOSUB 4455: REM MORTE
4560 :
4565 REM ** CANZONE GIUSTA **
4570 SN$="LO SPETTRO È PLACATO DAL TUO CANTO E"
4575 SN$=SN$+" SI DISSOLVE NELLA BRUMA DELLA FORESTA": GOSUB 5500
4580 RETURN
```

LIZ DIXON

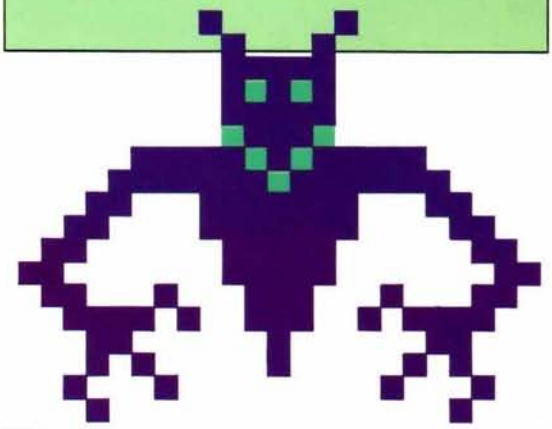
LISTATO DIGITAYA

```
2690 IF P=37 THEN 2780: REM TABELLA VETTORI
2700 IF P>7 THEN 2750: REM BUG CASUALE

2740 REM ** BUG CASUALE **
2750 RA=RND(TI)
2760 IF RA<0.05 THEN GOSUB 5420: REM CREA UN BUG
2770 RETURN
2780 REM ** TABELLA DEI VETTORI **
2790 SF=1
2800 SN$="TI SEI  SPOSTATO RAPIDAMENTE IN UN
     NUOVO AMBIENTE": GOSUB 5880
2810 FOR J=1 TO 1000: NEXT: REM PAUSA
2820 P=INT(RND(TI)*40+7)
2830 MF=1: RETURN

4550 REM **** UNITA' LOGICO-ARITMETICA (ALU) ****
4560 SF=1
4570 RN=INT(RND(TI)*3+1)
4580 IF RN=1 THEN CD$="AND"
4590 IF RN=2 THEN CD$="OR"
4600 IF RN=3 THEN CD$="NOT"
4610 SN$="SULLA PARETE CI SONO TRE PULSANTI
     CONTRASSEGNATI"
4620 SN$=SN$+" DALLE  SCRITTE 'AND', 'OR' E
     'NOT'. PREMENDO IL PULSANTE GIUSTO"
4630 SN$=SN$+" PUOI ENTRARE NELL'ACCUMULATORE"
4640 GOSUB 5880: REM VISUALIZZA MESSAGGIO
4650 :
4660 REM ** IMMISSIONE COMANDI **
4670 PRINT:INPUT "IL TUO COMANDO";IS$
4680 GOSUB 1700: GOSUB 1900: REM ANALISI COMANDO
4690 IF MF=1 THEN RETURN: REM USCITA
4700 IF VF=1 THEN 4670: REM COMANDO SUCCESSIVO
4710 IF VB$="USA" OR VB$="PREMI" THEN 4740
4720 PRINT "NON CAPISCO": GOTO 4670
4730 :
4740 REM ** COMANDO VALIDO **
4750 IF VB$="PREMI" THEN 4930
4760 REM ** COMANDO 'USA'? **
4770 GOSUB 5730: REM OGGETTO VALIDO?
4780 IF F=0 THEN PRINT NN$;" ESISTE":GOTO 4670:
     REM COMANDO SUCCESSIVO
4790 :
4800 REM ** L'OGGETTO E' UN MANUALE DEL
     CODICE? **
4810 IF F=7 THEN 4850: REM TUTTO OK
4820 SN$="QUI "+IV$(F,1)+" E' DEL TUTTO INUTILE":
     GOSUB 5880
4830 GOTO 4670: REM COMANDO SUCCESSIVO
4840 :
4850 OV=7: GOSUB 5830: REM IL MANUALE E'
     TRASPORTATO?
4860 IF HF=1 THEN 4900: REM TUTTO OK
4870 SN$="NON POSSIEDI "+IV$(7,1)
4880 GOSUB 5880: GOTO 4670: REM COMANDO
     SUCCESSIVO
4890 :
4900 SN$="APRI IL MANUALE E TROVA LA
     PAROLA '"+CD$+"'"
4910 GOSUB 5880: GOTO 4670: REM COMANDO
     SUCCESSIVO
4920 :
4930 REM ** IL COMANDO E' 'PREMI' **
4940 IF NN$="AND" OR NN$="OR" OR NN$="NOT"
     THEN 4970
4950 SN$=NN$+" NON ESISTE": GOSUB 5880: GOTO
     4670: REM COMANDO SUCCESSIVO
4960 :
4970 REM ** PULSANTE GIUSTO O SBAGLIATO? **
4980 IF NN$=CD$ THEN GOSUB 5100: RETURN
4990 GOSUB 5010: RETURN
5000 :
5010 REM ** S/R PULSANTE SBAGLIATO **
5020 SN$="SBAGLIATO! SI APRE  UNA BOTOLA E
     PRECIPITI"
5030 SN$=SN$+" TORNANDO NELLA MEMORIA
     PRINCIPALE"
5040 GOSUB 5880: REM VISUALIZZA MESSAGGIO
5050 IF RN=1 THEN P=39
5060 IF RN=2 THEN P=35
5070 IF RN=3 THEN P=29
5080 MF=1: RETURN
5090 :
5100 REM ** S/R PULSANTE GIUSTO **
5110 SN$="SI APRE LA PORTA DELL'ACCUMULATORE"
5120 SN$=SN$+" E TU L'ATTRAVERSI": GOSUB 5880
5130 P=30: MF=1: RETURN

5420 REM ** BUG CASUALE **
5430 SF=1
5440 SN$="DA DIETRO UN CHIP APPARE UN GRANDE E
     MINACCIOSO BUG"
5450 SN$=SN$+" CHE TI AGGREDISCE": GOSUB 5880
5460 :
5470 REM ** IMMISSIONE COMANDO **
5480 PRINT:INPUT "IL TUO COMANDO";IS$
5490 GOSUB 1700: GOSUB 1900: REM ANALISI
     COMANDO 5500 IF MF=1 THEN MF=0: PRINT "NON
     RIESCI A MUOVERTI ...": GOTO 5480
5510 IF VF=1 THEN 5480: REM COMANDO SUCCESSIVO
5520 IF VB$="UCCIDI" OR VB$="LOTTA"THEN 5550
5530 PRINT "NON CAPISCO": GOTO 5480
5540 :
5550 REM **IL COMANDO E' UCCIDERE O LOTTARE ? **
5560 RA=RND(TI)
5570 IF RA<0.5 THEN GOSUB 5600
5580 GOSUB 5670: RETURN
5590 :
5600 REM **** S/R VIENI UCCISO ****
5610 SN$="INIZIA LA LOTTA. IL BUG TI LANCIA CONTRO
     ERRORI 'PARASSITARI'"
5620 SN$=SN$+" CHE DISTRUGGONO IL TUO CERVELLO."
5630 SN$=SN$+" IL DOLORE  E' INSOPPORTABILE ED
     INFINE LA TUA TESTA ESPLODE."
5640 GOSUB 5880
5650 END
5660 :
5670 REM **** S/R UCCIDI IL BUG ****
5680 SN$="TI DIFENDI  DISPERATAMENTE ...."
5690 SN$=SN$+" INFINE RIESCI AD ABBATTERE IL BUG
     E SOPRAVVIVI.": GOSUB 5880
5700 RETURN
```



Basic & Basic

Spectrum:

In entrambi i programmi, sostituire SN$ con S$, IS$ con T$, IV$(,) con V$(,), VB$ con B$, CD$ con C$ e NN$ con R$. Sostituire le seguenti linee nel listato "Foresta Incantata":

```
207 RAND
4815 IF F<>2 THEN LET S$="QUI": LET A$=V$(F,1):
     GOSUB 7000
4816 IF F<>2 THEN LET S$=S$+"NON FUNZIONA": GO-
     SUB 5500: GOTO 4725
```

Sostituire queste linee nel listato "Digitaya":

```
2750 LET RN=RND(1)
2820 LET P=INT(RND(1)*40+7)
4570 LET RN=INT(RND(1)*3+1)
4820 LET SN$="QUI": LET A$=V$(F,1):GOSUB 8500
4825 LET SN$=SN$+"È INUTILE":GOSUB 5880
5560 LET RA=RND(1)
```

Micro BBC:

Sostituire queste linee nel listato "Foresta Incantata":

```
207 RND(-TIME)
4535 CR=RND(3)
```

Sostituire le seguenti linee nel listato "Digitatya":

```
2750 RN=RND(1)
2820 P=RND(40)+7
4570 RN=RND(3)
5560 RA=RND(1)
```

Disegnare cicloidi

Il piatto forte del LOGO è la grafica: vediamo come costruire immagini geometriche

In precedenza è stato fornito un semplice metodo per tracciare cerchi in LOGO:

```
TO CERCHIO
   REPEAT 360 [FORWARD 1 RIGHT 1]
END
```

Con questa procedura, l'approssimazione della figura è abbastanza buona, ma il tracciamento è penosamente lento, anche non visualizzando la tartaruga.
Tuttavia, su certi apparecchi, la distorsione è tale che occorre modificare la procedura per tentativi, fino a visualizzare un cerchio e non un'ellisse.
In realtà, CERCHIO non traccia una vera circonferenza, bensì un poligono con 360 lati. Nella maggior parte delle applicazioni più usuali, comunque, si può ridurre il numero di lati (e quindi il tempo necessario) senza rovinare troppo l'estetica: a volte bastano 60 o addirittura 30 lati. I "cerchi" considerati nel resto della trattazione hanno 60 o 120 lati, ma questo valore può essere adattato ai propri gusti.
Dopo questa premessa, passiamo a descrivere i "cicloidi". Queste figure geometriche vengono prodotte facendo idealmente ruotare un cerchio lungo un percorso rettilineo. Supponendo che su un punto del cerchio in movimento sia fissata una "penna", essa traccia un percorso detto appunto cicloide. Vediamo come realizzare un programma in LOGO per disegnare cicloidi.
Come prima approssimazione, supponiamo di fissare ogni immagine dopo una rotazione di 6° della penna rispetto al centro del cerchio. Lo spostamento lineare del centro corrisponde a (2 x pi x raggio : 60) unità lungo l'asse x. La coordinata y del centro, invece, rimane costante.
La procedura deve espletare alcuni semplici compiti:

1. muovere il centro del cerchio
2. collocarvi la tartaruga
3. orientare opportunamente la tartaruga
4. muoverla in avanti per la lunghezza del raggio

A questo punto la tartaruga "accende" un punto sullo schermo e l'intero processo viene ripetuto per la posizione successiva.
La prima procedura richiamata da CICLOIDE è PREPARASC, che inizializza una serie di parametri di visualizzazione. In particolare fissa la costante di aggiustamento dell'immagine (che dipende dal tipo di computer) e seleziona il modo NOWRAP, per fermare il programma quando la curva esce dallo schermo.

```
TO CICLOIDE
   PREPARASC
   MAKE "ANGOLO 6
   MAKE "PI 3.14
   MAKE "RAGGIO 15
   MAKE "CIRCONFERENZA 2 * :PI * :RAGGIO
   MAKE "PASSO :CIRCONFERENZA / (360 / :ANGOLO)
   MAKE "XCENTRO (-150)
   CIC 0
END

TO PREPARASC
   NOWRAP
   DRAW
   PENUP
   HT
END

TO CIC :ANG
   MUOVICENTRO
   SETXY :XCENTRO 0
   SETH :ANG
   FORWARD :RAGGIO
   PUNTO
   CIC :ANG + :ANGOLO
END

TO MUOVICENTRO
   MAKE "XCENTRO :XCENTRO + :PASSO
END

TO PUNTO
   PD
   FORWARD 1
   BACK 1
   PU
END
```

Se la tartaruga non viene collocata sul perimetro del cerchio, ma in un punto interno ad esso, la figura risultante è detta cicloide "accorciato". Viceversa, ponendola ad una distanza superiore al raggio, la figura che si ottiene è detta cicloide "allungato".
Per osservare tali effetti possiamo modificare CICLOIDE in modo da accettare un parametro che indichi la distanza del punto tracciante dal centro.
Valori positivi produrranno cicloidi accorciati, mentre valori negativi daranno cicloidi allungati.

```
TO CICLOIDE :SCARTO
   PREPARASC
   MAKE "ANGOLO 6
```

```
   MAKE "PI 3.14
   MAKE "RAGGIO 15
   MAKE "CIRCONFERENZA 2 * :PI * :RAGGIO
   MAKE "PASSO :CIRCONFERENZA / (360 / :ANGOLO)
   MAKE "XCENTRO (-150)
   MAKE "DISTANZA :RAGGIO - :SCARTO
   CIC 0
END

TO CIC :ANG
   MUOVICENTRO
   SETXY :XCENTRO 0
   SETH :ANG
   FORWARD :DISTANZA
   PUNTO
   CIC :ANG + :ANGOLO
END
```

Congiungere i punti

Adoperare una serie di punti per visualizzare l'andamento della curva è un sistema comodo, ma di scarso effetto: vediamo come unire i vari punti ed ottenere così una linea continua.
Questa procedura congiunge due punti della figura:

```
TO UNISCI :A :B
   POSIZIONA :A
   PD
   POSIZIONA :B
   PU
END

TO POSIZIONA :POS
   SETXY FIRST :POS LAST :POS
END
```

Nel richiamare questa procedura occorre passare le coordinate dei due punti da congiungere.
Un esempio di chiamata è UNISCI [12 34] [67 89]. Ciò significa, ovviamente, che occorre conservare le coordinate del punto precedente, per congiungerlo a quello corrente. La versione finale della procedura principale è quindi:

```
TO CICLOIDE :SCARTO
   PREPARASC
   MAKE "ANGOLO 6
   MAKE "PI 3.14
   MAKE "RAGGIO 15
   MAKE "CIRCONFERENZA 2 * :PI * :RAGGIO
   MAKE "PASSO :CIRCONFERENZA / (360 / :ANGOLO)
   MAKE "XCENTRO (-150)
   MAKE "DISTANZA :RAGGIO - :SCARTO
   MAKE "PRECED LIST :XCENTRO :DISTANZA
   CIC 0
END

TO CIC :ANG
   MUOVICENTRO
   SETXY :XCENTRO 0
   SETH :ANG
   FORWARD :DISTANZA
   MAKE "CORRENTE POSIZ
   JOIN:PRECED:CORRENTE
   MAKE "PRECED :CORRENTE
   CIC :ANG + :ANGOLO
END

TO POSIZ
   OUTPUT LIST XCOR YCOR
END
```

A questo punto, i più volenterosi possono cimentarsi in qualche esperimento. Ad esempio, i testi di matematica affermano che la lunghezza di un arco di cicloide è pari al perimetro del quadrato che circoscrive il cerchio "generatore". Si provi a modificare la procedura per controllare se ciò è vero.
Se previsti dalla versione LOGO adoperata, gli sprite possono facilitare eventuali esperimenti. Sostituendo il punto tracciante con uno sprite, infatti, si può conoscere la posizione di quest'ultimo ricorrendo a TELL e poi a XCOR e YCOR.
Una variazione molto interessante è quella di far rotolare il cerchio, anziché su di una linea retta, sul perimetro di un altro cerchio più grande.
La figura che si ottiene viene chiamata "ipercicloide".
Gran parte della procedura rimane identica a quella già vista: dopo aver posizionato il centro del cerchio generante, occorre spostare la penna su di un punto della sua circonferenza.
Adesso, tuttavia, è necessario tener conto di due spostamenti angolari: uno (ANGCEN) serve per stabilire lo spostamento del centro del cerchio generante, il secondo (ANGSEC) indica l'orientamento della penna rispetto al centro del medesimo cerchio. Mentre il centro del cerchio generante ruota in un senso, la penna ruota in senso inverso. La misura dei due spostamenti angolari è legata dalla seguente formula:

```
ANGSEC = ANGCEN * (RAGGIO1 / RAGGIO2 -1)
```

dove RAGGIO1 è il raggio del cerchio fisso e RAGGIO2 è quello del cerchio generante. La procedura IPERCICLOIDE, che utilizza alcune procedure di CICLOIDE, prevede come parametro in ingresso RAGGIO2, in modo da poter tracciare ipercicloidi di ampiezze diverse.

```
TO IPERCICLOIDE :RAGGIO2
   PREPARASC
   MAKE "PI 3.14
   MAKE "RAGGIO1 60
   MAKE "DIFFERENZA :RAGGIO1 - :RAGGIO2
   MAKE "ANGCEN 6
   MAKE "CIRCONFERENZA 2 * :PI - :DIFFERENZA
   MAKE "PASSO :CIRCONFERENZA / (360 / :ANGCEN)
   MAKE "ANGSEC :ANGCEN * (:RAGGIO1 / RAGGIO2 -1)
   MAKE "CENTRO LIST 0 :DIFFERENZA
   MAKE "TESTA 0
   MAKE "XCENTRO 0
   MAKE "PRECED LIST :XCENTRO :RAGGIO1
   IPCIC 0
END

TO IPCIC :ANG
```

```
    MUOVICENTRO2
    POSIZIONA POSIZ
    SETH :ANG
    FORWARD :RAGGIO2
    MAKE "CORRENTE POSIZ
    UNISCI :PRECED :CORRENTE
    MAKE "PRECED :CORRENTE
    IPCIC :ANG – :ANGSEC
END

TO MUOVICENTRO2
    SETXY 0 0
    SETH :TESTA
    FORWARD :DIFFERENZA
    MAKE "CENTRO POSIZ
    MAKE "TESTA :TESTA + :ANGCEN
END
```

Un caso molto interessante si presenta quando il raggio del cerchio generante è esattamente la metà di quello del cerchio fisso: l'ipercicloide diventa così una linea retta! In altre parole, un movimento circolare combinato viene trasformato in un movimento lineare.
Altri casi interessanti si possono sperimentare variando la distanza della penna dal centro del cerchio generante.
Infine, vediamo come produrre altre immagini geometriche molto decorative. Si disegnino due cerchi concentrici, suddividendoli in un certo numero di sezioni o fette, supponiamo 120. Si numeri ciascun punto così individuato sulle due circonferenze e si uniscano con la penna i vari punti in base ad un preciso metodo (ad esempio: congiungere x con 2x e via dicendo). I risultati sono spesso sorprendenti.

Ovviamente, non conviene fare tutto ciò manualmente, dato che abbiamo un servizievole LOGO a disposizione:

```
TO PREPARA
    MAKE "RAGGIOA 80
    MAKE "RAGGIOB 60
    DRAW
    HT
    PENUP
    TRACCIA 0 0
END

TO TRACCIA :A :B
    IF :A = 120 THEN STOP
    UNISCI PTA :A PTB :B
    MAKE "A :A + 1
    MAKE "B 2 * :A
    TRACCIA :A :B
END

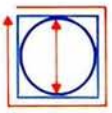
TO PTA :NUM
    SETXY 0 0
    SETH :NUM * 3
    FORWARD :RAGGIOA
    OUTPUT POS
END

TO PTB :NUM
    SETXY 0 0
    SETH :NUM * 3
    FORWARD :RAGGIOB
    OUTPUT POS
END
```

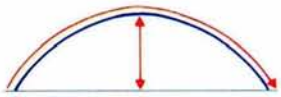
Per produrre altre immagini, si provi a modificare il criterio di congiunzione, ad esempio: x con 3x, oppure x con 4x e così via.

Rotolando e disegnando

Un cicloide è una figura geometrica tracciata dal movimento di un punto della circonferenza di un cerchio in rotazione lungo una linea retta. Variando la distanza del punto tracciante dal centro del cerchio si ottengono due variazioni della curva

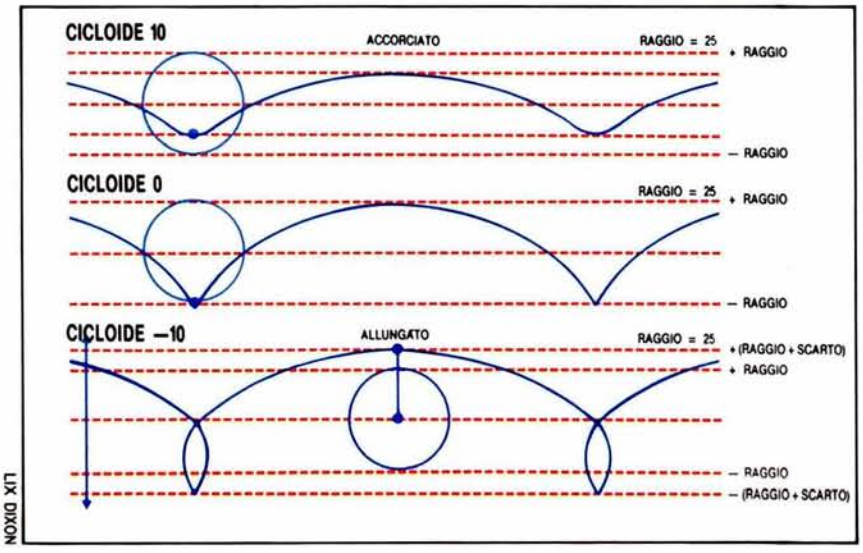


Quadrato che circoscrive il cerchio



Arco di cicloide





LIX DIXON

Logo & Logo

Per tutte le versioni LCSI:

La sintassi delle IF è diversa, ad esempio:IF :A = 120 [STOP]

Per POSIZIONA e POSIZ esistono due primitive, rispettivamente: SETPOS e POS.

Al posto di SETXY va usata SETPOS (ma il parametro in ingresso deve essere una lista).

Al posto di DRAW si usi CS.

Al posto di NOWRAP si usi FENCE (Nel LOGO dell'Atari FENCE non esiste, per cui si usi WINDOW e BREAK per interrompere la procedura).

Per definire la variabile .ASPECT si usi:

.SETSCR sugli Atari

SETSCRUNCH sugli Apple

SETSCRUNCH, seguito da una lista, sugli Spectrum

Il Link 480Z

Anatomia di un microcomputer a diffusione europea: tecnologie a confronto

l Link 480Z è disponibile sia nella versione a singola postazione che in quella "a rete". Qui viene esaminato l'apparecchio per singolo utente. Esteriormente il 480Z è notevolmente diverso dal suo predecessore: il 380Z era contenuto in un "cassone" metallico nero, assieme alle unità a dischetti, mentre il 480Z ha un elegante involucro plastico con tastiera incorporata. I dischetti sono opzionali e contenuti in un box a parte. Il 480Z non ha dimensioni molto contenute, ma la sua linea è veramente piacevole.
La tastiera, di tipo QWERTY, possiede ottimi tasti, che danno la certezza dell'avvenuta pressione (ideale per il word processing). I tasti di controllo, compreso un tasto Line feed e uno di Repeat, si trovano ai lati della tastiera. All'estremità destra si trova una tastierina per il controllo del cursore, i cui tasti, programmabili, assumono un diverso significato a seconda del programma applicativo in uso.
Una serie di interfacce, situate sul retro dell'apparecchio, consente il collegamento con un gran numero di periferiche. A sinistra c'è l'uscita RF, per collegare il computer ad un comune televisore, mentre a destra è situato il pulsante di Reset.
Le prese per monitor sono due: una DIN a 5 vie per i monitor della gamma Microvitec e una DIN a 8 vie per monitor di altro genere (TTL e RGB). Tra i connettori per monitor e quello per il registratore si trova l'interfaccia per gli accessori: una porta seriale di ingresso/uscita.
A destra della presa per il registratore c'è la porta parallela di ingresso/uscita, utile, ad esempio, per il collegamento con vari tipi di stampanti.
Questa porta è compatibile con lo standard Centronics, nel senso che i segnali per apparecchiature Centronics ci sono tutti, ma disposti con un criterio diverso. Il problema si può aggirare intervenendo sul cavo di collegamento connesso alla periferica.
Il Link 480Z possiede anche due porte seriali del tipo RS232 per il collegamento con periferiche e, in particolare, con le unità a dischetti. Accanto alle porte seriali ci sono 10 interruttori DIP. Il primo di essi, marcato R, consente la disabilitazione del pulsante di Reset. Il secondo disabilita invece l'uso dell'altoparlante incorporato, situato sotto la base dell'apparecchio.
I restanti otto interruttori servono per selezionare parametri di comunicazione, se è previsto un collegamento in rete: la loro posizione acceso/spento forma una configurazione binaria a 8 bit, corrispondente al numero di identificazione del computer nella rete.
Poiché le combinazioni sono 256, la rete può essere composta da un pari numero di apparecchi.
L'apposito cavo viene collegato ad una presa situata anch'essa nella parte posteriore, dove si trovano anche un ventilatore di raffreddamento, l'interruttore di accensione, il cavo di alimentazione e un fusibile.



CHRIS STEVENS

Un raffinato design

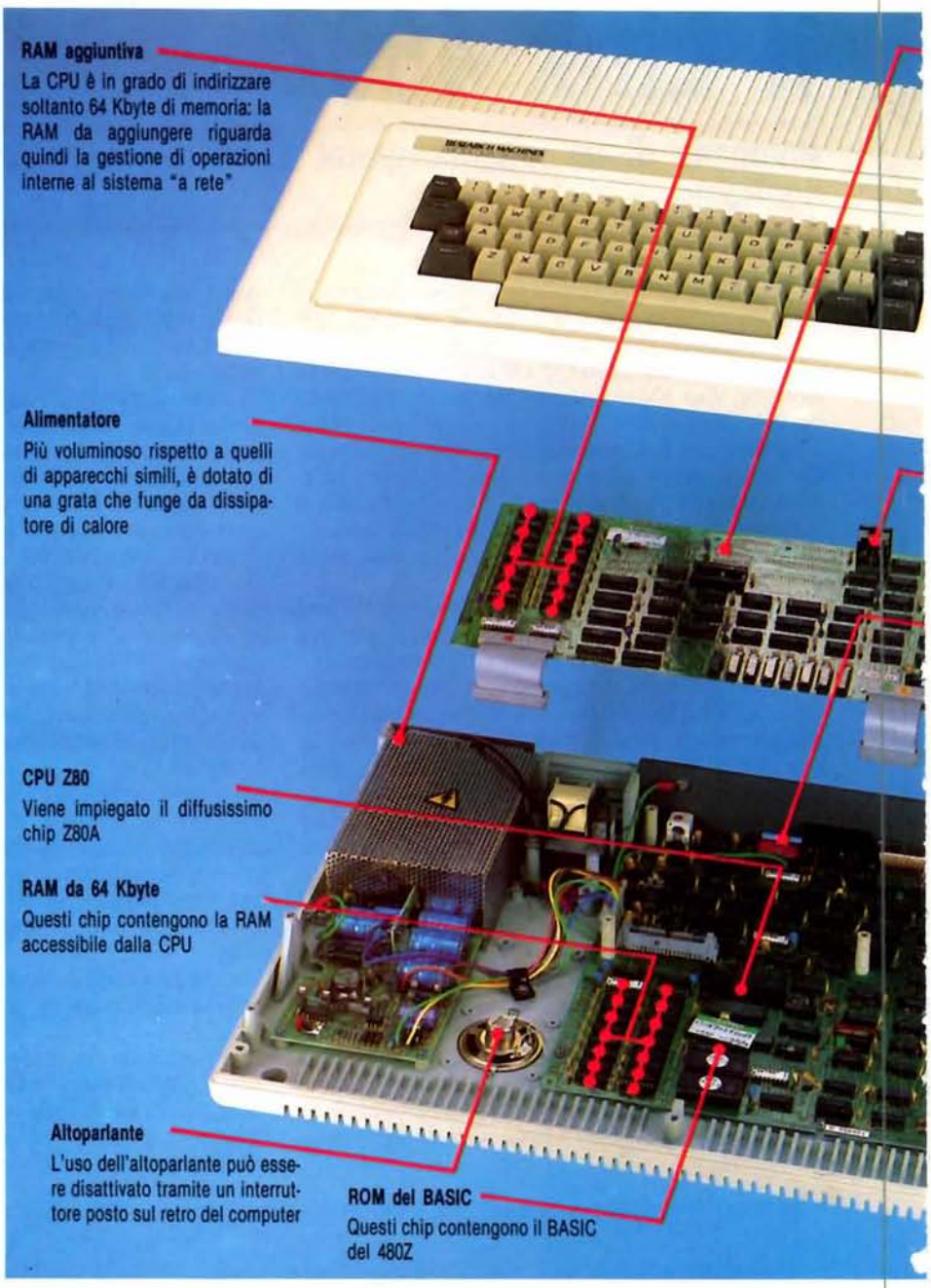
La tastiera inclinata e l'involucro in plastica conferiscono al 480Z un aspetto molto più elegante del suo predecessore 380Z; le dimensioni dell'apparecchio, però, sono superiori alla media. Ciò è dovuto alla presenza di due schede sovrapposte per i circuiti: la prima per le funzioni principali del computer, l'altra per il collegamento a rete

Le unità a dischetti

Il box MD2 contiene una doppia unità a dischetti da 5 1/4 pollici e, sorprendentemente per un computer così recente, il collegamento con l'unità centrale avviene tramite una porta seriale (RS232), anziché parallela. Ciò nonostante, la velocità di trasferimento è di 38,5 KBaud, che rientra nella norma. Le unità a dischetti, a doppia faccia/doppia densità, sono etichettate A, B, C e D.
L'involucro è in solido materiale plastico e il rumore emesso durante il funzionamento è quasi trascurabile, diversamente da quanto accade talvolta in apparecchi anche più costosi.
Nella parte posteriore il box è dotato di due connettori RS232. Uno serve per il collegamento al computer, l'altro per connettere "in cascata" altre unità. Il sistema di trasferimento dei file da e per il computer (sia la parte hardware sia quella software) è contenuto nel box. L'uso di unità a dischetti "intelligenti" significa che l'unità centrale può dedicarsi ad altri compiti, anche durante operazioni su disco, e disporre inoltre di una maggiore quantità di memoria.
All'accensione, si può entrare direttamente in BASIC (una versione estesa, conservata su ROM) oppure visionare il menu di HELP. Premendo il tasto H (per Help, appunto) viene visualizzato un elenco delle routine su ROM disponibili, dedicate soprattutto alla selezione delle funzioni di ingresso/uscita. L'operatore può scegliere, ad esempio, se caricare i programmi da nastro o da disco o se allacciarsi al network. Per le operazioni su nastro e su stampante si può anche selezionare la velocità di trasferimento. Infine, esiste un'opzione *Front panel* (che consiste essenzialmente in un programma monitor in codice macchina) per l'esame o la modifica di locazioni di memoria.

Risoluzione dello schermo

Le possibilità qui sono svariate. Si va dalla normale risoluzione testuale con 25 righe di 80 colonne, per arrivare alla "ultra-alta" risoluzione grafica di 640 x 192 pixel (con soli due colori però). Le tre risoluzioni grafiche intermedie prevedono 16 colori.
Come il 380Z anche il 480Z impiega il chip Z80.
La quantità di software disponibile, tra cui anche il CP/M ovviamente, è enorme e forse è stata proprio questa vasta disponibilità di software ad influenzare la Research Machines nella scelta della CPU. La ditta produttrice afferma che il 380Z e il 480Z sono apparecchi totalmente compatibili, tuttavia pare che in certi casi i programmi sviluppati sul primo e richiamati (in BASIC) sul secondo generino messaggi d'errore.
All'interno dell'apparecchio ci sono alcuni zoccoli liberi per chip aggiuntivi, che possono servire per utilizzare software su ROM (quale, ad esempio, un convertitore digitale/ana-





DOPPIA UNITÀ A DISCHETTI

Ciascun drive prevede l'uso di dischetti a doppia faccia, pertanto sono accessibili quattro facce etichettate A, B, C e D, secondo la convenzione del sistema operativo CP/M. Inoltre il drive funziona in doppia densità, aumentando così la capacità di dati memorizzabili su ciasun dichetto. La Reserach Machines ha prodotto anche un drive in quadrupla densità



logico da collegare a una delle prese esterne). L'apparecchio viene fornito con un dischetto di sistema che contiene diversi programmi dimostrativi, una versione di BASIC completa di comandi per i dischetti e il sistema operativo CP/M.

In conclusione, il 480Z è una versione riveduta del precedente 380Z: le innovazioni riguardano soprattutto il fattore estetico con risultati, va detto, apprezzabili. Considerato che la Research Machines orienta la propria produzione soprattutto verso il settore scolastico, è comprensibile che l'enorme quantità di software oggi disponibile per il chip Z80 abbia fatto preferire questa CPU a quelle più recenti, a 16 bit, quali l'Intel 8088. D'altro canto, però, gli studenti di oggi lavoreranno quasi certamente su apparecchi a 16 bit, quali il PC dell'IBM, per cui il 480Z può sembrare un apparecchio già obsoleto.

GLI "SCHOOL PACK"

Poiché la Research Machines orienta la propria produzione verso il settore scolastico, assieme al 480Z viene consegnato uno "School pack", che contiene una dozzina di package didattici di alto livello.

Il pacchetto contiene quattro linguaggi di programmazione. SBAS è un BASIC strutturato, considerato da molti una delle migliori versioni disponibili. Include varie strutture di controllo, tra le quali la WHILE...ENDWILE, la CASE...ENDCASE e la IF...ENDIF. Inoltre è possibile definire procedure, oltre a variabili globali e locali. Un altro linguaggio fornito, molto interessante a livello sia didattico sia applicativo, è il PASCAL. La documentazione di accompagnamento al linguaggio non raggiunge però vertici di chiarezza, sebbene sia molto dettagliata. Il terzo linguaggio è il LOGO, anch'esso in una eccellente versione. Tuttavia alcuni comandi non sono standard. Il caso più evidente è l'uso di BUILD al posto del consueto TO, per la creazione di procedure. Del LOGO è data anche una seconda versione molto più ridotta. Infine, non poteva mancare un Assembler per realizzare programmi in codice macchina dello Z80; la versione fornita si chiama ZASM.

Quattro programmi del pacchetto riguardano la dattilografia e l'elaborazione di testi. Un programma serve per l'apprendimento della dattilografia ("Touch'n'Go"). Un altro (il "Word") per apprendere l'uso e le possibilità offerte dai word processor. Un terzo è l'ormai conosciutissimo "WordStar" della MicroPro. Il quarto ("Txed"), infine, è un text editor adatto per la redazione sia di testi che di programmi.

"Quest-D" è un database appositamente studiato per l'insegnamento delle tecniche di archiviazione dei dati. Il programma "SIR" (School Information Retrieval), pur trattando di archiviazione, si rivolge invece ad un settore più specifico: la gestione di una biblioteca scolastica (il programma non è una semplice simulazione, ma è utilizzabile in pratica).

Infine c'è un programma di comunicazioni, "Telesoftware", per consentire l'allacciamento in rete di più computer, ma anche alla banca dati Prestel



CHRIS STEVENS

LINK 480Z

DIMENSIONI

mm. 520 x 330 x 80

CPU

Z80, frequenza di lavoro 4 MHz

MEMORIA

64 Kbyte di RAM

VIDEO

Modo testuale:
25 righe di 80 caratteri
Media risoluzione: 160 x 192 pixel (16 colori)
Alta risoluzione: 320 x 192 pixel (4 colori)
Altissima risoluzione: 640 x 192 pixel (2 colori)

INTERFACCE

Uscita RF, per monitor e per monitor TTL/RGB. Porta per dispositivi accessori, per registratore. Una porta parallela, due porte seriali e presa per collegamento a rete

LINGUAGGI FORNITI

BASIC, PASCAL e LOGO

TASTIERA

Tipo QWERTY, con 65 tasti, inclusi quelli funzione e quelli per il cursore

DOCUMENTAZIONE

I manuali sono molto esaurienti e trattano gli argomenti, fin nei minimi particolari, ad un livello adatto per i principianti. Talvolta, però, è difficile localizzare le informazioni cercate

PRO

Il Link 480Z è un apparecchio sviluppato principalmente (ma non esclusivamente) ad uso scolastico. La Research Machines ha sviluppato una notevole esperienza nel settore. La presenza del sistema operativo CP/M significa un'enorme disponibilità di software e la possibilità di collegare più apparecchi in rete rende questo computer molto versatile

CONTRO

Essendo basato sul "glorioso" (ma datato) chip Z80, questo computer è considerato un poco fuori moda, nonostante l'affidabilità e la robustezza

Potenza di calcolo

Un rapido sguardo a quattro utili package per home computer

Micro Swift, *Practicalc II*, *PS* e *Vizastar* appartengono tutti a una nuova generazione di programmi basati su spreadsheet, la cui struttura è stata senza dubbio ispirata sia dal *Lotus 1-2-3*, sia dal *Symphony* (vedere a pag. 1124).
Tuttavia, mentre questi ultimi due package sono destinati ai computer compatibili con il Personal della IBM (il primo necessita di 296 Kbyte, mentre il secondo di 320 Kbyte almeno), i programmi citati all'inizio sono progettati per la gamma degli home computer. In effetti, si può dire che siano stati compiuti dei veri miracoli per concentrare tante funzioni nei soli 30 Kbyte di memoria disponibili su apparecchi quali, ad esempio, il Commodore 64.
Ovviamente, in questi "mini-Symphony" qualche compromesso è stato necessario: infatti due delle quattro funzioni che rendono così utili (e costosi) i package originali non sono previste, almeno per il momento.
Del resto, il tentativo di incorporare un word processor, uno spreadsheet, un database e la programmabilità, in così poca memoria, avrebbe inevitabilmente condotto ad un risultato deludente, simile al software su ROM *Tre-Più-Uno* del Commodore Plus/4 (vedere a pag. 1189).

Potenza relativa

Tentiamo adesso un'analisi comparata dei quattro package. *PS*, *Micro Swift* e *Vizastar* sono tutti programmabili, in misura più o meno estesa. Questa è una caratteristica importante, poiché permette all'utente di rendere automatiche certe sequenze di lavoro che, altrimenti, richiederebbero noiose ripetizioni di comandi.
Questa possibilità assomiglia ai comandi "Macro" del *Lotus 1-2-3* (vedere a pag. 1264). Tuttavia, esistono sostanziali differenze nei tre package.
Nel *PS* i moduli, programmati per mezzo di normali comandi BASIC, vengono memorizzati premendo il tasto f3 ed eseguiti premendo il tasto U. È possibile anche un'esecuzione automatica dei moduli se, nell'eseguire la SAVE, il nome del programma è preceduto da un punto. Tra le caratteristiche di spicco di questo package c'è quella che consente di eseguire chiamate a subroutine del programma, semplicemente digitando GOSUB nella formula abbinata alla cella desiderata. Inoltre, tramite la FN del BASIC, si possono anche definire funzioni. Infine, è possibile scambiare stringhe, valori numerici, numeri di righe e di colonne dal programma allo spreadsheet e viceversa.
Micro Swift viene programmato inserendo una serie di comandi nella colonna Z dello spreadsheet: il primo comando consiste nel nome del programma, preceduto dal simbolo #, e l'ultima riga deve contenere il comando @QUIT.
Ecco un esempio:

```
Z1 #SUM
Z2 @SUM(A1,A3)
Z3 @ASSIGN(Z2,A4)
Z4 @QUIT
```

Questo programma somma i valori contenuti nelle celle A1, A2 e A3 per poi assegnare il risultato, che si trova in Z2, alla cella A4. Il programma è richiamato usando #SUM.
Dei quattro package qui esaminati, il più facile da programmare è forse il *Vizastar*, poiché i comandi vengono immessi usando la loro lettera iniziale. Per usare un particolare database, ad esempio, si adopera la sequenza: tasto Commodore, D (per Data), U (per Use) e D (per Database), immettendo per ultimo il nome del database. La sequenza va terminata con la pressione di RETURN.
Nella programmazione, il tasto Commodore viene simboleggiato dal segno /, per cui la sequenza /DUDnomeRETURN può essere abbinata a (e richiamata con) il tasto f8.
I tasti di funzione e quelli di editing vengono programmati premendo il tasto CTRL e quello desiderato (che appare poi nel programma quando si usa la particolare funzione definita). Fanno eccezione i tasti del cursore, che nei programmi appaiono come [up], [left], [right], [down].
Il database del *Vizastar* è particolarmente potente ed usa una sezione del "foglio" (dalla riga 1000 in poi), normalmente inaccessibile all'utente, per conservare il formato dei record. Ad ogni record, che può consistere di un massimo di nove schermi, si accede tramite vari comandi. È consentita, naturalmente, anche l'aggiunta, la modifica (o meglio la sostituzione) e la cancellazione dei record.
I campi sono individuati da lettere (da A a BK), corrispondenti alle colonne dello spreadsheet. Ciò consente di creare sequenze di ricerca dei dati. In tal caso, la chiave di ricerca deve trovarsi nella colonna A.
Practicalc è uno spreadsheet nel quale le etichette possono sconfinare ed occupare le celle vuote adiacenti. Questa possibilità consente di usare il programma come semplice

UNO SPREADSHEET NEI TRASPORTI

La maggior parte dei package commerciali specificamente destinati alla gestione di automezzi da trasporto (ad uso quindi di corrieri e aziende di trasporti) ha un costo molto alto e "gira" soltanto su computer di grandi dimensioni. Tra l'altro il parco automezzi previsto da tali programmi è generalmente dimensionato per circa un migliaio di veicoli. Considerando che in realtà molte ditte del settore hanno tutt'al più una decina di vetture, l'investimento necessario per computerizzare l'intera gestione risulta di gran lunga superiore agli eventuali benefici.

Tuttavia, in Gran Bretagna c'è stato chi, armatosi di un personal computer (e di buona volontà), ha tentato con notevole successo di risolvere il problema, giungendo ad una soluzione molto più economica. Si tratta di Terry Palmer, un consulente dei trasporti con qualche nozione di informatica.

Inizialmente Palmer ha realizzato il proprio progetto usando il "Lotus 1-2-3", ma in seguito è riuscito a comprimere la versione definitiva del programma nella limitata memoria di uno dei più diffusi home computer: il Commodore 64. Il materiale occorrente, oltre al C64, si limita al programma "Vizastar" (il database/spreadsheet descritto in queste pagine), per un costo totale di circa un quinto, rispetto alle soluzioni offerte dal mercato.

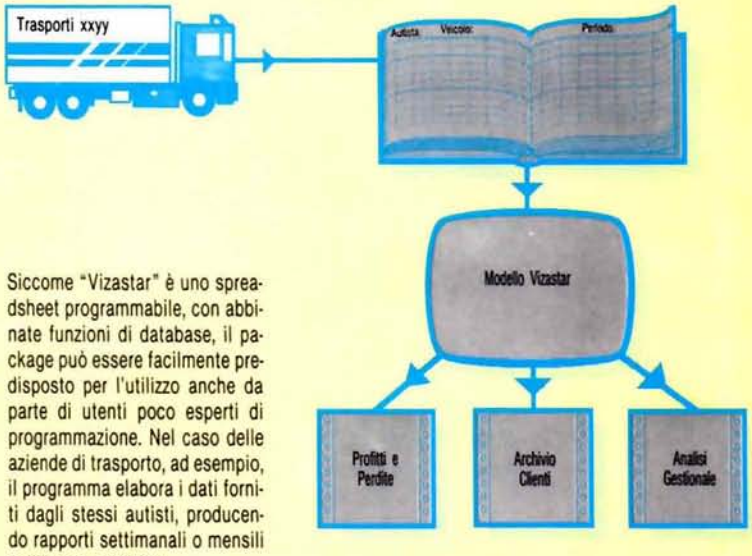
In primo luogo Palmer, collaborando con varie aziende di trasporti, ha formulato un elenco di tutti i dati necessari alla gestione. Il tradizionale "ruolino di marcia" usato dagli autisti, è stato ampliato, aggiungendovi una serie di annotazioni (alcune delle quali compilate dagli stessi autisti). Al termine di ogni viaggio (oppure anche settimanalmente), i dati contenuti in questi schedari vengono ricopiati nello spreadsheet, ottenendo un quadro generale e aggiornato di distanze percorse, spese sostenute, annotazioni sugli incarichi e via dicendo.

Lo spreadsheet calcola automaticamente, in base ai dati immessi, i profitti e le perdite relative al periodo in questione. Volendo, un secondo spreadsheet può essere impiegato per raccogliere i risultati suddivisi mensilmente e ottenere così un quadro d'insieme della gestione annuale.

Considerando che "Vizastar" svolge anche funzioni di database, è facile archiviare, ad esempio, anche i dati concernenti i clienti abituali (abbonamenti, condizioni di sconto ecc.)

Infine, partendo dai dati reali, è possibile valutare l'incidenza dei vari costi formulando ipotesi del tipo «Quali effetti produrrebbe un aumento di x lire della benzina?»

IN MOVIMENTO



Siccome "Vizastar" è uno spreadsheet programmabile, con abbinate funzioni di database, il package può essere facilmente predisposto per l'utilizzo anche da parte di utenti poco esperti di programmazione. Nel caso delle aziende di trasporto, ad esempio, il programma elabora i dati forniti dagli stessi autisti, producendo rapporti settimanali o mensili sull'intera gestione

IAN McKINNELL

word processor, in cui la lunghezza massima di ciascuna riga è di 100 caratteri. In effetti, sono previste molte delle funzioni tipiche dei word processor, tra le quali: il word-wrap e lo spostamento, l'inserimento e la cancellazione di blocchi.

È anche possibile richiamare ed inglobare uno spreadsheet, preparato in precedenza, in un testo durante la sua stesura.

Lo spreadsheet richiamato è funzionante (per eseguire calcoli "ad hoc" da includere nel documento redatto) senza per questo influire sulla versione del foglio memorizzata su dischetto.

Comunque, una caratteristica comune ai quattro package è quella di poter accedere ai file generati tramite un word processor o un database separato (purché della stessa marca).

Vizastar, ad esempio, può accedere ai file creati con *Vizawrite*; *Micro Swift* a quelli generati dal *Micro Magpie*; *Practicalc* e *PS* a quelli prodotti da *Practifile* della Practicorp. Tuttavia, dato che in tutti i casi si tratta di file sequenziali, è anche possibile un utilizzo "misto", con evidenti vantaggi per l'utente.

Micro Swift: Per il Commodore 64
Editore: Audiogenic (GB)
Formato: Su dischetto

Practicalc II: Per l'Apple II con 48K, il Micro BBC e il Commodore 64
Editore: Practicorp (GB)
Formato: Su dischetto

PS: Per il Commodore 64
Editore: Practicorp (GB)
Formato: Su dischetto

Vizastar: Per il Commodore 64
Editore: Viza Software (GB)
Formato: Su dischetto con cartuccia da 4 Kbyte

Taratura del robot

Il controllo dei movimenti e l'interazione tra sensori e motori

I motori passo passo sono l'ideale da impiegare con apparecchiature digitali, poiché la loro rotazione può essere accuratamente calibrata in base agli impulsi elettrici applicati. Tuttavia, per convertire le rotazioni dei motori sul nostro robot in quantità più adatte (distanza e angolazione del veicolo), occorrono alcuni tentativi sperimentali su percorsi campione. Solo così si possono ricavare i rapporti medi tra il numero di impulsi da applicare e, rispettivamente, l'effettivo spostamento lineare e quello angolare del robot. Questi due rapporti andranno poi utilizzati nei programmi applicativi. Più avanti nel corso, infatti, viene illustrato lo sviluppo del software che mette il robot in grado di rilevare e rappresentare, in modo digitale, la forma degli oggetti che lo circondano. Affinché il robot possa spostarsi agevolmente tra questi, occorrono proprio i valori desunti dagli esperimenti presentati in queste pagine.

Calibrazione lineare

Una prima approssimazione del rapporto tra impulsi e distanza percorsa si può ottenere a priori applicando semplici relazioni matematiche. Se ad ogni impulso corrisponde, ad esempio, una rotazione di 7,5° del motore il cui asse è collegato ad una demoltiplica 25:2, è facile ricavare la rotazione della ruota: 7,5 x 2/25 = 0,6°. Misurando il raggio delle ruote impiegate, si può calcolare, approssimativamente, anche la distanza corrispondente ad un impulso: 0,6 / 360 x (2 x π x raggio della ruota). Supponendo, come nel caso delle ruote della Lego, che il raggio sia di 30 mm., si ottiene che un impulso equivale a un percorso, in millimetri, di 0,1 x π. Il rapporto impulsi/distanza teorico vale quindi 1/0,31415, ossia 3,183.

Il seguente programma di taratura permette di collaudare il robot su più distanze: durante ogni prova sullo schermo compare il numero di impulsi applicati e la distanza teorica percorsa. Facendo muovere il robot a fianco di un metro adagiato sul pavimento, possiamo misurare la distanza effettiva percorsa in ciascuna prova. Il programma, infine, visualizza una tabella riassuntiva degli impulsi, delle distanze effettive e di quelle teoriche, calcolando anche la media del rapporto impulsi/distanza. Si prenda nota di quest'ultimo importante valore. I valori riportati a pag. 1375 mostrano i risultati di una serie di prove, da cui si vede che, nel caso specifico, il robot copre una distanza effettiva leggermente superiore a quella teorica. Lo scopo del programma è quello di ricavare un rapporto impulsi/distanza, il più accurato possibile, da usare nei successivi programmi applicativi.

```
 10 REM ••• TARATURA — BBC •••
 20 DDR = &FE62:REGDAT = &FE60
 30 ?DDR = 15:REM LINEE 0—3 IN USCITA
 50 avanti = 4:indietro = 2:DIM MD(12)
 60 FOR CC = 500 TO 1700 STEP 100
 70 ?REGDAT = 0
 80 ?REGDAT = (?REGDAT OR 1) OR avanti
 90 PRINT CC,INT(CC*π)/10
100 A$ = GET$
110 FOR I = 1 TO CC
120 PROCimpulsi
130 NEXT I
140 INPUT"DISTANZA MISURATA IN MM.";MD((CC—500)/100)
150 NEXT CC
160 ?REGDAT = 0:T = 0
180 PRINT" IMPULSI"," MM. MISURATI"," TEORICI"
190 PRINT
200 FOR CC = 500 TO 1700 STEP 100
210 PRINT CC,MD((CC—500)/100),INT(CC*π)/10
220 T = T + CC/MD((CC—500)/100)
230 NEXT CC
240 PRINT:PRINT "RAPPORTO IMPULSO/DISTANZA:",T/12
260 END
270 DEF PROCimpulsi
280 ?REGDAT = (?REGDAT OR 8)
290 ?REGDAT = (?REGDAT AND 247)
300 ENDPROC
```

```
 10 REM ••• TARATURA — CBM64 •••
 20 DDR = 56579:REGDAT = 56577
 30 POKE DDR,15:REM LINEE 0—3 IN USCITA
 50 AV = 4:ND = 2:DIM MD(12)
 60 FOR CC = 500 TO 1700 STEP 100
 70 POKE REGDAT,0
 80 POKE REGDAT,(PEEK(REGDAT) OR 1) OR AV
 90 PRINT CC,INT(CC*π)/10
100 GET A$:IF A$ = "" THEN 100
110 FOR I = 1 TO CC
120 GOSUB 270:REM S/R PER IMPULSI
130 NEXT I
140 INPUT"DISTANZA MISURATA IN MM.";MD((CC—500)/100)
150 NEXT CC
160 POKE REGDAT,0:T = 0
170 REM •• LE LINEE DA 180 A 260 SONO IDENTICHE ALLA ••
175 REM •• VERSIONE PER IL BBC, MA SOSTITUIRE PI CON π ••
270 REM ••• S/R PER IMPULSI •••
280 POKE REGDAT,PEEK(REGDAT) OR 8
290 POKE REGDAT,PEEK(REGDAT) AND 247
300 RETURN
```

Calibrazione angolare

Ecco come calcolare il rapporto tra impulso e angolo di rotazione del veicolo. Tenendo una delle due ruote ferma e l'altra in movimento, quest'ultima descrive una circonferenza che ha per raggio la distanza tra le due ruote. Supponendo che tale distanza sia di 140 mm., la circonferenza misura πx140. Un impulso, quindi, corrisponde ad un tragitto lungo il cerchio pari a 360x0,1xπ/(140xπ) gradi e il rapporto impulso/gradi vale dunque 3,846.

Qui il problema è misurare con accuratezza l'angolo della rotazione effettiva ma, poiché nella maggior parte dei casi esso sarà di 90° (o multipli di 90° facilmente misurabili, in base al rapporto teorico possiamo calcolare che per una rotazione ad angolo retto occorrono 346 impulsi.

Su di un foglio di carta si traccino due rette

IMPULSI	MM.MISURATI	TEORICI
500	160	157
600	195	188.4
700	225	219.9
800	258	251.3
900	293	282.7
1000	324	214.1
1100	352	345.5
1200	390	376.9
1300	421	408.4
1400	452	439.8
1500	488	471.2
1600	522	502.6
1700	553	534

RAPPORTO IMPULSO/DISTANZA: 3.34767511

Tabella valori teorici

Il programma di taratura del robot produce una tabella di questo tipo, che serve per verificare che la distanza calcolata in teoria sia sufficientemente vicina a quella effettivamente percorsa. In base alle 12 prove, viene calcolato un valore medio per il rapporto tra impulsi e distanza percorsa. Tale valore va accuratamente annotato, poiché serve per lo sviluppo dei programmi applicativi del robot

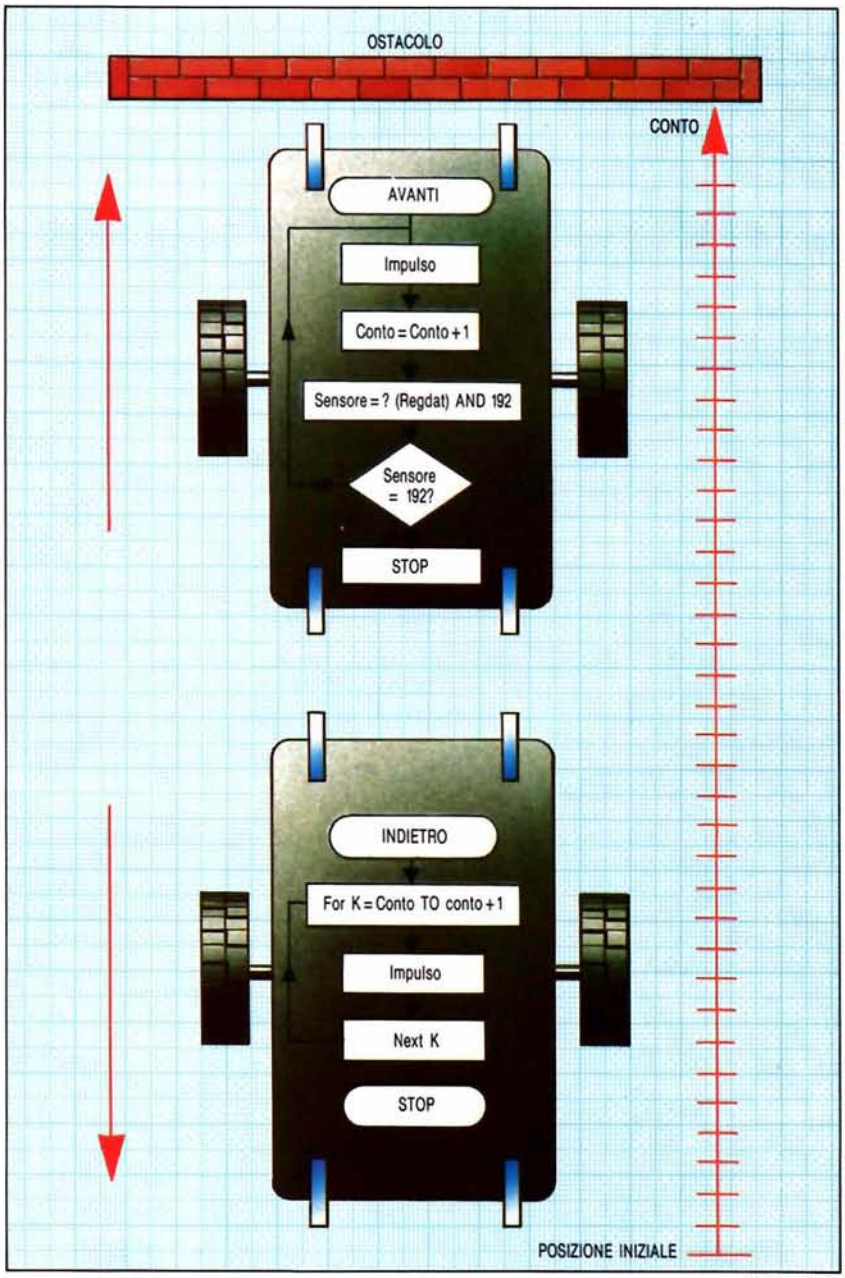
AZIONE RESPINGENTE

Il movimento in avanti è prodotto selezionando gli appropriati bit nel registro dati ed inviando impulsi ai motori. Il numero di tali impulsi (che corrisponde alla distanza percorsa) viene conteggiato e conservato in una variabile e, ad ogni passo, viene eseguita una verifica per constatare se i sensori segnalano una collisione. Queste operazioni vengono ripetute finché non si rileva una collisione. In tal caso viene invertita la direzione di moto e un ciclo FOR...NEXT provvede ad applicare il numero di impulsi occorrenti per tornare alla posizione di partenza. Poiché all'andata le operazioni che il BASIC deve svolgere sono più complesse di quelle per il ritorno, il robot è più svelto a retrocedere che non ad avanzare

perpendicolari. Su una delle perpendicolari si segni in modo chiaro la posizione di partenza corrispondente alle ruote del robot. Poi, usando il seguente programma, si faccia eseguire al veicolo una svolta di 90°. Il ciclo FOR...NEXT (linea 70) stabilisce il numero di impulsi da inviare ai motori. Negli esperimenti con il prototipo, il valore effettivo per una rotazione ad angolo retto è risultato essere 371. Il limite superiore del ciclo va aggiustato, per tentativi, in base alla risposta del robot realizzato, fino ad ottenere un posizionamento perfetto sulla linea perpendicolare tracciata sul foglio. In seguito si può verificare il corretto funzionamento anche per svolte in direzione opposta (modificando opportunamente la linea 60). Raddoppiando il limite superiore del ciclo la rotazione deve essere di 180°. Eventuali aggiustamenti possono essere fatti anche ravvicinando o allontanando leggermente le due ruote fra loro, prima di fissarle definitivamente nella posizione migliore.

```
10 REM **** SVOLTA — CBM64 ****
20 DDR=56579:REGDAT=56577
30 POKE DDR,15:REM LINEE 0—3 IN USCITA
40 SX=6:DX=0
50 POKE REGDAT,0
60 POKE REGDAT,(PEEK(REGDAT) OR 1) OR DX
70 FOR I = 1TO 371:GOSUB 90:NEXT I
80 POKE REGDAT,0:END
90 REM ** S/R PER IMPULSI **
100 POKE REGDAT,PEEK(REGDAT) OR 8
110 POKE REGDAT,PEEK(REGDAT) AND 247
120 RETURN
```

```
10 **** SVOLTA — BBC ****
20 DDR=&FE62:REGDAT=&FE60
30 ?DDR=15:REM LINEE 0—3 IN USCITA
40 sinistra=6:destra=0
50 ?REGDAT=0
60 ?REGDAT=(?REGDAT OR 1) OR destra
70 FOR I=1 TO 371:PROCimpulso:NEXT I
80 ?REGDAT=0:END
90 DEF PROCimpulso
100 ?REGDAT=(?REGDAT OR 8)
110 ?REGDAT=(?REGDAT AND 247)
120 ENDPROC
```



KEVIN JONES

Avendo dotato il robot di sensori a microinterruttori, possiamo sviluppare le routine di base per un controllo del veicolo condizionato dalle informazioni fornite da tali sensori. Il semplice programma che segue fa procedere il

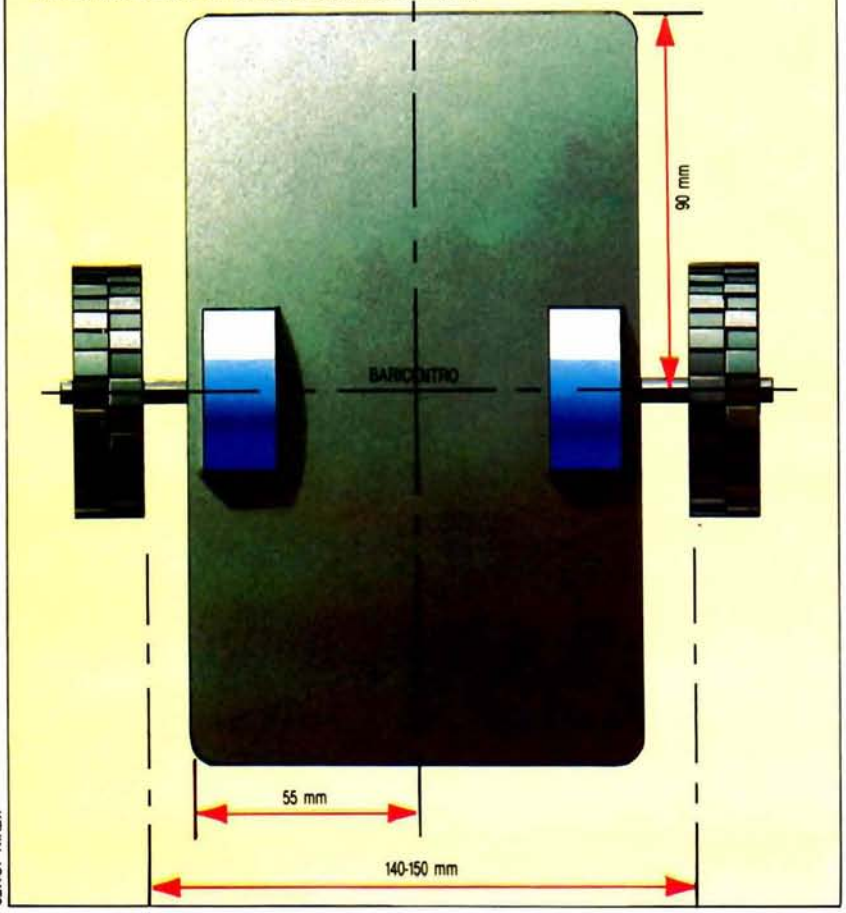
robot in avanti finché non incontra un ostacolo. Quando ciò accade il veicolo retrocede fino alla posizione di partenza.

```
10 REM •••• OSTACOLO — CBM64 ••••
20 DDR = 56579:REGDAT = 56577
30 POKE DDR,15:REM LINEE 0—3 IN USCITA
40 AV = 4:ND = 2
50 POKE REGDAT,(PEEK(REGDAT) OR 1) OR AV
60 REM ••• IMPULSI IN AVANTI •••
65 CC = 0
70 GOSUB 1000:CC = CC + 1:REM IMPULSI
80 IF (PEEK(REGDAT) AND 192) = 192 THEN 70
90 REM •••• TORNA INDIETRO ••••
95 POKE REGDAT,(PEEK (REGDAT) AND 1) OR ND
100 FOR I = 1 TO CC
110 GOSUB 1000:REM IMPULSI
120 NEXT I
130 POKE REGDAT,0:END
1000 REM •• S/R PER IMPULSI ••
1010 POKE REGDAT,(PEEK(REGDAT) OR 8)
1020 POKE REGDAT,(PEEK(REGDAT) AND 247)
1030 RETURN
```

```
10 •••• OSTACOLO — BBC ••••
20 DDR = &FE62:REGDAT = &FE60
30 ?DDR = 15:REM LINEE 0—3 IN USCITA
40 avanti = 4:indietro = 2
50 ?REGDAT = (?REGDAT OR 1) OR avanti
60 REM ••• IMPULSI IN AVANTI •••
65 conto = 0
70 REPEAT :PROCimpulsi:conto = conto + 1
80 UNTIL (?REGDAT AND 192) < > 192
90 REM •••• RITORNA ALL'INIZIO ••••
95 ?REGDAT = (?REGDAT AND 1) OR indietro
100 FOR I = 1 TO conto
110 PROCimpulsi
120 NEXT I
130 ?REGDAT = 0:END
1000 DEF PROCimpulsi
1010 ?REGDAT = (?REGDAT OR 8)
1020 ?REGDAT = (?REGDAT AND 247)
1030 ENDPROC
```

CALIBRAZIONE DELLE RUOTE

Durante gli esperimenti descritti in queste pagine può rendersi necessario correggere leggermente la distanza tra le due ruote motrici, in quanto il raggio della circonferenza descritta da una ruota, mentre l'altra resta ferma, influisce sulla precisione dei movimenti. Per facilitare il compito di centratura delle due ruote, occorre per prima cosa marcare con esattezza l'asse centrale del veicolo. In figura sono riportate le distanze ottimali. Quelle effettive vanno però ricavate in modo sperimentale, facendo attenzione a conservare una perfetta simmetria



KEVIN JONES

La logica del programma è presto descritta:

1. Inizializzazione del registro di direzione dati a 15 (bit 0—3 in uscita, bit 4—7 in ingresso)
2. Avviare il motore in avanti
3. Fornire impulsi ai motori finché i bit 6 o 7 non passano a livello basso, conteggiando il numero di impulsi applicati
4. Invertire il senso di marcia
5. Applicare il numero di impulsi (conteggiato in precedenza) ai motori
6. Azzerare il registro dati (arresto).

In questo programma i microinterruttori situati sul lato più lontano rispetto al pannello posto sul coperchio del veicolo sono considerati i "respingenti" frontali, collegando i ponticelli in modo da influire sulle linee d'ingresso 6 e 7. D'ora in poi viene adottato tale criterio per distinguere la parte anteriore da quella posteriore del veicolo.
Dei quattro bit "bassi" del registro dati che controlla il funzionamento dei motori, il bit 0 è quello di "riposo" (normalmente posto a 1), i bit 1 e 2 servono per controllare, rispettivamente, l'azionamento del motore destro e di quello sinistro e il bit 3 aziona simultaneamente i due motori, facendoli procedere di un "passo" ad ogni impulso applicato (passaggio del bit 3 dal livello basso a quello alto). Usando gli operatori logici AND e OR si può modificare il valore di ciascun bit del registro dati, senza alterare gli altri. Poiché i quattro bit alti del registro sono destinati all'ingresso di dati, essi sono normalmente posti a 1. Quando però si chiude il contatto di un microinterruttore, il corrispondente bit del registro dati passa al livello basso, ossia a 0. Isolando i bit 6 e 7, che valgono entrambi 1 in condizioni normali, si ottiene il valore 192 (in binario 11000000). Tale valore è impiegato nel ciclo che manda il veicolo in avanti (linee 70—80), per verificare quando uno dei sensori (o entrambi) segnala la presenza di un ostacolo. In tal caso il ciclo termina per passare al movimento in retromarcia. Avendo tenuto il conto degli impulsi applicati nel percorso in avanti, il robot è in grado di retrocedere fino alla posizione iniziale di partenza. Se l'angolo di rotazione del motore, per ciascun "passo", vale 7,5° un singolo impulso corrisponde ad uno spostamento della ruota di meno di un millimetro, quindi il posizionamento del veicolo è facilmente misurabile.
Infine, vale la pena di far notare che il movimento in avanti risulta più lento di quello all'indietro: ciò è dovuto alla lentezza del BASIC. Infatti, durante il percorso in avanti, il BASIC impiega tempo per eseguire il conteggio e per controllare l'eventuale collisione con l'ostacolo. Nel movimento inverso, invece, queste operazioni non vengono espletate.

Uso di vettori nel BBC

L'ultima parte dedicata al sistema operativo del BBC

Quasi tutte le routine del SO per il BBC sono dette "vettorizzate" (vedere a pag. 1358). Per ordinare al SO di chiamare, ad esempio, la routine OSCLI, un programma contiene una chiamata alla locazione &FFF7. Il codice a tale indirizzo non corrisponde effettivamente alla routine: l'indirizzo viene ricavato estraendolo da due particolari locazioni RAM della pagina 2. Tali locazioni vengono chiamate "vettori": la parte bassa del vettore è nella locazione numericamente minore, la parte alta nella locazione immediatamente superiore. Quindi per OSCLI, vettorizzata tramite le locazioni &208 e &209, la parte bassa dell'indirizzo si trova in &208 e la parte alta in &209, secondo il solito metodo di indirizzamento "basso-alto", usato dalla CPU 6502. Gli indirizzi nella tabella dei vettori vengono depositati dal SO ogni volta che si esegue un RESET dell'apparecchio. L'impiego di simili tabelle può sembrare, a prima vista, complicato ma in realtà offre notevoli vantaggi al programmatore.

Si sarà notato che nei programmi tutte le routine del SO vengono richiamate usando indirizzi compresi tra &FF00 e &FFFF. Ciò non è affatto accidentale, ma anzi ha un preciso fondamento: la posizione situata tra &FF00 e &FFFF nella tabella è identica in tutte le versioni del SO per il BBC. Ciò che cambia sono gli indirizzi delle routine di sistema nella tabella dei vettori cosicché, se la casa costruttrice ritiene necessario modificare la disposizione delle routine nelle ROM, tale fatto non influisce minimamente sul funzionamento dei programmi applicativi, purché questi non accedano in modo irregolare (con indirizzi esterni all'arco &FF00-&FFFF) alle routine del SO.

Un secondo vantaggio di adoperare una tabella vettoriale è che risulta facile modificare l'effetto di una chiamata al SO, sostituendo il vettore corrispondente con l'indirizzo di una routine in codice macchina scritta dall'utente.

Consideriamo, ad esempio, il vettore chiamato USERV, contenuto nelle locazioni &200 e &201. È un vettore "speciale", nel senso che, normalmente, il suo uso... non produce alcun effetto! È impiegato da due comandi "stella" (*CODE e *LINE). Impartendo questi da tastiera, il computer risponde con il messaggio d'errore Bad Command. Prima di precipitarsi a reclamare con la Acorn, però, si legga quanto detto qui di seguito.

USERV è appositamente previsto per poter "personalizzare" i due comandi citati. *CODE, ad esempio, è particolarmente utile per il passaggio di parametri a programmi in codice macchina, come illustrato dalla seguente tabella:

Registro	*CODE x,y	*LINE stringa
A	0	1
X	Contiene il valore del primo parametro che segue *CODE (arco 0-255)	Contiene il byte basso dell'indirizzo in memoria della stringa che segue *LINE
Y	Contiene il valore del secondo parametro di *CODE (arco 0-255)	Contiene il byte alto dell'indirizzo in memoria della stringa che segue *LINE

La tabella mostra lo stato dei tre registri della CPU all'ingresso della routine USERV. Il registro A può contenere 0 o 1, indicando così quale dei due comandi è stato impartito. Il significato dei registri X e Y varia in base a tale comando. Impartendo *CODE 3, 2, ad esempio, i valori dei registri saranno i seguenti: A=0, X=3 e Y=2. Il vettore corrispondente a USERV, naturalmente, va previamente modificato per contenere la routine in codice macchina alla quale si desidera passare tali parametri.

Nel riquadro a pagina seguente è riportato il listato di un programma dimostrativo sull'uso pratico di *CODE. La routine è assemblata con origine in &C00 (come stabilito dalla linea 40: la variabile P% ha questo effetto). Per modificare il vettore corrispondente a USERV si deposita il byte basso dell'indirizzo (ossia &00) in &200 e quello alto (&0C) in &201. La routine ha un effetto molto semplice: visualizza sullo schermo il carattere asc un numero di volte rep, poiché tali sono i parametri passati tramite il comando *CODE. Le linee 20 e 25 depositano l'indirizzo della nostra routine in codice macchina nel vettore USERV. Il ciclo compreso tra le linee 90-120 visualizza Y volte il carattere ASCII contenuto in A. Se la routine viene richiamata impartendo un comando *LINE, la verifica alle linee 60 e 70 se ne accorge, facendo terminare immediatamente la routine. Le linee 200-250 eseguono la chiamata a USERV con i parametri adatti.

L'uso del comando *LINE è meno frequente, ma rappresenta un modo molto elegante di passare variabili stringa dal BASIC a routine in codice macchina redatte dall'utente. Per il suo uso si consulti la tabella descritta poc'anzi, prestando attenzione all'ordine col quale vengono passate le due parti dell'indirizzo.

Interazione dell'utente

I principali sistemi di interazione tra utente e

computer sono basati sulla tastiera e sullo schermo. Vediamo come il SO del BBC gestisce tali importanti interazioni.
Esaminiamo per prima la routine del SO che consente di leggere un carattere dal corrente flusso in ingresso. La routine, chiamata OSRDCH, ha per indirizzo &FFE0 ed il suo vettore si trova in &210 e &211. In primo luogo occorre stabilire quale flusso deve essere considerato quello in ingresso. I due flussi possibili sono la tastiera e la porta seriale RS423. La selezione avviene tramite una chiamata OSBYTE oppure *FX, come riassunto in tabella:

Selezione del flusso in ingresso				
n	Tastiera	RS423	Assembly	BASIC
0	✓	X	LDA #2	*FX2,n
1	X	✓	LDX #n	
2	✓	✓	JSR &FFF4	

Pertanto, *FX2,1 disabilita la tastiera e abilita la RS423 come flusso d'ingresso. I dati ricevuti dalla porta seriale vengono trattati come se venissero digitati sulla tastiera. Lo stesso risultato si ottiene, in codice macchina, usando per n il valore 1.
Stabilito quale deve essere il flusso d'ingresso, si può accedere a OSRDCH, la cui utilità è principalmente sentita nei programmi in codice macchina, poiché il BASIC possiede già i comandi GET e INPUT per l'immissione dei caratteri. Eseguendo una chiamata all'indirizzo &FFE0, l'accumulatore riporta il carattere prelevato dal flusso. Se, durante la fase di lettura, non viene rilevato alcun errore il flag di carry vale zero, altrimenti esso viene posto a 1 dalla routine, per significare che il valore di A non va considerato valido. Un caso particolare si presenta quando, usando la tastiera come flusso d'ingresso, il tasto premuto è quello di Escape (codice ASCII 27). Il SO operativo considera questo un caso speciale (ad esempio come richiesta di uscire da un programma) quindi pone il flag di carry a 1. Il programma dell'utente deve tener conto di tale evenienza, poiché il SO aspetta da esso una precisa operazione.
Ciò si ottiene ricorrendo a una chiamata OSBYTE con A=126. Questa esegue un reset del SO, ripulendo le varie zone RAM di lavoro da esso usate. In BASIC, una simile reazione al codice Escape è automatica, ma in codice macchina va programmata. Ecco una semplice routine, per la lettura di carattere dalla tastiera, che tiene conto del codice Escape:

```
1000    .immiss  JRS &FFE0
1010             BCS errore
1020             RTS
1030    .errore  CMP #27
1040             BNE uscita
1050             LDA #126
1060             JRS &FFF4
1070    .uscita  RTS
```

La linea 1000 chiama la routine OSRDCH e la linea 1010 verifica il flag di carry. Se questo vale zero, la RTS provoca un ritorno al programma chiamante con il carattere in A. Altrimenti, la linea 1030 verifica se l'errore è causato dalla pressione di Escape e, in tal caso, le linee 1050 e 1060 provvedono alla necessaria "reazione". A questo punto si può essere indotti a credere che per immettere una stringa di caratteri occorra realizzare un'apposita routine in codice macchina, ma non è così: una delle chiamate OSWORD, analizzate meglio più avanti, è proprio adatta a questo scopo.
L'indirizzo &FFF1 è quello da usare per le chiamate OSWORD, depositando in A il codice della particolare routine desiderata. In tutte le chiamate OSWORD i registri X e Y puntano ad un particolare blocco di memoria, chiamato "blocco di controllo", che deve contenere i parametri previsti dalla particolare routine chiamata. X contiene la parte bassa dell'indirizzo del blocco, Y quella alta. Ecco come è strutturato il blocco di controllo per la routi-

COMANDI DEFINITI DA UTENTE (USERV)

```
5 DIM C 100
10 USERV=&200
20 ?USERV=&00
25 ?(USERV+1)=&0C
30 FOR I%=0 TO 2 STEP 2
40 P%=&C00
50 [  OPT I%
60     CMP  #0
70     BNE  notcode
80     TXA
90     .ciclo
95     JSR &FFE3
100    DEY
110    CPY  #0
120    BNE  ciclo
130    RTS
140    .notcode
145    RTS
150 ] : NEXT I%
160 :
200 FOR rep=1 to 10
210 FOR asc=33 to 48
220 asc$=STR$(asc)
230 rep$=STR$(rep)
240 code$="*CODE  "+asc$+","+rep$
250 $C=code$
260 X%=C MOD 256
270 Y%=C DIV 256
280 CALL &FFF7
290 NEXT : NEXT
```

GRAFICA CON OSWRCH

```
 10 MODE 1
 20 FOR I%=0TO2STEP2
 30 P%=&C00
 40 [OPT I%
 50 LDA #18
 55 JSR &FFEE
 60 LDA #0
 65 JSR &FFEE
 70 LDA #1
 75 JSR &FFEE
 80
 90 LDA #25
 95 JSR &FFEE
100 LDA #5
105 JSR &FFEE
110 LDA #0
115 JSR &FFEE
120 LDA #100
125 JSR &FFEE
130 LDA #0
135 JSR &FFEE
140 LDA #100
145 JSR &FFEE
150 RST
160 ]:NEXT I%
170 CALL &C00
```

ne OSWORD relativa all'immissione di una stringa:

Blocco di controllo per OSWORD	
Elemento	**Funzione**
0	Parte bassa dell'indirizzo di memoria ove depositare i caratteri letti
1	Parte alta dell'indirizzo di memoria ove depositare i caratteri letti
2	Lunghezza massima della stringa
3	Minimo codice ASCII accettabile
4	Massimo codice ASCII accettabile

Durante l'immissione dei caratteri, il tasto Delete opera regolarmente per la cancellazione di caratteri.
La routine termina se viene premuto Escape o Return. Per fare un esempio, questo blocco di controllo:

Esempio di blocco di controllo OSWORD	
Elemento	**Valore**
0	&00
1	&0C
2	&07
3	32
4	96

fissa le seguenti caratteristiche:

1. Il primo carattere immesso viene depositato nella locazione &C00, il secondo in &C01 e così via.
2. Vengono accettati soltanto 7 caratteri, provocando un "bip" ad ogni ulteriore tentativo d'immissione.
3. Vengono accettati soltanto i caratteri il cui codice è compreso tra 32 (=spazio) e 96 (= simbolo della sterlina): gli altri vengono ignorati.

La routine, dunque, offre la comoda possibilità di filtrare i caratteri indesiderati. Al suo ritorno, tramite il flag di carry, la routine comunica che ha terminato l'immissione: se carry=1 è stato premuto il tasto Escape, se carry=0 è stato usato il normale Return. Il registro Y contiene la lunghezza della stringa immessa, ritorno carrello finale incluso. Si ricordi che tale routine può essere adoperata anche sul flusso d'ingresso alternativo.
Avendo considerato l'immissione dei caratteri, adesso è la volta delle routine per la loro emissione. Ancora una volta, la prima cosa da fare è la selezione del flusso, stavolta in uscita.
Il comando è *FX3,n (dove n specifica il flusso). *FX3,1, ad esempio, seleziona l'uscita seriale, quella del video e quella della stampante (consentendo anche l'uso del comando *SPOOL).

Tabella dei parametri di controllo per flussi in uscita						
Bit / Valore	0	1	2	3	4	6
1	RS423 ON	Video OFF	Stampante OFF	Stampante ON*	'Spool' OFF	Stampante OFF**
0	OFF	ON	ON	OFF*	ON	ON**

* La stampante è abilitata sempre
** La stampante è disabilitata, salvo il carattere sia preceduto da un CHR$(1)

La routine principale, per l'invio di caratteri al flusso in uscita, è chiamata OSWRCH è il suo indirizzo è &FFEE, vettorizzato in &20E e &20F. L'uso è molto semplice: basta caricare A con il codice ASCII del carattere da emettere e poi chiamare la routine. Tutte le seguenti combinazioni visualizzano una A sullo schermo:

```
1000 VDU 65

1000 PRINT CHR$(65)

1000 LDA #65
1010 JRS &FFEE
```

Il comando VDU del BASIC è virtualmente identico alla chiamata OSWRCH. Tutti i codici ASCII compresi tra 32 a 255 visualizzano qualcosa sullo schermo, fatta eccezione per il codice 127, che corrisponde al carattere di Delete (cancellazione). I codici da 0 a 31, detti "di controllo", hanno un particolare significato e permettono di usare la OSWRCH per generare grafica su schermo, per eseguire i comandi COLOUR e GCOL, per definire caratteri e controllare il chip video 6845 del BBC.
L'emissione di caratteri (sullo schermo o altrove) tramite la routine OSWRCH viene spesso definita un'operazione di scrittura sui driver VDU. La tabella dei codici di controllo ASCII riportata più avanti riassume gli effetti dei codici compresi tra 0 e 31, quando sono usati dalla OSWRCH.

La grafica tramite OSWRCH

Tutti i normali comandi grafici sono eseguibili anche tramite la OSWRCH. Il programma dimostrativo nel riquadro a pagina precedente traccia una riga rossa sullo schermo. Le linee 50-75 corrispondono al comando GCOL 0,1 col quale si seleziona il colore rosso per la riga. Le linee 90-150 eseguono un comando PLOT 5,100,100 (equivalente a DRAW 100,100). la linea 100 invia ai driver VDU il tipo di PLOT (5 in questo caso), seguito da due byte per la coordinata x e da altri due per la coordinata y (byte basso per primo). Sostituendo il valore 5 con un 4 si ottiene il comando MOVE (poiché esso equivale a PLOT 4,x,y). Anche altre funzioni grafiche, quale la PLOT 85 che traccia triangoli, sono accessibili con questo criterio. È importante rammentare, nell'invio di comandi PLOT, che occorre rispettare il numero e la lunghez-

za dei parametri previsti (5 byte in tutto, dopo il codice 25 inviato inizialmente).
La definizione di nuovi caratteri avviene tramite il comando VDU 23. Ad esempio:

```
VDU23,224,255,255,255,255,255,255,255,255
```

ridefinisce il carattere con codice 224, normalmente definito come un quadratino pieno. Questo comando BASIC consente di ridefinire, nei MODE da 0 a 6, i caratteri con codice da 224 a 255. Tuttavia, usato in congiunzione a una chiamata OSBYTE, si può estendere l'arco dei caratteri ridefinibili a tutta l'intera gamma. Infatti, tutte le chiamate VDU23 non riconosciute dal SO vengono fatte passare dal vettore collocato in &226 e &227. Modificando l'indirizzo di questo vettore, l'utente può ampliare l'uso del comando VDU23.
Un particolare impiego del comando VDU23 è quello che consente di operare sul chip video 6845. Il comando segue il formato:

```
VDU23,0,registro,valore,0,0,0,0,0,0
```

dove "registro" corrisponde al registro del 6845 nel quale si desidera scrivere il numero indicato da "valore". Come esempio pratico di questo comando, il seguente programma modifica i due registri del chip 6845 che fissano l'indirizzo della parte di RAM da adibire a memoria video. Il programma porta tale indirizzo a &0000, visualizzando così l'area di lavoro del SO. Si provi, a questo punto, ad aggiungere qualche linea di programma oppure a dimensionare una matrice. La sequenza di comandi è facilmente convertibile in codice macchina:

```
10 MODE 0
20 VDU23,0,12,0,0,0,0,0,0
30 VDU23,0,13,0,0,0,0,0,0
40 VDU28,0,10,30,0:REM definisce una finestra di testo
50 CLS
```

Esistono altre due chiamate al SO simili ad OSWRCH: OSNEWL e OSASCII. La prima (indirizzo &FFE7) emette sullo schermo la combinazione "ritorno carrello/avanzamento linea", mentre la seconda (indirizzo &FEE3) è una variante di OSWRCH, utile nella gestione di testi. Quando il codice del carattere inviato allo schermo tramite questa routine è il 13 (ritorno carrello) la routine aggiunge automaticamente un "line feed" (avanzamento di linea). Questa chiamata, pertanto, non va adoperata se si tratta di inviare immagini grafiche ai driver VDU, in quanto l'emissione automatica del codice 10 può compromettere la regolarità del disegno.
Per concludere, anche *SPOOL ed *EXEC sono due comandi che si riferiscono ad operazioni su flussi di ingresso/uscita. *EXEC nomefile provoca l'apertura del file citato (ovviamente solo se questo esiste già) e ne legge il contenuto come se si trattasse di un'immissione da tastiera. *SPOOL, invece, esegue l'operazione inversa: scrive caratteri nel file specificato, trattando questo come flusso di uscita.

TABELLA CODICI DI CONTROLLO ASCII

Codice	Ulteriori byte occorrenti	Descrizione
0	0	(nessun effetto)
1	1	invia il successivo carattere solamente alla stampante
2	0	abilita la stampante
3	0	disabilita la stampante
4	0	emette caratteri usando il cursore testuale
5	0	emette caratteri usando il cursore grafico
6	0	consente ai driver VDU di scrivere caratteri nel flusso in uscita
7	0	genera l'emissione di un breve suono
8	0	muove il cursore a sinistra di un posto
9	0	muove il cursore a destra di un posto
10	0	muove il cursore in basso di un posto
11	0	muove il cursore in alto di un posto
12	0	ripulisce l'area testuale dello schermo
13	0	posiziona il cursore all'inizio della riga corrente
14	0	abilitazione modo "impaginato"
15	0	disabilitazione modo "impaginato"
16	0	ripulisce l'area grafica dello schermo
17	1	seleziona il colore del testo in base al valore del successivo byte
18	2	forma un comando GCOL con i successivi due byte. Ad esempio 18,0,3 equivale al comando GCOL 0,3
19	5	definizione dei colori logici (vedere manuale BBC)
20	0	ristabilisce i colori logici originali
21	0	disabilita la scrittura di caratteri sui flussi in uscita da parte dei driver VDU
22	1	seleziona il MODE dello schermo in base al byte seguente. La sequenza 22 7, ad esempio, seleziona il MODE 7. HIMEM non viene alterato
23	9	invio di comandi al chip video 6845. Consente la ridefinizione di tutti i caratteri
24	8	definisce una finestra grafica
25	5	esegue il comando PLOT
26	0	ripristina le finestre testuale e grafica originali
27	0	(nessun effetto)
28	4	definisce una finestra testuale
29	4	fissa l'origine per la grafica
30	0	posiziona il cursore in alto a destra sullo schermo
31	2	posiziona il cursore in base ai due byte successivi. La sequenza 31,10,10, ad esempio, posiziona il cursore nella cella 10,10 dello schermo

Telematica per tutti

Un'applicazione di crescente rilievo: l'intercomunicazione fra più computer

Collegando il proprio computer a un modem, si schiudono orizzonti vastissimi per quanto concerne la comunicazione con altri possessori di computer o con banche dati appositamente progettate. In questo modo si rendono possibili tante applicazioni fra le quali troviamo ad esempio la posta elettronica (recapitata istantaneamente), lo scambio di software, il collegamento con grossi centri di elaborazione.
Per prima cosa, però, occorre chiarire i principi che rendono possibile la trasmissione di dati tra computer.
Come è facile intuire, le basi della tecnologia delle comunicazioni si sono formate, prima dell'avvento dei micro e dei personal computer, su elaboratori "mainframe". Agli inizi, la configurazione tipica prevedeva l'impiego di un grosso computer esclusivamente impiegato per lo smistamento delle comunicazioni, collocato in un ambiente a temperatura e umidità costanti.
Da questo computer si diramavano tutti i cavi ai quali venivano allacciati i vari terminali (schermo più tastiera), situati in più parti di un edificio.
La trasmissione seriale via cavo, però, non consente l'utilizzazione di terminali posti a più di duecento metri circa dal computer principale: su distanze maggiori i segnali si deteriorano ed impediscono il corretto funzionamento del sistema.
I modem sono stati realizzati per superare tali impedimenti.
Un MODEM (da MOdulatore/DEModulatore) è un apparecchio che consente di inviare dati computerizzati, opportunamente convertiti in segnali di frequenze udibili, lungo le normali linee telefoniche.
Questo processo prende il nome di modula-



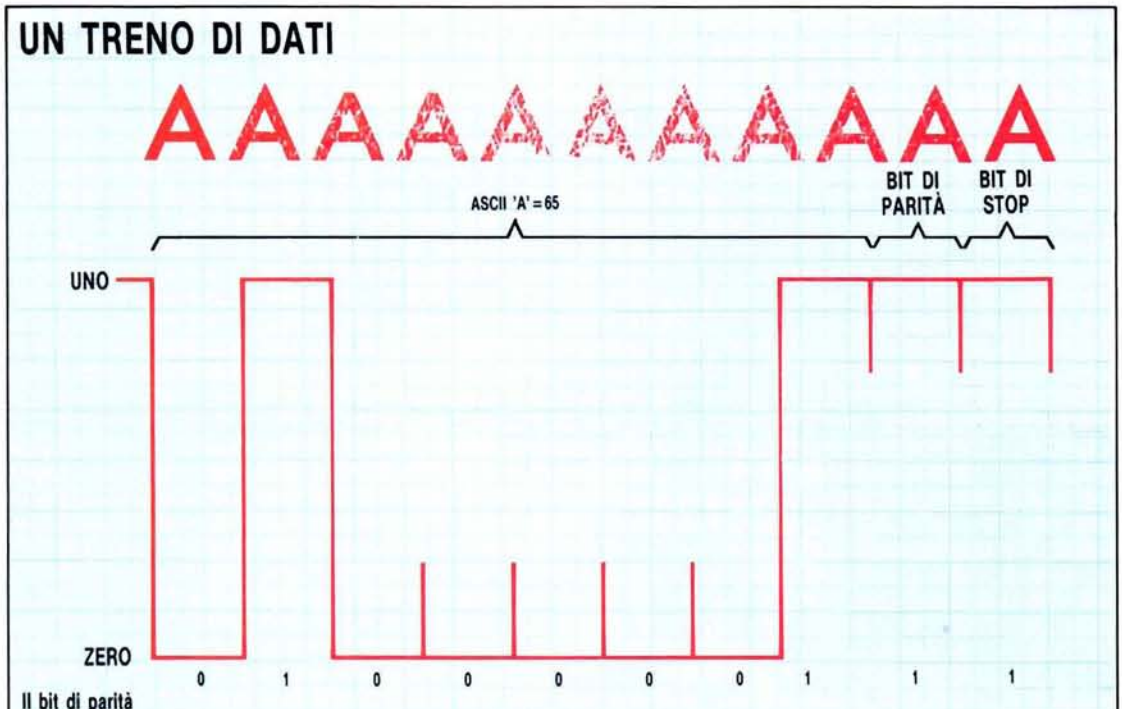
Un tono di comando

Originariamente concepiti per le comunicazioni tra grossi centri di elaborazione, i modem convertono i segnali digitali in frequenze udibili (e pertanto utilizzabili sulle normali linee telefoniche). In trasmissione il modem "modula" un segnale detto "portante" in base ai valori binari da inviare. In ricezione l'apparecchio "demodula" il segnale per riconvertirlo in formato binario, prima di passarlo al computer ricevente. Nei modem "acustici" il collegamento avviene senza alcun intervento sull'impianto elettrico del telefono: basta appoggiare la cornetta in appositi ricettacoli di gomma. Nei modem "diretti", invece, il collegamento è di tipo elettrico, tramite una speciale presa o scatola di derivazione

zione ed è complementare a quello di demodulazione, operato dal modem situato all'altro capo della linea telefonica e collegato al computer ricevente.
Per rendere possibile tale comunicazione è necessario impiegare un segnale costante di riferimento, detto "portante" (il "carrier tone").
L'impiego di modem permette quindi l'accesso ad un computer principale, qualunque sia la distanza tra esso e il terminale, in modo pressoché istantaneo.
Anche in questo caso i terminali possono essere del tipo "non intelligente" (un semplice schermo dotato di tastiera), ma è sempre più diffuso l'impiego di microcomputer, usati normalmente come piccole unità indipendenti di elaborazione che, collegate all'occorrenza al computer principale, utilizzano software per comunicazioni.
Questo secondo tipo di terminale viene spesso definito "intelligente", per distinguerlo dagli altri che non sono dotati di capacità elaborative proprie.

Tipi di modem

Esistono due tipi di modem: quelli "acustici" e quelli "a collegamento diretto". Gli accoppiatori acustici sono due ricettacoli gomma costruiti in modo da potervi appoggiare la cornetta del telefono, che provvedono a garantire una certa stabilità e protezione dai rumori ambientali. Essendo questo un collegamento "volante", sono stati realizzati anche degli accoppiatori acustici a pile, da poter usare perfino da un telefono pubblico, assieme ai sempre più diffusi computer portatili.
Questo tipo di modem, poco costoso, ma non è però del tutto affidabile, poiché il rumore ambientale può talvolta interferire con i dati trasmessi.
Il secondo tipo di modem, invece, viene collegato direttamente alla linea telefonica, per mezzo di un apposito spinotto o scatola di derivazione.
Questi modem, oltre ad essere molto più affidabili dei primi, offrono in genere molte più possibilità d'impiego.
La SIP, in Italia, offre ai propri utenti la possibilità di noleggiare alcuni modelli di modem ma, poiché si tratta di intervenire sull'impianto del telefono, occorre comunque una speciale autorizzazione della compagnia telefonica per adoperare modem a collegamento diretto acquistati altrove, che naturalmente, devono essere conformi a quanto prescritto dalla SIP (in altre parole devono essere di tipo "approvato").
I modem diretti offrono spesso alcune comode opzioni, quali l'"auto-answering" e l'"auto-dialling". I modem dotati della prima opzione rispondono automaticamente al telefono e, qualora venga individuata la portante di un modem sulla linea, passano il controllo al computer cui sono collegati, altrimenti... riattaccano!
L'opzione "auto-dialling" consiste semplice-



LIZ DIXON

Il bit di parità

I dati vengono inviati sulla linea telefonica come sequenza binaria di livelli "alti" e "bassi", rispetto al segnale della portante. La sequenza corrisponde al formato binario del codice ASCII del carattere trasmesso. Per controllare l'esattezza dei dati, vengono adoperati alcuni bit aggiuntivi: quello di parità, quello di start e quello di stop. Nell'esempio, viene illustrato l'invio del carattere A (ASCII 65, ossia 01000001 in binario) con controllo di parità "dispari" e un solo bit di stop. Poiché il numero di 1 nel dato è pari (sono solo 2) il bit di parità va posto a 1 per produrre un risultato dispari. L'ultimo bit è quello di stop, per indicare il termine del dato. La trasmissione di un dato di 8 bit, in questo caso, implica l'uso di 10 bit in tutto

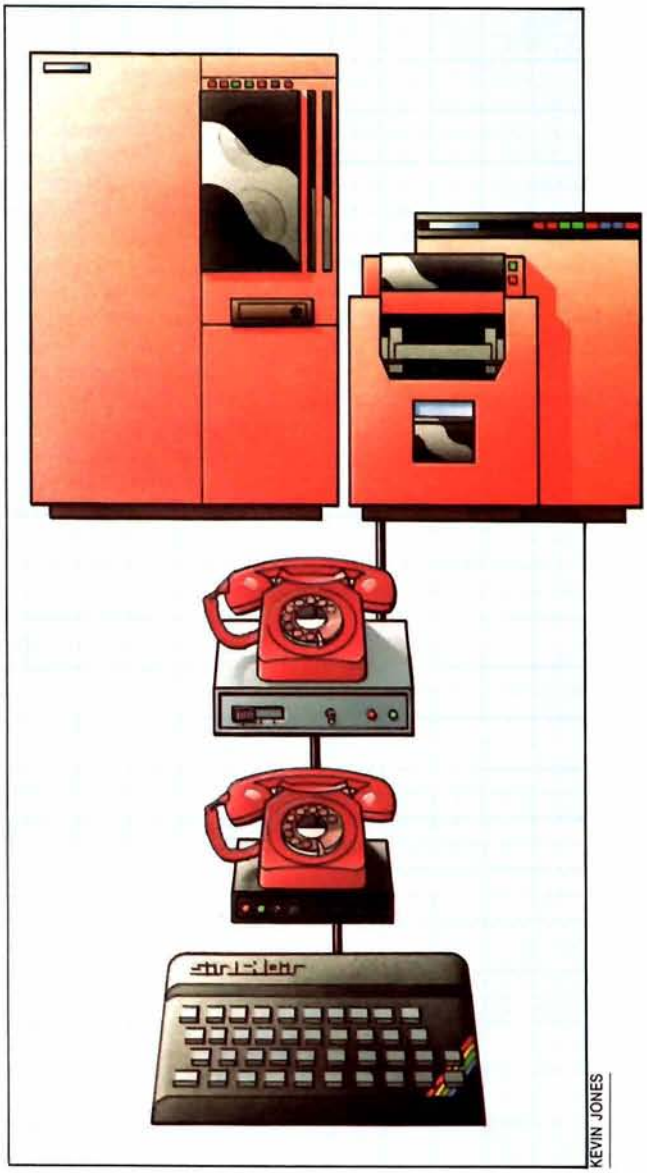
mente nel far formare un numero telefonico direttamente al computer.
Per misurare la velocità impiegata per trasmettere i dati si usa un'unità di misura detta "Baud".
Comunemente si crede che il numero di Baud corrisponda al numero di bit trasmessi al secondo, ma ciò non è del tutto preciso (vedere oltre).
Le velocità più usate sono due: 300 e 1200/75 Baud. Quest'ultima (con 1200 Baud in ricezione e 75 in trasmissione) è usata spesso quando la quantità di dati inviati in una direzione supera di molto quelli inviati in direzione opposta.
Detto per inciso, velocità simili sono comuni anche nella registrazione di dati su nastri a cassetta.
Come al solito, non esiste un unico standard nella frequenza usata per le trasmissioni: i due principali sistemi sono comunque il CCITT (o V21) usato in Europa e il Bell, usato negli Stati Uniti.

Tutto sui bit

I dati vengono trasmessi sotto forma di segnali binari. Pertanto, ogni carattere viene prima convertito nel suo equivalente ASCII e quindi in formato binario. La lettera A, per esempio, viene prima convertita in 65 (ASCII) e poi in 01000001 (binario). Ad ogni 1 corrisponde una frequenza "alta", rispetto alla portante, mentre ad ogni 0 una frequenza "bassa".
Poiché il computer ricevente deve in qualche modo sapere quando inizia o quando termina un carattere trasmesso serialmente, vengono anche usati degli speciali bit aggiuntivi, detti di "start" (inizio) e di "stop" (fine).
Il "protocollo" (così viene chiamato il formato dei dati trasmessi) più comune sui micro è di otto bit di dati seguiti da un bit di stop oppure, in alternativa, sette bit di dati e due di stop.
Inoltre, poiché una trasmissione può essere disturbata da fattori esterni, occorre anche un semplice sistema per controllare l'esattezza dei dati ricevuti. In genere si usa un "controllo di parità", che consiste nel contare tutti gli 1 presenti nel dato, sommandovi o meno un bit aggiuntivo (detto appunto "di parità") per ottenere un risultato pari ("parity even") oppure dispari ("parity odd").
Tornando all'esempio della lettera A: il numero totale di bit "alti" in 01000001 è due (pari). Volendo usare la convenzione parity odd, è necessario porre a 1 il bit di parità. Nella lettera C, invece, la sequenza 01000011 implica un bit di parità a zero.
Questo metodo di controllo è molto rudimentale: può intercettare la presenza di un bit sbagliato, ma non può segnalare di quale si tratti.
Inoltre, se i bit errati sono due, il sistema spesso non funziona.
Si è detto in precedenza che il numero di Baud non indica con esattezza il numero di bit di dati trasmessi al secondo: bisogna infatti, tenere conto della presenza, nel protocollo usato, dei bit di controllo appena descritti.
Supponendo che in un sistema venga adottato il controllo di parità e venga usato il solo bit di stop, su 10 bit trasmessi solo otto rappresentano dati effettivi, quindi la velocità effettiva, per esser precisi, è 240 Baud (ossia 80% di 300 Baud).

FLUSSO DI INFORMAZIONI

La velocità di trasmissione dei dati viene espressa in Baud (dal nome di J.M.E. Baudot, il pioniere francese delle telecomunicazioni) per indicare il numero di bit inviati al secondo.
Negli attuali sistemi di intercomunicazione tra computer la velocità più usata è di 300 Baud.
Se però la quantità di dati che fluiscono in una direzione è molto superiore rispetto a quella dei dati in direzione opposta, viene talvolta impiegato il sistema misto 1200/75 (1200 Baud in ricezione e 75 in trasmissione).
Queste velocità appartengono alla stessa gamma impiegata nella registrazione di dati su nastri a cassette


KEVIN JONES

Avventura grafica

Come arricchire con schermi di grafica il gioco d'avventura

Il Micro BBC offre una serie di funzioni grafiche con le quali è facile creare immagini per i giochi. Il programma *Digitaya*, sviluppato in precedenza, è un gioco basato su testi: viene usata una serie di frasi per illustrare gli scenari nei quali il giocatore si muove. Ma, giostrando adeguatamente l'occupazione di memoria, invece di usare una frase del tipo: «Ti trovi nella Sala del Trono», per descrivere un particolare ambiente, è possibile disegnare una stanza al cui interno si trova un trono.
Le immagini che ci accingiamo a progettare riproducono due importanti locazioni del programma *Digitaya*, ossia l'accesso alla porta del joystick e l'Unità Logico-Aritmetica (ULA). Il numero di scenari realizzabili è ovviamente limitato dalla quantità di memoria a disposizione: talvolta realizzare immagini efficaci significa occupare memoria che potrebbe essere altrimenti usata per accrescere la complessità del gioco stesso.

Progettazione dell'immagine ALU

Prima di passare alla creazione di un'immagine per il Micro BBC, occorre stabilire una serie di importanti premesse:

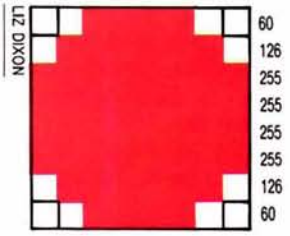
1) Quanta memoria è disponibile?
2) Quanti colori si intendono usare?
3) Quale risoluzione grafica impiegare?

In effetti, sul BBC, tutti questi interrogativi sono contenuti in una sola domanda: «Quale modo intendiamo usare ?». L'impiego di una risoluzione più alta e di un maggior numero di colori porta l'area video ad occupare una quantità superiore di RAM. Il nostro progetto usa il MODE 1, che prevede 4 colori, un formato dello schermo di 40 x 25 caratteri ed una risoluzione media. Per selezionare il modo, occorre inserire all'inizio del programma la seguente linea:

```
1095 MODE 1
```

Scelto il modo, possiamo tracciare un primo abbozzo dell'immagine che intendiamo creare, fissando su carta le relative coordinate. Il progetto della routine prevede lo scorrimento verticale sullo schermo delle lettere maiuscole A, L ed U. Durante il gioco, il giocatore preme uno dei tre pulsanti AND, OR e NOT che devono quindi comparire sullo schermo assieme ad altri particolari dell'immagine, quali il bordo dello schermo e lo sfondo di scorrimento. Le tre lettere, formate da una serie di linee parallele, vengono tracciate con il comando PLOT 1 a partire dal punto in cui esse iniziano a muoversi. In questo modo è possibile farle scorrere semplicemente modificando il comando iniziale MOVE. Per cancellare le lettere dallo schermo è sufficiente visualizzarle nella stessa posizione, specificando però una procedura di tracciamento OR-Esclusiva con il comando GCOL 3.
L'immagine dei pulsanti viene creata ridefinendo il carattere CHR$(240) con la procedura pulsante, ottenendo così la sagoma mostrata a fianco. Si noti che CHR$(240) viene assegnato alla variabile pulsante$, usata nella sezione principale della routine. Scritte e pulsanti vengono riprodotti sullo schermo in base alle coordinate indicate nel comando TAB.
Per creare le figure in primo piano vengono adoperate le primitive di colorazione triangolare, fornite dal comando PLOT 85: questo permette di congiungere il punto specificato agli ultimi due punti tracciati e di colorare il triangolo così ottenuto. La figura rettangolare che fa da sfondo può essere disegnata e colorata usando due primitive di questo tipo.
Il codice per la visualizzazione dell'immagine è contenuto in una speciale subroutine della routine che controlla la locazione ALU durante il gioco. Il comando A$=GET$ (linea 7560) attende che il giocatore prema un tasto prima di ripristinare il colore originale di primo piano, cancellare lo schermo e restituire il controllo (Return) alla routine principale dell'immagine ALU per la prosecuzione del gioco. La subroutine grafica viene chiamata dalla seguente linea del programma principale:

```
4565 GOSUB 7000: REM S/R IMMAGINE ALU
```

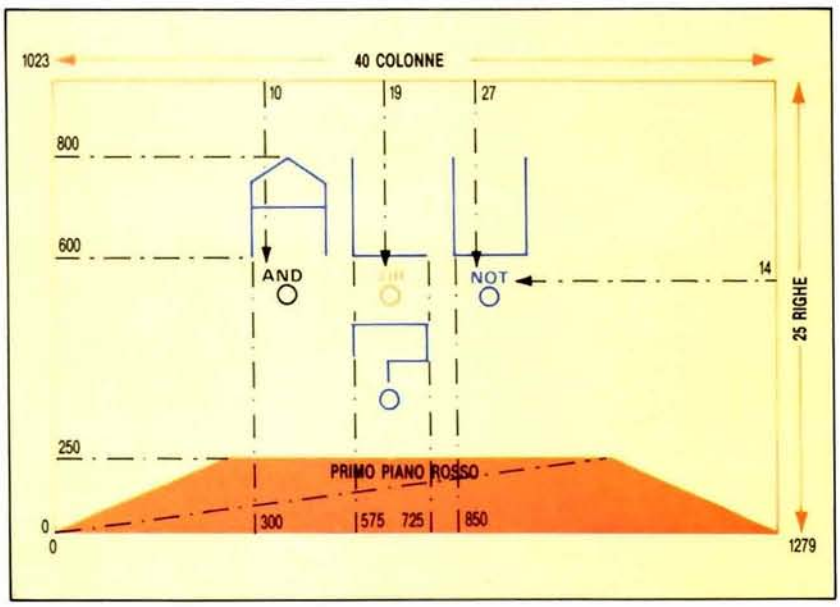


Il pulsante

Nell'immagine ALU per il Micro BBC, a ciascuna scelta viene associato un pulsante. In mancanza del comando CIRCLE o di altri caratteri adatti, è necessario ridefinire un carattere BBC. In questo caso si è modificato il carattere CHR$ (240), usando il comando VDU 23 ed 8 numeri decimali per creare l'immagine del pulsante su di una griglia 8x8

Modalità d'uso

Per scrivere un programma per il BBC, si devono fare delle scelte: l'alta risoluzione consuma una discreta quantità di memoria ed offre una ridotta gamma di colori, mentre i modi "testuali" usano meno memoria e consentono l'uso di più colori ma possiedono solo grafica a media o bassa risoluzione. Il MODE 1 procura la risoluzione necessaria, ma occupa ben 20 Kbyte di memoria video



Nel gioco *Digitaya*, quando il giocatore raggiunge la locazione della porta del joystick, corre il rischio di essere colpito da un raggio laser. La subroutine grafica per questa immagine deve quindi tracciare una porta del joystick dal cui centro origina un raggio laser. La porta del joystick compare nell'angolo superiore destro dello schermo: per creare i contatti interni sono state usate due istruzioni PRINT seguite da una serie di caratteri "punto", mentre il caratteristico profilo "tipo D" viene disegnato in alta risoluzione con istruzioni PLOT. Queste seguono un comando MOVE, che indica la posizione iniziale, e sono tutte istruzioni PLOT 1, ossia comandi di tracciamento "relativo". Ciò consente di cambiare la posizione della porta del joystick modificando solo la prima istruzione MOVE.

Il primo piano dell'immagine è composto da un rettangolo colorato, disegnato usando due primitive di riempimento triangolare. Per dare un senso di profondità, sul rettangolo viene tracciata una serie di linee convergenti tramite un ciclo FOR...NEXT (linee 8170-8200). Il ciclo seleziona valori di X compresi tra 0 e 1280, ossia la larghezza dello schermo espressa in unità grafiche. Le linee originano da punti successivi dell'orizzonte, la cui distanza aumenta al crescere di X, e terminano sul margine inferiore dello schermo. La distanza tra due linee successive, che sul margine inferiore dello schermo è pari a 32 unità grafiche, decresce a 4 sulla linea dell'orizzonte (perché ciascun valore di X viene diviso per 8 nel comando MOVE che calcola il punto di origine di ogni linea).

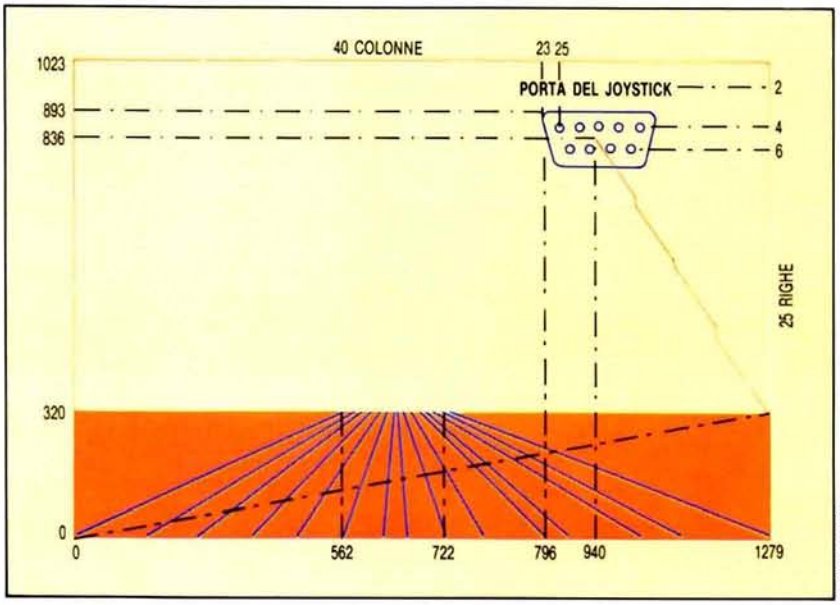
Il raggio laser è rappresentato da una linea colorata in modo casuale, che parte dal centro della porta del joystick e termina sull'orizzonte. La linea viene poi cancellata, senza danneggiare lo sfondo, ritracciandola nel modo OR Esclusivo, selezionato dal comando GCOL 3. Il tracciamento e la cancellazione vengono eseguiti all'interno di un ciclo REPEAT...UNTIL, assieme ad un test che verifica se è stato premuto un tasto. L'impiego di INKEY$ al posto di GET$ consente di non interrompere l'esecuzione del programma durante la verifica. Quando il giocatore preme un tasto, il ciclo si conclude, viene cancellata l'immagine e ripristinato il colore originale per i testi e il controllo torna alla routine della porta del joystick.

Per chiamare la subroutine grafica inserire la seguente linea:

```
3845 GOSUB 8000: REM IMMAGINE PORTA PER JOYSTICK
```

Tecniche di tracciamento

Per entrambe le immagini è stata usata una procedura di tracciamento relativo, che facilita la cancellazione e lo spostamento di intere figure. Per tracciare e colorare blocchi in alta risoluzione si è ricorsi ad una diversa funzione di tracciamento



KEVIN JONES

SCENARIO ALU

```
7000 REM **** S/R SCENARIO ALU ****
7010 CLS
7015 REM ** CURSORE SPENTO **
7017 VDU23,1,0;0;0;0;
7020 REM ** BORDO **
7030 GCOL 0,1
7040 MOVE 0,0
7050 DRAW 0,1023
7060 DRAW 1279,1023
7070 DRAW 1279,0
7080 DRAW 0,0
7090 :
7100 REM ** SENTIERO **
7110 MOVE 1279,0
7120 PLOT 85,500,250
7130 PLOT 85,800,250
7140 REM ** LETTERA A **
7150 GCOL 3,2
7160 FORI=0 TO 300 STEP 10
7170 FOR I=1 TO 2
7180 PROCletteraA
7190 NEXT I,X
7200 PROCletteraA
7210 :
7220 REM ** LETTERA L **
7230 FOR Y=300 TO 590 STEP 10
7240 FOR I=1 TO 2
7250 PROCletteraL
7260 NEXT I,Y
7270 PROCletteraL
7280 :
7290 REM ** LETTERA U **
7300 FOR X=1280 TO 850  STEP -10
7310 FOR I=1 TO 2
7320 PROCletteraU
7330 NEXT I,X
7340 PROCletteraU
7350 :
7360 REM ** PULSANTI **
7370 PROCpulsanti
7380 COLOUR3
7390 PRINT TAB (11,15) pulsanti$
7400 COLOUR1
7410 PRINT TAB (20,15) pulsanti$
7420 COLOUR2
7430 PRINT TAB (28,15) pulsanti$
7440 REM ** COMANDI **
7450 COLOUR3
7460 PRINT TAB(10,14)"AND"
7470 COLOUR1
7480 PRINT TAB(19,14)"OR"
7490 COLOUR2
7500 PRINT TAB(27,14)"NOT"
7510 :
7520 MOVE 575,400
7530 PROCpuntointerr
7540 :
7550 REM ** ATTESA PER BATTUTA TASTO **
7560 A$=GET$
7562 REM ** CURSORE ACCESO **
7564 VDU23,1,1;0;0;0;
7566 COLOUR3:CLS
7570 RETURN
7580:
7590 DEF PROCletteraA
7600 MOVE X,600
7610 PLOT 1,0,150
7620 PLOT 1,75,50
7630 PLOT 1,75,-50
7640 PLOT 1,0,-150
7650 PLOT 0,0,80
7660 PLOT 1,-150,0
7670 ENDPROC
7680 :
7690 DEF PROCletteraL
7700 MOVE 725,Y
7710 PLOT 1,-150,0
7720 PLOT 1,0,200
7730 ENDPROC
7740 :
7750 DEF PROCletteraU
7760 MOVE X,800
7820 DEF PROCpulsanti
7830 VDU23,240,60,126,255,255,255,255,126,60
7840 pulsanti$=CHR$(240)
7850 ENDPROC
7860 :
7870 DEF PROCpuntointerr
7880 PLOT 1,0,60
7890 PLOT 1,150,0
7900 PLOT 1,0,-70
7910 PLOT 1,-75,0
7920 PLOT 1,0,-50
7930 PLOT 0,-8,-30
7940 PLOT 1,16,0
7950 PLOT 1,0,-16
7960 PLOT 1,-16,0
7970 PLOT 1,0,16
7980 ENDPROC
8000 REM **** IMMAGINE PORTA DEL JOYSTICK ****
8010 :
8020 REM ** CURSORE SPENTO **
8030 VDU23,1,0;0;0;0;
8040 CLS
8050 REM ** BORDO **
8060 GCOL 0,1
8070 MOVE 0,0
8080 DRAW 0,1023
8090 DRAW 1279,1023
8100 DRAW 1279,0
8110 DRAW 0,0
8120 :
8130 REM ** ORIZZONTE **
8140 PLOT 85,1279,319
8150 PLOT 85,0,319
8160 GCOL,2
8170 FOR X=0 TO 1280 STEP 32
8180 MOVE 562+X/8,320
8190 DRAW X,0
8200 NEXT X
8210 :
8220 REM ** JOYSTICK **
8230 COLOUR 2
8240 PRINT TAB (23,2) "PORTA JOYSTICK"
8250 PRINT TAB (25,4) ". . . . ."
8260 PRINT TAB (25,6) ". . . . ."
8270 GCOL 0,2
8280 MOVE 796,893
8290 PLOT 1,288,0
8300 PLOT 1,4,-4
8310 PLOT 1,4,-4
8320 PLOT 1,0,-4
8330 PLOT 1,-4,-4
8340 PLOT 1,-30,-81
8350 PLOT 1,-4,-4
8360 PLOT 1,-4,-4
8370 PLOT 1,-218,0
8380 PLOT 1,-4,4
8390 PLOT 1,-4,4
8400 PLOT 1,-30,81
8410 PLOT 1,-4,4
8420 PLOT 1,0,4
8430 PLOT 1,4,4
8440 :
8450 REM ** RAGGIO LASER **
8460 REPEAT
8470 A$=INKEY$(10)
8480 X=RND (1279):Y=320
8490 GCOL 3,RND(3)
8500 FOR I=1 TO 2
8510 MOVE 940,836
8520 DRAW X,Y
8530 NEXT I
8540 UNTIL A$<>"": REM ATTESA PER BATTUTA TASTO
8550 :
8560 REM ** CURSORE ACCESO **
8570 VDU23,1,1;0;0;0;
8575 COLOUR 3:CLS
```

Software e didattica

Sfruttiamo nella produzione di software per l'infanzia le esperienze compiute all'estero

Il fine didattico di tutti i package qui esaminati va dal semplice riconoscimento di colori e forme a sofisticati tentativi per ampliare le capacità artistiche dei giovanissimi utenti, senza dimenticare le basi dell'aritmetica e della lettura.
Tutti i package sono piuttosto chiari nell'esporre cosa l'operatore debba fare per conseguire i risultati migliori: questa è una caratteristica di primaria importanza in un package didattico, poiché il bambino deve sapere esattamente cosa gli viene richiesto. Inoltre, deve esistere una chiara indicazione del risultato raggiunto, eventualmente abbinato ad immaginarie "ricompense", se le risposte o i risultati sono corretti.
In secondo luogo, i programmi devono essere facili da usare: è del tutto inutile se un package rivolto all'insegnamento della lettura inizia con lunghe paginate di testo. Nei programmi migliori le istruzioni sono ridotte al minimo essenziale.
Un programma didattico deve inoltre catturare l'interesse del bambino: qualunque siano i pregi didattici del package, se esso richiede capacità non commisurate a quelle del bambino o se diviene noioso e ripetitivo il suo valore è praticamente nullo.
Infine, la "prova del fuoco" per un programma è che raggiunga effettivamente lo scopo educativo prefissato.
Può sembrare superfluo citare questa caratteristica, ma purtroppo la pratica insegna che troppo spesso le software house privilegiano gli aspetti ricreativi a scapito di quelli didattici.
I package qui esaminati sono tutti prodotti negli Stati Uniti, dalla Spinnaker e dalla Fisher-Price. Sul mercato americano i programmi sono disponibili su cartucce (più facili da usare per un bambino), mentre in Europa essi vengono distribuiti su cassetta, il che implica la necessaria presenza di un adulto per il loro utilizzo.
Il più attraente dei programmi esaminati è forse il *Dance Fantasy* della Fisher-Price, adatto a bambini tra i 4 e gli 8 anni. Lo schermo iniziale mostra un palcoscenico e varie figure di danza, che il bambino deve collocare sullo schermo per comporre una coreografia. Prima di iniziare, va scelto il sesso dei due ballerini.
In basso sullo schermo sono riprodotte le varie pose di danza. Il bambino, tramite joystick, seleziona la sequenza di movimenti desiderata, componendo via via un intero balletto. Il programma provvede ad aggiungere fluidità ai movimenti e, al termine del lavoro, l'intera sequenza può essere memorizzata su nastro. Ciò consente di riesaminare con continuità tutta la coreografia.
Lo sviluppo creativo di quello che il bambino considera un gioco, ha però delle affinità teoriche con i processi logici della programmazione.
Infatti, nel creare un balletto (il programma), il bambino adopera una serie di sequenze di base (le routine o procedure), sulle quali viene poi eseguita una SAVE o, successivamente, una LOAD. Ciò serve per familiarizzare con l'uso dei programmi e con la loro conservazione.
Aegean Voyage (Viaggio nell'Egeo), della Spinnaker, per bambini di età leggermente

Dance Fantasy

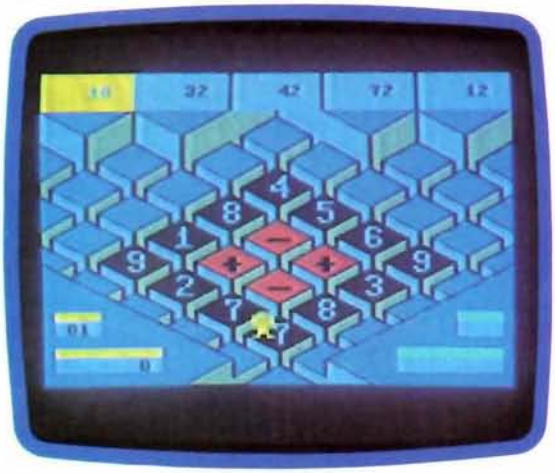


Aegean Voyage



superiore, ricorre a personaggi della mitologia greca per formulare una semplice avventura. L'obiettivo è quello di far viaggiare una nave sul mare Egeo, navigando tra scogli e burrasche. Quando si approda su un'isola, ne compare il nome e, in basso sullo schermo, viene visualizzato un criptico messaggio. A questo punto occorre decidere se esplorare l'isola o salpare per nuovi lidi.

Number Tumblers



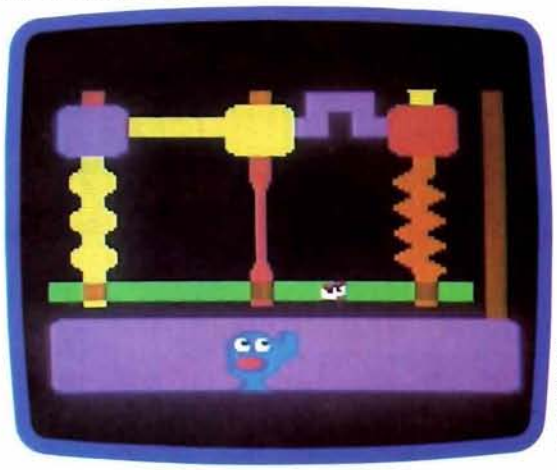
L'esplorazione può comportare ricompense oppure, nella peggiore delle ipotesi, l'affondamento della nave da parte di mitiche creature.
Tuttavia, vanno rilevate alcune imprecisioni introdotte dagli ideatori del programma: il Minotauro, ad esempio, può indifferentemente comparire a Creta o a Delo. Inoltre, il programma non tenta di spiegare il significato dei vari nomi mitologici adoperati, quindi non può certo essere considerato educativo per quanto riguarda una materia così classica.
La grafica (abbastanza ripetitiva) e l'intero sviluppo del gioco sono tali che raramente un bambino desidera giocare a lungo.
Number Tumblers (della Fisher-Price per bambini di età tra gli 8 e i 12 anni) è orientato allo sviluppo di capacità aritmetiche e possiede un ritmo da gioco "arcade". In alto sullo schermo compare una serie di numeri: il giocatore deve disporre numeri e segni aritmetici, su "dadi" collocati nella parte inferiore dello schermo, per formulare equazioni che diano per risultato uno dei numeri della serie.
Il gioco è piacevole, veloce, cattura l'interesse e, soprattutto, riesce effettivamente a stimolare il bambino nel collegare logicamente le varie componenti, migliorando così le sue capacità aritmetiche.
Kindercomp (della Spinnaker, età 3-8 anni) ha lo scopo di introdurre il concetto di computer e di sviluppare le qualità artistiche del bambino. Il package, originariamente creato dal dottor Doug Davis per la propria figlia, comprende un certo numero di esercizi diversi. Tuttavia, *Kindercomp* dà l'impressione di consistere soprattutto in trucchetti di programmazione (quelli, per intendersi, che ogni principiante sperimenta muovendo i primi passi in BASIC).
L'opzione Nomi, ad esempio, non fa che riprodurre la parola immessa (massimo 15 caratteri) in più posizioni, dimensioni e colori sullo schermo. L'effetto è divertente per un bambino, ma l'utilità didattica di un simile esercizio è discutibile, dato che non esiste alcun criterio nell'immissione delle parole (qualsiasi

Kindercomp



combinazione di lettere produce risultati sullo schermo).
Kindercomp non si presta ad un uso prolungato, poiché le varie opzioni diventano presto ripetitive e quindi l'interesse decresce rapidamente.
L'ultimo package esaminato è progettato per bambini molto piccoli: *Alf In The Color Caves* (della Spinnaker) è un gioco in cui il bambino, tramite joystick, muove un simpatico personaggio attraverso percorsi tortuosi e riccamente colorati.

Alf in the Color Caves



IAN McKINNELL

Se, alla fine, il personaggio riesce a sfuggire ai vari pericoli (tra i quali un paio di minacciosi occhi) esso improvvisa una curiosa danza di gioia.
Questi sono soltanto alcuni esempi di programmi didattici (o pseudo-didattici); altri, sempre indirizzati ad utenti giovanissimi, vengono esaminati in seguito.

DATABASE

Ecco la terza tabella sulle funzioni delle routine ROM del Micro BBC, assieme ai relativi indirizzi

DESCRIZIONE	INDIRIZZO
COS function routine routine funzione COS	**A98D**
COUNT function routine routine funzione COUNT	**AEF7**
Check FAC # 1 controllo FAC # 1	**A1DA**
Check for a 'Bad program' controllo per 'Bad program'	**BE6F**
Check for a comma at PTR # 2 controllo per virgola a PTR # 2	**8AAE**
Check for end of statement verifica fine istruzione	**9857**
Check if A is alphanumeric verifica se A è numerico	**8926**
Clear a new variable azzera una nuova variabile	**9531**
Clear stacks ripulisce gli stack	**BD3A**
Clear variables azzera le variabili	**BD20**
Cold start avvio del sistema a 'freddo'	**8ADD**
Compare two items confronto tra due elementi	**9A9E**
Convert a line number to binary converte numero di linea in binario	**889C**
Convert number to a string conversione da numero a stringa	**9EDF**
Convert string to a number conversione da stringa a numero	**A07B**
Create a PROC/FN catalogue entry creazione elemento PROC/FN nel catalogo	**94ED**
Create a catalogue entry creazione elemento nel catalogo	**94FC**
DATA statement routine routine istruzione DATA	**8B7D**
DEF statement routine routine istruzione DEF	**8B7D**
DEG function routine routine funzione DEG	**ABC2**
DELETE command routine routine comando DELETE	**8F31**
DIM statement routine routine istruzione DIM	**912F**
DIV operator routine routine operatore DIV	**9E0A**
DRAW statement routine routine istruzione DRAW	**93E8**
Decode hexadecimal number decodifica numero esadecimale	**AE6D**

INDIRIZZO	DESCRIZIONE
8F31	**DELETE command routine** routine comando DELETE
8FA3	**RENUMBER command routine** routine comando RENUMBER
90AC	**AUTO command routine** routine comando AUTO
912F	**DIM statement routine** routine istruzione DIM
9222	**Increment the 1AC** incrementa 1AC
9236	**Set (&2A) = (&2A)*(&3F)** pone (&2A) = (&2A)*(&3F)
925D	**HIMEM statement routine** routine istruzione HIMEM
926F	**LOMEM statement routine** routine istruzione LOMEM
9283	**PAGE statement routine** routine istruzione PAGE
928D	**CLEAR statement routine** routine istruzione CLEAR
9295	**TRACE statement routine** routine istruzione TRACE
92C9	**TIME statement routine** routine istruzione TIME
9304	**PROC statement routine** routine istruzione PROC
9323	**LOCAL statement routine** routine istruzione LOCAL
9356	**ENDPROC statement routine** routine istruzione ENDPROC
937A	**GCOL statement routine** routine istruzione GCOL
938E	**COLOUR statement routine** routine istruzione COLOUR
939A	**MODE statement routine** routine istruzione MODE
93E4	**MOVE statement routine** routine istruzione MOVE
93E8	**DRAW statement routine** routine istruzione DRAW
93F1	**PLOT statement routine** routine istruzione PLOT
942F	**VDU statement routine** routine istruzione VDU
945B	**Search for FN/PROC in catalogue** cerca FN/PROC nel catalogo
9469	**Search for name in catalogue** cerca un nome nel catalogo
94ED	**Create a PROC/FN catalogue entry** creazione elemento PROC/FN nel catalogo

Foto computerizzate

Le più moderne macchine fotografiche sono dotate di minuscoli computer

Prima di scattare un'istantanea, un fotografo deve scegliere la corretta "esposizione". Ciò implica giudicare attentamente quanta luce, proveniente dalla scena da fotografare, colpirà la pellicola: troppa luce "brucia" il negativo e l'immagine risultante sarà completamente bianca; se troppo poca produrrà invece una foto completamente nera. Per stabilire l'esatta esposizione, il fotografo può regolare due fattori: l'apertura del diaframma (che regola la quantità di luce) e la durata dell'esposizione.
Pertanto, occorre in primo luogo misurare la quantità di luce generata o riflessa dal soggetto da fotografare, poi tener conto della sensibilità della pellicola e regolare di conseguenza diaframma e tempo di esposizione. Finalmente (se il soggetto non si è stancato di attendere la messa a punto di tutte queste operazioni) si può scattare. Nel corso degli anni sono stati realizzati diversi tipi di esposimetri, per valutare la quantità di luce proveniente da una scena. In alcuni casi tali apparecchi venivano (e vengono tuttora) addirittura incorporati nella macchina fotografica, lasciando però al fotografo il compito di regolare diaframma e tempo d'esposizione.

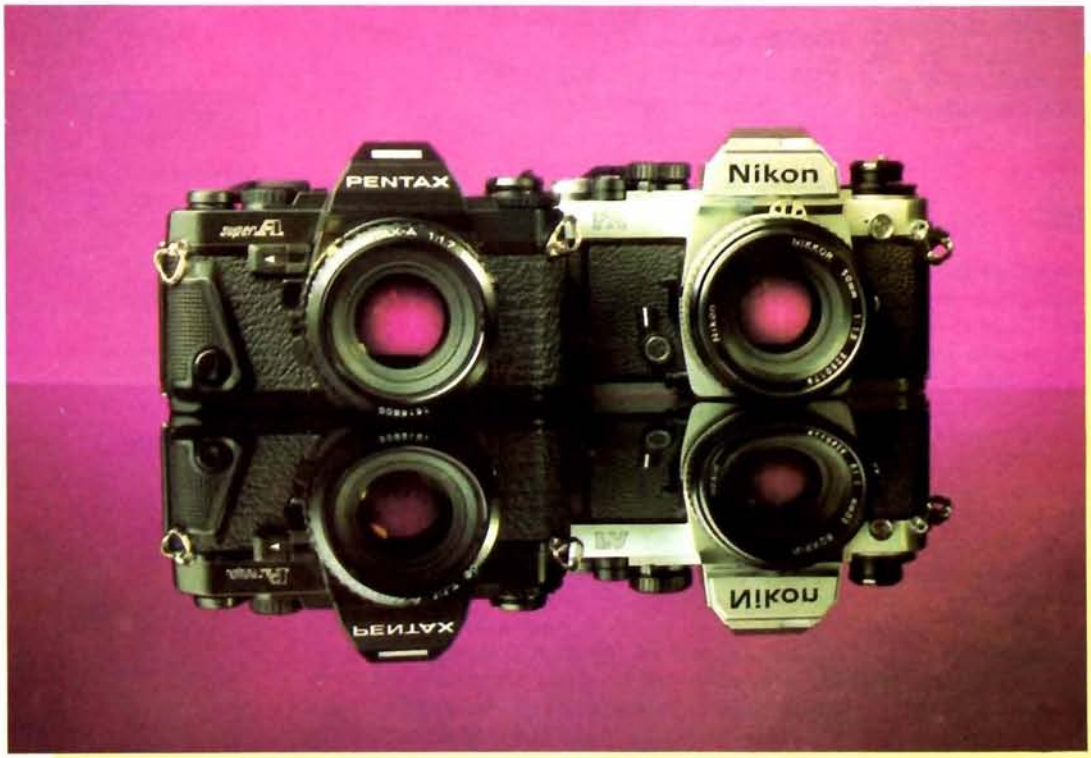
Il rapido progresso della miniaturizzazione elettronica, tuttavia, consente oggi di far sì che la lettura dell'esposimetro, convertita in opportuni segnali elettrici, determini automaticamente sia l'apertura del diaframma sia il tempo di posa. Essendo questo processo automatico pressoché istantaneo, ottenere belle foto, catturando immagini o particolari espressioni dei soggetti, è molto facile.
La Canon A1 è un apparecchio fotografico molto sofisticato, che permette ben sei opzioni di "scatto":

1) Priorità del tempo di posa: il fotografo fissa il tempo di posa, la macchina regola automaticamente il diaframma.

2) Priorità del diaframma: il fotografo fissa l'apertura del diaframma e la macchina regola automaticamente il tempo di posa.

3) Programmazione automatica: la macchina si occupa di entrambe le operazioni citate, in base ad una combinazione ottimale delle due.

4) Flash automatico: se vengono utilizzati



IAN McKINNELL

Basta scattare...

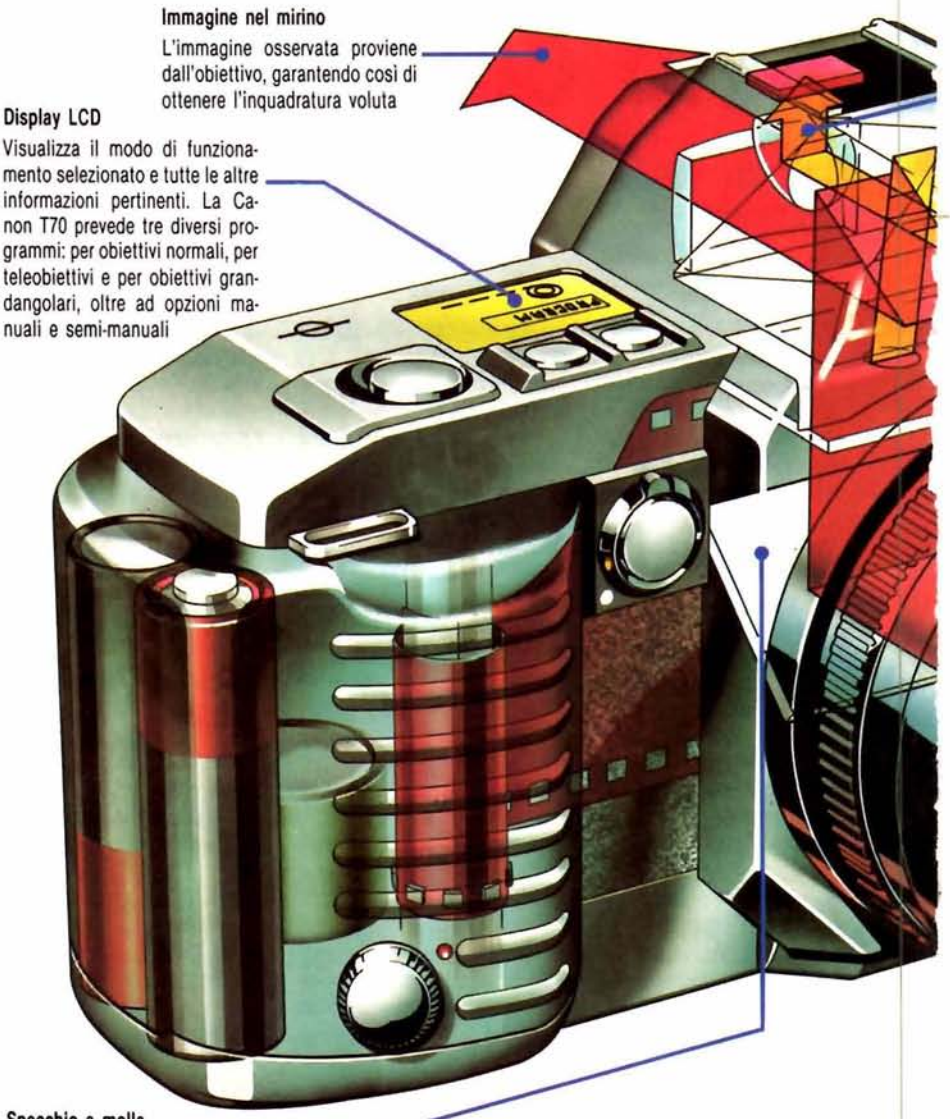
Fin dagli anni '70 sono stati impiegati microprocessori per controllare l'esatto rapporto tra quantità di luce e apertura di diaframma. Oggi, però, le funzioni offerte dalle più moderne macchine sono di gran lunga superiori, poiché riescono a coordinare le informazioni lette e a regolare la macchina in conseguenza, anche in condizioni molto particolari (uso di flash, ecc.). Ciò consente anche ai più inesperti di produrre fotografie di grande effetto. Due macchine altamente sofisticate sono la Nikon FA e la Pentax Super A, qui illustrate, programmate per scegliere l'esposizione ottimale nelle più disparate condizioni ambientali

speciali flash, la macchina seleziona automaticamente il tempo di esposizione con flash (1/60 di secondo) e la corretta apertura di diaframma. Un apposito circuito di controllo, nel flash, regola la luce prodotta in modo che sia sufficiente per la foto.

5) Priorità di apertura "forzata": questa serve per utilizzare certi vecchi sistemi a lenti o altri accessori, in cui l'apertura del diaframma rimane fissa sul valore scelto. Nei dispositivi più moderni l'apertura, salvo al momento dello scatto, è sempre massima, consentendo così di esaminare dal mirino un'immagine sempre alla massima luminosità.

6) Manuale: in tal caso il fotografo fissa sia l'apertura del diaframma che il tempo di posa. È utile per ottenere speciali effetti di luce.

Un display elettronico, visibile guardando nel mirino della Canon A1, indica all'operatore l'opzione selezionata e quali valori sono stati fissati per il diaframma e per il tempo di posa. Siccome il display funziona a LED, esso è visibile anche con poca luce ambientale.
Nonostante l'ovvia utilità dell'opzione "Programmazione automatica" della Canon A1, possono presentarsi occasioni in cui il semplice algoritmo impiegato per calcolare l'esposizione "ottimale" non sia poi così efficace. Se, ad esempio, si scattano fotografie di notte (senza flash) può darsi che la macchina selezioni un tempo di posa di 1/30 di secondo e un'apertura di f/2.8. I valori sono tecnicamente giusti, ma la foto riesce solo se la mano del fotografo è ben ferma. Infatti, con tempi al di sotto di 1/60 di secondo il rischio di foto "mosse" è abbastanza alto: il display lampeggia un avviso in tal senso, ma la macchina non compie alcun tentativo di aumentare il diaframma per poter così usare un tempo di posa più veloce.

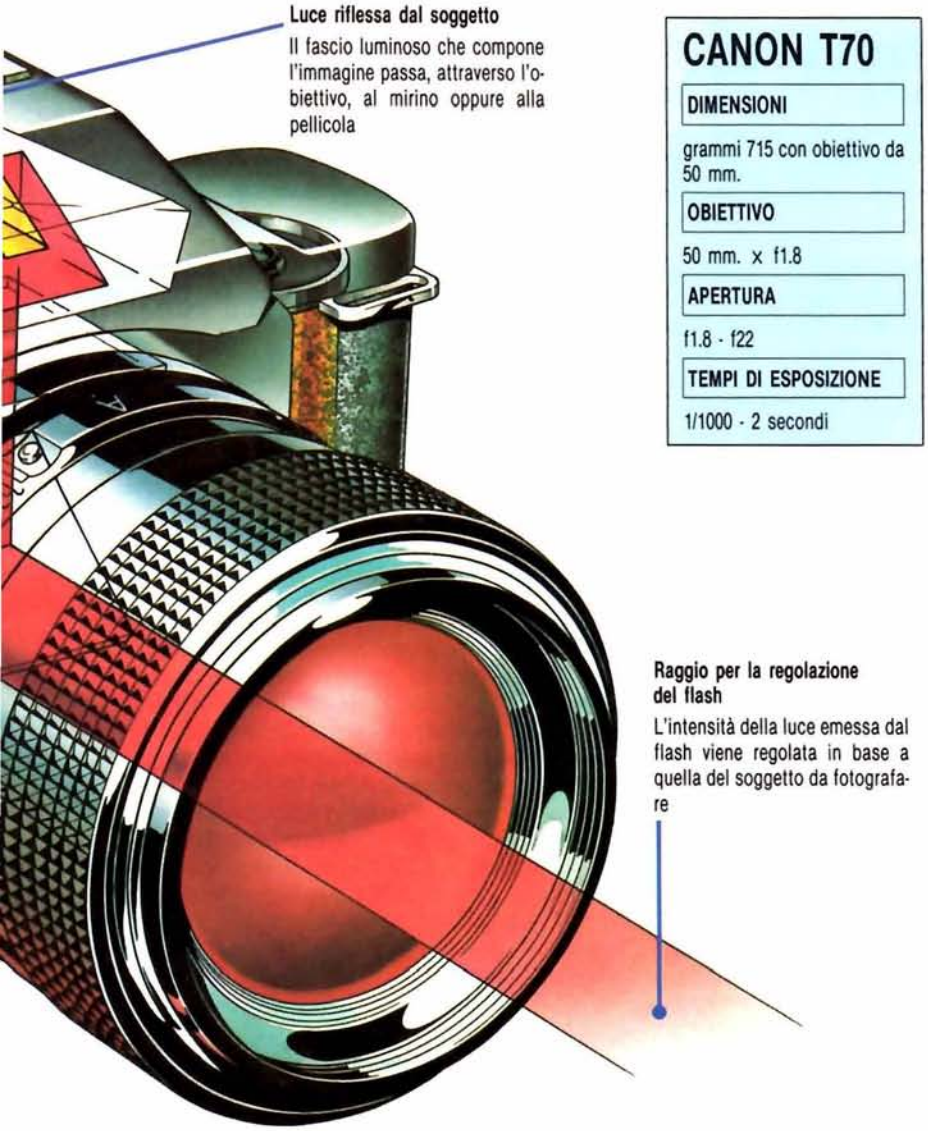


Apparecchi concorrenti

Anche altre ditte hanno realizzato macchine fotografiche multifunzione con microprocessore incorporato. La Pentax Super A, ad esempio, è dotata di un programma leggermente più sofisticato di quello della Canon A1, in grado di scegliere combinazioni migliori di diaframma/tempo. Nella situazione descritta poc'anzi (foto di notte) la Pentax Super A sceglierebbe un tempo di poco inferiore a 1/60, riducendo così il rischio di foto mosse. La Nikon FA, invece, riesce ad "accorgersi" se viene impiegato un teleobiettivo da 135 mm. o più, selezionando di conseguenza un programma alternativo, che consente alla macchina di selezionare combinazioni di diaframma/tempo più adatte.
Forse traendo spunto dalla Nikon, la Canon ha realizzato un nuovo modello, il T70, che prevede addirittura tre diversi programmi: uno per obiettivi normali, uno per teleobiettivi e uno per obiettivi grandangolari. Tuttavia la selezione del programma adatto va fatta manualmente, perché la macchina non si accorge della presenza del tipo di obiettivo montato. Ciò non rappresenta necessariamente uno svantaggio: selezionando il programma per teleobiettivo, ma montando un grandangolare, si ottengono tempi di posa più brevi, con maggiori garanzie di "fissare" ogni singolo elemento di un'immagine in movimento.
Un problema apparentemente irrisolvibile, nelle macchine automatizzate, deriva dal fatto che l'esposizione viene regolata in base alla luminosità media della scena. Ciò significa che fotografando una motocicletta sullo sfondo di un luminoso tramonto, la macchina tende a regolarsi sulla luminosità del sole, facendo apparire la moto come una macchia scura quasi indistinguibile. Viceversa, fotografando la moto contro uno sfondo completamente nero, si corre il rischio di ottenere una foto sovraesposta, poiché il soggetto

UNA CHIP-FOTOGRA

Una CPU a 8 bit, appositamente progettata, provvede al funzionamento della Canon T70, assieme ad altri chip destinati al controllo della luminosità e dei dispositivi di ingresso/uscita. Un oscillatore al quarzo genera gli impulsi di clock necessari al funzionamento del sistema e al conteggio del tempo di esposizione. Un complicato sistema di rimandi, controllato elettronicamente, provvede a fissare la giusta apertura del diaframma. Un dispositivo aggiuntivo permette di imprimere su ciascun negativo i dati sull'esposizione



CANON T70

DIMENSIONI

grammi 715 con obiettivo da 50 mm.

OBIETTIVO

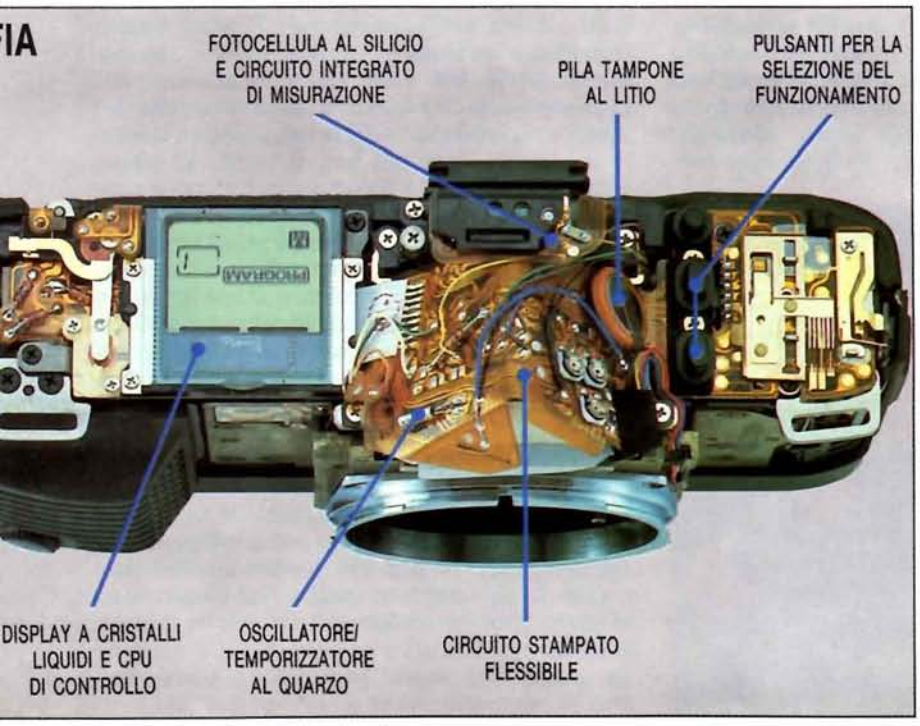
50 mm. x f1.8

APERTURA

f1.8 - f22

TEMPI DI ESPOSIZIONE

1/1000 - 2 secondi



viene considerato molto meno luminoso di quanto lo sia realmente.
La Nikon FA aggira l'ostacolo misurando la luminosità di ben cinque punti diversi dell'immagine, confrontando poi le letture con una serie di scene "standard" memorizzate, per scegliere l'esposizione ottimale. I modelli memorizzati sono stati selezionati dalla Nikon analizzando migliaia di fotografie.
La più sofisticata (elettronicamente parlando) tra tutte le macchine fotografiche qui considerate è la Canon T70. La T70 ha un prezzo ragionevole e non possiede controlli di tipo meccanico: tutto viene regolato tramite pulsanti, uno dei quali (posto in alto a sinistra sulla macchina) aziona il display a cristalli liquidi (situato sull'altra estremità), che riporta tutte le informazioni necessarie. Quelle più importanti, comunque, appaiono anche guardando attraverso il mirino.

La T70 in azione

Il tempo di posa viene fissato agendo su due appositi pulsanti e il display visualizza il valore via via scelto. Anche la sensibilità della pellicola (il numero di ASA o ISO) viene fissato in modo analogo. Sul display appare anche il numero del fotogramma che si sta per scattare.
Un motore incorporato provvede all'avanzamento automatico della pellicola dopo lo scatto e al riavvolgimento della stessa dopo l'ultimo fotogramma. La T70 adopera due comuni pile, il cui stato di carica è anch'esso indicato sul display. Se si usa l'autoscatto, sul display appare il "conto alla rovescia" dei secondi che mancano allo scatto.
I microprocessori adoperati nelle macchine fotografiche sono molto meno potenti di quelli usati nei computer.
La T70 possiede un chip a 8 bit, appositamente progettato e realizzato con tecnologia CMOS (per contenere il consumo). Il processore lavora a soli 32 KHz: una velocità 100 volte inferiore a quella di un computer. Se l'apparecchio non viene usato, la frequenza di lavoro viene automaticamente selezionata ad appena 8 KHz.
Il chip è contenuto in un involucro a 60 piedini, assieme ad una vasta ROM e a 16 Kbyte di RAM. Altri quattro chip lavorano "in complemento"; il più importante è quello che presiede alle operazioni di ingresso/uscita, e controlla l'azionamento delle varie parti tramite magneti e un motore. Esso si occupa anche di convertire il segnale analogico emesso dall'esposimetro in segnali digitali, adatti al microprocessore.
Anche il campo della fotografia, dunque, è stato invaso da chip, da RAM e ROM: scattare foto non comporta più le relative difficoltà di una volta, in quanto basta premere il pulsante di scatto al momento giusto. Tuttavia, per quanto perfetta tecnicamente possa essere una macchina fotografica, occorreranno sempre l'estro e la creatività umana per realizzare foto di alto pregio artistico.

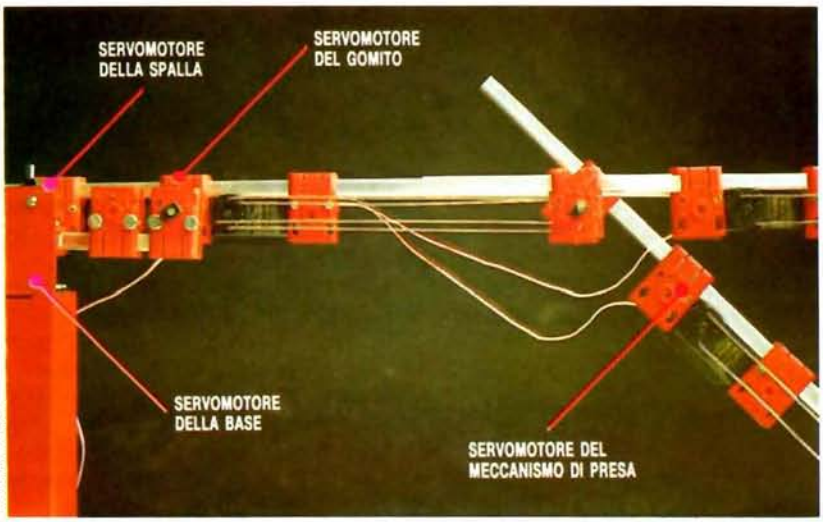
I servomotori

Le applicazioni di controllo per i servomotori collegati ad un computer

Esistono tre tipi di motori elettrici: quelli a corrente continua, quelli passo passo e i servomotori. Un motore a corrente continua può essere facilmente controllato da un computer, ma tende ad essere impreciso se incontra una certa resistenza. In tal caso la velocità di rotazione del motore si riduce e il computer non è più in grado di conoscerne l'esatta posizione.
Un motore passo passo non presenta questo inconveniente perché ruota di un dato angolo (ad esempio, 7,5°) ogni volta che riceve un impulso. Tenendo il conto degli impulsi trasmessi e presumendo che il motore non venga mai sovraccaricato, il computer può calcolarne l'esatta posizione. I motori passo passo vengono comunemente applicati a sistemi controllati da computer, quali: bracci robot, torni, selezionatrici, ecc. Poiché gli impulsi di controllo alimentano anche il sistema, tali motori richiedono dispositivi di pilotaggio, che li rendono abbastanza costosi.
I piccoli servomotori digitali sono facilmente reperibili nei negozi di modellismo perché vengono comunemente usati nei modelli radiocomandati di aerei, navi ed automobili. Le loro dimensioni variano da metà a circa dieci volte il formato di una scatola di fiammiferi. Alcuni sono eccezionalmente potenti, capaci di imprimere più forza di quanta riesca ad esercitarne una persona usando un grosso cacciavite. Anche i servomotori più economici sono quindi adatti alla realizzazione di piccoli bracci robot, ecc.
Un piccolo servomotore digitale come il Futuba FP-S126 o l'Axoms AS-1, contiene un potenziometro a retroazione ed un piccolo motore a corrente continua. Questo è collegato da una serie di ingranaggi ad una barretta sporgente dal motore a cui possono essere attaccate leve, ruote dentate o altro. Tali servomotori, sia che incorporino tutti i circuiti necessari sia che possiedano un chip integrato di controllo del circuito, sono ideali per la messa a punto di sistemi a feedback collegati a un computer.
Un tipico servomotore digitale per modellismo è alimentato a 5 volt e la sua angolazione viene regolata tramite una linea di controllo indipendente. Un impulso della durata di un millisecondo (1/1000) ruota la barretta in una direzione, mentre un impulso di due millisecondi la ruota dello stesso angolo ma in direzione opposta. Le variazioni di angolazione sono infatti proporzionali alla durata dell'impulso.
Purtroppo però, dopo la ricezione del segnale, il motore resta in azione solo per circa 20 millisecondi per poi tornare alla posizione originale. Per mantenere l'angolazione desiderata è necessario ripetere l'impulso di controllo con una frequenza di circa 50 Hz.
I servomotori vengono abitualmente usati per azionare leve, ma consentono anche movimenti lineari. Se il potenziometro viene staccato dall'ingranaggio e disposto centralmente, il motore gira senza interruzioni, ma la sua velocità di rotazione dipende ancora dalla durata dell'impulso.

I servomotori del Beasty

Il braccio robot Beasty (vedere a pag. 1250) è azionato da tre servomotori (movimento della base, della spalla e del gomito) e da un quarto servomotore opzionale per il controllo dell'effetto finale. I servomotori sono dispositivi ideali per i bracci robot, perché consentono loro di raggiungere e mantenere una particolare posizione



IAN McKINNELL

Collegamento al computer

Se colleghiamo i servomotori direttamente alla porta utente del computer, un errore di allacciamento può danneggiare i delicati circuiti interni dell'apparecchio. È bene quindi introdurre un sistema di separazione, composto dal buffer box e dai box di uscita costruiti in precedenza. Il cavetto di alimentazione del motore va collegato alla presa positiva (rossa) di una delle linee del box di uscita di basso voltaggio, mentre il filo di massa va collegato alla presa negativa (nera).
Dopo aver connesso il cavetto per il controllo del motore alla linea 0 per i dati della porta utente, possiamo azionare il motore inviando l'appropriata sequenza di impulsi sulla linea 0. Per trasmettere un impulso il bit 0 per i dati va posto a 1, ottenendo così il passaggio di una tensione di 5 volt sulla linea per i dati 0. Un contatore di ritardo permette di introdurre una pausa della durata richiesta prima di riportare il bit 0 per i dati a zero.
Il Micro BBC ed il Commodore 64 usano un versatile chip d'interfaccia per configurare la porta utente, che può funzionare indifferentemente in ingresso o in uscita. Nel nostro caso la porta viene predisposta per l'uscita con un programma BASIC che, tramite una serie di istruzioni POKE, modifica direttamente il registro di controllo per la direzione dei dati.

Passiamo adesso all'invio degli impulsi sulla linea dati della porta per l'utente. Comunicando alla porta il numero esadecimale 88 (equivalente al decimale 136 ed al binario 10001000), otteniamo sul piedino per i dati i seguenti voltaggi: 5v, 0v, 0v, 0v, 5v ,0v, 0v, 0v. Questa sequenza permane inalterata finché non viene modificata dal programmatore. L'impulso viene quindi generato inserendo in rapida sequenza i valori esadecimali 00, 88, 00.
Per rendere l'impulso veloce è necessario usare il codice macchina. L'algoritmo per l'azionamento di un servomotore è il seguente:

1) Specificare l'angolo della leva usando un programma BASIC che immagazzina un valore compreso tra 0 e 255 in un byte (chiamato ANGOLO).
2) Porre a 1 la linea per i dati 0 (5v), quindi trasmettere l'impulso.
3) Creare una pausa di un millisecondo mediante un contatore di ritardo.
4) Introdurre un'ulteriore pausa di durata compresa tra 0 e 1 millisecondo. In questo caso il valore iniziale del contatore di ritardo (quindi del numero di cicli) è uguale ad ANGOLO.
5) Azzerare la linea dati 0 (0v) per terminare l'emissione dell'impulso.

Se ANGOLO = 0, l'impulso dura un millisecondo; se ANGOLO = 128 (sul Micro BBC e sul Commodore 64) l'impulso dura 1,5 millisecondi e la leva assume una posizione intermedia.

Un "treno" di impulsi

Un solo impulso non è sufficiente per mantenere un servomotore nella posizione desiderata, occorre così inviare un "treno" d'impulsi con una frequenza di circa 20 millisecondi. Il treno può essere generato in due modi:

1) Con un ciclo di ritardo che introduca una pausa tra impulsi successivi. In questo caso il computer non può eseguire altre operazioni durante il ciclo.
2) Usando un sistema di interrupt, che consentono al computer di eseguire contemporaneamente un altro programma (di solito in BASIC). Questo programma secondario può controllare il movimento dei motori.

Tanto il Micro BBC quanto il Commodore 64 usano processori della serie 6500 dotati di due piedini per le interrupt: NMI e IRQ. Il secondo, corrispondente alla linea delle interrupt, viene impiegato per operazioni di temporizzazione. Quando sulla linea IRQ compare una interrupt, il processore interrompe il proprio lavoro e passa a gestire l'interrupt. Infine torna a riprendere il lavoro originale, dal punto in cui era stato interrotto.
I sistemi operativi di entrambi gli apparecchi ricorrono alle interrupt per il proprio funzionamento. Il Micro BBC genera 100 interrupt per secondo (uno ogni 10 millisecondi), il Commodore 60 per secondo. Durante la loro esecuzione il computer aggiorna i temporizzatori del sistema, esegue una scansione della tastiera, ecc. Quindi i sistemi operativi di queste macchine possiedono clock che generano interrupt e routine di sistema per la loro gestione.
Le interrupt di sistema di entrambi gli apparecchi possono essere usate per creare programmi che generano impulsi. Per captare le interrupt del Commodore è necessario però modificare il vettore d'interruzione, ossia un indirizzo registrato in due byte consecutivi, che comunica al processore la locazione della routine di gestione delle interrupt, inserendovi l'indirizzo della routine in codice macchina aggiunta. Al termine di questa, occorre eseguire un salto alla normale routine di gestione delle interrupt di sistema. Otteniamo così un sistema in cui il processore genera un impulso ogni volta che si verifica un interrupt, ossia 60 volte al secondo. I listati che seguono sono le versioni in BASIC e in linguaggio Assembly di un programma che controlla un servomotore collegato ad un Commodore 64. (In seguito vengono riportati i listati per il controllo combinato di più motori e le versioni di entrambi i programmi per il Micro BBC).

MECCANISMO DI FUNZIONAMENTO



KEVIN JONES

L'inclinazione della barretta collegata al perno di un servomotore è proporzionale alla durata dell'impulso del computer di controllo: più esso è lungo, maggiore è l'inclinazione. Se il segnale viene ripetuto ogni 20 millisecondi, il motore mantiene la posizione raggiunta. Questo tipo di "bloccaggio" consente di utilizzare i servomotori in apparecchiature di controllo elettromeccanico. La maggior parte dei microcomputer genera interrupt con una frequenza compresa tra 10 e 20 millisecondi, utilizzabili per ripetere il segnale senza ricorrere ad un programma BASIC di controllo. Un altro vantaggio è che la trasmissione e la ripetizione del segnale avvengono su un'unica linea dati, quindi ad ogni servomotore corrisponde un solo bit della porta dati

CONTROLLO DI UN SERVOMOTORE SUL COMMODORE 64

La prima parte del listato per il Commodore 64 per il controllo di un servomotore mostra come vengono modificati i vettori d'interruzione (locazioni 788-789). Questa operazione non può essere eseguita in BASIC, perché la comparsa di un'interrupt durante la correzione potrebbe causare la "caduta" del sistema. Si rammenti che le interrupt vengono disabilitate con l'istruzione SEI, prima di effettuare la modifica e successivamente riabilitate con l'istruzione CLI.
Il programma di chiamata in BASIC carica le routine in codice macchina, configura la porta utente ed inserisce nella locazione di memoria $3000 (12288) i valori assegnati ai tasti di controllo compresi tra 1 a 9. Questi inducono rotazioni del motore proporzionali ai valori loro assegnati. Premere il tasto E per uscire dal programma.
Se si possiede un assemblatore, si trascriva il listato sorgente e lo si assembli in un file "oggetto", da far successivamente caricare dal programma di chiamata. In alternativa, si usi il programma di caricamento ("loader") in BASIC (che contiene il codice macchina sotto forma di DATA) e lo si esegua per caricare il codice in memoria. Si impartisca un comando NEW, prima di caricare ed eseguire il programma di chiamata. Se si usa il "loader" si possono omettere le linee 30 e 40 dal programma di chiamata.

Nota: È importante sottolineare che, se il programma che usa le interrupt contiene errori, esso può provocare la "caduta" dell'intero sistema: il computer non subisce alcun danno, ma in genere è necessario spegnere l'apparecchio e ripetere l'immissione del programma ripartendo da capo. Per evitare ciò conviene salvare (SAVE) il programma prima di eseguirlo

CODICE SORGENTE

```
;++++++++++++++++++++++++++++
;++++++++++++++++++++++++++++
;++                        ++
;++ Programma di controllo ++
;++    per servomotore     ++
;++                        ++
;++++++++++++++++++++++++++++
;++++++++++++++++++++++++++++
;
PORTA = 56579    ;  REG. DATI DELLA PORTA UTENTE
ANGOLO = 12288   ;  LOCAZ. VALORE DELL'ANGOLO
;
      *=$0334
;
      SEI         ;DISABILITA INTERRUPT
      LDA $0314   ;VETTORE IRQ CORRENTE
      LDX $03C4
      STA $03C4
      STA $0314
      LDA $0315
      LDX $03C5
      STA $03C5
      STX $0315
;
      CLI         ;RIABILITA INTERRUPT
      RTS
;
;++++ GESTIONE EVENTI ++++
;
      PHP
      PHA
      TYA         ; APPOGGIA I REGISTRI
      PHA         ; SULLO STACK
      TXA
      PHA
      LDA #$FF
      STA PORT
      LDY #$FF
LOOP
      DEY         ; CICLO DI RITARDO
      BNE LOOP    ; CIRCA 1 MILLISEC
;
      LDY ANGLE
LOOP1
      DEY         ; CONTEGGIA IMPULSI
      BNE LOOP1
;
      LDA #$00
      STA PORT    ; REGISTRO ZERO PER I DATI
;
      PLA
      TAX         ; RIPRISTINA I VALORI
      PLA         ; DEI REGISTRI
      TAY
      PLA
      PLP
;
      JMP $EA31
```

PROGRAMMA DI CARICAMENTO IN BASIC

```
10 REM ******************************
11 REM ******************************
12 REM **        PROGRAMMA DI        **
13 REM **       CARICAMENTO PER      **
14 REM **        PROGRAMMA DI        **
15 REM **  CONTROLLO SERVOMOTORE     **
16 REM ******************************
20 REM ******************************
30 :
40 FOR  I=820 TO 882
50 READ A:POKE I,A
60 CC=CC+A
70 NEXT I
80 READ CS: IF CC<>CS THEN PRINT
   "CHECKSUM  ERROR": STOP
100 DATA120,173,20,3,174,196,3,1,41,196
110 DATA3,141,20,3,173,21,3,174,197,3
120 DATA141,197,3,142,21,3,88,96,8,72
130 DATA152,72,138,72,169,255,141,3
140 DATA221,160,255,136,208,253,172,0
150 DATA48,136,208,253,169,0,141,3,221
160 DATA104,170,104,168,104,40,76,49
170 DATA234
180 DATA7170:REM * CHECKSUM *
```

PROGRAMMA DI CHIAMATA

```
10 REM ***** UN SOLO SERVOMOTORE *****
20 :
30 DN=8: REM PER CASSETTA DN=1
40 IF A=0 THEN A=1: LOAD"SINGSERV.HEX"
   ,8,1
50 POKE 964,79: POKE 965,3: REM PUNTA
   ALLA LOCAZ. PER IRQ
60 DDR=56577: POKE DDR,255: REM TUTTE
   LE LINEE IN USCITA
70 MC=820: SYS MC: REM SELEZIONA IL
   VETTORE IRQ
80 POKE 53265, PEEK (53265) AND 239:
   REM CANCELLA LO SCHERMO
90 :
100 GET K$: IF K$="" THEN 100: REM
   ATTENDE LA BATTUTA DI UN TASTO
110 REM **** MODIFICA LA POSIZIONE
   DEL MOTORE ****
120 IF ASC(K$)>48 AND ASC(K$)<58
   THEN POKE 12288, VAL(K$)*20
130 IF K$<>"E" THEN 80: REM PREMERE
   "E" PER USCIRE
140 END
```

Trasformazioni spaziali

Approfondiamo alcune applicazioni grafiche del LOGO

Su di una figura bidimensionale è possibile applicare quattro tipi di trasformazioni, inalterando la figura originale (ma non necessariamente la sua posizione). Queste sono: la traslazione, la rotazione, la riflessione e la riflessione slittata (vedere illustrazione).
Una figura è detta simmetrica se è possibile riprodurla, tramite una o più trasformazioni, ma lasciando inalterata sia la sua forma sia la sua posizione. Figure "finite" (quali poligoni o lettere dell'alfabeto) devono avere simmetrie basate sulla rotazione e sulla riflessione, poiché una traslazione o una riflessione slittata ne modificherebbero la posizione.
Per esplorare queste simmetrie è comodo disporre di procedure LOGO in grado di traslare e di ruotare una sagoma. Iniziamo da una procedura per ottenere una figura riflessa rispetto ad un asse con una data origine e direzione.
La procedura di disegno della figura ha maggiore utilità se è di tipo "trasparente" (ossia se la tartaruga, dopo aver tracciato il disegno, torna nella sua posizione originale, orientata nella direzione originale). Il compito può essere così suddiviso in due parti: in primo luogo occorre trovare le coordinate e la direzione iniziali della figura riflessa rispetto alla posizione originale. In secondo luogo occorre cambiare, nel tracciamento della figura, tutti gli spostamenti verso destra in spostamenti verso sinistra e viceversa. Un sistema per ottenere ciò può essere quello di sostituire tutte le istruzioni RT e LT con una procedura GIRA, così definita:

```
TO GIRA :A
   RT :DIR * :A
END
```

Adesso possiamo definire un quadrato con:

```
TO QUADRATO
   REPEAT 4 [FD 50 GIRA 90]
END
```

ma prima di usare questa procedura occorre assegnare alla variabile globale DIR il valore 1, cosicché MAKE "DIR 1 QUADRATO disegni un quadrato. Per riflettere il quadrato lungo l'asse y basta digitare MAKE "DIR (–1) e poi QUADRATO.
Per la procedura che posiziona la tartaruga prima di tracciare il disegno riflesso occorre qualche nozione di trigonometria:

```
TO RIFLETTERE :A
   MAKE "H HEADING
MAKE "XOLD XCOR
MAKE "ANGOLO (ATAN :YOLD :XOLD) –90 + :A
   MAKE "R SQRT (:XOLD * :XOLD + :YOLD * :YOLD)
   PU
   SETXY 0 0
   SETH :A + :ANGOLO
   FD :R
   SETH 2 * :A – :H
   PD
   MAKE "DIR :DIR * (–1)
END
```

Questa procedura può essere impiegata per riflettere linee che passano per l'origine. Si provi:

```
MAKE "DIR 1
PU SETXY 40 70 PD
QUADRATO
RIFLETTERE 60
QUADRATO
```

Se l'immagine riflessa coincide con quella originale, allora si parla di "simmetria riflettente" attorno all'asse usato. Si provi:

```
MAKE "DIR 1
PU SETXY 0 0 PD
QUADRATO
RIFLETTERE 45
QUADRATO
```

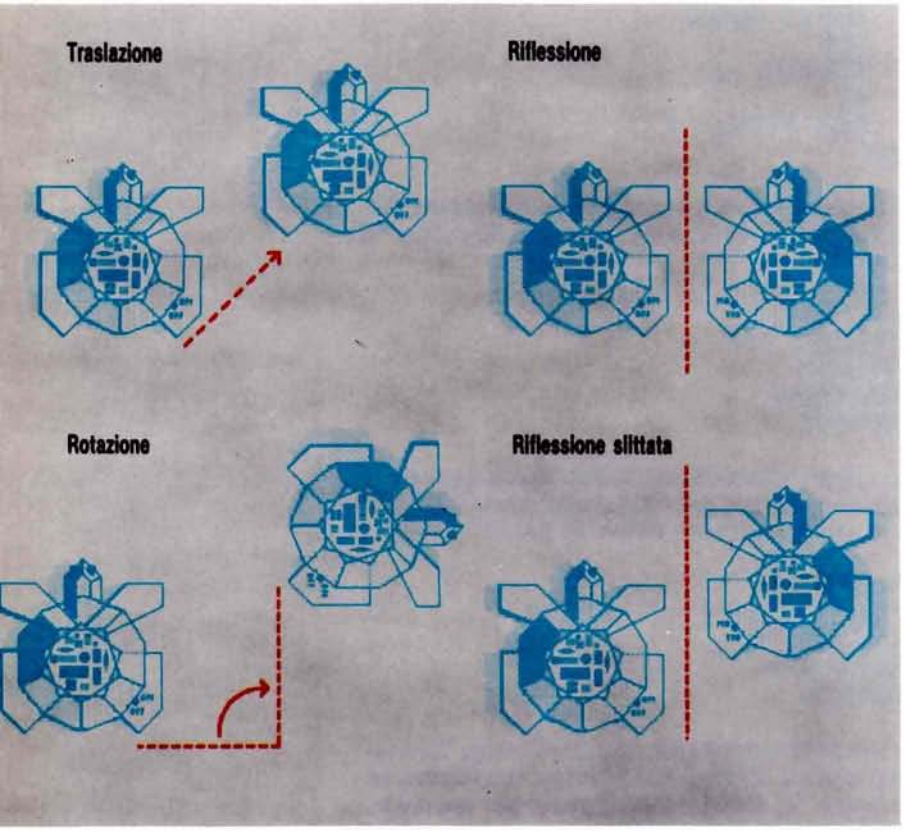
Una procedura simile può servire per far ruotare, di un angolo determinato, una figura attorno a un dato punto. Si provi, per esercizio, a scrivere tale procedura.
Alcuni disegni, come quelli sulla carta da parati, contengono elementi ripetitivi: è possibile effettuare traslazioni e riflessioni slittate su di loro, lasciando inalterata la conformazione del disegno originale. Per il momento, però, conviene dedicarsi alla traslazione di figure rispetto a una sola linea (i disegni bidimensionali vengono trattati in seguito).
Combinando le quattro trasformazioni fondamentali, si ottengono in tutto 7 variazioni, come mostrato nell'illustrazione delle "gambette". Alle sette combinazioni corrispondono altrettante procedure in cui vengono adoperate le procedure MUOVI per la traslazione, RUOTA per la rotazione e R.MOTIVO che scambia le svolte RT in LT (e viceversa) in MOTIVO.
Per redigere R.MOTIVO, partendo da MOTIVO, si sono usate le operazioni su liste del LOGO. Ecco la procedura:

```
TO RISCRIVI :PROC
   OUTPUT RISCRIVI.PROC TEXT :PROC
END
```

RISCRIVI esamina il testo di una procedura, lo

altera ed emette la procedura sotto diverso nome. È dato per scontato che la procedura da elaborare contenga primitive LOGO ma non chiamate a sottoprocedure. RISCRIVI contiene una chiamata alla seguente procedura:

```
TO RISCRIVI.PROC :TEXT
   IF :TEXT = [] THEN OUTPUT []
   OUTPUT FPUT RISCRIVI.LINEA FIRST :TEXT
   RISCRIVI.PROC BUTFIRST :TEXT
END
```



Movimenti isometrici

Le trasformazioni che mutano la posizione, ma non la forma, di una figura sono dette isometrie. Esistono quattro tipi fondamentali di trasformazioni isometriche: la traslazione, la rotazione, la riflessione e la riflessione slittata. La traslazione consiste semplicemente in uno scorrimento della figura. La rotazione gira la figura intorno ad un punto specificato. La riflessione implica la riproduzione della figura riproducendone ciascun punto in posizione simmetrica rispetto ad un asse prestabilito. La riflessione slittata non è che una particolare combinazione di riflessione e traslazione. Mentre la traslazione e la rotazione mantengono lo stesso senso della figura originale, la riflessione e la riflessione slittata lo ribaltano, così come avviene nella riflessione in uno specchio

Questa procedura, a sua volta, ne richiama un'altra:

```
TO RISCRIVI.LINEA :LINEA
   IF :LINEA = [] THEN OUTPUT []
   IF LIST? FIRST :LINEA THEN OUTPUT FPUT
   RISCRIVI.LINEA FIRST :LINEA RISCRIVI.LINEA BUTFIRST
      :LINEA OUTPUT FPUT
   CAMBIA.PAROLA FIRST :LINEA
   RISCRIVI.LINEA BUTFIRST :LINEA
END
```

RISCRIVI.LINEA elabora ciascuna linea, passando le singole parole a CAMBIA.PAROLA. La linea che inizia IF LIST? è necessaria nel caso la procedura MOTIVO da elaborare contenga una REPEAT. La procedura CAMBIA.PAROLA è la seguente:

```
TO CAMBIA.PAROLA :PAROLA
   IF (ANYOF :PAROLA = "RT :PAROLA = "RIGHT) THEN
      OUTPUT "LEFT
   IF (ANYOF :PAROLA = "LT :PAROLA = "LEFT) THEN
      OUTPUT "RIGHT
   OUTPUT :PAROLA
END
```

Questa procedura esamina ciascuna parola e provvede alle sostituzioni necessarie. Avendo trascritto tutte queste procedure, vediamole adesso al lavoro. Prima di tutto occorre definire una figura:

```
TO TRI
   REPEAT 3 [FD 50 RT 120]
END
```

Adesso si digiti DEFINE "RIFL RISCRIVI "TRI e si richiami RIFL.
La sua forma dovrebbe essere:

```
TO RIFL
   REPEAT 3 [FD 50 LEFT 120]
END
```

È senz'altro possibile scrivere una procedura RISCRIVI, di più generica applicazione, in grado di gestire anche le chiamate a sottoprocedure, ma sarebbe di gran lunga più complessa di quanto occorre in queste semplici dimostrazioni.

Motivi a sette strisce

È possibile (e matematicamente molto elegante) costruire configurazioni ripetitive mediante procedure che richiamano le quattro trasformazioni fondamentali. Le procedure, qui chiamate semplicemente CONFIG1, CONFIG2 ecc., adoperano queste tre semplici sottoprocedure:

```
TO POSIZIONA
   HT
   PU
   SETXY -125 0
   PD
END
```

Questa posiziona convenientemente la tartaruga nella parte sinistra dello schermo, pronta a disegnare.

```
TO MUOVI
   PU
   RT 90
   FD 50
   LT 90
   PD
END
```

MUOVI esegue la traslazione richiesta

```
TO GIRA
   RT 180
END
```

GIRA esegue la rotazione necessaria.
Per usare le procedure descritte nella pagina a fianco occorre prima definire una procedura per la figura (chiamandola, ad esempio, FIGURA) che sia trasparente e priva di chiamate a sottoprocedure. Poi si può disegnare la prima configurazione digitando CONFIG1 "FIGURA.

LOGOMOTIVO

CONFIGURAZIONE A SETTE STRISCE

Per poter adoperare le procedure CONFIG occorre avere in memoria le seguenti procedure: RISCRIVI.PROC, RISCRIVI.LINEA, CAMBIA.PAROLA, POSIZIONA, MUOVI e GIRA. Ecco le sette procedure di configurazione:

```
TO CONFIG1 :PROC
   DEFINE "MOTIVO TEXT :PROC
   POSIZIONA
   REPEAT 6 [MOTIVO MUOVI]
END

TO CONFIG2: PROC
   DEFINE "MOTIVO TEXT :PROC
   DEFINE "R.MOTIVO RISCRIVI.PROC TEXT :PROC
   POSIZIONA
   REPEAT 3 [MOTIVO MUOVI GIRA R.MOTIVO GIRA MUOVI]
END

TO CONFIG3: PROC
   DEFINE "MOTIVO TEXT :PROC
   DEFINE "R.MOTIVO RISCRIVI.PROC TEXT :PROC
   POSIZIONA
   REPEAT 6 [MOTIVO R.MOTIVO MUOVI]
END

TO CONFIG4: PROC
   DEFINE "MOTIVO TEXT :PROC
   DEFINE "R.MOTIVO RISCRIVI.PROC TEXT :PROC
   POSIZIONA
   REPEAT 6 [MOTIVO GIRA MOTIVO GIRA MUOVI]
END

TO CONFIG5: PROC
   DEFINE "MOTIVO TEXT :PROC
   DEFINE "R.MOTIVO RISCRIVI.PROC TEXT :PROC
   POSIZIONA
   REPEAT 3 [MOTIVO R.MOTIVO MUOVI GIRA MOTIVO
      R.MOTIVO GIRA MUOVI]
END

TO CONFIG6: PROC
   DEFINE "MOTIVO TEXT :PROC
   DEFINE "R.MOTIVO RISCRIVI.PROC TEXT :PROC
   POSIZIONA
   REPEAT 6 [MOTIVO GIRA R.MOTIVO GIRA MUOVI]
END

TO CONFIG7: PROC
   DEFINE "MOTIVO TEXT :PROC
   DEFINE "R.MOTIVO RISCRIVI.PROC TEXT :PROC
   POSIZIONA
   REPEAT 6 [MOTIVO R.MOTIVO GIRA MOTIVO
      R.MOTIVO GIRA MUOVI]
   END
```

Queste procedure vanno eseguite avendo definito un MOTIVO adatto. Quello qui impiegato è:

```
TO GAMBA
   FD 50
   RT 90
   FD 20
   BK 20
   LT 90
   BK 50
END
```

Un motivo alternativo può essere:

```
TO FIG
   RT 30
   FD 20
   LT 50
   FD 20
   RT 90
   FD 10
   REPEAT 4 [FD 20 RT 90]
   BK 10
   LT 90
   BK 20
   RT 50
   BK 20
   LT 30
END
```

Logo & Logo

Nel LOGO Atari non esistono le primitive ATAN e TOWARDS, né un facile modo per sostituirle. Ciò influisce sulle procedure per la RIFLESSIONE e la ROTAZIONE, ma non le sette CONFIG.
Nel LOGO Atari non esistono le primitive TEXT e DEFINE, ma il manuale Atari spiega come definirle. In alternativa non resta che riscrivere R.MOTIVO partendo da MOTIVO



Sette gambette a spasso

Le quattro trasformazioni isometriche fondamentali possono essere combinate in sette modi diversi, come illustrato da questa figura. In ciascun caso si parte dalla figura di una gambetta e tutte le traslazioni vengono fatte in direzione dell'asse x

Schermi a volontà

Come conservare in memoria e visualizzare otto schermi grafici sul Commodore 64

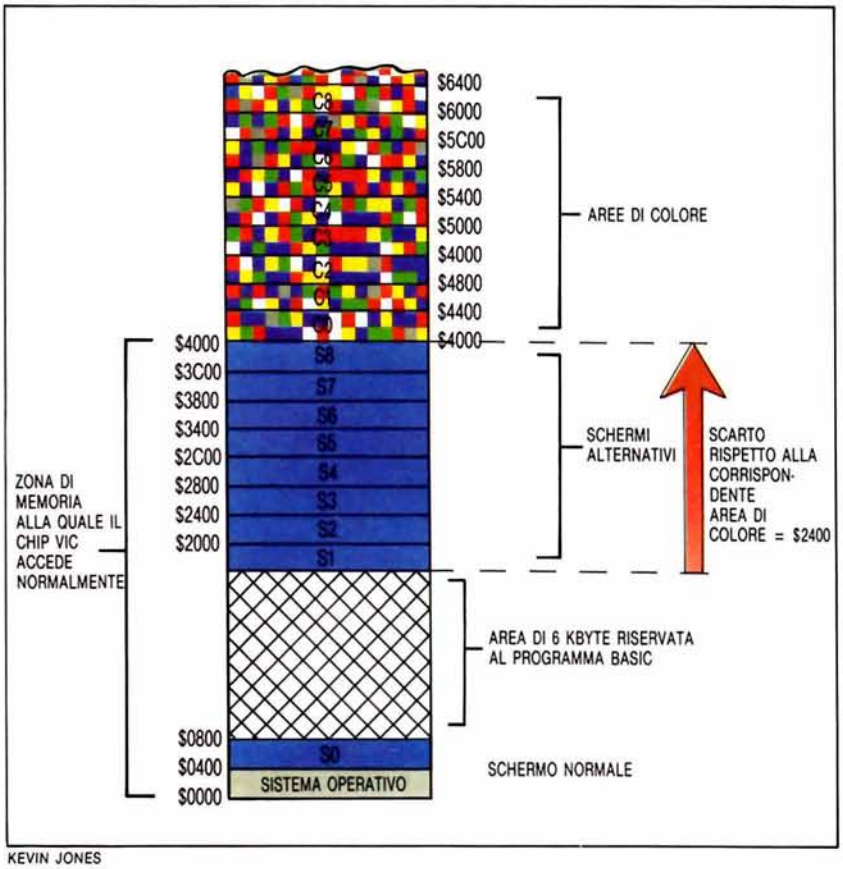
La gestione dello schermo e degli sprite, sul Commodore 64, è affidata a uno speciale chip, chiamato VIC II. Questo chip accede a varie sezioni di memoria per estrarre informazioni con le quali costruisce l'immagine sullo schermo. Queste sezioni di memoria includono la ROM dei caratteri (dove sono conservate le sagome dei vari caratteri), la RAM del colore (che contiene le informazioni sulla colorazione dello schermo) e la RAM del video. Quest'ultima contiene le informazioni sui caratteri da visualizzare in ognuna delle 1000 locazioni (25 righe per 40 colonne) che formano lo schermo.
All'accensione, il chip VIC presume che la memoria del video occupi 1000 byte a partire dalla locazione 1024 ($0400) e accede a tale zona di memoria per estrarre le informazioni iniziali per la visualizzazione. Tuttavia, alterando il contenuto di un registro nel chip VIC, possiamo far sì che esso indirizzi una qualsiasi altra zona di memoria (in genere vengono usati i primi 16 Kbyte di RAM). I quattro bit più significativi del registro di controllo sul VIC, situato all'indirizzo 53272 ($D018) determinano quale blocco da 1 Kbyte, nell'area di 16 Kbyte accessibili, sia da considerare "memoria video". La tabella che segue mostra le possibili configurazioni di bit nel registro di controllo e le zone di memoria corrispondenti:

Selezione del VIC

Il chip VIC del Commodore 64, che gestisce la visualizzazione dello schermo, "vede" una zona di memoria di 16 Kbyte per volta. All'accensione del computer, questa zona è compresa tra $0000 e $3FFF, ma è possibile alterare il contenuto di un registro del VIC in modo da utilizzare un diverso blocco di 16 Kbyte. Il programma qui presentato consente di creare e gestire ben otto diversi schermi, oltre a quello regolare. Per ogni nuova area video viene inoltre creata la corrispondente area di colore, in una zona situata subito sopra gli otto schermi

Configurazione	Inizio memoria video	
0000XXXX	0	$0000
0001XXXX	1024	$0400*
0010XXXX	2048	$0800
0011XXXX	3072	$0C00
0100XXXX	4096	$1000
0101XXXX	5120	$1400
0110XXXX	6144	$1800
0111XXXX	7168	$1C00
1000XXXX	8192	$2000
1001XXXX	9216	$2400
1010XXXX	10240	$2800
1011XXXX	11264	$2C00
1100XXXX	12288	$3000
1101XXXX	13312	$3400
1110XXXX	14336	$3800
1111XXXX	15360	$3C00

* = Zona selezionata all'accensione



KEVIN JONES

Per forzare l'uso di un diverso schermo basta quindi alterare la metà superiore della locazione 53272 ($D018), in base ai valori della tabella. Tuttavia, i quattro bit meno significativi di tale locazione (indicati con XXXX nella tabella) non vanno assolutamente modificati, poiché controllano altre funzioni. Per azzerare la metà superiore della locazione, senza alterare quella inferiore, si opera un AND logico del contenuto con 15 (00001111 in binario). Il risultato di questa operazione viene posto in OR logico con il valore da depositare nel registro, ottenendo l'effetto voluto. Se volessimo, ad esempio, far usare l'ultima zona di memoria disponibile (quella che inizia alla locazione 15360, ossia $3C00), il registro va posto in OR con 240 (11110000 in binario). Ecco, in una sola linea di BASIC, la procedura completa:

```
POKE 53272,(PEEK(53272)AND15)OR240
```

Prima di scrivere sul nuovo schermo, però, occorre informare anche il sistema operativo del Commodore 64 dell'avvenuto cambiamento. Ciò si ottiene depositando la metà più significativa dell'indirizzo del nuovo schermo nella locazione 648 ($0288). In BASIC, tale valore si ottiene facilmente, in quanto è sufficien-

te dividere l'indirizzo del nuovo schermo per 256: nel nostro caso si ottiene $3C.
Avendo modificato il contenuto di questi due registri, possiamo adoperare il nuovo schermo così come faremmo con quello regolare.
La possibilità di usare più schermi, sul Commodore 64, torna molto utile per variare rapidamente le immagini visualizzate. Tuttavia c'è un piccolo problema che riguarda la RAM del colore che, a differenza della RAM del video, è unica e non riproducibile col sistema appena descritto. La soluzione consiste nel depositare le informazioni di colore in tante sezioni di memoria da 1000 byte, quante sono quelle occupate dagli schermi. Prima di usare un particolare schermo, la corrispondente sezione va trasferita nella RAM del colore, avendo cura di salvare le informazioni in essa contenute.

Organizzazione della memoria

Il programma presentato qui di seguito si occupa di riorganizzare la configurazione della memoria del Commodore 64, tenendo conto di tutte le premesse appena enunciate: poiché il massimo numero di schermi indirizzabili dal chip VIC è otto, il programma gestisce otto sezioni per le informazioni di colore, provvedendo a scambiarle opportunamente assieme al corrispondente schermo selezionato. La suddivisione della memoria è mostrata nella figura a pag. 1398.
Naturalmente, per assicurarsi che il programma BASIC non sconfini nelle zone di memoria necessarie agli schermi, occorre abbassare il limite superiore dell'area BASIC. La linea del nostro programma che si occupa di ciò è:

```
POKE 55,0:POKE 56,32:CLR
```

L'indirizzo iniziale di ciascun schermo si ricava da questa formula:

Inizio Schermo = $1C00 + ($0400 * Numero schermo)

L'indirizzo della corrispondente area del colore si calcola semplicemente aggiungendo uno scarto fisso al valore precedente:

Inizio Area Colore = $2400 + Inizio Schermo

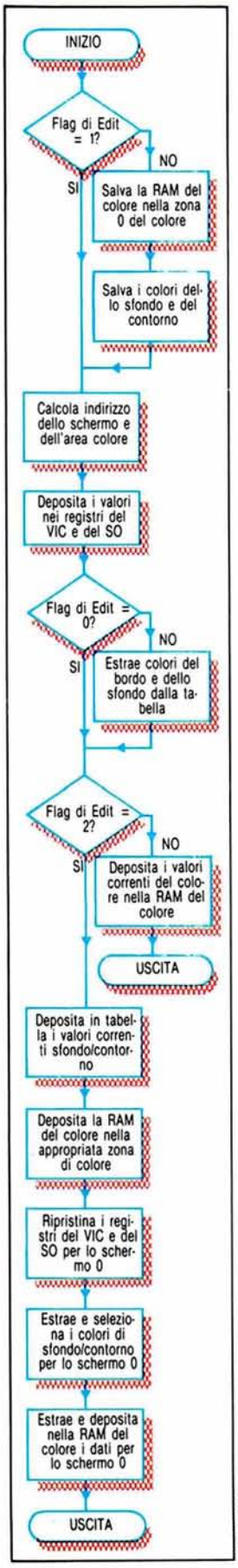
Le aree di colore possono essere collocate ovunque in memoria, ma una posizione molto conveniente è quella che segue immediatamente le zone riservate ai vari schermi. Si noti che viene riservata un'area di colore anche per lo schermo 0, ossia quello regolare. Per tale area di colore lo scarto è diverso dagli altri e il programma deve necessariamente tenerne conto.
I valori per i due registri di controllo (quello del chip VIP e quello del sistema operativo) si ricavano da:

registro VIC = $70 + ($10 * Numero Schermo)
registro SO = Parte alta di Inizio Schermo

Il programma, oltre ad aggiustare i valori dei registri e a trasferire le varie informazioni, conserva in una tabella anche il colore dello sfondo e del contorno di ciascuno schermo. Questi due elementi sono controllati da una coppia di registri del chip VIC, rispettivamente le locazioni: 53281 ($D021) e 53280 ($D020).

Funzionamento del programma

Il programma prevede due modi di funzionamento: il modo edit consente di creare e modificare uno schermo, mentre il modo display permette di visualizzarne uno già definito.
La selezione dello schermo avviene depositando, con una POKE, il numero dello schermo nella locazione 49152 ($C000). Per poter adoperare un'unica chiamata SYS, si ricorre all'uso di un flag, contenuto nella locazione 49153 ($C001). In questa locazione si deposita il valore 0 per accedere al modo display oppure il valore 1 per il modo edit.
Il modo edit, funziona in modo insolito, onde permettere di sfruttare tutte le possibilità dello screen editor (quali cambiare il colore dei testi, selezionare il modo reverse e cancellare lo schermo): il flag che stabilisce il modo viene selezionato a 1, ma dopo la chiamata alla routine. Solo in questa maniera si obbliga la routine a salvare il contenuto dello schermo originale, dell'area del colore e degli altri parametri. La routine torna quindi al programma BASIC, che posiziona il cursore in alto a sinistra e, tramite l'istruzione INPUT, consente l'immissione dei comandi di editing. Lo spostamento del cursore avviene premendo il tasto Return (carattere ASCII 13).
Al termine, il programma BASIC deve richiamare nuovamente la routine, ma stavolta ponendo il flag del modo a 2, forzando la routine a salvare i valori dell'appropriata area del colore e quelli dello sfondo e del contorno. Volendo modificare questi ultimi due valori, occorre eseguire una POKE, rispettivamente, nelle locazioni 49155 ($C003) e 49154 ($C002).
Benché la routine qui presentata sia orientata ad un preciso compito, essa contiene subroutine di applicazione generica, che possono essere utilizzate in altri programmi.
Il programma BASIC principale visualizza un menu relativo alle opzioni di edit e display.
Questa seconda opzione consente di visualizzare, in sequenza, ciascuno degli otto schermi: per passare dall'uno all'altro basta premere un tasto. Visualizzato l'ottavo schermo si riparte dal primo, finché non si preme Return.
A questo punto il programma ripristina lo schermo originale e torna al menu. Se viene scelta l'opzione edit, l'operatore può selezionare il colore dello sfondo e del contorno e poi usare i normali comandi di editing per creare un'immagine.
Premendo Return viene memorizzato il lavoro appena fatto e ripristinato lo schermo principale.



CARICATORE IN BASIC

```
10 REM ******************************
20 REM ** CARICA ROUTINE IN C.M. **
30 REM ** PER SCHERMI AGGIUNTIVI **
40 REM ******************************
50 :
60 FOR I=49152 TO 49439
65 READA:POKE I,A
70 CC=CC+A
75 NEXT
80 READ CS:IFCS<>CC THENPRINT "ERRORE CHECKSUM":STOP
100 DATA255,255,0,0,191,255,0,0,255
110 DATA191,0,0,191,191,0,0,255,191,0
120 DATA0,255,191,0,0,255,255,0,173,1
130 DATA192,201,1,208,30,169,0,141,6
140 DATA192,169,64,141,7,192,32,224
150 DATA192,162,0,32,246,192,173,2,192
160 DATA141,32,208,173,3,192,141,33,
170 DATA 208,173,0,192,208,6,32,32,193
180 DATA76,139,192,169,0,141,4,192,141
190 DATA5,192,174,0,192,173,5,192,24
200 DATA105,4,202,208,250,24,105,28
210 DATA141,5,192,24,105,36,141,7,192
220 DATA173,0,192,162,4,10,202,208,252
230 DATA24,105,112,141,8,192,173,24
240 DATA208,41,15,13,8,192,141,24,208
250 DATA173,5,192,141,136,2,173,1,192
260 DATA208,6,174,0,192,32,211,192,201
270 DATA2,240,4,32,189,192,96,174,0
280 DATA192,32,246,192,32,224,192,32
290 DATA32,193,169,0,141,6,192,169,64
300 DATA141,7,192,162,0,32,211,192,32
310 DATA189,192,96,173,6,192,133,251
320 DATA173,7,192,133,252,169,0,133
330 DATA253,169,216,133,254,32,3,193
340 DATA96,189,9,192,141,32,208,189,18
350 DATA192,141,33,208,96,173,6,192
360 DATA133,253,173,7,192,133,254,169
370 DATA0,133,251,169,216,133,252,32,3
380 DATA193,96,173,32,208,157,9,192
390 DATA173,33,208,157,18,192,96,162,3
400 DATA160,0,177,251,145,253,136,208
410 DATA249,230,252,230,254,202,48,10
420 DATA208,240,177,251,145,253,160
430 DATA231,208,232,96
440 DATA36606:REM*CHECKSUM*
```

SCHERMI AGGIUNTIVI

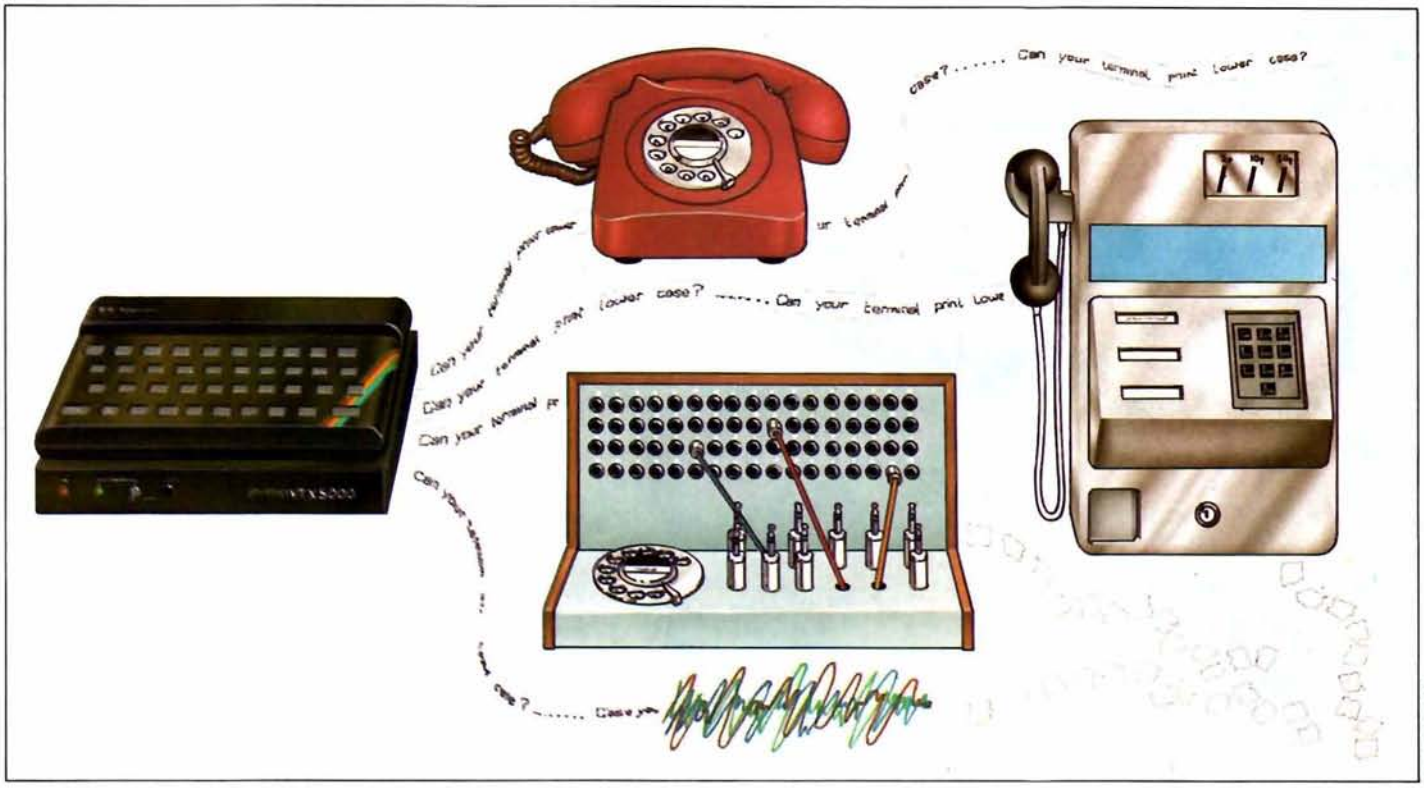
```
  3 REM ******************************
  5 REM **                            **
 10 REM **   SCHERMI AGGIUNTIVI       **
 11 REM **                            **
 12 REM ******************************
 13 :
 15 DN=8:REM PER CASSETTE DN = 1
 20 IFA=0 THEN A=1:LOAD "SCHERMI.HEX",DN,1
 25 POKE55,0:POKE56,32:CLR:REM  ABBASSA  LIMITE MEM.
 40 SCNUMB=49152:     REM NUMERO DELLO SCHERMO
 70 EDITFG=49153:     REM 0=MODO "DISPLAY"
 75 :                 REM 1=MODO "EDIT"
 77 :                 REM 2=COPIA RAM COLORE
 80 ALT=49179:        REM INIZIO COD. MACCHINA
 85 BRDCOL=49154:     REM COLORE CONTORNO
 87 PAPCOL=49155:     REM COLORE SFONDO (PAPER)
 90 :
 95 REM ***** MENU PRINCIPALE ****
 96 :
100 PRINTCHR$(147):REM RIPULISCE LO SCHERMO
105 PRINTCHR$(154):REM LETTERE IN BLU
110 DN$="■■":REM ■ = CARATT. CURSORE IN BASSO
130 PRINTTAB(8);DN$;"CBM 64 - SCHERMI AGGIUNTIVI"
135 PRINTTAB(8);"----------------------------"
140 PRINTTAB(5);DN$;"F1 - MODO EDIT"
160 PRINTTAB(5);DN$;"F3 - MODO DISPLAY"
200 :
210 GETA$:IFA$="" THEN210:REM ATTENDE PRESSIONE
220 IFA$="■" THEN GOSUB 1000
230 IFA$="■" THEN GOSUB 1500
260 GOTO 100
999 :
1000 REM **** EDIT DI UNO SCHERMO ****
1002 :
1005 EF=1:REM SELEZIONA MODO EDIT
1010 PRINTCHR$(147)
1020 PRINTTAB(16);DN$;"MODO EDIT"
1030 PRINTDN$;:INPUT"SCHERMO NUMERO";SN$
1050 IF ASC(SN$)<48 OR ASC (SN$)>56 THEN 1030
1060 PRINTDN$;:INPUT"COLORE BORDO";BC$
1070 IFVAL(BC$)=0 AND BC$<>"0"THEN 1060
1080 PRINTDN$;:INPUT"COLORE SFONDO";PC$
1090 IFVAL(PC$)=0 AND PC$<>"0"THEN 1080
1100 :
1110 POKEEDITFG,EF
1120 POKESCNUMB,VAL(SN$)
1130 POKEBRDCOL,VAL(BC$)
1140 POKEPAPCOL,VAL(PC$)
1150 SYS ALT
1155 REM **** ATTENDE UN RETURN ****
1162 INPUT " ■";X$:REM S = CURSORE "HOME"
1165 REM **** SALVA LO SCHERMO  ****
1170 EF=2
1175 POKE EDITFG,EF
1180 SYS ALT
1185 RETURN
1190 :
1500 REM **** DISPLAY DEGLI SCHERMI ****
1510 EF=0
1520 PRINT CHR$(147)
1545 SN=1
1550 POKE EDITFG,EF
1555 POKE SCNUMB,SN
1560 SYS ALT
1570 GET X$:IFX$=""THEN1570:REM ATTENDE TASTO
1572 IFX$=CHR$(13)THEN 1580:REM SCHERMO NORMALE
1575 SN=SN+1:IF SN<9 THEN 1555
1577 SN=1:GOTO1555
1580 POKESCNUMB,0:SYS ALT
1600 RETURN
```

COMMODORE 64

```
;**********************************
;**      SCHERMI AGGIUNTIVI      **
;**       SUL COMMODORE 64       **
;**********************************
;
FROM  =$FB            ;PUNTATORI
TO    =$FD            ;IN PAGINA 0
;
CRAMLO=$00            ;INIZIO DELLA RAM COLORE
CRAMHI=$D8
NCOLLO=$00            ;COLORE NORMALE
NCOLHI=$40            ;INDIRIZZO INIZIALE
NSCRHI=$04            ;IND. DELLO SCHERMO NORMALE
NVCPOK=$10            ;VALORE NORMALE REGISTRO VIC
;
BLOCKS=$03            ;N. DI BLOCCHI DA 256 BYTE
EXTRA =$E7            ;ZONA EXTRA DI 1000 BYTE
;
COLOFF=$24            ;OFFSET COLORE (PARTE ALTA)
SCROFF=$1C            ;OFFSET SCHERMO (PARTE ALTA)
VICOFF=$70            ;OFFSET REG. CONTROLLO DEL VIC
VCMASK=$0F            ;MASCHERA PER PARTE BASSA REG. VIC
VICREG=$D018          ;REG. CONTROLLO LOCAZ. SCHERMO
EDREG=$0288           ;REG. KERNAL DELLO SCREEN EDITOR
BORDER=$D020          ;REG. COLORE DEL BORDO
PAPER =$D021          ;REG. COLORE DELLO SFONDO
;
*=$C000               ;INIZIAL. PUNTATORE PER LOAD
;
SCNUMB *=*+1          ;NUMERO DELLO SCHERMO
EDITFG *=*+1          ;FLAG DEL MODO EDIT
BRDCOL *=*+1          ;COLORE DEL BORDO
PAPCOL *=*+1          ;COLORE DELLO SFONDO
;
SCBASE *=*+2          ;DEPOSITO PER INIZIO SCHERMO
CLBASE *=*+2          ;DEPOSITO PER INIZIO COLORE
VPOKE  *=*+1          ;DEP. TEMPORANEO PER NUMERO VIC
BRDTAB *=*+9          ;TABELLA COLORI BORDO
PAPTAB *=*+9          ;TABELLA COLORI SFONDO
;
;** SALVA SCHERMO 0 **
        LDA EDITFG
        CMP #$01      ;SE 0 OPPURE 2
        BNE CALC      ;ALLORA NIENTE SALVATAGGIO
;
        LDA #NCOLLO
        STA CLBASE
        LDA #NCOLHI
        STA CLBASE+1
        JSR SAVE      ;SALVA RAM COLORE IN CO
;
        LDX #$00
        JSR SAVEBP    ;SALVA BORDO/SFONDO
;
        LDA #DCOL
        STA BORDER    ;DEPOSITA NUOVI COLORI
        LDA PAPER     ;PER BORDO/SFONDO
        STA PAPER
;
;** CALCOLA INIZIO AREA COLORE **
;
CALC
        LDA SCNUMB
        BNE NOZERO
        JSR RESET     ;RIPRISTINA REG. ORIGINALI
        JMP TESTFG
;
NOZERO
        LDA #$00
        STA SCBASE
        STA SCBASE+1  ;INIZIAL. INIZIO SCHERMO
;
        LDX SCNUMB    ;= NUMERO DELLO SCHERMO
        LDA SCBASE+1  ;SERVE SOLO LA PARTE ALTA
MULT
        CLC
        ADC #$04
        DEX
        BNE MULT
;
        CLC
        ADC #SCROFF   ;SOMMA OFFSET PER LO SCHERMO
        STA CLBASE+1
;
        CLC
        ADC #COLOFF   ;SOMMA OFFSET PER AREA COLORE
        STA SCBASE+1
;
;** RIPRISTINA I REGISTRI VIC E EDIT **
        LDA SCNUMB
        LDX #$04
MORE
        ASL A
        DEX
        BNE MORE      ;MOLTIPL. PER 16
;
        CLC
        ADC #VICOFF   ;A66. OFFSET
        STA VPOKE
        LDA VICREG
        AND #VCMASK
        ORA VPOKE
        STA VICREG    ;ASSEGNA REG. VIC
;
        LDA SCBASE+1
        STA EDREG     ;ASSEGNA REG. EDITOR
;
;** VERIFICA VALORE FLAG DI EDIT **
TESTFG
        LDA EDITFG
        BNE NOLOAD    ;MODO DISPLAY SE = 0
        LDX SCNUMB
        JSR LOADBP    ;CARICA REG. CON BORDO/SFONDO
;
;
NOLOAD
        CMP #$02
        BEQ CONT      ;SALVA RAM SE = 2
        JSR LOAD      ;CARICA COLORE IN CRAM
        RTS
;
CONT
        LDX SCNUMB
        JSR SAVEBP    ;SALVA REGISTRI BORDO/SFONDO
        JSR SAVE      ;SALVA CRAM IN COLORE
        JSR RESET     ;SELEZ. REG. NORMALI
        LDA #NCOLLO
        STA CLBASE    ;CARICA CBASE CON INIZIO CO
        LDA #NCOLHI
        STA CLBASE+1
        LDX #$00
        JSR LOADBP    ;RIPRISTINA BORDO/SFONDO ORIG.
        JSR LOAD      ;E NORMALI COLORI DI SCHERMO
        RTS
;
;** S/R TRASFERIMENTO NELLA RAM **
LOAD
        LDA CLBASE
        STA FROM      ;CARICA PUNTATORI PAG. 0
        LDA CLBASE+1
        STA FROM+1
;
        LDA #CRAMLO
        STA TO
        LDA #CRAMHI
        STA TO+1
;
        JSR COPY      ;COPIA AREA RAM
        RTS
;
;** S/R CARICAM. COLORI BORDO/SFONDO **
LOADBP
        LDA BRDTAB,X
        STA BORDER
        LDA PAPTAB,X
        STA PAPER
        RTS
;
;** S/R TRASFERIMENTO DA RAM **
SAVE
        LDA CLBASE
        STA TO        ;CARICA PUNTATORI PAG.0
        LDA CLBASE+1
        STA TO+1
;
        LDA #CRAMLO
        STA FROM
        LDA #CRAMHI
        STA FROM+1
;
        JSR COPY      ;COPIA AREA RAM
        RTS
;
;** S/R SALVATAGGIO COLORI BORDO/SFONDO **
SAVEBP
        LDA BORDER
        STA BRDTAB,X
        LDA PAPER
        STA PAPTAB,X
        RTS
;
;** S/R PER COPIA DI 1000 BYTE **
;
COPY
        LDX #BLOCKS
        LDY #$00
NEXT
        LDA (FROM),Y
        STA (TO),Y
        DEY
        BNE NEXT
NYBLOC  INC FROM+1
        INC TO+1
        DEX
        BMI FINISH
        BNE NEXT
        LDA (FROM),Y
        STA (TO),Y
        LDY #EXTRA    ;BYTE AGGIUNTIVI
        BNE NEXT
FINISH
        RTS
;
;** S/R PER IL RESET DEI REGISTRI VIC/EDITOR **
RESET
        LDA #NCOLLO
        STA CLBASE
        LDA #NCOLHI
        STA CLBASE+1
        LDA VICREG
        AND #VCMASK
        ORA #NVCPOK
        STA VICREG    ;RIPRISTINA REG. VIC
        LDA #NSCRHI
        STA EDREG     ;RIPRISTINA REG. EDITOR
        RTS
```

Comunicare tramite computer

Ulteriori aspetti della comunicazione tra microcomputer



STEVE CROSS

Il termine "duplex" indica un sistema in grado di trasmettere e ricevere informazioni simultaneamente. Contrariamente ai sistemi "half-duplex", un sistema "full-duplex" consente il contemporaneo trasferimento bidirezionale dei dati. Quest'ultimo è paragonabile ad una comunicazione telefonica, nella quale i due interlocutori possono parlare e ascoltare contemporaneamente; il primo invece assomiglia alle comunicazioni tra radioamatori, alternate dall'azionamento di un pulsante di trasmissione che esclude automaticamente la ricezione. Il principale vantaggio dei sistemi full-duplex è la possibilità di interrompere un computer in fase di trasmissione. In questo modo è possibile, ad esempio, selezionare direttamente l'opzione 1 di un lungo menu senza dover attendere la fine dell'intera trasmissione, come invece accade con i sistemi half-duplex.

I sistemi full-duplex adoperano differenti frequenze di trasmissione a seconda del computer usato. Il computer che effettua la chiamata impiega una frequenza detta "di emissione" (originate), mentre l'elaboratore principale usa una frequenza "di risposta" (answer).

Protocolli di comunicazione

Quando un computer trasmette un'informazione attraverso una linea telefonica, non conosce le caratteristiche del computer ricevente. Questo può essere uno Spectrum con un display a 32 colonne, una telescrivente a 132 colonne a colori o meno, oppure può essere privo di caratteri minuscoli. In breve può trattarsi sia di un piccolo micro sia di una grossa unità.

Esistono tre tecniche principali per comunicare. La prima è quella adottata dal linguaggio BASICODE (vedere a pag. 721), in cui viene impiegato un ristretto numero di codice di base, comprensibili anche ai terminali meno sofisticati; nella seconda tecnica, il computer "interroga" l'apparecchio ricevente per ottenere una serie di informazioni sulle sue caratteristiche e modifica di conseguenza il formato dei dati trasmessi; nella terza, il sistema presume di essere collegato ad un terminale o ad una serie di terminali dalle caratteristiche note e lascia al software scritto dall'utente il compito di armonizzare le informazioni trasmesse.

Il primo metodo è il meno comune, dato che il

Problemi di installazione

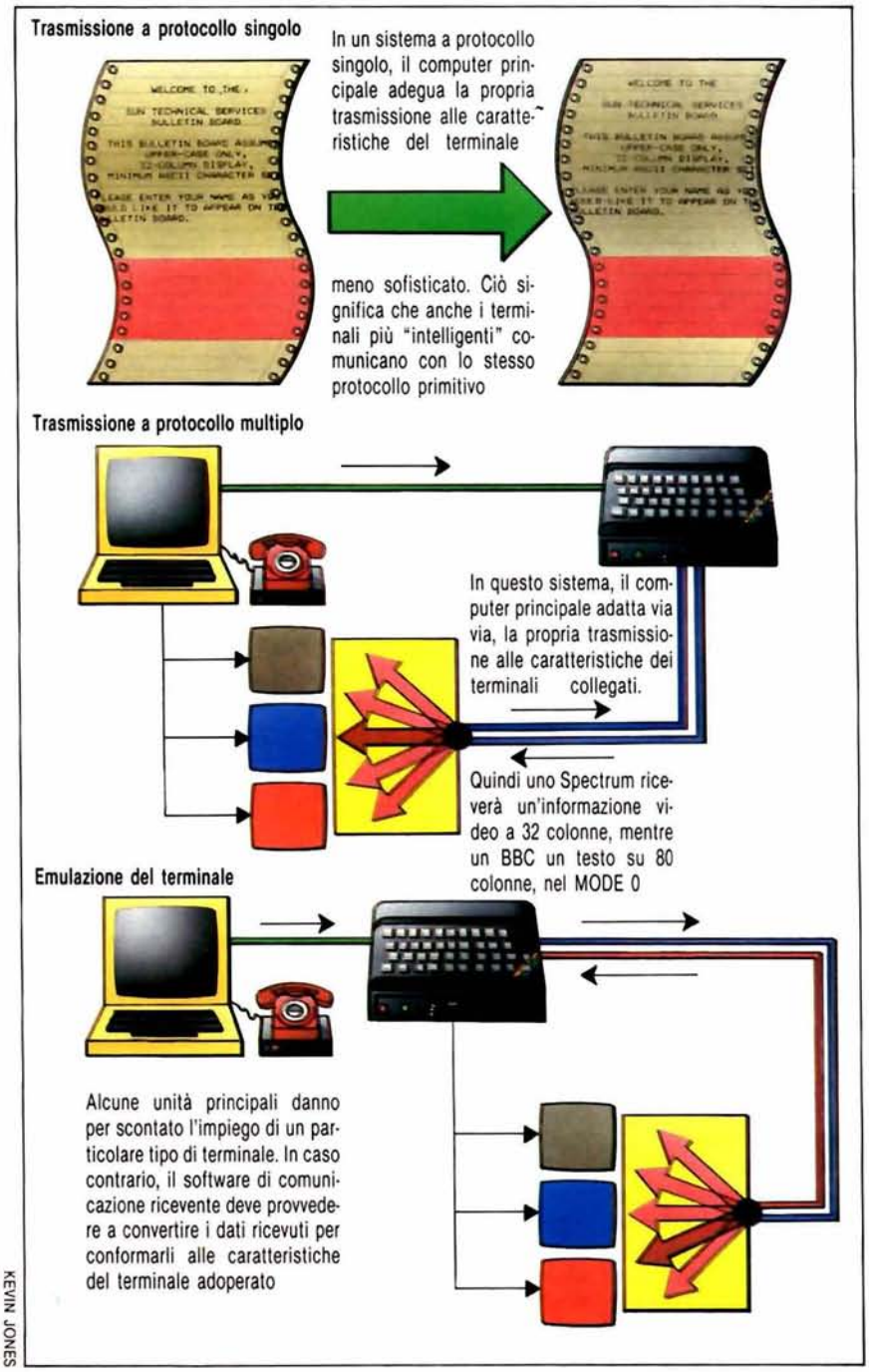
Nella pubblicità di molti modem si afferma che i micro portatili ed i modem acustici a pile possono essere impiegati in qualsiasi situazione. In realtà, la loro utilizzazione in alcuni casi si dimostra piuttosto difficile, se non addirittura impossibile: non sempre le cornette dei telefoni pubblici sono adatte, né lo sono quelle, di alcuni impianti interni quali uffici, alberghi, ecc.

Tipi di protocollo

Le tecniche di protocollo maggiormente usate sono tre: la prima impiega alcuni parametri di base comuni a più terminali (velocità di trasmissione ridotta, numero di colonne limitato, nessuna funzione grafica o di colore). La seconda adegua il protocollo di uscita del computer principale alle caratteristiche tecniche dei diversi terminali collegati (sistema usato dalla maggior parte dei Bulletin Board). La terza, tramite una particolare tecnica software detta "emulazione del terminale", costringe un terminale ad imitare il comportamento di un altro

protocollo di base usato dai terminali corrisponde al formato standard della telescrivente. Questo possiede solo le lettere maiuscole, non esistono comandi di formattazione della stampa, è privo delle funzioni grafiche e di colore ed è piuttosto lento. Il secondo è più diffuso e viene adottato dalla maggior parte dei "Bulletin Board". Il terzo, tipico dei sistemi d'elaborazione universitari e commerciali, si basa su una tecnica software detta "emulazione del terminale".

Come suggerisce lo stesso nome, questa tecnica induce un micro a simulare il comportamento di un terminale. L'apposito software "di emulazione" converte i caratteri di controllo inviati dal computer (ad esempio, quelli per posizionare il cursore) in comandi comprensibili all'apparecchio ricevente e viceversa.



Quasi tutti i sistemi sono in grado di interpretare un sottoinsieme di caratteri di controllo ASCII (vedere la tabella dei caratteri ASCII a pag. 557). I caratteri usati con maggior frequenza sono: Control-S (cod. ASCII 19, detto XOFF), che interrompe temporaneamente la ricezione dei dati; Control-Q (cod. ASCII 17, detto XON) riabilita la ricezione; Control-J (cod. ASCII 10, detto LF), che inserisce un'interlinea senza ritorno-carrello; Control-G (cod. ASCII 7, detto BELL), che aziona il segnalatore acustico della consolle per richiamare l'attenzione dell'operatore al terminale ricevente.

Il protocollo XModem

Esaminata la trasmissione di caratteri ASCII, vediamo adesso come avviene l'invio del software. Per i programmi BASIC la procedura è molto semplice. La maggior parte dei micro possiede particolari istruzioni per decodificare i token delle istruzioni in uno dei formati ASCII: il Micro BBC usa *SPOOL, il Commodore 64 il comando LIST per i file su disco o nastro, mentre il Tandy ricorre all'istruzione CSAVE <nomefile>,A e così via. Il file così decodificato viene quindi trasmesso e in seguito ricodificato dal computer ricevente.

Purtroppo i file binari eseguibili (ad esempio i .COM del CP/M) non possono essere trasformati in formato ASCII ed il tentativo di trasmetterli provoca risultati imprevedibili. Per rimuovere questo ostacolo è stato sviluppato un protocollo, detto "XModem", col quale i file CP/M letti dal dischetto vengono trasmessi usando il formato binario standard. Naturalmente, anche il terminale ricevente deve possedere il protocollo XModem. Le applicazioni del software di comunicazione vengono descritte in seguito: prima occorre esaminare le operazioni consentite dall'impiego di un modem.

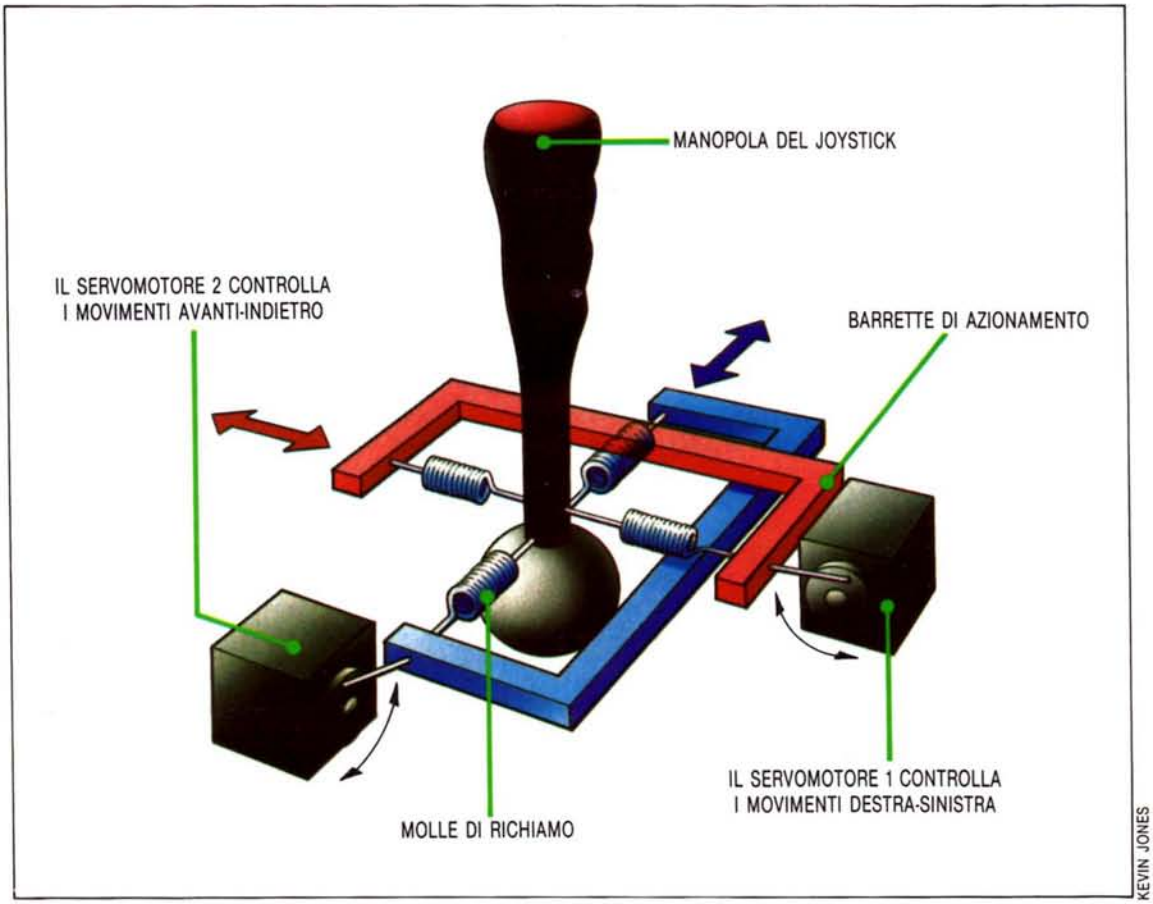
Il termine "posta elettronica" indica quei sistemi che consentono agli utenti di scambiarsi messaggi, inviandoli ad un computer centrale che li conserva in memoria finché non vengono richiesti dai destinatari.

I possessori di micro trovano particolarmente utili due delle applicazioni consentite da questi sistemi: la prima riguarda lo scambio di programmi via telefono, che si dimostra molto più rapido, funzionale ed economico dell'invio di un nastro a cassetta per posta. La seconda interessa gli appassionati dei "Bulletin Board". Questi consentono di inviare comunicazioni agli altri utenti, chiedere aiuti di carattere tecnico, raccontarsi barzellette, ricevere programmi, partecipare a giochi e così via. I Bulletin Board sono in genere gestiti da semplici appassionati e di solito sono gratuiti (eccetto alcuni che obbligano al versamento di una modesta quota per le spese di funzionamento).

Nella prossima parte dedicata all'argomento viene fatta una breve panoramica sui modem e sul software di comunicazione, oltre ad un approfondimento in materia di Bulletin Board.

Usare i servomotori

Come controllare simultaneamente più di un servomotore



Feedback dal joystick

Di solito i joystick vengono usati per comunicare informazioni direzionali al software del sistema. Un'applicazione possibile dei servomotori è quella di impiegarli per consentire un flusso di informazioni nel senso opposto: dal software di controllo alla manopola del joystick. L'azione combinata dei due servomotori mostrati in figura permette al computer di un simulatore di volo di muovere la cloche in tutte le direzioni, fornendo una serie di importanti informazioni di feedback. Una cloche che interagisce con la mano del pilota in risposta ai movimenti di controllo che egli esegue migliora il processo di simulazione e fornisce una serie di informazioni visive e tattili

Il buffer box può usare solo quattro delle otto linee per i dati collegabili ai motori; tuttavia è possibile adoperare tutte e otto le linee duplicando la circuiteria del box di uscita precedentemente realizzato.

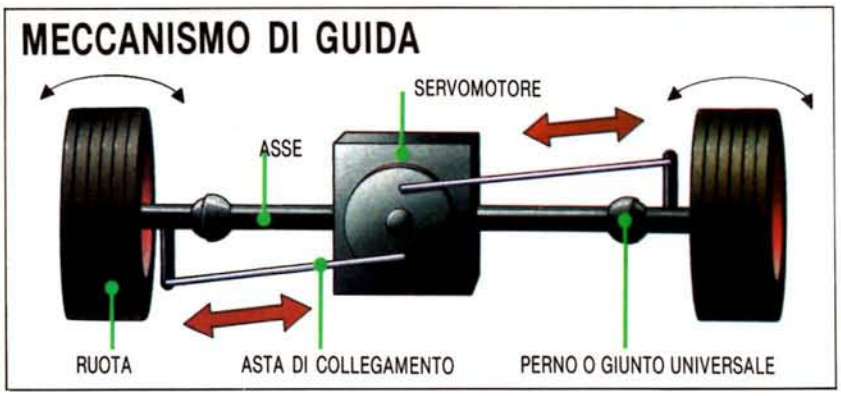
Per controllare simultaneamente otto motori, occorre qualche lieve modifica all'algoritmo, sviluppato a pag. 1374, per il controllo di un singolo motore.

La trasmissione di tutti i segnali inizia simultaneamente, ma il secondo ciclo di attesa va sostituito con la consultazione di una tabella, per la quale vengono qui riservate 255 locazioni di memoria, tutte inizialmente poste a 255 ($FF).

Nella tabella vengono registrate le configurazioni di bit occorrenti quando un motore viene spento. Se, ad esempio, disattiviamo la linea 2 dopo un conteggio di 20, il valore binario 11111111 ($FF) contenuto nella ventesima locazione della tabella viene sostituito da 11111011 ($FB). Per operare questa sostituzione il programma Assembly pone il nuovo valore in AND rispetto a quello conservato nella particolare locazione. Dopo l'inserimento in tabella di tutte e otto le configurazioni di bit, inizia il ciclo d'attesa. Questa volta, però, gli elementi della tabella vengono posti in AND rispetto alla porta utente.

L'algoritmo per il controllo multiplo dei motori è il seguente:

1) Specificare l'angolo di ciascun motore memorizzando i valori angolari in otto byte (da ANGOLO+0 ad ANGOLO+7).
2) Porre a 1 tutti i bit dei dati nella porta utente, per trasmettere i segnali contemporaneamente.
3) Inserire le configurazioni nella tabella di consultazione.
4) Eseguire una pausa di un millisecondo.
5) Caricare il valore binario 11111111 ($FF) nell'accumulatore, quindi (a turno) porre l'accumulatore in AND con ciascun elemento della tabella. Quando viene rilevata un'eccezione (ossia un bit della configurazione vale 0), è azzerato il bit di dati corrispondente. Poiché la procedura è ripetuta fino alla fine della tabella, il bit resta "spento" fino al termine.



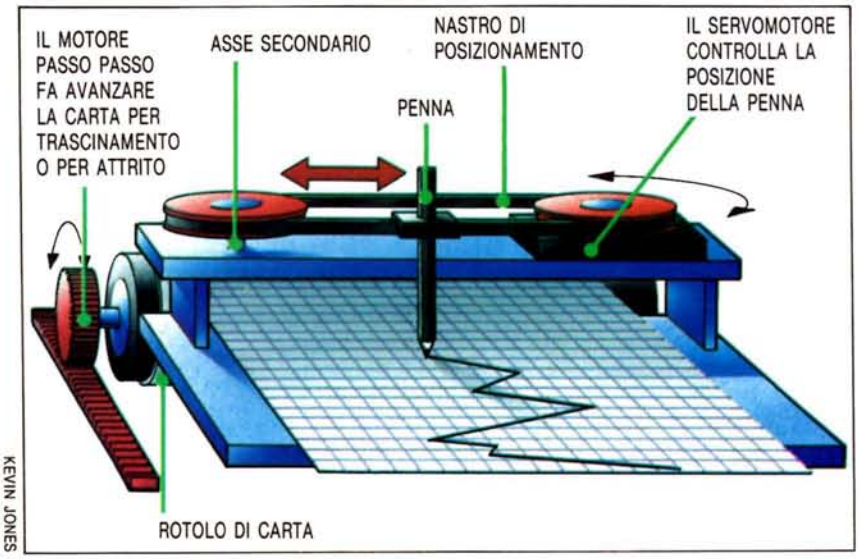
Il meccanismo di guida si basa sul controllo indipendente di due motori passo passo bidirezionali. Una diversa soluzione può essere quella di adottare un motore passo passo per azionare il veicolo ed un servomotore per controllare la direzione del movimento. In figura: un servomotore montato sull'asse delle ruote, dotato di due aste che agiscono sull'inclinazione delle stesse. Altrimenti si può usare una ruota dentata, fissata al perno centrale del servomotore, che azioni una catena di trasmissione

Registratore su carta

Un registratore su carta si può realizzare usando un servomotore, per muovere una penna, e un motore passo passo per il trascinamento della carta. La rotazione del perno centrale del servomotore viene trasformata in un movimento lineare della penna tramite un nastro di posizionamento ad anello. Il movimento della penna viene comandato in base a una variabile, che contiene i valori dell'asse verticale, ad esempio: la temperatura o la pressione barometrica. Il trascinamento della carta, invece, viene regolato in base al tempo

6) Riportare le eccezioni in tabella al valore binario 11111111, preparandosi a ricevere il segnale successivo.

I listati forniti per il Micro BBC presentano la stessa routine iniziale (linee 10-280) in entrambi i programmi. Il primo listato controlla un solo servomotore, collegato ad una delle linee della porta utente. Una procedura di inizializzazione assembla la routine di gestione degli eventi, prima che il programma principale la esegua, abilitando l'evento 5.
Il sistema operativo del Micro BBC possiede un temporizzatore che scandisce il tempo in centesimi di secondo. Selezionandolo per 2 centesimi di secondo (due interrupt di 10 millisecondi ciascuna) ed utilizzando il vettore di "evento" (consultare il Manuale del Micro BBC), al momento opportuno il processore salta alla sezione del programma che genera i segnali. Poiché il sistema operativo è predisposto per la gestione di eventi, il programma usa una semplice istruzione RTS, e non RTI, per recuperare il controllo al termine della subroutine.
Il secondo listato consente di controllare più motori contemporaneamente. Un ciclo iterativo in BASIC inizializza la tabella di consultazione con valori $FF. Se gli elementi della tabella vengono inviati in successione alla porta utente, i segnali hanno tutti una durata di due millisecondi. Tuttavia possono presentarsi delle eccezioni (ossia particolari configurazioni di bit), nel qual caso i corrispondenti motori vengono spenti. Dette eccezioni vengono inserite nella tabella, iniziando dal motore 7, in base ad un offset (contenuto nel registro X) proporzionale alla durata del segnale. La tabella viene quindi trasmessa alla porta utente, accedendo sequenzialmente ai suoi elementi. L'indirizzamento avviene in modo indiretto, ricorrendo questa volta ad un indirizzamento indiretto post-indicizzato (il processore somma il valore contenuto nel registro X all'indirizzo trovato in un vettore di pagina 0).
Le configurazioni di bit (qui chiamate eccezioni) trasmesse alla porta utente per controllare i motori sono le seguenti: il binario 01111111 spegne il motore 7, 10111111 il motore 6, 11011111 il motore 5 e così via. Queste vengono generate caricando $FF nel registro A e azzerando il flag di riporto (Carry). Quindi, poiché la routine elabora le eccezioni l'una dopo l'altra, i bit vengono ruotati di una posizione verso destra. Alla prima rotazione, il Carry viene spostato nel bit 7, il bit 7 va nel 6 e così via per terminare col bit 0, che finisce nel flag di Carry. Le diverse configurazioni necessarie per spegnere i motori vengono temporaneamente salvate sullo stack. Ogni elemento della tabella viene posto in AND rispetto al precedente per assicurarsi che un segnale, una volta spento, rimanga tale.
Dopo aver creato il codice macchina, l'evento 5 viene cancellato. Infine, per selezionare i motori, si devono premere contemporaneamente il tasto Shift ed uno dei tasti numerati, mentre per posizionarli nel modo desiderato basta premere i tasti da 1 a 9.



KEVIN JONES

Uso dei listati

Per il BBC, si trascrivano i relativi listati per il controllo singolo e multiplo dei servomotori, si salvino su nastro o disco e poi si eseguano. Entrambi i programmi vengono adoperati in congiunzione alla Routine Iniziale Comune (linee 10-750).
Sul Commodore 64, il secondo algoritmo usa gli stessi tasti di controllo del listato BBC. Il programma di chiamata in BASIC consente di regolare in modo indipendente l'angolazione di ciascun motore: anche in questo caso si usa il tasto Shift assieme a uno dei tasti numerici (1-8) per selezionare il motore. Il posizionamento è controllato con la pressione dei tasti (1-9).
Se si possiede un programma assembler, si può impiegarlo per convertire il listato sorgente in file di codice macchina eseguibile. Questo verrà successivamente caricato in memoria usando il programma BASIC di chiamata. In alternativa, si può trascrivere, salvare ed eseguire il programma caricatore in BASIC, nelle cui DATA si trova per il codice macchina occorrente (in tal caso eliminare le linee 30 e 40 del programma di chiamata in BASIC). Si impartisca un NEW, prima di caricare e si esegua il programma BASIC di chiamata.

CONTROLLO MULTIPLO DEI SERVOMOTORI SU COMMODORE 64

CODICE SORGENTE

```
1000 ;++++++++++++++++++++++++++++++
1010 ;++++++++++++++++++++++++++++++
1020 ;++                            ++
1030 ;++ CONTROLLO MULTIPLO SERVO ++
1040 ;++    MOTORI SUL COMMODORE    ++
1050 ;++                            ++
1060 ;++++++++++++++++++++++++++++++
1070 ;++++++++++++++++++++++++++++++
1080 ;
1090 PORTA  = 56577    ;REG. DATI DELLA PORTA UTENTE
1100 ANGOLO = 12288    ;LOCAZIONE DEL VALORE ANGOLARE
1110 PZERO  = $FB      ;PUNTAT. DI PAGINA O PER TABELLA
1120 ;
1130 *=$0334
1140 ;
1150     SEI           ;DISATTIVA GLI INTERRUPT
1160     LDA $0314     ;ATTUALE VETTORE IRQ
1170     LDX $03C4
1180     STA $03C4
1190     STX $0314
1200     LDA $0315
1210     LDX $03C5
1220     STA $03C5
1230     STX $0315
1240 ;
1250 ;++++ INIZIALIZZA LA TABELLA ++++
1260 ;
1270     LDA #$FF
1280     LDY #$00
1290 TABELLA
1300     STA (PZERO),Y
1310     DEY
1320     BNE TABELLA
1330     CLI           ;RIATTIVA GLI INTERRUPT
1340     RTS
1350 ;
1360 ;++++ ROUTINE DI GESTIONE DELL'EVENTO ++++
1370 ;
1380     PHP
1390     PHA           ;SALVA I REGISTRI
1400     TYA           ;SULLO STACK
1410     PHA
1420     TXA
1430     PHA
1440 ;
1450 ;++ INVIA IL SEGNALE, CON ALCUNI MOTORI E' POSSIBILE
1460 ;INIZIARE A TRASMETTERE PRIMA DI CARICARE
1470 ;LA TABELLA RIDUCENDO IL CICLO D'ATTESA ++
1480 ;
1490     LDA #$FF
1500     STA PORT
1510 ;++ CARICA LE ECCEZIONI IN TABELLA ++
1520     LDX #$07
1530     LDA #$FF
1540     CLC
1550 ECCEZ
1560     ROR A
1570     PHA           ;CONFIGURAZIONE DEI BIT
1580     LDY ANGOLO,X  ;RICEVE L'OFFSET DEL MOTORE X
1590     AND (PZERO),Y ;MANTIENE CONFIGURAZ. ESISTENTE
1600     STA (PZERO),Y ;MA MODIFICATA PER IL MOTORE X
1610     PLA
1620     DEX
1630     BPL ECCEZ
1640 ;++ TABELLA CARICATA ++
1650     LDY #$30
1660 ATTESA
1670     DEY           ;TEMPO DI
1680     BNE ATTESA    ;ATTESA
1690 ;
1700     LDA #$FF      ;TUTTI I SEGNALI ACCESI
1710     LDY #$00
1720 CICLO
1730     AND (PZERO),Y ;MA PONE IN AND CON CIASCUN ELEMENTO
1740     STA PORTA     ;DELLA TABELLA IN SUCCESSIONE
1750     INY
1760     BNE CICLO
1770 ;
1780     LDX #$07
1790     LDA #$FF
1800 CLEAR
1810     LDY ANGOLO,X  ;CANCELLA TUTTE LE ECCEZIONI
1820     STA (PZERO),Y
1830     DEX
1840     BPL CLEAR
1850 ;++ TUTTI I SEGNALI DOVREBBERO ESSERE TERMINATI ++
1860     PLA
1870     TAX
1880     PLA           ;REGISTRI ALLO STATO INIZIALE
1890     TAY
1900     PLA
1910     PLP
1920     JMP $EA31
```

PROG. BASIC DI CARICAMENTO

```
10 REM **** CARICATORE BASIC PER ****
20 REM **** SERVO MOTORI MULTIPLI ****
30 :
40 FOR I=820 TO 922
50 READ A:POKE I,A
60 CC=CC+A
70 NEXT I
80 READ CS:IF CS<>CC THEN PRINT
   "ERRORE DI CHECKSUM":STOP
100 DATA120,173,20,3,174,196,3,1,41,196
110 DATA3,142,20,3,173,21,3,174,197,3
120 DATA141,197,3,142,21,3,169,2,55,160
130 DATA0,145,251,136,208,251,88,96,8
140 DATA72,152,72,138,72,169,255,141,1
150 DATA221,162,7,169,255,24,106,72
160 DATA188,0,48,49,251,145,251,104
170 DATA202,16,243,160,48,136,20,8,253
180 DATA169,255,160,0,49,251,141,1,221
190 DATA200,208,248,162,7,169,25,5,188
200 DATA0,48,145,251,202,16,248,104
210 DATA170,104,168,104,40,76,49,234
220 DATA13072:REM*CHECKSUM*
```

PROG. BASIC DI CHIAMATA

```
10 REM ** SERVO MOTORI MULTIPLI **
20 :
30 DN=8: REM PER CASSETTA DN=1
40 IF A=0 THEN A=1:LOAD"MULTISER
   V.HEX",DN,1
50 POKE 778,88: POKE779,3: REM
   PUNTAT. ALLA ROUTINE DI
   GESTIONE EVENTO
60 POKE 251,0:POKE252,49: REM
   PREPARA I PUNTAT. DI PAGINA 0
70 DDR=56579:POKE DDR,255: REM
   TUTTE LINEE IN USCITA
80 MC=820: SYS MC
90 :
100 GET K$:IF K$="" THEN 100: REM
    ATTENDE CHE VENGA PREMUTO
    UN TASTO
110 REM ** MODIFICA LA POSIZIONE
    DI UN MOTORE **
115 AK=ASC(K$)
120 IF AK>48 AND AK<58 THEN POKE
    12288+SERVO, VAL(K$)*20
130 IF AK>32 AND AK<40 THEN
    SERVO=ASC(K$)-35
140 IF K$<>"E" THEN 100: REM BATTERE
    "E" PER USCIRE
```

Controllo di un solo servomotore

```
  755 REM ************************
  756 REM  ASSEMBLA IL CODICE SUCCESSIVO
  757 REM ************************
10000 DEF PROCinizializza
10010 DIM spazioI 200
10020 FOR I=0 TO 2 STEP 2
10030   porta=&FE60
10040   P%=spazioI
10050   angolo=P%
10060   P%=P%+1
10070   OPT I
10080   .gestioneeventi
10090   \salva i registri
10100   PHP:PHA:TYA:PHA:TXA:PHA
10110   LDA #&04
10120   LDE &ttimer
10130   LDY &ytimer
10140   JSR &FFF1 \azzera il temporizzatore
10150   LDA #&FF
10160   STA porta
10170   \attende circa 1 millisec.
10180   LDY #&FF
10190   .LOOP DEY
10200       BNE LOOP
10210   \esclude il segnale
10220   LDY angolo
10230   .LOOP1 DEY
10240       BNE LOOP1
10250   \arresta tutti i segnali in uscita
10260   LDA #&0
10270   STA porta
10280   PLA:TAX:PLA:TAY:PLA:PLP
10290   RTS
10300   ]
10310   NEXT I
10320 REM punta a gestioneeventi
10330 !&220=gestioneeventi OR (!&220 AND &FFFF0000)
10340 ENDPROC
```

Routine iniziale comune

```
  10 MODE 0
  15 REM ****************************************
  16 REM  METTE A PUNTO IL TEMPORIZZATORE ECC.
  17 REM ****************************************
  20 osbyte=&FFF4
  30 A%=&97 :X%=&62 :Y%=&FF
  40 CALL osbyte: REM porta B in uscita
  50 CLS
  60 DIM p%(8)
  70 DIM timer%12, read% 12
  80 xtimer=timer% MOD 256
  90 ytimer=timer% DIV 256
 100 xread=read% MOD 256
 110 yread=read% DIV 256
 120 PROCinizializza
 130 FOR I%=angolo TO angolo+8: angolo?I%=128: NEXT
 140 t=.02 :REM secondi tra i segnali
 150 tempo%=&FFFFFFFF -(t*100)+1
 160 timer1?4=&FF :REM carica il byte piu' alto
 170 !timer1=tempo% :REM prepara il temporizzatore, abilita gli
     eventi
 180 =PI/4,5
 190 A%=4 :I%=xtimer :Y%=ytimer :CALL &FFF1
 195 REM ****************************
 196 REM  TASTI DI CONTROLLO (BASIC)
 197 REM ****************************
 200 CLS
 210 PRINT "Premi SHIFT+NUMERO per selezionare il motore"
 220 PRINT "Premi un NUMERO per selezionare l'angolo"
 225 motore=1
 230 REPEAT
 240   A=GET
 250   IF A%&2F AND A<&3A THEN a=(A-&30)*10+(255/90): angolo?
       (motore-1)=a: PRINT TAB (10,motore);"motore " motore
       " angolo "a*(90/255)
 260   IF A%&20 AND A<&2A THEN motore=A-&20
 270 UNTIL 0
 280 END
 750
```

Controllo di più servomotori

```
 755 REM ************************
 756 REM  Assembla il codice macchina
 757 REM ************************
 760 DEF PROCinizializza
 770 DIM spazioI 600
 780 FOR I%=0 TO 3 STEP 3
 790   pagina=&70  :REM a disposizione dell'utente
 800   porta=&FE4b: numero=&FFF1
 810   P%=spazioI
 820   angolo=P% :P%=P%+8 :REM 8 motori
 830   tabella=P% :P%=P%+256 :REM 256 possibili lunghezze segnale
 835   FOR I%=tabella TO tabella+&100: ?I%=&FF: NEXT
 840   tabloI=tabella MOD 256
 850   tabhiI=tabella DIV 256
 860   ?pagina=tabloI: pagina?1=tabhiI
 870   OPT I%
 880   .gestioneeventi
 890   PHP:PHA:TYA:PHA:TXA:PHA
 900   LDA #&04
 910   LDE &ttimer
 920   LDY &ytimer
 930   JSR numero
 940     \invia il segnale; con alcuni motori e' possibile
 950     \iniziare a trasmettere prima di caricare la tabella
 955     \riducendo il ciclo d'attesa
 960   LDA #&FF:STA porta
 970     \carica le eccezioni in tabella
 980   LDE #&7 :LDA #&FF :CLC \ignora la configurazione di bit
 990   .eccezioni
1000       ROR A :PHA \configurazione di bit
1010       LDY angolo,I \riceve l'offset corrisp. all'angolo del motore I
1020       AND (pagina),y \mantiene la  config. esistente
1030       STA (pagina),y \ma modificata per il motore I
1040       PLA :DEX
1050       BPL eccezioni
1060     \tabella caricata, tempo di attesa
1070   LDY #&60
1080   .attesa DEY
1090       BNE attesa
1100   LDA #&FF  \tutti i segnali accesi
1110   LDY #&0
1120   .ciclo  AND (pagina),y \ma pone in AND con ciascun
1130           STA porta \elemento della tabella in successione
1140           INY
1150           BNE ciclo
1160   LDE #&7 :LDA #&FF
1170   .cancella
1180           LDY angolo,I \cancella tutte le eccezioni
1190           STA (pagina),Y
1200   DEX
1210   BPL cancella
1220     \tutti i segnali dovrebbero essere terminati
1230   PLA:TAX:PLA:TAY:PLA:PLP
1240   RTS
1250   ]
1260 NEXT%
1270 !&220=gestioneeventi OR (!&220 AND &FFFF0000)
1280 ENDPROC
```

Didattica d'oltremanica

La seconda parte della breve panoramica sul software per l'infanzia

Lo *Story Machine* della Spinnaker è dedicato a bambini di età compresa tra 5 e 9 anni. Questo package didattico ha come obiettivo l'insegnamento della sintassi e dell'ortografia della lingua inglese.
Il programma contiene un dizionario di 52 parole, cinque delle quali sono riservate a nomi propri: il nome di un ragazzo, di una ragazza, di un gatto, di un cane e di un personaggio immaginario chiamato "bumpus".
I nomi propri devono essere scelti dall'utente all'inizio del programma.
Usando le parole del dizionario, il bambino scrive brevi frasi che vengono poi "recitate" sullo schermo dai personaggi. Ad esempio, la frase «Maurizio bacia i fiori» genera l'immagine di un ragazzo e di alcuni fiori. Quando il ragazzo tocca i fiori, sullo schermo compaiono piccoli cuori lampeggianti che rappresentano i baci.

Story machine



Il computer rifiuta parole contenenti errori ortografici o usate impropriamente. In tal caso visualizza un messaggio che fa notare al bambino l'errore e lo invita a correggere la parola o a cambiarla. Alla fine di ogni frase (ossia dopo ogni punto), il programma passa alla rappresentazione.
Una particolare opzione consente di visualizzare in sequenza più frasi che compongono una breve storia.
Lo *Story Machine* offre altre due opzioni: una consente di scrivere una storia con l'aiuto del programma (il bambino ed il computer scelgono a turno le parole), l'altra lascia che il computer scriva l'intera storia da solo. Ciò è possibile perché le parole del dizionario sono suddivise per gruppi (nomi, verbi, preposizioni, ecc.) e il programma possiede un rigoroso algoritmo grammaticale per la scrittura di frasi.
La maggior parte delle parole del dizionario e delle strutture grammaticali usate dal programma è abbastanza semplice, soprattutto i verbi che descrivono azioni facilmente traducibili in immagini, ad esempio "salta" oppure "vai".
A parte il verbo immaginario "zot", che sembra significare "colpire" o "battere", il pacchetto si attiene strettamente alla filosofia della Spinnaker, che rifiuta qualsiasi riferimento anche generico ad azioni violente verso esseri umani o animali.
Il programma segue due fondamentali regole grammaticali: una frase deve contenere un soggetto, un verbo ed un complemento oggetto disposti in modo logico. Le concordanze del singolare e del plurale devono essere rispettate. Dal punto di vista sintattico, il package presenta qualche lacuna, del resto sarebbe troppo complesso prevedere tutte le variazioni presenti nella lingua inglese. Questi limiti portano saltuariamente alla mancata individuazione di errori di lieve entità, tuttavia un programma con finalità didattiche dovrebbe essere sempre corretto.
Un secondo pacchetto della Spinnaker, il *Kids On Keys*, è dedicato ai bambini di età compresa tra i 3 e i 9 anni.
Il programma mira ad insegnare ai piccoli utenti come riconoscere le lettere dell'alfabeto, le parole ed i numeri. È composto da tre diversi giochi didattici di vario livello di difficoltà.
Il primo gioco fa scorrere una lettera sullo schermo e chiede al bambino di battere il tasto corrispondente, prima che la lettera scompaia dal video.
Dopo quindici lettere il programma visualizza un palloncino contenente una parola: il bambino deve riscriverla in modo esatto prima che il palloncino raggiunga il margine inferiore dello schermo. Oltre al riconoscimento delle lettere e delle parole, questo programma permette al bambino di imparare a conoscere la tastiera divertendosi.
Il secondo gioco fa scorrere sullo schermo una serie di figure (sprite espansi): ancora una volta il bambino deve scriverne il nome prima che esse raggiungano il margine inferiore dello schermo. Segue quindi una prova speciale, che mostra le figure solo per metà, rendendo il loro riconoscimento molto più difficile.
A questo punto il gioco sembra diventare un test per riflessi, nel quale il giocatore cerca di

Kids on keys



premere il più rapidamente possibile le lettere giuste. Poiché il programma manca della funzione di cancellazione, non è possibile correggere le parole scritte in modo inesatto. Purtroppo, poiché la grafica di alcuni sprite lascia a desiderare e complica notevolmente il riconoscimento delle figure, c'è il rischio che il programma confonda le idee al bambino. Ad esempio, può accadere che il bambino, riconoscendo una figura umana, scriva correttamente la parola "uomo": l'immagine, tuttavia, continua a muoversi verso il basso, perché il programma abbina a quello sprite la parola "ragazzo". In questo caso egli può pensare di aver commesso un errore d'ortografia dato che il programma non fornisce alcuna spiegazione sul tipo di errore commesso.
Inoltre, quando lo sprite scompare dallo schermo, il programma non visualizza la risposta esatta, lasciando così il bambino nel dubbio.
Nel terzo gioco vengono visualizzate cinque figure e una parola: il bambino deve abbinare ogni parola all'immagine corrispondente. Questo gioco punta meno dei precedenti alla rapidità della risposta, ma presenta anch'esso una grafica piuttosto grossolana.

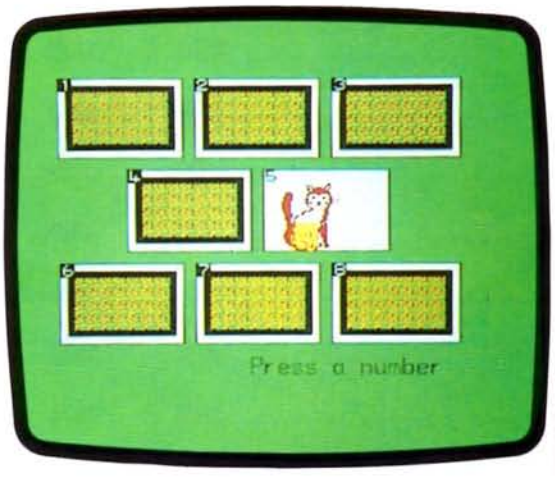
Impariamo a leggere

La casa editrice inglese MacMillan, specializzata in testi didattici, ha sviluppato, assieme alla Sinclair Research, una serie di package per lo Spectrum. Il gruppo di cinque programmi, chiamato *Learn To Read* (impariamo a leggere), deriva da un corso didattico tradizionale chiamato *Gay Way*, pubblicato con grande successo dalla stessa MacMillan, e adottato in molte scuole elementari inglesi. Il *Gay Way* è un corso decisamente orientato verso le esignenze dello studente e insegna a leggere senza mai forzare i tempi di apprendimento individuali.
I cinque programmi del *Learn To Read* rappesentano un logico sviluppo del corso originale: l'uso del computer ne esalta le caratteristiche di individualità.
I cinque pacchetti compongono un corso completo di approccio alla lettura. Il primo programma presenta sei animali (il topo Deb, la volpe Sam, ecc.) che vivacizzano lo studio ed accompagnano lo studente per tutto il corso.
I programmi partono dal presupposto che il bambino non sappia né leggere né usare un computer. Per questo motivo la scelta delle diverse opzioni all'interno di un programma è stata semplificata al massimo. Il primo package, ad esempio, visualizza un menu di cinque opzioni sulle quali compare a rotazione un grosso rettangolo lampeggiante. L'utente attende che il rettangolo si sposti sull'opzione prescelta e quindi preme un tasto qualsiasi per selezionarla.
A questo punto il computer rivolge al bambino una domanda e attende la sua risposta. Se il bambino commette un errore, ha la possibilità di ripetere la risposta più volte, prima che il computer visualizzi quella esatta.
Le immagini dei sei animali vengono create con grafica in alta risoluzione e sono molto più curate di quelle descritte nei pacchetti precedenti. La serie *Learn To Read* è forse meno divertente di altri pacchetti, ma il suo valore didattico è indiscutibile e con l'andar del tempo si dimostra molto più gratificante per il bambino.

Learn to read



Learn to read



IAN McKINNELL

DATABASE

Ecco la quarta tabella sulle funzioni delle routine ROM del Micro BBC, assieme ai relativi indirizzi

DESCRIZIONE	INDIRIZZO
Decrement MAN#1 decrementa MAN#1	A4C7
Delete a line from program cancella una linea di programma	BC2D
Delete bytes in buffer cancella i byte nel buffer	887C
Discard the IAC from stack elimina IAC dallo stack	BDFF
END statement routine routine istruzione END	8AC8
ENDPROC statement routine routine istruzione ENDPROC	9356
ENVELOPE statement routine routine istruzione ENVELOPE	B472
EOF function routine routine funzione EOF	ACB8
EOR operator routine routine operatore EOR	9B55
EQUX in assembler direttiva EQUX (assembler)	883A
ERL function routine routine funzione ERL	AF9F
ERR function routine routine funzione ERR	AFA6
EVAL function routine routine funzione EVAL	ABE9
EXP function routine routine funzione EXP	AA91
EXT function routine routine funzione EXT	BF46
Error handler routine trattamento errori	B402
Evaluate expression calcolo di espressione	9B29
Evaluate integer expression calcolo espressione intera	8821
Evaluate operand calcolo operando	ADEC
Exchange FAC#1 and (&4B) scambio tra FAC#1 e (&4B)	A4D6
Exponentiation routine routine esponenziale	9E35
Expression evaluator (integer) calcolo espressioni (intere)	8821
FALSE function routine routine funzione FALSE	AECA
FN function routine routine funzione FN	B195
FOR statement routine routine istruzione FOR	B7C4

INDIRIZZO	DESCRIZIONE
94FC	**Create a catalogue entry** creazione elemento nel catalogo
9531	**Clear a new variable** azzera una nuova variabile
9559	**Get an array/PROC/FN name** prende un nome di matrice/PROC/FN
95A5	**! unary operator** operatore di unità !
95A7	**? unary operator** operatore di unità ?
95B0	**$ unary operator** operatore di unità $
95DD	**Get a variable name** prende un nome di variabile
967B	**? binary operator** operatore binario ?
967F	**! binary operator** operatore binario !
97EB	**Line number decoding** decodifica numeri di linea
9857	**Ceck for end of statement** verifica fine istruzione
9880	**Move to next statement** passaggio all'istruzione successiva
98C2	**IF statement routine** routine istruzione IF
991F	**Print the IAC** routine istruzione IAC
9970	**Search for line program** cerca una linea nel programma
99BE	**Integer division** divisione tra interi
9A50	**Real comparison** confronto tra numeri reali
9A9E	**Compare two items** confronto tra due elementi
9AA2	**Integer comparison** confronto tra interi
9AE7	**String comparison** confronto tra stringhe
9B29	**LEVEL 1 of expression evaluator** calcolo espressioni LIVELLO 1
9B3A	**OR operator routine** routine operatore OR
9B55	**EOR operator routine** routine operatore EOR
9B72	**LEVEL 2 of expression evaluator** calcolo espressioni LIVELLO 2
9B7A	**AND operator routine** routine operatore AND

Due Sharp da taschino

Casio e Sharp: due ditte giapponesi in forte concorrenza nei computer tascabili

La strategia di mercato della Sharp è antitetica a quella della Casio: la prima fabbrica due computer tascabili diversi tra loro, mentre la seconda produce tre apparecchi sostanzialmente simili.
I due computer Sharp non sono destinati a farsi reciproca concorrenza: il PC-1251 è il più piccolo computer sul mercato ed ha un prezzo contenuto mentre il PC-1500, assai più grande e potente, è il più costoso computer tascabile tra quelli commercializzati dalle due compagnie.
Lo Sharp PC-1251 pesa appena 115 grammi e misura 135x70x10 mm. Le dimensioni della tastiera sono meno della metà di quelle di una tastiera comune: i tasti sono larghi solo 4 mm. ma ben distanziati e di facile impiego. A destra della tastiera alfabetica c'è una tastierina numerica, che permette di usare il computer come una comune calcolatrice tascabile.
Il PC-1251 possiede un display a cristalli liquidi (a 24 caratteri) la cui angolazione può essere variata, a seconda dell'illuminazione esterna, ruotando un dischetto posto sul retro dell'apparecchio. A destra del display c'è l'interruttore "Mode Selector", per la selezione dei tre modi di funzionamento: il primo (RSV) viene usato per definire le funzioni di alcuni tasti, il secondo (PRO) per la programmazione ed il terzo per l'elaborazione dei programmi BASIC, quando il tascabile funziona come computer. L'interruttore serve anche per spegnere la macchina.
Il dialetto BASIC adottato non raggiunge lo standard dei comuni micro, ma è più che adeguato per un apparecchio così ridotto. Possiede alcuni comandi (quali ASC e CHR$) assenti in alcuni tascabili Casio, ma la struttura IF...THEN è priva dell'opzione ELSE. Come sui più piccoli computer prodotti dalla Casio, il BASIC del PC-1251 usa gli stessi nomi a lettera singola per stringhe e variabili. Quindi, se si usa la variabile A come variabile numerica, non è possibile usare una variabile stringa A$.
Questo particolare dialetto BASIC dispone solo di nove messaggi d'errore: un po' pochi per essere di significativo aiuto in un programma.
Una funzione accessoria, il comando ADD, permette di proteggere i programmi con una "password" (parola d'ordine). Il BASIC consente l'impiego di numeri di linea compresi tra 0 e 999.
Dopo aver selezionato il modo per l'immissione dei programmi, due tasti per il cursore consentono di far scorrere verticalmente sul display le linee del programma.
Due altri tasti permettono lo scorrimento laterale in tutti i modi di funzionamento.
Poiché il software disponibile è molto limitato, gli utenti devono sviluppare per conto proprio i programmi di cui hanno bisogno. Il manuale di corredo facilita la programmazione in due maniere: innanzi tutto si dimostra una buona ed esauriente guida alla comprensione del BASIC (sebbene manchi una sezione didattica per principianti); secondariamente contiene i listati di nove brevi programmi applicativi, alcuni dei quali di carattere matematico (per il calcolo delle radici quadrate, della deviazione standard in statistica, ecc..), altri più generici, come un programma per l'uso pratico della tastiera ed un gioco del tipo "atterraggio morbido". Infine la Sharp offre tre microcassette contenenti una serie di programmi.
Mentre i computer della Casio possono contenere in memoria fino a 10 programmi contemporaneamente, il PC-1251 può immagazzinare un solo programma alla volta. È tuttavia possibile usare programmi composti da più sottoprogrammi separati da istruzioni END,

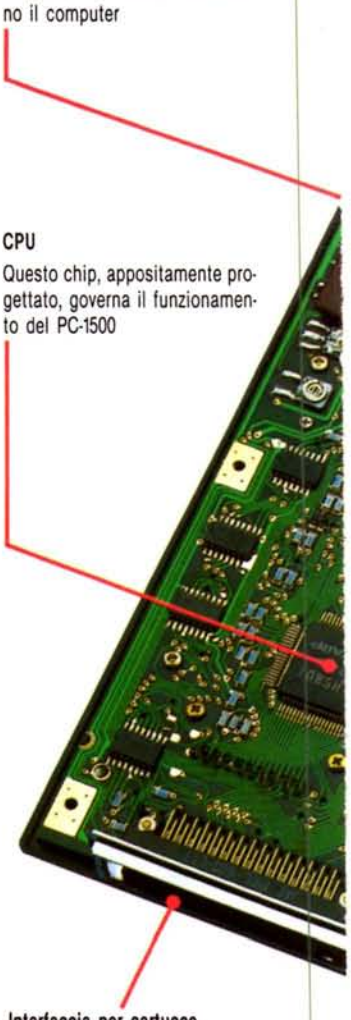


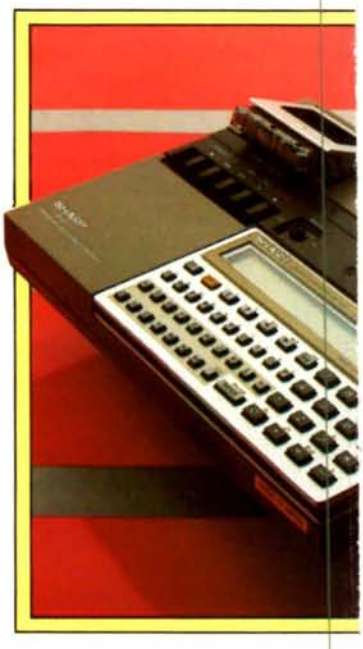
Un potente tascabile
Questi modelli Sharp, grazie alla RAM CMOS ed alla CPU a 8 bit, alla tastiera QWERTY, al BASIC ed alla vasta gamma di periferiche disponibili, possono essere considerati veri e propri microcomputer tascabili

che vengono eseguiti inserendo il comando GOTO seguito dal relativo numero di linea.
La presenza in memoria di più programmi è estremamente utile perché, mancando la funzione di caricamento, si devono riscrivere i programmi tutte le volte che se ne ha bisogno.
Date le ridotte dimensioni della sua memoria (solo 4 Kbyte), il computer ha spazio appena sufficiente per 4 brevi programmi BASIC.
Quindi, per un impiego di tipo professionale, è necessario acquistare anche la stampante e il lettore di cassette.

Lo Sharp PC-1500A

Il PC-1500A, l'altro portatile della Sharp, deriva dal precedente PC-1500, ma è dedicato a un'utenza più specializzata. Il modello base misura 195x85x25 mm. e pesa 375 grammi, circa tre volte di più del PC-1251, rispetto al quale offre una migliore tastiera ed un display più grande (26 caratteri invece di 24), anche se è privo della manopola per l'orientamento.
La versione del BASIC è notevolmente migliore: le variabili possono essere definite con nomi a due lettere ed è consentito l'impiego di uno stesso nome per stringhe e variabili numeriche; inoltre vi sono anche funzioni grafiche e di suono.
Con il comando GPRINT, che permette di definire colonne a 7 punti, è possibile creare immagini con una risoluzione di 7 righe per 156 colonne. Il comando BEEP controlla invece il tono e la durata delle note che hanno tra l'altro una discreta sonorità.
Il computer possiede un orologio e un calendario incorporati, accessibili tramite la variabile TIME, che segnano il tempo anche quando l'apparecchio è spento.
Il PC-1500A dispone di molte altre istruzioni BASIC, che lo pongono in posizione preminente rispetto a numerosi micro: tra queste, il comando ON ERROR GOTO e le funzioni di debugging TRON e TROFF. La versione base possiede ben 39 messaggi d'errore. Altri 16 sono abbinati ai comandi destinati agli accessori opzionali.
Purtroppo, come sul PC-1251, i messaggi d'errore vengono espressi come codici numerici, costringendo così a consultare il manuale.
I 10 tasti alfabetici della prima riga corrispondono anche a una serie di parole chiave BASIC, riportate su di un'apposita mascherina da sovrapporre alla tastiera. I sei tasti posti al di sopra della tastiera possono essere programmati con 18 diverse funzioni.
Anche il PC-1500A manca di un software adeguato: solo poche software house hanno sviluppato programmi per l'apparecchio e la stessa Sharp offre una sola cassetta di programmi vari. Un opuscolo di corredo alla macchina contiene 53 programmi applicativi dedicati a cinque argomenti principali: matematica, statistica, elettrologia, attività d'ufficio e giochi. Questi programmi, scritti originariamente per il PC-1500, funzionano perfettamente anche sul PC-1500A, poiché i due apparecchi differiscono solo per le dimensioni della RAM: il primo ha 3,5 Kbyte di memoria mentre il secondo ne possiede 8,5.
Un portacartucce, posto sotto l'apparecchio, consente di utilizzare i quattro tipi di espansioni di memoria (piuttosto costose) vendute dalla Sharp.
Due delle cartucce (da 4 e 8 Kbyte) conservano i dati immagazzinati solo quando sono innestate sull'apparecchio. Le altre due cartucce (da 8 e 16 Kbyte) sono invece alimentate da batterie interne e conservano i dati anche una volta estratte.
L'unità aggiuntiva, composta dall'interfaccia per cassetta e dalla stampante/plotter, è un accessorio di elevata qualità. La prima consente di salvare e caricare programmi da un comune registratore a cassette (la Sharp tuttavia offre un proprio registratore); la seconda, simile a molte altre stampanti/plotter per home computer, usa quattro piccole penne a sfera per produrre testi e grafica in quattro colori. Purtroppo il formato della carta, appena 75 mm. di larghezza, è piuttosto ridotto.
Il BASIC possiede una serie di comandi dedicati alla stampante/plotter: CSIZE produce lettere di varie dimensioni, ROTATE stampa caratteri in posizione normale laterale o rovesciata, COLOR seleziona una delle quattro penne colorate, LT fa scorrere la carta verticalmente (in entrambe le direzioni, a scelta), LPRINT stampa testi, LCURSOR e GLCURSOR azionano la penna nei modi testuale e grafico, SORGN seleziona il punto d'origine e, infine, LINE e RLINE usano rispettivamente coordinate assolute e relative per congiungere due punti con una linea.
La stampante/plotter e l'interfaccia per cassette sono riunite in un'unica custodia, dotata di un alimentatore per la ricarica delle batterie.
Altri due importanti accessori del PC-1500A sono le interfacce Centronics e RS232, che permettono al portatile di comunicare con computer e stampanti più grandi, ed una periferica detta "software board". Questa consiste in una grande tastiera a sfioramento, dotata di 140 tasti che possono essere programmati per eseguire operazioni di uso frequente (come il calcolo automatico dei totali nelle applicazioni di lavagna elettronica).
Il suo funzionamento richiede la presenza di un apposito software che ne eleva notevolmente il costo.
Grazie al gran numero di accessori disponibili, il PC-1500A può trasformarsi in un potente sistema di elaborazione ma, a parità di prezzo, il mercato offre altri microsistemi più versatili e dotati di un maggior corredo software.
Con questo tascabile la Sharp mira a conquistare il primo posto nel mercato dei computer della categoria "peso piuma", ma il PC-1500A deve tener conto del rischio di essere superato da apparecchi di maggior peso, ma decisamente più potenti.

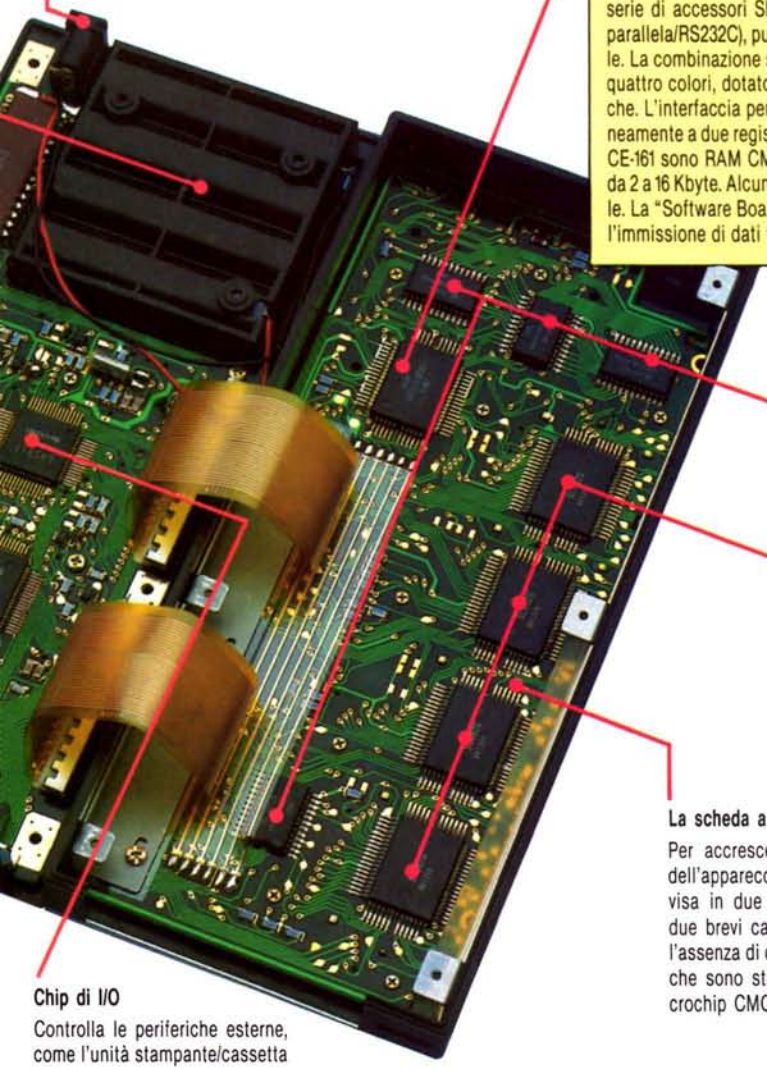




Presa d'alimentazione
In alternativa alle batterie l'alimentazione può provenire dalla normale rete, attraverso un apposito trasformatore

ROM BASIC
Contiene il BASIC Microsoft impiegato dal computer

Accessori Sharp
Lo Sharp PC-1251 viene innestato su di un'unità mista stampante/microcassetta, che misura mm. 205x149x23. Questa unità si compone di una stampante termica (linee di 24 caratteri) e di un registratore a microcassette. Il PC-1500 è il "cuore" di una nutrita serie di accessori Sharp e tramite l'unità CE-158 (un'interfaccia parallela/RS232C), può collegarsi a qualsiasi micro o unità centrale. La combinazione stampante/cassetta CE-150 è un plotter X-Y a quattro colori, dotato di 9 formati di carattere e di funzioni grafiche. L'interfaccia per cassetta può essere collegata contemporaneamente a due registratori. Le 4 cartucce CE-151, CE-155, CE-159 e CE-161 sono RAM CMOS che consentono espansioni di memoria da 2 a 16 Kbyte. Alcune contengono anche una ROM programmabile. La "Software Board" CE-153 è invece una tastiera a lamina per l'immissione di dati formattati



Chip RAM
Forniscono 8,5 Kbyte di memoria, 6,6 dei quali a disposizione dell'utente

Microprocessori del display
Controllano il display a cristalli liquidi (LCD)

La scheda a circuiti stampati
Per accrescere la compattezza dell'apparecchio, la scheda è divisa in due parti, collegate da due brevi cavi a nastro. Si noti l'assenza di componenti discreti, che sono stati integrati nei microchip CMOS

Chip di I/O
Controlla le periferiche esterne, come l'unità stampante/cassetta

PC-1500A

DIMENSIONI
mm. 195 x 85 x 25

PESO
375 grammi

MEMORIA
8,5 Kbyte di RAM

VIDEO
A cristalli liquidi (LCD) con 1 riga di 26 caratteri

NOTE
Orologio e calendario incorporati; buona versione di BASIC, memoria espandibile, ampia gamma di periferiche

PC-1251

DIMENSIONI
mm. 135 x 70 x 10

PESO
115 grammi

MEMORIA
4 Kbyte di RAM

VIDEO
A cristalli liquidi (LCD) con 1 riga di 24 caratteri

NOTE
Tre modi di funzionamento, 18 tasti di funzione programmabili, unità cassette/stampante opzionale



LO SHARP PC-1251

Questo tascabile, pubblicizzato dalla Sharp come sistema gestionale polivalente, probabilmente incontra maggior favore presso ingegneri e scienziati che non tra gli operatori economici. Infatti le piccole dimensioni dei tasti alfabetici della tastiera QWERTY complicano notevolmente la redazione di testi

LO SHARP PC-1500A

È un computer tascabile che abbina alle ridotte dimensioni di una calcolatrice un'eccezionale flessibilità operativa. La sua potenza d'elaborazione e il vasto corredo di accessori lo rendono uno strumento particolarmente adatto alle attività d'ufficio, ma può considerarsi anche un "giocattolo" per executive e una capace calcolatrice



CHRIS STEVENS

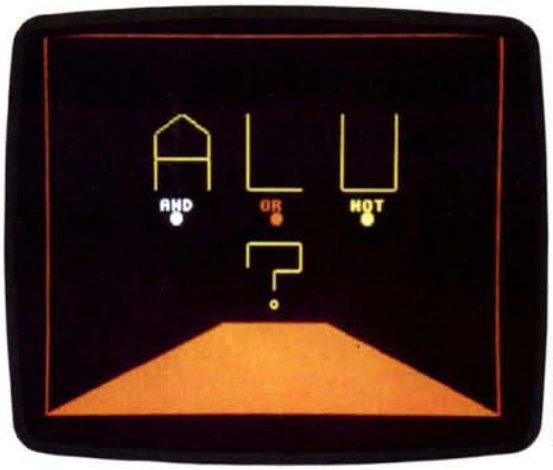
Più schermi sullo Spectrum

È la volta di produrre la grafica sullo Spectrum

La progettazione dell'ambiente ALU in alta risoluzione prevede lo scorrimento fino al centro dello schermo delle lettere A, L ed U. La versione BBC del programma *Digitaya* disegna le lettere mediante una serie di comandi di tracciamento relativo, riferiti ad un'origine prefissata, quindi le cancella dallo schermo e sposta in basso l'origine, per poi ripetere l'intera procedura (vedere a pag. 1384). La versione Spectrum adotta la stessa tecnica.
Il comando DRAW dello Spectrum, che fra parentesi consente solo procedure di tracciamento relativo, si dimostra particolarmente adatto al tipo di scorrimento qui desiderato. Dopo aver definito (con PLOT) il punto iniziale, possiamo tracciare l'immagine di una lettera con una serie di comandi DRAW e muoverla attraverso lo schermo modificando semplicemente le coordinate del suo punto iniziale. Per cancellare la lettera è sufficiente ritracciarla nella stessa posizione usando i colori invertiti. Le istruzioni INVERSE 0 ed INVERSE 1 presiedono rispettivamente alla doppia procedura di tracciamento e di cancellazione per ogni posizione assunta dalla lettera.
Prendiamo ad esempio la lettera A che si muove dal margine sinistro verso il centro dello schermo. Le istruzioni che la tracciano si trovano all'interno di un ciclo FOR...NEXT, che incrementa ad ogni passaggio il valore della coordinata x del punto iniziale di tracciamento. Un secondo ciclo FOR...NEXT, nidificato all'interno del primo, ripete semplicemente le istruzioni di tracciamento. L'ultimo valore di x è 55 e corrisponde alla posizione finale della lettera sullo schermo. Per evitare che quest'ultima immagine venga cancellata, il programma contiene una procedura di verifica che cancella la lettera (abilitando l'istruzione INVERSE 1) solo se la coordinata x è minore di 55. La stessa tecnica di tracciamento viene applicata alle lettere L ed U, che si portano al centro dello schermo partendo rispettivamente dall'alto e dal margine destro.

Fasi di progettazione dell'ALU

Prima di passare alla programmazione vera e propria dell'immagine grafica, è consigliabile tracciarne un abbozzo su carta per definire i valori delle coordinate da assegnare a ciascuna figura sullo schermo. Inoltre è necessario indicare la posizione delle lettere da visualizzare, facendo riferimento alle righe e alle colonne. La figura seguente mostra l'immagine ALU finale.



IAN McKINNELL

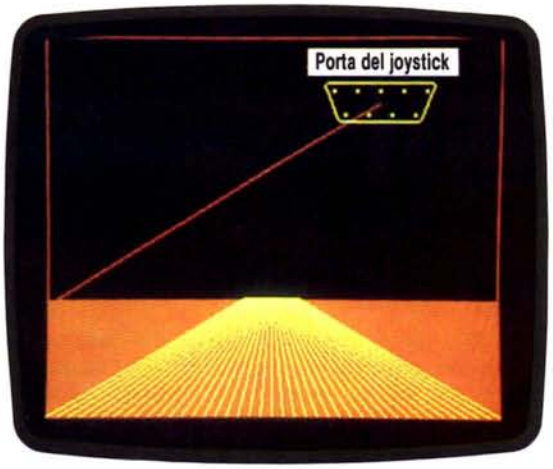
Le parole AND, OR e NOT vengono posizionate con il comando PRINT AT r,c, dove r indica il numero di righe a partire dalla sommità dello schermo, e c quello delle colonne a partire dal margine sinistro. I pulsanti vengono tracciati con l'istruzione CIRCLE x,y,r in cui vanno specificate le coordinate del centro e la lunghezza del raggio.
Dopo aver visualizzato l'intera immagine, il programma attende che il giocatore prema un tasto (istruzione di verifica INKEY$), prima di ripristinare i colori iniziali di INK e PAPER e di cancellare lo schermo. Quindi restituisce il controllo alla routine principale. Se il giocatore non preme alcun tasto la procedura di verifica viene ripetuta.
Per chiamare la subroutine che disegna l'immagine ALU, dobbiamo inserire nel programma *Digitaya* la seguente linea:

```
4565 GOSUB 7000: REM S/R IMMAGINE ALU
```

La porta per il joystick

L'ambiente della porta per il joystick presenta una presa per joystick dal cui centro origina un raggio laser. I piedini della presa sono rappresentati da caratteri "punto", mentre il caratteristico profilo "tipo D" è ottenuto con grafica in alta risoluzione. Per accentuare la profondità dell'immagine, in primo piano vengono tracciate (PLOT) una serie di linee divergenti, che originano dall'orizzonte e terminano sul margine inferiore dello schermo. I punti iniziali delle linee sono spaziati di un'unità grafica, quelli finali di 7 unità. Poiché il comando DRAW dello Spectrum è un comando di tracciamento relativo, non è possibile specificare un punto finale assoluto: ciò rende la

struttura della routine più complicata. Se la coordinata x del punto iniziale della linea più a sinistra è 1119, lo scarto relativo del suo punto finale viene calcolato sulla base di questo valore. Il calcolo viene eseguito dal ciclo FOR...NEXT compreso tra le linee 8030 ed 8060.
Anche in questo caso conviene progettare sulla carta le dimensioni e le coordinate dell'ambiente prima di passare alla scrittura del programma. La figura seguente mostra l'immagine video finale.



Il colore INK dei raggi laser, che originano dal centro della porta per il joystick e terminano su un punto qualunque dell'orizzonte, viene selezionato in modo casuale. Ripetendo la procedura di tracciamento con INVERSE 1, possiamo cancellare il raggio e visualizzarlo nuovamente dopo un breve intervallo. Tuttavia questa tecnica di cancellazione presenta un inconveniente. Poiché il raggio origina dal centro della porta, ne incrocia ovviamente il profilo tracciato in precedenza. Quando il raggio viene cancellato, scompare anche una piccola sezione del profilo della porta che deve essere ritracciata.
Un inconveniente simile si verifica sulla linea dell'orizzonte, sebbene il raggio laser termini a breve distanza da essa. A causa della particolare tecnica di controllo del colore adottata sullo Spectrum, la parte di orizzonte più vicina all'estremità finale del raggio assume il suo stesso colore. Poiché lo Spectrum accetta, all'interno di una cella di carattere, solo un colore INK ed uno PAPER, qualsiasi carattere grafico presente nella cella assume il colore di primo piano del nuovo colore INK usato al suo interno. Quindi dopo ogni cancellazione del raggio è necessario ritracciare anche la linea dell'orizzonte. La subroutine continua a lanciare raggi laser finché il giocatore non preme un tasto, quindi, dopo aver ripristinato i colori originali INK e PAPER, restituisce il controllo alla routine principale della porta per il joystick.
La seguente linea, inserita nel programma principale, chiama la subroutine della porta per il joystick:

```
3845 GOSUB 8000: REM PORTA PER IL JOYSTICK
```

Nella prossima parte viene esaminata la realizzazione di entrambe le subroutine grafiche sul Commodore 64.

AMBIENTE ALU

```
7000 REM ** S/R Immagine ALU **
7010 INK 6: PAPER 0: CLS
7015 :
7017 REM ** Lettera A **
7020 FOR x=0 TO 55 STEP 5
7030 INVERSE 0
7040 FOR i=1 TO 2
7050 PLOT x,100
7060 DRAW 0,30
7070 DRAW 15,20
7080 DRAW 15,-20
7090 DRAW 0,-30
7095 DRAW 0,20
7096 DRAW -30,0
7110 IF x<55 THEN INVERSE 1
7115 NEXT i
7120 NEXT x
7130 :
7140 REM ** Lettera L **
7150 FOR y=100 TO 150 STEP 5
7152 INVERSE 0
7155 FOR i=1 TO 2
7160 PLOT 113,y
7170 DRAW 0,-50
7180 DRAW 30,0
7190 IF y<150 THEN INVERSE 1
7200 NEXT i
7210 NEXT y
7220 :
7230 REM ** Lettera U **
7240 FOR x=225 TO 170 STEP -5
7250 INVERSE 0
7260 FOR i=1 TO 2
7270 PLOT x,150
7280 DRAW 0,-50
7290 DRAW 30,0
7300 DRAW 0,50
7310 IF x>170 THEN INVERSE 1
7320 NEXT i
7330 NEXT x
7340 :
7350 REM ** Pulsanti **
7360 PRINT AT 10,7;"AND"
7370 PRINT AT 10,15;"OR"
7380 PRINT AT 10,22;"NOT"
7390 INK 3: CIRCLE 70,80,5
7400 INK 4: CIRCLE 128,80,5
7410 INK 5: CIRCLE 185,80,5
7420 :
7430 REM ** Punto interr. **
7435 INK 6
7440 PLOT 113,45
7450 DRAW 0,15
7460 DRAW 30,0
7470 DRAW 0,-20
7480 DRAW -15,0
7490 DRAW 0,-7
7500 FOR r=6 TO 0 STEP -2
7510 CIRCLE 128,23,r
7520 NEXT r
7530 :
7540 IF INKEY$="" THEN GO TO 7540
7550 INK 0: PAPER 7: CLS
7560 RETURN
```

AMBIENTE PORTA JOYSTICK

```
8000 REM ** S/R Porta del joystick **
8010 INK 6: PAPER 0: CLS
8020 REM ** Primo piano **
8030 FOR n=1 TO 31
8040 PLOT 112+n,50
8050 DRAW 7*n-112,-50
8060 NEXT n
8070 :
8080 REM ** Orizzonte **
8085 INK 6: INVERSE 0
8090 PLOT 0,50
8100 DRAW 255,0
8110 :
8120 REM ** Porta **
8130 PRINT AT 1,18;"PORTA DEL JOYSTICK"
8140 PRINT AT 3,20;". . . . ."
8150 PRINT AT 5,21;". . . ."
8160 PLOT 158,152
8170 DRAW 75,0
8180 DRAW 1,-1
8190 DRAW 1,-1
8200 DRAW 0,-1
8210 DRAW -1,-1
8220 DRAW -10,-25
8230 DRAW -2,-2
8240 DRAW -52,0
8250 DRAW -2,2
8260 DRAW -10,25
8270 DRAW -1,1
8280 DRAW -1,1
8290 DRAW 0,1
8300 DRAW 1,1
8310 :
8320 REM ** Raggio laser **
8340 INK RND*7
8350 LET x=RND*255-194
8360 LET y=-86
8365 INVERSE 0
8367 FOR i=1 TO 2
8370 PLOT 194,136
8380 DRAW x,y
8385 INVERSE1
8387 NEXT i
8390 :
8400 REM ** Verifica per il tasto **
8410 IF INKEY$="" THEN GO TO 8080
8415 INVERSE 0
8420 INK 0: PAPER 7: CLS
8430 RETURN
```

Altre trasformazioni

Dopo aver realizzato motivi grafici lineari, passiamo alle due dimensioni

Dopo aver prodotto una semplice traslazione lineare del motivo definito a pag. 1395, proviamo adesso ad eseguire simultaneamente due traslazioni non parallele, per ottenere una configurazione bi-dimensionale. Iniziamo lo studio di queste nuove configurazioni usando come modulo un singolo punto.

```
TO PUNTO
   PD FD 1 BK 1 PU
END
```

La seguente procedura LOGO effettua due traslazioni simultanee:

```
TO GRIGLIA: XINIZ :YINIZ :PASSOX :PASSOY :ANGOLO
   DRAW HT PU
   SETXY :XINIZ :YINIZ SETH 0
   REPEAT 3 [LINEA :PASSOX BASSO :PASSOY :ANGOLO]
END
```

Questa procedura traccia una griglia composta da 9 punti. I valori in ingresso rappresentano le coordinate del punto di partenza, l'incremento di X e Y e la direzione dei punti della riga successiva relativamente a quelli della linea corrente. La procedura LINEA:

```
TO LINEA :X
   REPEAT 3 [MODULO SETX XCOR + :X]
   SETX XCOR — 3 * :X
END
```

disegna sullo schermo una linea composta da tre moduli e riporta la tartaruga sul punto di partenza. Si rammenti che, per il momento, il modulo è un punto:

```
TO MODULO
   PUNTO
END
```

Una seconda procedura:

```
TO BASSO :Y :A
   SETH :A
   FD :Y
   SETH 0
END
```

sposta la tartaruga sulla riga successiva (ossia in basso di una riga), quindi ripristina la direzione iniziale. I cinque tipi di griglia bi-dimensionale ottenibili sono mostrati a pag. 1416, assieme alle rispettive procedure di tracciamento LOGO.
Combinando queste griglie di base è possibile creare svariate configurazioni, ma si provi anche a modificare la procedura di tracciamento, per ottenere linee con moduli posti a diverse angolazioni piuttosto che sulla stessa linea.
Vediamo adesso come rendere più sofisticata la simmetria delle griglie, sostituendo il modulo puntiforme sin qui usato con altre forme di simmetria. Le 17 configurazioni ottenibili in questo modo sono mostrate nella pagina seguente: per realizzarle occorre sostituire al comando MODULO, nella procedura LINEA, una procedura che tracci il nuovo modulo.

Il modulo di simmetria

I moduli sono composti da un motivo base, sottoposto a varie riflessioni e rotazioni. Come esempio, creiamo un nostro motivo base e chiamiamolo MBASE (si rammenti che la procedura deve essere trasparente e non deve chiamare sottoprocedure).

```
TO MBASE
   PD
   FD 15
   RT 90
   FD 5
   BK 5
   LT 90
   BK 15
   PU
END
```

Possiamo utilizzare le procedure sviluppate in precedenza per creare due nuove procedure: MOTIVO e la sua immagine speculare R.MOTIVO.

```
DEFINE "MOTIVO TEXT "MBASE
DEFINE "R.MOTIVO RISCRIVE "MBASE
```

Definiamo adesso, come esempio, i moduli per le configurazioni 7 e 17:

```
TO MODULO7
   LT 90 MOTIVO RT 90
END

TO MODULO17
   RT 30
   REPEAT 6 [MOTIVO R.MOTIVO RT 60]
   LT 30
END
```

Tornando ad esaminare la procedura LINEA, vediamo che essa esegue la procedura MODULO. Quindi, per tracciare la configurazione 7, dobbiamo sostituire MODULO7 a MODULO per mezzo della procedura DEFINISCE.MODULO 7:

```
TO DEFINISCE.MODULO :NUM
```



```
    DEFINE "MODULO TEXT PAROLA "MODULO :NUM
END
```

La seguente procedura, chiamata CONF, traccia contemporaneamente la griglia e il modulo di simmetria di una configurazione:

```
TO CONF :GRIGLIA :NUM :PROC
    DEFINE "MOTIVO TEXT :PROC
    DEFINE "R.MOTIVO RISCRIVE :PROC
    DEFINISCE.MODULO :NUM
    RUN (LISTA :GRIGLIA)
    ERASE MOTIVO
    ERASE R.MOTIVO
    ERASE MODULO
END
```

Per tracciare la configurazione 17 dobbiamo scrivere:

```
CONF "ESAG 17 "MBASE
```

che impiega MBASE come motivo base e traccia una griglia esagonale i cui elementi sono MODULO17.
Questa procedura di tracciamento non si adatta però alle configurazioni 4, 6, 7 e 12. In esse il modulo non è lo stesso in ogni punto ma subisce una trasformazione (una riflessione, una rotazione o entrambe). Per realizzare tali griglie è necessario inserire le trasformazioni necessarie nelle procedure LINEA e BASSO. Definiamo TRASX e TRASY come le trasformazioni da applicarsi rispettivamente alla traslazione di base lungo le colonne e le righe. Otteniamo così le seguenti procedure modificate LINEA e BASSO:

```
TO LINEA :X
    REPEAT 3 [MODULO SETX XCOR XCOR + :X TRASX]
    SETX XCOR — 3 * :X
END

TO BASSO :Y :A
    SETH :A
    FD :Y
    SETH 0
    TRASY
END
```

Definiamo adesso la configurazione 7:

```
TO CONFIG7 :PROC
    DEFINE "TRASX [[ ] [RIFLETTERE RT 180]]
    DEFINE "TRASY [[ ] [ ]]
    CONF "RETT7 :PROC
    ERASE TRASX
    ERASE TRASY
END
```

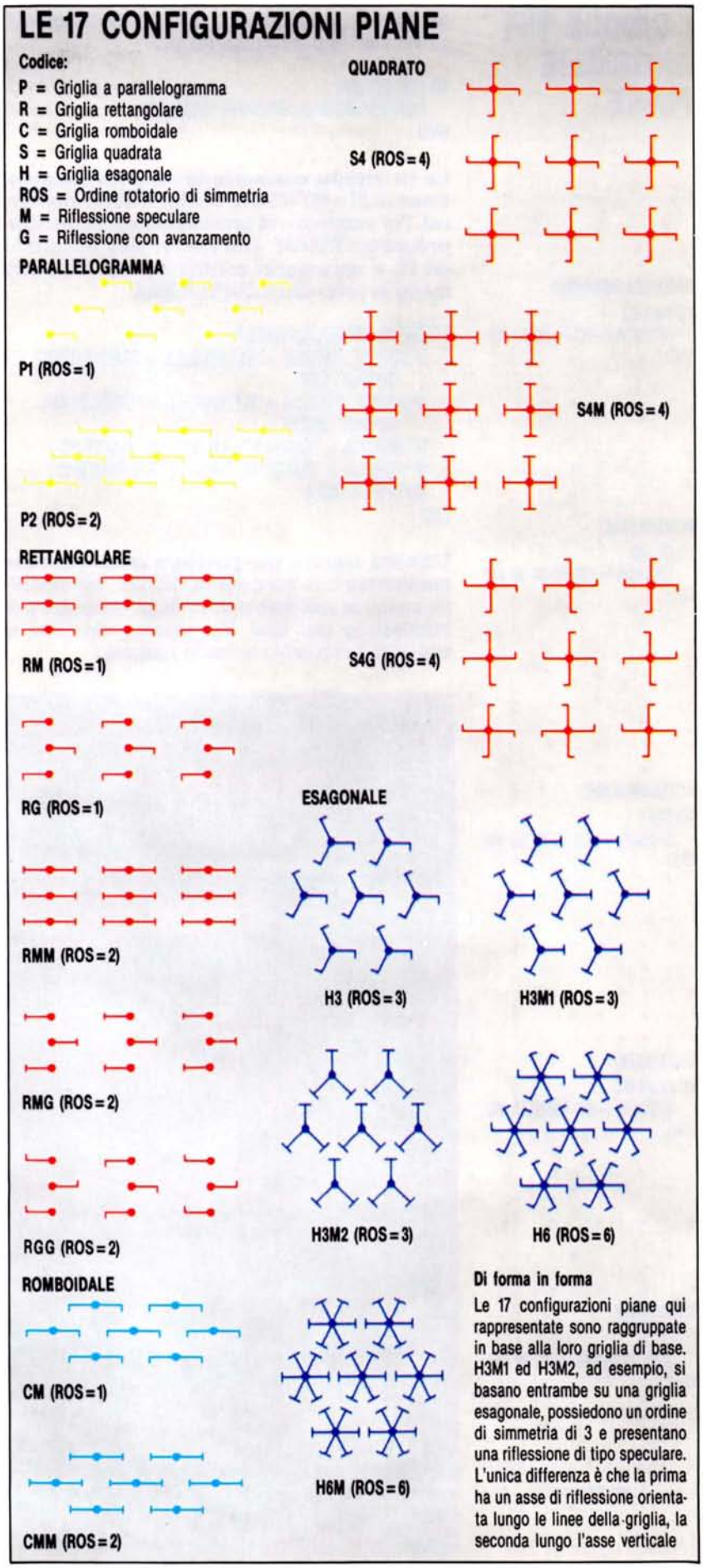
Dopo aver eseguito questa procedura per mezzo del comando CONFIG7 "GAMBA, definiamo TRASX:

```
TO TRASX
    RIFLETTERE
    RT 180
END
```

La procedura RIFLETTERE, usata per riflettere il

LE 17 CONFIGURAZIONI PIANE

Codice:
P = Griglia a parallelogramma
R = Griglia rettangolare
C = Griglia romboidale
S = Griglia quadrata
H = Griglia esagonale
ROS = Ordine rotatorio di simmetria
M = Riflessione speculare
G = Riflessione con avanzamento



Di forma in forma

Le 17 configurazioni piane qui rappresentate sono raggruppate in base alla loro griglia di base. H3M1 ed H3M2, ad esempio, si basano entrambe su una griglia esagonale, possiedono un ordine di simmetria di 3 e presentano una riflessione di tipo speculare. L'unica differenza è che la prima ha un asse di riflessione orientata lungo le linee della griglia, la seconda lungo l'asse verticale

IAN McKINNELL

I CINQUE TIPI DI GRIGLIE PIANE



PARALLELOGRAMMA

```
TO PARALL
   GRIGLIA (—60) 90 80 50 205
END
```



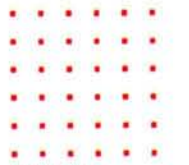
ROMBOIDALE

```
TO ROMB
   GRIGLIA (—30) 90 80 80 225
END
```



RETTANGOLARE

```
TO RETT
   GRIGLIA (—80) 90 80 50 180
END
```



QUADRATA

```
TO QUADR
   GRIGLIA (—80) 90 80 80 180
END
```



ESAGONALE

```
TO ESAG
   GRIGLIA (—30) 90 80 80 210
END
```

modulo della configurazione, viene definita riscrivendo la procedura MODULO:

```
TO RIFLETTERE
   DEFINE "MODULO RISCRIVE "MODULO
END
```

La riscrittura comporta la rispettiva sostituzione di RT e MOTIVO con LT e R.MOTIVO (e viceversa). Per cambiare la precedente versione della procedura RISCRIVE, che interveniva solo su RT ed LT, è necessario modificare nel seguente modo la procedura CAMBIA.PAROLA:

```
TO CAMBIA.PAROLA :PAROLA
   IF (ANYOF :PAROLA = "RT :PAROLA = "RIGHT) THEN
      OUTPUT "LEFT
   IF (ANYOF :PAROLA = "LT :PAROLA = "LEFT) THEN
      OUTPUT "RIGHT
   IF :PAROLA = "MOTIVO THEN OUTPUT "R.MOTIVO
   IF :PAROLA = "R.MOTIVO THEN OUTPUT "MOTIVO
   OUTPUT:PAROLA
END
```

Un'altra tecnica per risolvere questo problema è usare una versione di RISCRIVE che modifica anche le sottoprocedure della procedura di immissione dei dati. Per questa versione si consulti il riquadro a fondo pagina.

Nella maggior parte delle configurazioni, il movimento da un punto all'altro dipende da una semplice traslazione e non implica altre trasformazioni. Quindi TRASX e TRASY non svolgono alcuna funzione. CONFIG17 ne è un tipico esempio:

```
TO CONFIG17 :PROC
   DEFINE "TRASX [[] []]
   DEFINE "TRASY [[] []]
   CONF "ESAG 17 :PROC
   ERASE TRASX
   ERASE TRASY
END
```

Logo & Logo

Per tutte le versioni LCSI:

Usare CS al posto di DRAW
Usare OR al posto di ANYOF
Usare SETPOS, seguito da una lista, al posto di SETXY
IF possiede una diversa sintassi:

IF :PAROLA = MOTIVO [OUTPUT "R.MOTIVO]

Il LOGO Atari manca delle primitive TEXT e DEFINE, ma il manuale Atari offre un metodo che permette di definirle

Risposte agli esercizi

1) Procedura per ruotare un modulo con coordinate (X,Y) di un angolo di A gradi:

```
TO ROTAZIONE :X :Y :A
   PU
   MAKE "H HEADING
   MAKE "XOLD XCOR
   MAKE "YOLD YCOR
   MAKE "R SQRT (:XOLD —:X) * (:XOLD —:X) +
   (:YOLD —:Y) * (:YOLD — :Y)
   PU
   SETXY :X :Y
   SETH TOWARDS :XOLD :YOLD
   RT :A
   FD :R
   SETH :H + :A
   PD
END
```

2) Una procedura di riscrittura che modifica anche le sottoprocedure:

```
MAKE "PROC.FATTE []

TO RISCRIVE :PROC
   MAKE "PROC.FATTE FPUT :PROC :PROC.FATTE
   OUTPUT RISCRIVE.PROC TEXT :PROC
END

TO RISCRIVE.PROC :TEXT
   IF :TEXT = [] THEN OUTPUT []
   OUTPUT FPUT RISCRIVE.LINEA FIRST :TEXT
   RISCRIVE.PROC BUTFIRST :TEXT
END

TO RISCRIVE.LINEA :LINEA
   IF :LINEA = [] THEN OUTPUT []
   IF LIST? FIRST :LINEA THEN OUTPUT FPUT
   RISCRIVE.LINEA FIRST :LINEA RISCRIVE.LINEA
      BUTFIRST :LINEA
   OUTPUT FPUT CAMBIA.PAROLA FIRST :LINEA
   RISCRIVI.LINEA BUTFIRST :LINEA
END

TO CAMBIA.PAROLA :PAROLA
   IF (ANYOF :PAROLA = "RT :PAROLA = "RIGHT) THEN
      OUTPUT "LEFT
   IF (ANYOF :PAROLA = "LT :PAROLA = "LEFT) THEN
      OUTPUT "RIGHT
   IF PROCEDURA? :PAROLA THEN SOTTOPROC :PAROLA
      OUTPUT :PAROLA "£ :PAROLA
   OUTPUT :PAROLA
END

TO PROCEDURA? :NOME
   IF NUMBER? :NOME OUTPUT "FALSE
   IF LIST? :NOME OUTPUT "FALSE
   TEST WORD? :NOME
   IF TRUE IF WORD? TEXT :NOME OUTPUT "FALSE ELSE IF
      NOT (TEXT :NOME = [])
   OUTPUT "TRUE
   OUTPUT "FALSE
END

TO SOTTOPROC :PAROLA
   IF MEMBER? :PAROLA :PROC.FATTE THEN STOP
   DEFINE (WORD "£ :PAROLA) RISCRIVE :PAROLA
END
```

Movimenti scorrevoli

Il Commodore 64 consente lo scorrimento dell'immagine video di un pixel alla volta

Il chip VIC, che si occupa della gestione del video nel Commodore 64, può fare scorrere lo schermo del Commodore di un massimo di 8 pixel in entrambe le direzioni. Lo scorrimento orizzontale è controllato dai tre bit più bassi del registro del VIC posto alla locazione 53270 ($D016). Se carichiamo in essi valori da 7 a 0 in sequenza, otteniamo lo scorrimento progressivo dell'immagine di un pixel a sinistra. In BASIC, si può usare una sequenza del tipo:

```
POKE 53270, (PEEK(53270) AND 248) + P
```

dove P contiene un valore da 0 a 7.
Abbinando a questa funzione una routine in codice macchina che sposti tutte le informazioni per lo schermo di una cella a sinistra e inserisca una nuova colonna di dati sul margine destro del video, otteniamo un effetto di scorrimento ("scrolling") uniforme. Affinché l'immagine possa scorrere in modo omogeneo, è necessario ridurre le dimensioni dello schermo del Commodore da 40 a 38 colonne. Per passare al modo a 38 colonne dobbiamo azzerare il bit 3 del registro di scrolling orizzontale. L'istruzione BASIC per questa operazione è:

```
POKE 53270, PEEK(53270) AND 247
```

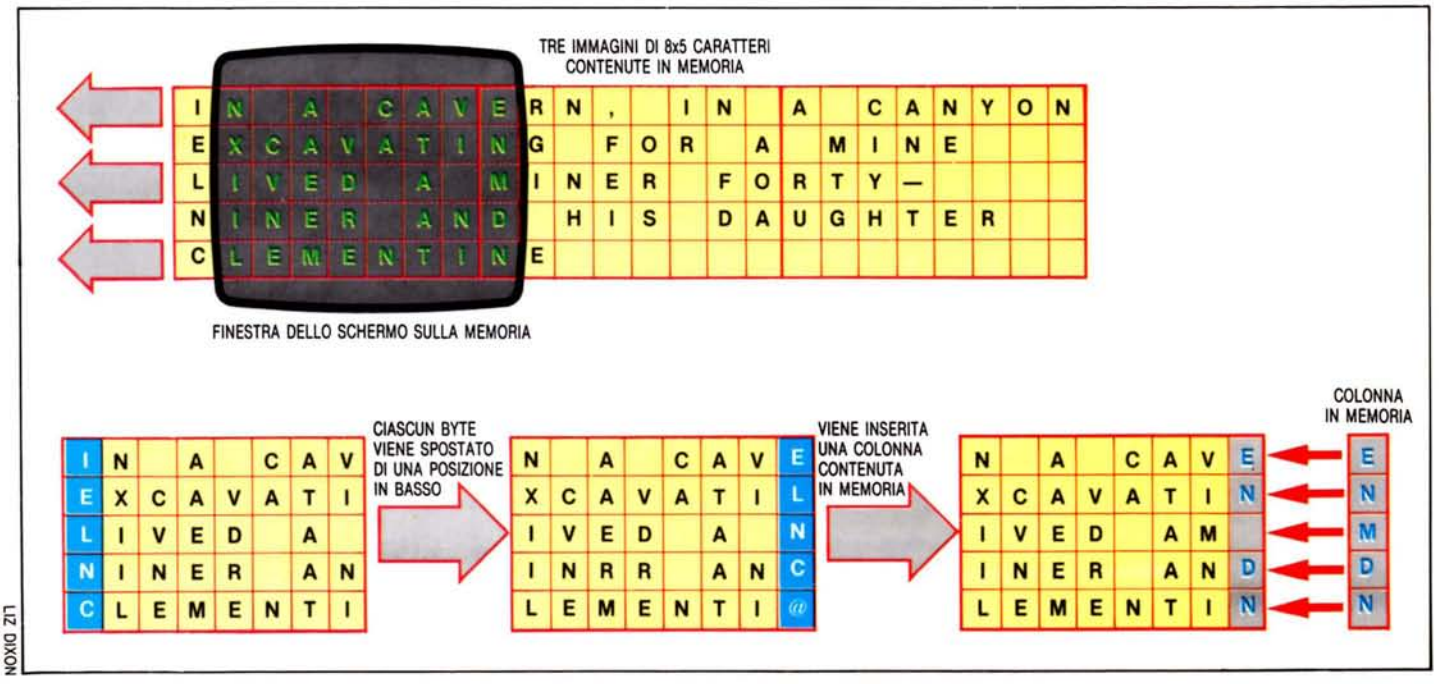
Lo schermo viene riportato alle abituali 40 colonne ponendo il bit 3 a uno.
Il diagramma di flusso (vedere alla pagina seguente) illustra le operazioni necessarie per ottenere uno scrolling orizzontale uniforme. Va ricordato infine che, quando spostiamo o inseriamo le informazioni per lo schermo, occorre contemporaneamente modificare quelle per il colore.

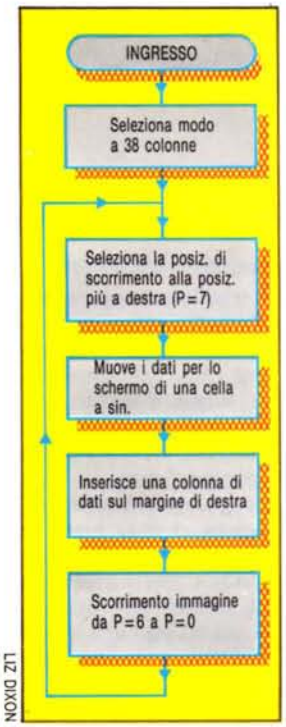
Scorrimento dei dati per lo schermo

In teoria, l'intero procedimento è abbastanza semplice. Le informazioni per lo schermo vengono di solito conservate in un blocco di 1000 byte consecutivi, che iniziano alla locazione 1024 ($0400): i primi 40 byte compongono la prima riga dello schermo, i 40 byte successivi la seconda e così via. Per ottenere lo scorrimento a sinistra di una posizione, ciascun byte di dati va spostato in basso di una locazione rispetto alla posizione originale. Questa sezione della routine usa una serie di puntatori alla pagina zero e il metodo di indirizzamento indiretto per abbassare di un byte in memoria le informazioni per lo schermo ed il colore.
Chiamando SB l'indirizzo di base dell'area di memoria per lo schermo, l'ultima cella della prima riga sarà SB+39, quella della seconda riga SB+79 e così via. Poiché le informazioni che devono scorrere sullo schermo possono essere immagazzinate in memoria in modo simile allo schermo reale, cioè in "pacchetti" di 1000 byte, i dati inseriti sul margine destro

Meccanismo di scorrimento

Lo scorrimento sullo schermo di un'immagine conservata in memoria si sviluppa in tre fasi principali. Innanzi tutto vengono spostati in basso di una posizione i byte contenuti nell'area di memoria video. Poiché le righe che compongono l'immagine sono immagazzinate in memoria come sequenze di byte, durante lo scrolling ogni carattere sullo schermo si sposta a sinistra di una posizione, con l'eccezione dei caratteri posti nella colonna più a sinistra, che scompaiono per riapparire, scalati di un posto, sulla prima colonna di destra. In questa fase, quando il primo carattere in alto a sinistra scompare, contemporaneamente compare un carattere "spurio" nell'angolo inferiore destro dello schermo (qui rappresentato dal simbolo @).
La seconda fase consiste nel trasferire nella prima colonna di destra dello schermo la relativa colonna di caratteri contenuti nella memoria video.
Eseguita questa operazione, viene modificato il valore contenuto nel registro di scorrimento del chip VIC, producendo lo scorrimento di un pixel alla volta dell'immagine sullo schermo





LIZ DIXON

Diagramma di flusso per lo scorrimento

Per ottenere uno scorrimento omogeneo, si può ricorrere ad una particolare funzione del chip per la gestione del video del Commodore. Il VIC possiede speciali registri di scrolling che, tramite l'accensione successiva di singoli pixel, consentono di ottenere spostamenti orizzontali e verticali dell'immagine sullo schermo. Abbinando questo effetto al trasferimento di caratteri in codice macchina è possibile generare scorrimenti su uno schermo ridotto a 38 colonne

corrispondono al primo, al quarantunesimo, all'ottantunesimo ecc. byte dell'area di memoria loro riservata.
Un puntatore viene inizialmente indirizzato sul primo byte dell'area di memoria da inserire sullo schermo. Dopo l'immissione della prima colonna, viene incrementato di uno, per inserire una seconda colonna sul margine destro: da qui la colonna scorre poi verso sinistra. La procedura, ripetuta per 40 volte, porta al completo scorrimento di un'immagine sullo schermo. Il puntatore alla memoria deve essere incrementato di 960 (1000-40) per puntare al primo byte dell'immagine successiva.
L'intero processo viene ripetuto per la corrispondente area contenente le informazioni per il colore. Per semplificare questa operazione, conviene che l'indirizzo di ciascun byte della mappa di colore abbia un offset costante rispetto all'indirizzo del relativo byte della mappa per lo schermo. Tale procedimento può essere ripetuto per tutte le immagini contenute in sequenza nella memoria.
Prima di chiamare ed eseguire la routine di scorrimento, occorre fornire al programma le seguenti informazioni:

1) L'indirizzo iniziale dell'area di memoria con i dati da trasferire sullo schermo.
2) L'offset per le informazioni di colore.
3) Il numero delle immagini da visualizzare.
4) Un valore di ritardo, usato per rallentare la velocità di scorrimento.

Queste informazioni vengono depositate con istruzioni POKE nelle locazioni riservate dal programma in codice macchina.

PROGRAMMA DI CHIAMATA BASIC

```
10 REM ********************************
20 REM ********************************
30 REM **                            **
40 REM **   PROG. DI CHIAMATA PER    **
50 REM ** LA ROUTINE DI SCORRIMENTO **
60 REM **                            **
70 REM ********************************
80 REM ********************************
90 :
95 DN=8: REM PER CASSETTA DN=1
100 IF A=0 THEN A=1: LOAD "SCROLL.HEX",DN,1
110 POKE 55,0: POKE 56,32: CLR: REM ABBASSA LA MEMORIA
115 REM GOSUB 1000: REM PREPARA LO SCHERMO
120 :
130 LMEM  =49664:     REM INIZIO DELL'AREA
140 HMEM  =49665:     REM DI MEMORIA
150 LCOFF =49666:     REM OFFSET PER LA MAPPA
160 HCOFF =49667:     REM DEL COLORE
170 NMSCR =49668:     REM NUM. DELLE IMMAGINI
190 DELAY =49669:     REM VALORE DI RITARDO
200 SCROLL=49670:     REM INDIRIZZO INIZIALE DEL PROG.
210 :
220 REM PRINT CHR$(147): REM CANCELLA LO SCHERMO
230 INPUT "INDIRIZZO DECIMALE DI PARTENZA";SA
240 HS=INT (SA/256): LS=SA-HS*256
250 POKE LMEM,LS: POKE HMEM,HS
260 :
270 INPUT "NUMERO DELLE IMMAGINI";NS
290 POKE NMSCR,NS
300 :
310 INPUT "OFFSET DECIMALE PER LA MAPPA DI COLORE";OS
320 HO=INT (OS/256): LO=OS-HO*256
330 POKE LCOFF,LO: POKE HCOFF,HO
340 :
350 INPUT "VALORE DI RITARDO < 256";DV
360 IF DV>255 OR DV<0 THEN 350
370 POKE DELAY,DV
380 :
390 SYS SCROLL
400 POKE 53270, PEEK (53270) OR 8
```

Il programma carica il codice macchina in memoria e attende l'introduzione delle informazioni necessarie (tramite istruzioni INPUT). Quindi le suddivide, se necessario, secondo il formato byte-basso/byte-alto e le memorizza (con POKE) nell'area riservata all'inizio del programma in codice macchina. Infine chiama la routine in codice macchina. È possibile specificare qualsiasi indirizzo iniziale, offset e numero di immagini, ma non si otterranno risultati significativi se nell'area di memoria indicata non vengono inserite opportune informazioni sulle immagini. Per verificare il buon funzionamento del programma, si carichi in memoria e si esegua il breve programma BASIC riportato qui sotto, che visualizza due semplici immagini con indirizzo iniziale alla locazione 8192. L'offset per l'area delle informazioni di colore è 3000 byte. Per ottenere lo scrolling delle immagini si deve rispondere alle domande del programma nel modo seguente:

1) Indirizzo decimale di partenza: 8192
2) Offset per il colore: 3000
3) Numero delle immagini: 2
4) Ritardo: 255

CARICATORE BASIC

```
 10 REM ****************************
 15 REM **  CARICATORE BASIC PER  **
 20 REM ** ROUTINE DI SCORRIMENTO **
 30 REM **      ORIZZONTALE       **
 40 REM ****************************
 50 :
 60 FOR I=49670 TO 49945
 70 READ A:POKEI,A
 80 CC=CC+A
 90 NEXT
 92 READ CS:IF CS<>CC THEN PRINT "ERRORE DI CHECKSUM":STOP
100 DATA173,22,208,41,247,141,22,208
110 DATA174,4,194,160,40,138,72,152,72
120 DATA173,22,208,41,248,24,105,7,141
130 DATA22,208,169,0,133,251,169,4,133
140 DATA252,169,0,133,253,169,216,133
150 DATA254,162,3,160,1,177,251,,136
160 DATA145,251,200,177,253,136,145
170 DATA253,200,200,208,241,230,252
180 DATA230,254,177,251,198,252,136
190 DATA145,251,200,177,253,198,254
200 DATA136,145,253,230,252,230,254
210 DATA202,208,213,160,1,177,251,136
220 DATA145,251,200,177,253,136,145
230 DATA253,200,200,192,232,144,239
240 DATA173,0,194,133,253,173,1,194
250 DATA133,254,169,39,133,251,169,4
260 DATA133,252,32,241,194,173,0,194
270 DATA24,109,2,194,133,253,173,1,194
280 DATA109,3,194,133,254,169,39,133
290 DATA251,169,216,133,252,32,241,194
300 DATA162,6,173,22,208,41,248,141,22
310 DATA208,138,24,109,22,208,141,22
320 DATA208,172,5,194,136,208,253,202
330 DATA16,231,173,0,194,24,105,1,141
340 DATA0,194,173,1,194,105,0,141,1
350 DATA194,104,168,104,170,136,240,3
360 DATA76,19,194,173,0,194,24,105,192
370 DATA141,0,194,173,1,194,105,3,141
380 DATA1,194,202,240,3,76,17,194,96
390 DATA162,25,160,0,177,253,145,251
400 DATA202,240,29,165,251,24,105,40
410 DATA133,251,165,252,105,0,133,252
420 DATA165,253,24,105,40,133,253,165
430 DATA254,105,0,133,254,76,245,194
440 DATA96
450 DATA40227:REM*CHECKSUM*
```

ROUTINE MESSA A PUNTO SCHERMO

```
1000 REM **** PREPARA LO SCHERMO *****
1010 CL=3000: REM OFFSET PER LA MAPPA DEL COLORE
1020 SS=8192: REM INIZIO DELLA MAPPA PER L'IMMAGINE
1030 FOR I=SS TO SS+479
1040 POKE I,1: REM COD. DI SCHERMO PER "A"
1050 POKE I+CL: REM BIANCO
1060 POKE I+480,2: REM COD. DI SCHERMO PER "B"
1070 POKE I+CL+480,14: REM BLU
1080 NEXT
1085 FOR I=SS+960 TO SS+999
1090 POKE I,3: REM COD. DI SCHERMO PER "C"
1100 POKE I+CL,3: REM AZZURRO
1110 NEXT
1999 :
2020 SS=9192: REM INIZIO IMMAGINE SUCC.
2030 FOR I=SS TO SS+479
2040 POKE I,3: REM COD. DI SCHERMO PER "C"
2050 POKE I+CL,5: REM VERDE
2060 POKE I+480,4: REM COD. DI SCHERMO PER "D"
2070 POKE I+CL+480,0: REM NERO
2080 NEXT
2085 FOR I=SS+960 TO SS+999
2090 POKE I,5: REM COD. DI SCHERMO PER "E"
2100 POKE I+CL,2: REM ROSSO
2110 NEXT
```

SCORRIMENTO ORIZZONTALE

```
;++++++++++++++++++++++++++++++++
;++++++++++++++++++++++++++++++++
;++                            ++
;++ SCORRIMENTO  ORIZZONTALE ++
;++      PER COMMODORE 64      ++
;++                            ++
;++++++++++++++++++++++++++++++++
;++++++++++++++++++++++++++++++++
;
;SCRPTR=$FB          ;PAGINA 0
;PUNTCO=$FD          ;PUNTAT. DI TRASFERIMENTO
;
;PUNTME=$FD          ;PUNTAT. ALLA PAGINA 0
;RGSCOR=$D016        ;REG. DI SCORRIMENTO ORIZZ.
;BYBAIM=$00          ;BYTE BASSO INIZIO IMMAGINE
;BYALIM=$04          ;BYTE ALTO INIZIO IMMAGINE
;BYBACO=$00          ;BYTE BASSO INIZIO COLORE
;BYALCO=$D8          ;BYTE ALTO INIZIO COLORE
;BLOCC=$03           ;BLOCCHI BYTE 3*256
;EXTRA  =$E8         ;BYTE EXTRA
;NUMCOL =$28         ;NUM. DELLE COLONNE
;NUMRIG =$19         ;NUM. DELLE RIGHE
;
*=$C200
;
MEMLO   *=*+1        ;INIZIO DELLA MEMORIA
MEMHI   *=*+1        ;DA FARE SCORRERE
COFFLO  *=*+1        ;OFFSET PER LA MAPPA DI COLORE
COFFHI  *=*+1
NUMIMM  *=*+1        ;NUM. DELLE IMMAGINI
RITARD  *=*+1        ;VALORE DEL CICLO DI RITARDO
;
;++++ SELEZIONA MODO A 38 COLONNE ++++
;
        LDA RGSCOR
        AND #$F7
        STA RGSCOR
;
;++++ SELEZIONA PRIMA POSIZ. SCORRIMENTO ++++
;
        LDX NUMIMM
NEXSCR
        LDY #NUMCOL
START
        TXA
        PHA          ;APPOGGIA I REGISTRI X,Y
        TYA          ;SULLO STACK
        PHA
;
        LDA RGSCOR
        AND #$FB
        CLC
        ADC #$07
        STA RGSCOR
;
;++++ SCHERMO & COLORE UNA POSIZ. A SINISTRA ++++
;
        LDA #BYBAIM
        STA SCRPTR   ;PREPARA I PUNTATORI
        LDA #BYALIM
        STA SCRPTR+1 ;ALLA PAGINA 0 PER IL
        LDA #BYBACO
        STA PUNTCO   ;TRASFERIMENTO
        LDA #BYALCO
        STA PUNTCO+1
;
        LDX #BLOCC
AGAIN
        LDY #$01
NEXT
        LDA (SCRPTR),Y
        DEY
        STA (SCRPTR),Y
        INY
        LDA (PUNTCO),Y
        DEY
        STA (PUNTCO),Y
        INY
        INY
        BNE NEXT
;
;++++ COPIA SOPRA IL LIMITE DELLA PAGINA ++++
        INC SCRPTR+1     ;INCR. I BYTE ALTI DEI
        INC PUNTCO+1     ;PUNTAT. ALLA  PAGINA 0
        LDA (SCRPTR),Y
        DEC SCRPTR+1
        DEY
        STA (SCRPTR),Y   ;COPIA SOPRA LA PAGINA
        INY
        LDA (PUNTCO),Y
        DEC PUNTCO+1
        DEY
        STA PUNTCO,Y
        INC SCRPTR+1     ;INCR. NUOVAMENTE I PUNTAT.
        INC PUNTCO+1     ;ALLA PAGINA 0
        DEX
        BNE AGAIN
;
;++++ SELEZIONA I BYTE EXTRA ++++
        LDX #$01
ANTHER
        LDA (SCRPTR),Y
        DEY
        STA (SCRPTR),Y
        INY
        LDA (PUNTCO),Y
        DEY
        STA (PUNTCO),Y
        INY
        CPY #EXTRA
        BCC ANTHER       ;RIPETE SE Y < EXTRA
;
;++++ INSERISCE LA COLONNA DI DESTRA PER L'IMMAGINE ++++
;
        LDA MEMLO
        STA PUNTME
        LDA MEMHI        ;PREPARA LA PAGINA 0
        STA PUNTME+1     ;PUNTAT. ALLA MEMORIA
        LDA #NUMCOL-1
        STA SCRPTR       ;PREPARA LA PAGINA 0
        LDA #BYALIM      ;PUNTAT. ALL'AREA PER LO SCHERMO
        STA SCRPTR+1
        JSR COPY40       ;TRASFERISCE LA COLONNA
;
;++++ INSERISCE LA COLONNA DI DESTRA PER IL COLORE ++++
;
        LDA MEMLO
        CLC
        ADC COFFLO       ;SOMMA L'OFFSET ALLA
        STA PUNTME       ;MAPPA PER IL COLORE
        LDA MEMHI        ;E SELEZIONA I PUNTAT.
        ADC COFFHI       ;ALLA PAGINA 0
        STA PUNTME+1
        LDA #NUMCOL-1    ;PREPARA LA PAGINA 0
        STA SCRPTR       ;PUNTAT
        LDA #BYALCO      ;ALLA RAM PER
        STA SCRPTR+1     ;IL COLORE
        JSR COPY40       ;ESEGUE IL TRASFERIMENTO
;
;++++ FA SCORRERE LE POSIZIONI DA 6 A 0 ++++
;
        LDX #$06
MORE1
        LDA RGSCOR
        AND #$F8
        STA RGSCOR
        TXA
        CLC
        ADC RGSCOR
        STA RGSCOR
;
        LDY RITARD       ;CONTEGGIO ALLA ROVESCIA
MORE2
        DEY              ;VALORE DI RITARDO
        BNE MORE2
        DEX
        BPL MORE1
;
;++++ INCREMENTA IL PUNTAT. ALLA MEMORIA ++++
;
        LDA MEMLO
        CLC
        ADC #$01
        STA MEMLO
        LDA MEMHI
        ADC #$00
        STA MEMHI
;
;++++ VERIFICA PER FINE AREA DI MEMORIA ++++
;
        PLA
        TAY              ;RECUPERA DALLO STACK I
        PLA
        TAX              ;REGISTRI X,Y
        DEY
        BEQ NOJMP
        JMP START
NOJMP
        LDA MEMLO
        CLC
        ADC #$C0         ;SOMMA 1000-40 AL
        STA MEMLO        ;PUNTAT. ALLA MEMORIA
        LDA MEMHI
        ADC #$03
        STA MEMHI
        DEX
        BEQ RETRN
        JMP NEXSCR
RETRN
        RTS
;
;++++ S/R DI TRASFERIMENTO OGNI 40 BYTE ++++
;
COPY40
        LDX #NUMRIG
        LDY #$00
REPEAT
        LDA (PUNTME),Y
        STA (SCRPTR),Y
        DEX
        BEQ FINISH       ;FATTE TUTTE LE RIGHE?
;
        LDA SCRPTR
        CLC
        ADC #NUMCOL      ;INCR. PUNTAT. DI 40
        STA SCRPTR
        LDA SCRPTR+1
        ADC #$00
        STA SCRPTR+1
;
        LDA PUNTME
        CLC
        ADC #NUMCOL
        STA PUNTME
        LDA PUNTME+1
        ADC #$00
        STA PUNTME+1
        JMP REPEAT
FINISH
        RTS
```

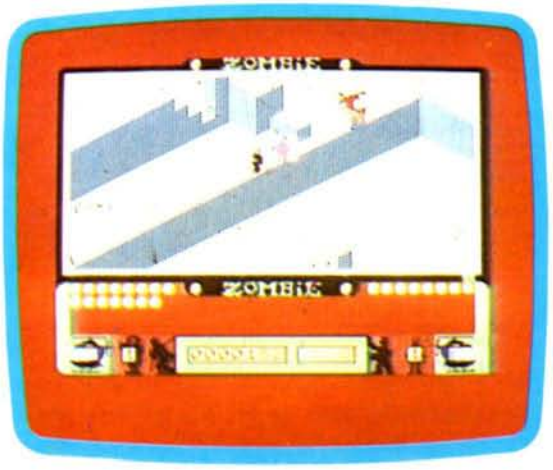
Software in vetrina

Una selezione tra i programmi disponibili per i computer più diffusi

È questa la volta di due giochi, entrambi realizzati per lo Spectrum della Sinclair. I giocatori più accaniti riconosceranno nel primo un rifacimento di un gioco ormai divenuto un classico.

"Zombie Zombie"

Prodotto dalla Quicksilva, questo gioco si svolge nello scenario di Antescher, quello del famosissimo *Ant Attack*. Qui, però, gli avversari del giocatore sono dei personaggi piuttosto inconsueti: Zombie, ossia morti viventi. Solitamente colorati di verde, diventano rossi di rabbia quando si apprestano ad attaccare chi li avvicina.
Per ripulire la città da questa invasione di fan dei Duran Duran, il giocatore dispone di un elicottero. Tuttavia, l'eliminazione degli Zombie avviene con pericolosi "corpo a corpo". Rispetto ad *Ant Attack*, questo gioco offre qualche opzione in più e, forse, è anche più piacevole e divertente. Ad esempio, l'elicottero può trasportare mattoni per personalizzare le mura della città sulle quali, inoltre, è consentito tracciare graffiti.





"Eddie Kidd Jump Challange"

Questo può essere considerato un gioco di abilità: si tratta di saltare, in sella ad una motocicletta, un certo numero di automobili, per poi atterrare sani e salvi sulla pedana di arrivo. Un'errata impostazione produce conseguenze spettacolari, tuttavia (a differenza di quanto accade in realtà) qui il motociclista si rialza sempre in ottima forma per salutare il pubblico!
Come simulazione lascia un po' a desiderare, in quanto poco conta la rincorsa presa: basta portare la moto al massimo dei giri nella marcia più alta e... spiccare il volo. Nella parte inferiore dello schermo è visualizzato un pannello di controllo con tutti i parametri ed il punteggio raggiunto. Il gioco (prodotto dalla Martech) prevede anche un'opzione di "allenamento" con una bicicletta MBX, ma la facilità con la quale si ottengono successi (quasi sempre) la rende scarsamente divertente.

Strumenti per comunicare

Qualche suggerimento sulla scelta di un modem e del relativo software di comunicazione

Prima di acquistare un computer conviene decidere quale software si intende usare e quindi scegliere l'hardware più adatto a tale tipo di software. La stessa riflessione vale per l'acquisto di un sistema di comunicazioni. Tuttavia, in linea di principio, è sempre consigliabile scegliere modem e software della stessa casa che produce il computer.

La prima cosa da stabilire è l'uso che si vuol fare del sistema. Per collegarsi ad un sistema "viewdata", come il britannico Prestel, occorre un modem con una velocità di trasmissione di 1200/75 baud; per la maggior parte dei sistemi di posta elettronica e dei "bulletin board" è necessario un modem da 300 baud. Per accedere al Compunet si deve invece ricorrere ad un particolare modem Compunet, in quanto una parte del software di collegamento risiede nella ROM del modem stesso.

Una seconda caratteristica da considerare è quella delle frequenze impiegate: in Gran Bretagna e nei paesi europei, ad esempio, si usano frequenze CCITT, negli Stati Uniti frequenze BELL. Quindi, per effettuare trasmissioni trans-oceaniche occorrono modem capaci di generare entrambe le frequenze.

Per comunicazioni "da utente a utente" è consigliabile adottare un modem commutabile tra frequenze di "emissione" (originate) e di "risposta" (answer). Se uno dei due modem collegati lavora solo in emissione, l'altro deve poter funzionare in entrambi i modi.

A chi desideri collegarsi a un bulletin board conviene invece acquistare un modem "auto-dial", poiché la maggior parte di tali sistemi impiega una sola linea telefonica e consente l'accesso di un solo utente alla volta. Dato il notevole traffico spesso presente sulla linea, usare un modem che consenta la ripetizione automatica della chiamata (auto-dial, appunto) facilita enormemente l'accesso. Un modem auto-dial è controllato da un apposito software, che consente la composizione del numero (tramite la tastiera oppure attraverso un database di numeri telefonici) e il suo invio al modem in un formato adeguato. Sfortunatamente, ogni modello di modem auto-dial adotta un proprio formato di comunicazione, quindi il modem e il software devono essere compatibili: questo è un ulteriore motivo per acquistare modem e software prodotti da una stessa compagnia.

Se si vogliono ricevere informazioni direttamente da altri utenti, può essere conveniente acquistare un modem dotato anche di "auto-answer" (risposta automatica). Adoperando un tale tipo di modem sul telefono di casa, però, è consigliabile avvisare i propri amici e parenti che, se chiameranno durante una trasmissione in atto, udranno un fischio di dieci secondi e la comunicazione verrà interrotta; ciò non è affatto un tentativo di offenderli, bensì il regolare sistema adottato dal modem!

I modem auto-answer sono controllati da un apposito software, che può avere diverse finalità: dai pacchetti che aprono un nuovo file ad ogni chiamata e lo salvano su disco al termine della comunicazione ai più sofisticati programmi per bulletin board come il *TBBS*. Un interessante programma per auto-answer, ideale per gli utenti di micro portatili o da tavolino, che girano sotto il sistema opera-

Sistema in funzione

Il gran numero di apparecchi e di software di comunicazione in commercio indubbiamente non facilita la scelta del modem: il sistema "ideale", mostrato in figura, fonde le caratteristiche più interessanti dei modem oggi disponibili. Ai più inesperti conviene quindi spiegare le proprie esigenze di comunicazione al fornitore di fiducia e lasciare a quest'ultimo la scelta del pacchetto più adatto



STEVE CROSS



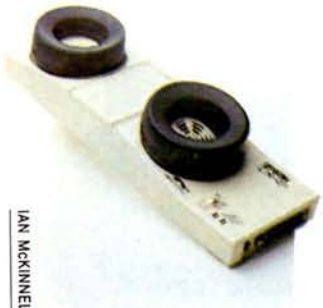
Protek 1200
Tipo: modem acustico
Velocità di trasmissione: 1200/75 e 1200/1200 baud



Prism 1000
Tipo: Modem viewdata a collegamento diretto
Velocità di trasmissione: 1200/75 baud



Modem di comunicazione Commodore
Tipo: Modem viewdata a collegamento diretto per Compunet
Velocità di trasmissione: 1200/75 e 1200/1200 baud



IAN McKINNELL

Epson CX-21
Tipo: Modem acustico
Velocità di trasmissione: 300 baud

tivo CP/M, è il *Remote CP/M*. Questo pacchetto consente di eseguire una chiamata con un micro CP/M e di eseguire qualsiasi programma CP/M via telefono.
Il software scelto per il proprio sistema deve possedere come minimo la capacità di caricare in memoria (download) e di trasmettere (upload) file ASCII. Inoltre deve essere compatibile con le memorie di massa in uso. I programmi BASIC possono essere trasmessi in formato ASCII (vedere a pag. 1401) ma per l'invio dei file binari (ad esempio, i file .COM del CP/M) sono necessari particolari protocolli di trasmissione binaria: quello usato più di frequente è lo XMODEM.
Se il sofwtare adottato prevede la creazione di file di ingresso automatico (auto-log on) per sistemi di altro tipo, al momento del collegamento con un diverso sistema è sufficiente caricare l'appropriato file, corredandolo di codice d'identificazione, parola chiave e così via. Alcuni sistemi auto-dial collegano tali file a un database di numeri telefonici; è sufficiente che l'utente digiti il nome dell'applicazione desiderata: il software compone quindi il numero telefonico, esegue la chiamata ed entra automaticamente nel sistema.
Definite le funzioni base del sistema, è necessario reperire un modem e il software capace di eseguirle: il modo migliore consiste nel redigere un piccolo elenco delle caratteristiche desiderate, consegnandolo al negoziante. Egli potrà così consigliare la combinazione hardware/software più adatta.

Scelta del terminale

Se già si possiede un micro, lo si vorrà probabilmente usare come terminale. Qualsiasi computer può svolgere questo lavoro, anche uno ZX81 dalle capacità limitate, ma alcuni apparecchi si dimostrano più adatti di altri. La seguente panoramica prende in esame alcuni dei più diffusi home computer.
L'apparecchio che meglio si adatta a tale trasformazione è il Micro BBC. A questo scopo, è sufficiente scrivere un banale programma per terminale in linguaggio BASIC:

```
100 REM Programma per BBC/terminale
110 *FX2,2
120 *FX3,1
130 REPEAT: GET A$: IF$ = CHR$(13) THEN PRINT
140 PRINT A$;: UNTIL FALSE
```

Purtroppo i numerosi pacchetti di comunicazione sviluppati per il Micro BBC, alcuni dei quali su ROM, sono piuttosto costosi. Tuttavia, molti di essi offrono tutte le funzioni che un utente possa desiderare e la maggior parte del lavoro viene svolta dal programma sul computer: l'utente si limita ad impartire qualche semplice comando.
Lo Spectrum presenta invece alcuni problemi di adattabilità. Innanzi tutto il BASIC pone alcuni limiti al sistema di comunicazione: con un programma BASIC non si possono ottenere velocità di trasmissione superiori ai 10 baud, né operazioni come la registrazione dei caratteri in RAM. L'Interfaccia 1 resta fuori gioco perché, non essendo una porta RS232, è di scarso aiuto nelle comunicazioni. Infine lo Spectrum non dispone di alcun software di comunicazione.
Il Commodore 64 possiede un'interfaccia seriale non standard, quindi la maggior parte dei modem per questo apparecchio viene collegata alla porta utente. Anche in questo caso la programmazione in BASIC offre vantaggi limitati. Dato che il CBM 64 usa una serie di caratteri ASCII non standard, il software di comunicazione deve essere tradotto dall'ASCII standard in quello Commodore depositando e consultando in memoria una tabella di conversione. Inoltre non esiste software di comunicazione "omologato" dalla Commodore. Si può ricorrere però al *Termulator*, prodotto dalla Chris Townsend Computer, oppure a uno dei tanti package che circolano tra gli "User Group" o pubblicati in riviste specializzate.
La Tandy ha sviluppato, per i suoi apparecchi TRS-80 con unità a dischetti, un software per terminale che funziona con vari sistemi operativi. Tale programma viene adoperato principalmente per il trasferimento diretto tra due computer, ma può essere usato anche con i modem. I micro della Tandy sono stati i primi ad essere impiegati per controllare i bulletin board, quindi dispongono di un buon numero di programmi di comunicazione basati su disco e su cassette. Infine, gli apparecchi TRS-80 non consentono operazioni di tipo viewdata (1200/75 baud).
Quasi tutti i micro gestionali possono essere usati nei sistemi di comunicazione, ma a questo proposito sono necessarie alcune distinzioni. Sempre più spesso questi apparecchi vengono venduti con modem e relativo software di comunicazione già incorporati. In tal caso è sufficiente collegarli ad una presa telefonica e il gioco è fatto.
Qualora il modem non sia incorporato, conviene scegliere micro dotati di almeno due porte RS232, collegabili contemporaneamente ad un modem e ad una stampante seriale. Alcuni modelli sono forniti di porte per modem non standard e indipendenti, allacciabili al modem per mezzo di un apposito cavo.
Per quanto riguarda il software, non si dovrebbero avere problemi con apparecchi che lavorano sotto CP/M, MS-DOS o PC-DOS. Nel caso di sistemi operativi non standard, i pacchetti di comunicazione creano difficoltà pari a quelle riscontrate con qualsiasi altro tipo di software.
Le macchine maggiormente indicate per i sistemi di comunicazione restano tuttavia i portatili cosiddetti "da viaggio", come il Tandy Modello 100, il NEC PC8201A e l'M10 della Olivetti, che, abbinati ad un modem a pile, si dimostrano eccellenti terminali portatili di ridotte dimensioni.
Tutti e tre gli apparecchi incorporano un text editor ed un software per il terminale e le loro pile formato AA garantiscono un tempo di lavoro di circa 20 ore.

Programmi per il robot

Proviamo a misurare oggetti usando il nostro veicolo robot

Affinché un robot possa individuare e misurare il lato di un oggetto, occorre dotarlo di un software discretamente sofisticato. Per esplorare l'oggetto in questione il robot si serve dei sensori a microinterruttori descritti a pag. 1356. Il metodo di misurazione è all'incirca il seguente:

1) Individua l'oggetto.
2) Localizza un'estremità del lato da misurare.
3) Esplora il lato dell'oggetto fino ad incontrare l'estremità opposta.

Per eseguire con successo la prima operazione, il robot va opportunamente orientato verso il lato dell'oggetto da misurare, prima di avviare il programma. Tuttavia può accadere che il robot entri in contatto con un'estremità del lato attraverso un solo sensore frontale e non con entrambi. La figura in basso mostra le diverse situazioni possibili. In questo caso è possibile chiarire se il sensore in contatto con l'oggetto è il destro o il sinistro e seguire poi la strategia più appropriata.

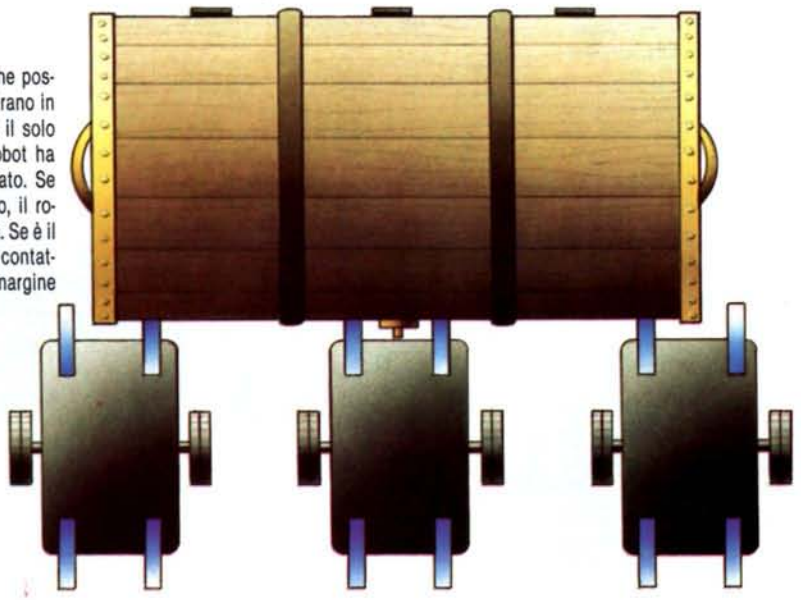
Il programma richiede che il robot venga inizialmente orientato perpendicolarmente (a 90°) rispetto al lato da misurare. In questo modo è possibile evitare contatti obliqui.

La seconda fase prevede che il robot si muova sempre verso l'estremità destra dell'oggetto, prima di iniziare la misurazione. Per localizzare il margine destro, la macchina si sposta lungo il lato, eseguendo una serie di piccoli "passi". Tale scorrimento si interrompe quando solo il sensore sinistro resta in contatto con l'oggetto. Per "esplorare" il lato, il robot esegue una complessa serie di manovre, ciascuna delle quali implica 5 movimenti. Supponiamo che, inizialmente, il robot si trovi in contatto con l'oggetto: per prima cosa deve arretrare, quindi ruotare di 90° verso destra, muoversi in avanti di una certa distanza, ruotare di 90° verso sinistra ed infine avanzare, finché i sensori non toccano di nuovo l'oggetto. La figura a pagina seguente illustra l'intera manovra. La lunghezza del passo, ossia la distanza tra due punti di contatto successivi con l'oggetto, corrisponde a quella parte della manovra durante la quale il robot si muove parallelamente al lato da misurare.

Per localizzare con precisione il margine destro, non è necessario che il robot esplori il lato eseguendo per tutto il tragitto passi di pochi millimetri. L'apparecchio compie passi piuttosto lunghi finché non supera il margine: a questo punto torna indietro esplorando il lato con una serie di piccoli passi fino ad

DA UN MARGINE ALL'ALTRO

La figura mostra le tre possibilità che possono verificarsi quando i sensori entrano in contatto con un oggetto. Quando è il solo sensore di destra a "toccare", il robot ha localizzato l'estremità sinistra del lato. Se entrambi i sensori toccano l'oggetto, il robot occupa una posizione intermedia. Se è il solo sensore di sinistra ad entrare in contatto, l'apparecchio ha raggiunto il margine destro



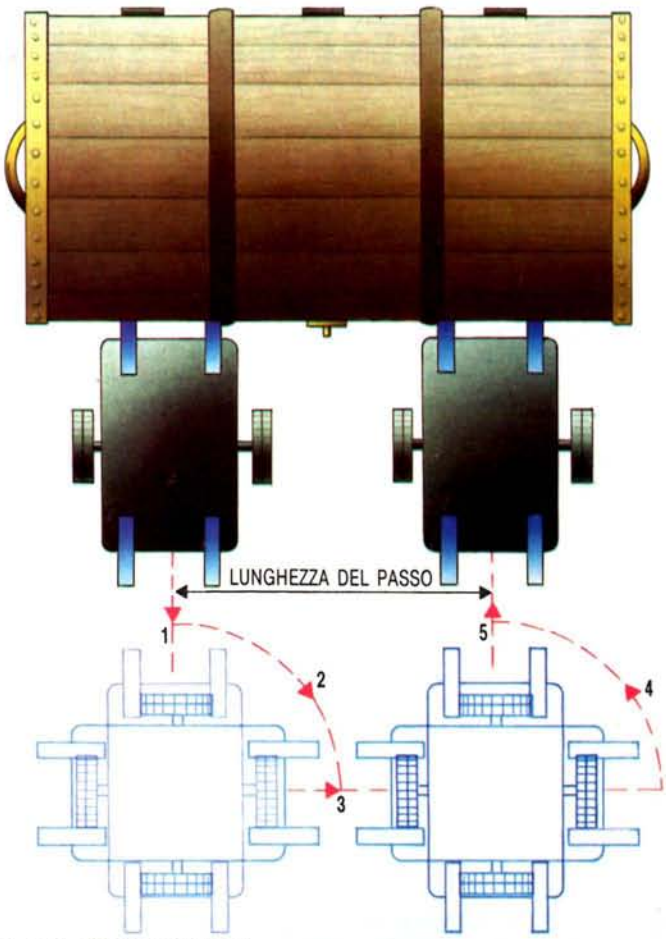
Solo il sensore di sinistra | Entrambi i sensori | Solo il sensore di destra

KEVIN JONES

individuare con precisione l'estremità destra dell'oggetto. Una lunghezza di passo pari alla distanza tra i due sensori frontali, ossia 60 mm. circa, dovrebbe consentire al robot di raggiungere l'estremità senza mai oltrepassarla del tutto, mantenendo almeno un sensore in contatto con l'oggetto.
La terza fase prevede un metodo di esplorazione simile, ma questa volta il robot si sposta verso sinistra e conta i passi che gli occorrono per raggiungere il margine sinistro. Al termine, la lunghezza del lato dell'oggetto si trova registrata nella variabile Contatore e può essere visualizzata.

Programma di misurazione

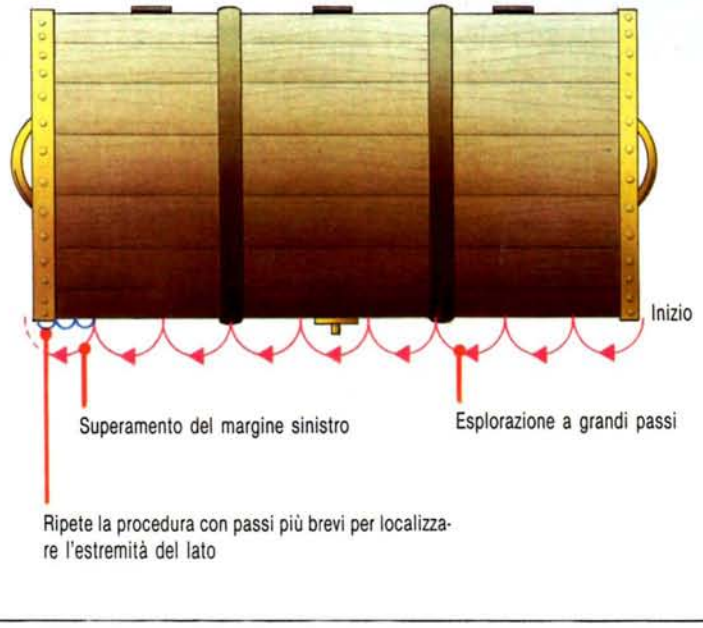
I listati presentati sono quelli per il Micro BBC e per il Commodore 64. I rapporti impulso/distanza e impulso/angolo, calcolati in modo sperimentale a pag. 1374, devono tener conto delle caratteristiche del robot impiegato. L'impostazione procedurale del BASIC BBC è ideale per scrivere un programma di questo tipo, poiché consente di sviluppare un software altamente strutturato, controllando i diversi movimenti del robot con una serie di procedure indipendenti. L'impiego di nomi "estesi" di variabili e la possibilità di trasferire parametri da una procedura ad un'altra rendono il programma chiaro e facilmente comprensibile. La versione Commodore presenta invece maggiori difficoltà di strutturazione dovute alle specifiche caratteristiche della versione BASIC e il programma appare meno comprensibile.
Dopo aver isolato le diverse operazioni che il programma deve eseguire, possiamo scrivere procedure separate per i diversi movimenti e combinarle per ottenere una procedura "di esplorazione", per poi collegare le procedure di esplorazione ottenendo una procedura di "misura" globale. Nel programma è abbastanza agevole distinguere i vari livelli procedurali, che vanno da una semplice procedura di basso livello per l'invio di un impulso ai motori, alla procedura di misurazione di livello massimo.
Un'errata orientazione iniziale del robot può causare notevoli problemi. Se un solo sensore fa contatto, il programma ne deduce che il robot si trova ad un'estremità dell'oggetto. In questo caso è necessario interrompere il programma, collocare il robot perpendicolarmente al lato da misurare e far ripartire il programma dall'inizio.
È inevitabile che il programma commetta alcuni errori di misurazione: la superficie di contatto del sensore è larga circa 5 mm., quindi la misura del lato può presentare un margine d'errore massimo di 10 mm. Inoltre, dato che il robot esplora le due estremità dell'oggetto eseguendo passi di 5 mm., può verificarsi un ulteriore errore di 10 mm. Una serie di prove eseguite con il prototipo, su un oggetto con un lato di 410 mm., ha permesso di calcolare un margine d'errore medio di circa 20 mm., ossia del 5%.



AREA DI MANOVRA

La figura in alto illustra il procedimento di base seguito dal robot durante l'esplorazione. Quando entrambi i sensori fanno contatto, il robot retrocede (1) per poter girare senza urtare l'oggetto. Eseguita una rotazione a destra di 90° (2), avanza di un passo (3). Quindi il robot ruota a sinistra di 90° (4) e si sposta in avanti (5) per verificare se ha ancora di fronte l'oggetto. Questa manovra viene ripetuta per tutta la lunghezza del lato da misurare. Inizialmente il robot compie dei passi piuttosto lunghi per calcolare la lunghezza approssimativa del lato. Quando un solo sensore fa contatto, il robot torna sui propri passi esplorando l'oggetto con una serie di piccoli movimenti

PROCEDURA DI ESPLORAZIONE



KEVIN JONES

LISTATO PER IL MICRO BBC

```
1000 REM **** PROG. DI MISURAZIONE BBC ****
1010 MODE 7
1020 PROCinizializza
1030 PROCmisura
1040 PROCstampa
1050 END
1060 DEF PROCmisura
1070 PROCtrova
1080 REM ** SOLO UN SENSORE ? **
1090 PROCverifica_sensori
1100 REM **** TROVA IL MARGINE****
1110 REPEAT: PROCesplora(destra,ampiezza)
1120 UNTIL (?REGDAT AND 192)=sensore_sin
1130 REM ** MANOVRA PER TROVARE IL MARGINE**
1140 PROCesplora(sinistra,ampiezza)
1150 REPEAT: PROCesplora(destra,piccola_ampiezza)
1160 UNTIL (?REGDAT AND 192)=sensore_sin
1170 PRINT "INDIVIDUA IL MARGINE DESTRO"
1180 PRINT "INIZIO  DELLA MISURAZIONE"
1190 REM ** COMINCIA A MISURARE **
1200 contatore=ampiezza
1210 REPEAT: PROCesplora(sinistra,ampiezza)
1220 contatore=contatore+ampiezza
1230 UNTIL (?REGDAT AND 192)=sensore_des
1240 REM ** MANOVRA PER TROVARE IL MARGINE **
1250 contatore=contatore-ampiezza
1260 PROCesplora(destra,ampiezza)
1270 REPEAT: PROCesplora(sinistra,piccola_ampiezza)
1280 contatore=contatore+piccola_ampiezza
1290 UNTIL (?REGDAT AND 192)=sensore_des
1300 ?REGDAT=0
1310 ENDPROC
1320 :
1330 DEF PROCstampa
1340 CLS
1350 PRINT TAB (5,12) "IL LATO DELL'OGGETTO MISURA "
     ;conteggio;" mm."
1360 ENDPROC
1370 :
1380 DEF PROCinizializza
1390 DDR=&FE62: REGDAT=&FE60
1400 ?DDR=15: REM LINEE 0-3 IN USCITA
1410 ?REGDAT=1: REM BIT DI AZZERAMENTO POSTO A 1
1420 avanti=4: indietro=2: sinistra=6: destra=0
1430 rapp_id=3.34446: rapp_ia=375/90
1440 sensore_des=128: sensore_sin=64
1450 entrambi_sensori=0: nessun_sensore=192
1460 ampiezza=60: piccola_ampiezza=5
1470 ENDPROC
1480 :
1490 DEF PROCcerca(contatto)
1500 REPEAT: PROCesplora(contatto,ampiezza)
1510 UNTIL (?REGDAT AND 192)=entrambi_sensori
1520 ENDPROC
1530 :
1540 DEF PROCtrova
1550 REPEAT: PROCmuove(avanti,8)
1560 UNTIL (?REGDAT AND 192)<>nessun_sensore
1570 ENDPROC
1580 :
1590 DEF PROCverifica_sensori
1600 IF (?REGDAT AND 192)=sensore_des THEN
     PROCcerca(destra): ENDPROC
1610 IF (?REGDAT AND 192)=sensore_sin THEN
     PROCcerca(sinistra): ENDPROC
1620 ENDPROC
1630 :
1640 DEF PROCesplora(movim,passo)
1650 IF modo=destra THEN movim_opp=sinistra ELSE
     movim_opp=destra
1660 PROCmuove(indietro,30)
1670 PROCruota(movim,90)
1680 PROCmuove(avanti,passo)
1690 PROCruota(movim_opp,90)
1700 REPEAT: PROCmuove(avanti,8)
1710 UNTIL (?REGDAT AND 192)<>nessun_sensore
1720 ENDPROC
1730 :
1740 DEF PROCmuove(dir,distanza)
1750 ?REGDAT=(?REGDAT AND 1) OR dir
1760 impulsi=rapp_id*distanza
1770 FOR I=1 TO impulsi: PROCimpulso: NEXT I
1780 ENDPROC
1790 :
1800 DEF PROCruota(dir,angolo)
1810 ?REGDAT=(?DATAERG AND 1) OR dir
1820 impulsi=rapp_ia*angolo
1830 FOR I=1 TO impulsi: PROCimpulso: NEXT I
1840 ENDPROC
1850 DEF PROCimpulso
1860 ?REGDAT=(?REGDAT OR 8)
1870 ?REGDAT=(?REGDAT AND 247)
1880 ENDPROC
```

LISTATO PER IL COMMODORE 64

```
10 REM **** PROG. DI MISURAZONE CBM64 ****
20 GOSUB 1000 : REM INIZIALIZZA
30 GOSUB 2000 : REM MISURA
40 GOSUB 3000 : REM STAMPA
50 END
60 :
1000 REM **** INIZIALIZZA ****
1010 DDR=56579:REGDAT=56577
1020 POKE DDR,15: REM LINEE 0-3 IN USCITA
1030 POKE REGDAT,1: REM BIT DI AZZERAMENTO = 1
1040 AV=4: IN=2: SN=6: DS=0
1050 PD=3.34446:PA=375/90
1060 RB=128:LB=64:BB=0:NB=192
1070 WD=80:SW=5
1080 RETURN
1090 :
2000 REM **** MISURA ****
2010 GOSUB 3500: REM TROVA L'OGGETTO
2020 GOSUB 4000: REM VERIFICA I SENSORI
2030 REM ** TROVA IL MARGINE **
2040 WY=DS: SP=WD: GOSUB 6000: REM ESPLORA
2050 IF(PEEK(REGDAT)AND192)<>LB THEN 2040
2060 REM ** MANOVRA PER TROVARE IL MARGINE **
2070 DR=SN: DS=WD: GOSUB 6000: REM ESPLORA
2080 DR=DS: DS=SW: GOSUB 6000: REM ESPLORA
2090 IF (PEEK(REGDAT)AND 192)<>LB THEN 2080
2100 PRINT "TROVA IL MARGINE DESTRO"
2110 PRINT "INIZIO DELLA MISURAZIONE"
2120 REM ** COMINCIA A MISURARE **
2130 CC=WD
2140 DR=SN:DS=WD:GOSUB6000:CC=CC+WD
2150 IF(PEEK(DATREG)AND192)<>RB THEN 2140
2160 REM ** MANOVRA PER TROVARE IL MARGINE **
2170 CC=CC-WD
2180 DR=DS:DS=WD:GOSUB6000:CC=CC+SW
2190 IF(PEEK(REGDAT)AND192)<>RB THEN 2180
2200 POKE REGDAT,0
2210 RETURN
2220 :
3000 REM **** STAMPA ****
3010 PRINTCHR$(147)
3020 PRINT "L'OGGETO MISURA "¡CC¡" MM."
3030 RETURN
3040 :
3500 REM **** TROVA ****
3510 DR=AV: DS=5: GOSUB 7000: REM MUOVE
3520 IF(PEEK(REGDAT)AND 192)=NB THEN 3510
3530 RETURN
3540 :
4000 REM **** VERIFICA I SENSORI ****
4010 IF(PEEK(REGDAT)AND192)=RB THEN SS=DS:GOSUB5000:RETURN
4020 IF(PEEK(REGDAT)AND192)=LB THEN SS=SN:GOSUB5000:RETURN
4030 RETURN
4040 :
5000 REM **** CERCA (SS) ****
5010 DR=AV: DS=8: GOSUB 7000: REM MUOVE
5020 IF(PEEK(REGDAT)AND192)=NB THEN 5010
5030 RETURN
5040 :
6000 REM **** ESPLORA (WY,SP) ****
6010 IF WY=DS THEN OW=SN
6020 IF WY=SN THEN OW=DS
6030 DR=IN: DS=30: GOSUB 7000: REM MUOVE
6040 DR=WY: AG=90: GOSUB 7500: REM RUOTA
6050 DR=AV: DS=SP: GOSUB 7000: REM MUOVE
6060 DR=OW: AG=90: GOSUB 7500: REM RUOTA
6070 DR=AV: DS=8: GOSUB 7000: REM MUOVE
6080 IF(PEEK(REGDAT)AND192)=NB THEN 6070
6090 RETURN
6100 :
7000 REM **** MUOVE (DR,DS) ****
7010 POKE REGDAT,(PEEK(REGDAT)AND 1)OR DR
7020 PL=PD*DS
7030 FOR I=1 TO PL:GOSUB8000:NEXT I
7040 RETURN
7050 :
7500 REM **** RUOTA (DR,AG) ****
7510 POKE DATREG,(PEEK(REGDAT)AND 1)OR DR
7520 PL=PA*AG
7530 FOR I=1 TO PL:GOSUB8000:NEXT I
7540 RETURN
7550 :
8000 REM **** IMPULSO ****
8010 POKE REGDAT,PEEK(REGDAT)OR 8
8020 POKE REGDAT,PEEK(REGDAT)AND 247
8030 RETURN
```

Il package "Human Edge"

Prosegue la panoramica sul software "verticale"

La Human Edge Software, una ditta di programmazione californiana, ha compiuto un ulteriore passo verso l'intelligenza artificiale: i sofisticati pacchetti gestionali, da essa prodotti, sono capaci di suggerire, in tempi ragionevolmente brevi, le migliori decisioni da prendere in base alle informazioni fornite dall'utente. Non si tratta, però, di veri programmi "pensanti": la notevole massa di dati viene elaborata ad alta velocità e valutata secondo criteri precedentemente memorizzati, fino a consigliare precise strategie operative. Il package *Human Edge*, per apparecchi IBM e IBM-compatibili, si compone di 4 programmi: *Communications Edge, Sales Edge, Management Edge* e *Negotiation Edge*. La ditta produttrice afferma che il pacchetto permette all'utente di sviluppare le proprie capacità professionali nelle diverse aree trattate dai programmi. Esiste anche una versione ridotta del pacchetto (chiamata *Mind Prober*) adatta per il Commodore 64, l'Apple II ed il Macintosh e basata sugli stessi principi di funzionamento.

I programmi, frutto di oltre 10 anni di ricerche da parte di studiosi del comportamento ed esperti di economia, applicano nuove tecniche quali l'analisi dei fattori umani, la tecnologia dei sistemi avanzati e la matematica della teoria decisionale.

Questa presentazione di tipo specialistico e l'alto prezzo dei programmi possono intimidire il potenziale utente ma, in realtà, i programmi sono facili da usare e basta meno di un'ora per apprenderne, da soli, il funzionamento. Il package è interamente pilotato da menu ed è basato su lunghi questionari, composti da una serie di asserzioni, con le quali l'utente deve dichiararsi d'accordo o meno. I questionari esplorano i tratti più significativi della personalità dell'utente, ma esaminano anche le sue capacità di vendita, i clienti attuali, le persone a lui sottoposte e quelle a lui superiori: insomma un po' tutti i diversi aspetti delle relazioni commerciali che l'utente desideri approfondire.

Il programma valuta le risposte ottenute e produce una dettagliata relazione che contiene alcuni consigli di comportamento. Ad esempio, il miglior modo per prender contatto con un nuovo cliente, un'efficace strategia di liquidazione se l'azienda si trova in difficoltà economiche oppure i diversi modi di condurre i negoziati aziendali con impiegati ed impiegate.

Ogni programma inizia con un questionario di auto-valutazione, composto da frasi del tipo: «Cerco di avere il sopravvento durante le riunioni», «Discuto con gli altri più del necessario», «Sono piuttosto impulsivo», e così via. L'utente seleziona le affermazioni che meglio descrivono la sua personalità. Il questionario formula anche numerose domande "incrociate", che garantiscono una certa validità alla valutazione finale. Ad esempio, le risposte «Sono piuttosto impulsivo» e la successiva «Spesso agisco senza pensare» vengono confrontate per verificarne la coerenza. Al termine, le risposte vengono memorizzate su disco e possono essere aggiornate e richiamate in qualsiasi momento.

Dopo aver risposto al questionario, l'utente deve esprimere il proprio parere su una serie di aggettivi riferiti all'oggetto dell'indagine, ad esempio un cliente. Il programma ricorre a termini come chiacchierone, apprensivo, indipendente, determinato, ambizioso, cortese, vanitoso ecc. per aiutare l'utente a valutare il carattere del cliente.

L'utente può scorrere la lista degli aggettivi e cambiare risposta in qualsiasi momento. Anche in questo caso la lista viene archiviata su disco e può essere modificata alla luce di esperienze successive.



DAVID HIGHAM

Il questionario di auto-valutazione viene compilato una sola volta; in seguito può essere richiamato dal disco e messo a confronto con una delle numerose liste di valutazione della cosiddetta "controparte". Solo otto di queste liste possono essere archiviate sul dischetto. Dopo che l'utente ha richiamato dal disco due liste definitive, il programma esegue un'analisi delle risposte e stampa una relazione che sintetizza le caratteristiche dell'utente e della sua controparte.
I seguenti brani, creati (in lingua inglese) dal programma *Communication Edge*, si riferiscono ad un padre (l'utente) e al suo giovane figlio (la controparte), indicato col termine generico di Sig. T. Il computer si rivolge in modo diretto all'utente del programma:

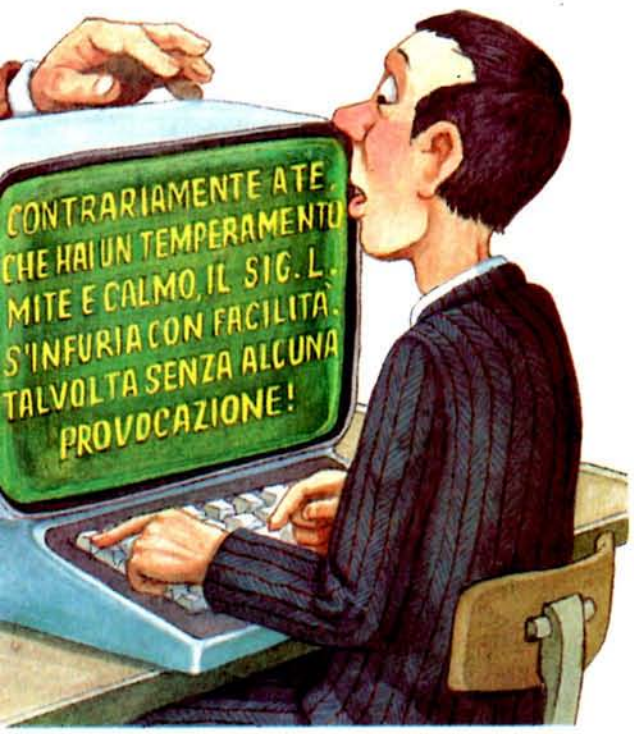
«Per poter dialogare col Sig. T devi tenere un atteggiamento flessibile e aperto. Ha un carattere molto riservato e solitario, non ama parlare né socializzare con gli altri. Probabilmente risponderà alle tue sollecitazioni con un atteggiamento cinico e sospettoso. Quando parli con lui, cerca di usare frasi facilmente comprensibili ma, soprattutto, concise e dirette.»

«Contrariamente a te, che possiedi un pacato modo di esprimerti, il Sig. T si adira facilmente e prima di iniziare la conversazione può sembrare nervoso. Probabilmente cercherà di importi le sue opinioni. Mantieniti cordiale, nonostante la sua aggressività e imprevedibilità. Le sue frasi possono essere ora impulsive ora attentamente meditate. Cerca di guidare sempre la conversazione. Parafrasa le sue affermazioni cercando di raggiungere la maggior chiarezza possibile ed una concordanza di vedute.»

I due brani descrivono abbastanza bene il tipico atteggiamento di un adolescente sebbene il programma *Communication Edge* non rivolga alcuna domanda sull'età del soggetto (ma, curiosamente, sul suo sesso sì!).
I termini usati con maggior frequenza nelle relazioni sono caratteristici di molti test di personalità.
L'utente viene solitamente descritto con aggettivi del tipo: flessibile, ragionevole, pacato, mentre per la controparte vengono usati termini meno lusinghieri come nervoso, imprevedibile, cinico, sospettoso ecc.
Questa diversità, che rispecchia forse il modo di pensare tipico del manager americano (l'applicabilità di tali programmi in una realtà italiana è alquanto discutibile) è ancor più accentuata nel programma *Negotiation Edge*, che consiglia comportamenti di questo genere:

SERVITI DEL SIG. T PER I TUOI SCOPI
BLANDISCI IL SIG. T CON QUALCHE CONCESSIONE
METTILO IN CONDIZIONE DI DOVERTI RISPONDERE "SI"
VELA LE TUE MINACCE
PONI L'ACCENTO SULLE TUE DIFFICOLTÀ
MINIMIZZA I TUOI PROFITTI
OFFRI UNA GRATIFICA

La presenza di questionari di auto-valutazione quasi identici in tutti e quattro i programmi indica che la Human Edge Software intende vendere i pacchetti anche separatamente. Questa scelta probabilmente è dovuta al loro alto prezzo, ma una programmazione meglio coordinata avrebbe consentito l'impiego di un unico questionario di auto-valutazione per tutti e quattro i programmi. In tal caso, però, dato che i programmi sono piuttosto simili, all'utente sarebbe bastato acquistare il meno costoso, ossia il *Communication Edge*, e adattare le relazioni che esso produce a una varietà di situazioni diverse.
L'uso dei programmi come strumenti gestionali evidenzia tuttavia un grosso limite: la loro incapacità di modificarsi in base a precedenti esperienze. Per esempio, all'utente potrebbe far comodo introdurre i risultati di una strategia commerciale precedentemente suggerita e modificarli alla luce dell'esperienza, soprattutto per quanto concerne la valutazione delle controparti. Inoltre, per molte delle domande che i programmi pongono, non è sufficiente una sintetica risposta di accettazione o di rifiuto: una più efficace strutturazione "ad albero" dei questionari avrebbe consentito una più ampia verifica delle risposte. Per migliorare le prestazioni dei programmi si potrebbe infine introdurre la risposta «non so», che aprirebbe la strada a domande di livello inferiore a cui rispondere sinteticamente «si» o «no».
In conclusione, l'uso più corretto di questo sofisticato pacchetto è forse quello per il miglioramento della preparazione dell'utente prima di un incontro o di una riunione, ricordando però di non prendere troppo alla lettera i consigli ricevuti: almeno fino a quando i computer non diverranno vere e proprie macchine pensanti!



Human Edge
Una serie di quattro pacchetti (acquistabili separatamente) per computer MS-DOS e PC-DOS.
Attualmente solo in lingua originale (inglese)

Autori
Human Edge Software, California, USA

Formato
Dischetto

Mind Prober
Per Commodore 64, Apple II e Macintosh. Attualmente solo in lingua originale (inglese)

Autori
Human Edge Software, California, USA

Formato
Dischetto

DATABASE

Ecco la quinta tabella sulle funzioni delle routine ROM del Micro BBC, assieme ai relativi indirizzi

DESCRIZIONE	INDIRIZZO
Find ERL trova linea contenente errore (ERL)	B3C5
Fix FAC#1 fissa FAC#1	A3FE
Floating point addition somma di numeri reali	9C8B
Floating point comparison confronto tra numeri reali	9A50
Floating point multiplication moltiplicazione tra numeri reali	9D20
Floating point subtraction sottrazione tra numeri reali	9CE1
GCOL statement routine routine istruzione GCOL	937A
GET function routine routine funzione GET	AFB9
GET$ function routine routine funzione GET$	AFBF
GOSUB statement routine routine istruzione GOSUB	B888
GOTO statement routine routine istruzione GOTO	B8CC
Get a variable name prende un nome di variabile	95DD
Get an array/PROC/FN name prende un nome di matrice/PROC/FN	9559
Get character from PTR#1 prende un carattere da PTR#1	8A97
Get character from PTR #2 prende un carattere da PTR#2	8A8C
Get file handle routine gestione file	BFA9
Get the value of a variable estrae valore di una variabile	B32C
HIMEM function routine routine funzione HIMEM	AF03
HIMEM statement routine routine istruzione HIMEM	925D
IF statement routine routine istruzione IF	98C2
INKEY function routine routine funzione INKEY	ACAD
INKEY$ function routine routine funzione INKEY$	B026
INPUT statement routine routine istruzione INPUT	BA44
INPUT# statement routine routine istruzione INPUT#	B9CF
INSTR function routine routine funzione INSTR	ACE2

INDIRIZZO	DESCRIZIONE
9B9C	**LEVEL 3 of expression evaluator** calcolo espressioni LIVELLO 3
9BAE	**= operator routine** routine operatore =
9BC0	**< operator routine** routine operatore <
9BD4	**< = operator routine** routine operatore < =
9BDF	**<> operator routine** routine operatore <>
9BE8	**> operator routine** routine operatore >
9BFA	**> = operator routine** routine operatore > =
9C15	**String concatenation** concatenazione di stringhe
9C42	**LEVEL 4 of expression evaluator** calcolo espressioni LIVELLO 4
9C4E	**+ operator routine** routine operatore +
9C53	**Integer addition** somma di interi
9C8B	**Real addition** somma di numeri reali
9CB5	**— operator routine** routine operatore —
9CBA	**Integer subtraction** sottrazione tra interi
9CE1	**Real subtraction** sottrazione tra numeri reali
9D20	**Real multiplication** moltiplicazione tra numeri reali
9D3C	*** operator routine** routine operatore *
9D41	**Integer multiplication** moltiplicazione tra interi
9DD1	**LEVEL 5 of expression evaluator** calcolo espressioni LIVELLO 5
9DE5	**/ operator routine** routine operatore /
9E01	**MOD operator routine** routine operatore MOD
9E0A	**DIV operator routine** routine operatore DIV
9E20	**LEVEL 6 of expression evaluator** calcolo espressioni LIVELLO 6
9E35	**^ operator routine** routine operatore ^
9E90	**Print number in hexadecimal** visualizza un numero in esadecimale

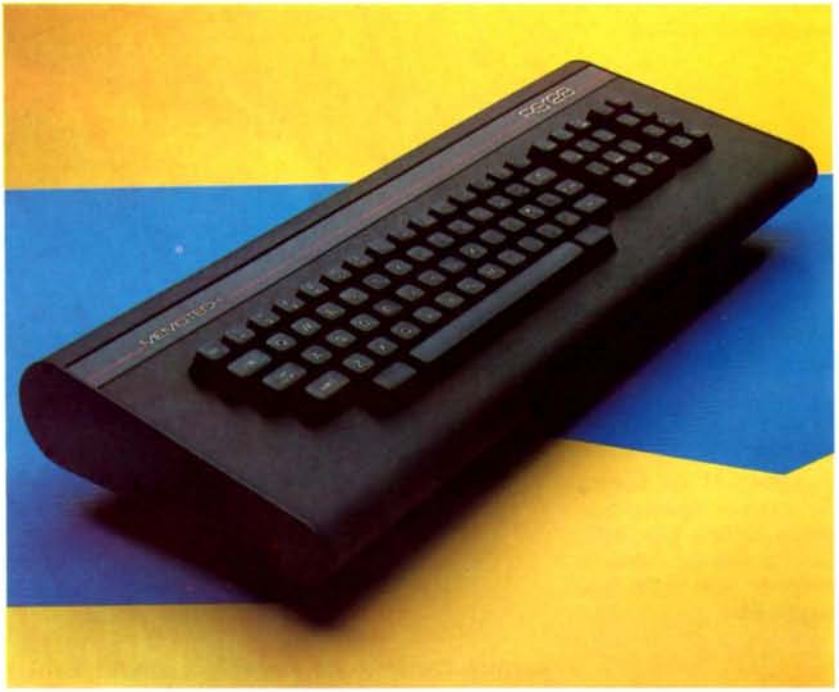
Memotech RS 128

Anatomia di un microcomputer a diffusione europea: tecnologie a confronto

Sebbene siano apparecchi di ottima qualità, dotati di grafica in alta risoluzione e di un assembler incorporato, di un BASIC sofisticato e di un linguaggio per la gestione dei testi, chiamato NODDY (unico nel suo genere), i micro MTX 500 e MTX 512 della serie Memotech 500 sono stati piuttosto trascurati dagli appassionati di home computer. Lo scarso successo commerciale è dovuto probabilmente alla loro anomala collocazione sul mercato degli home computer.
Da un lato il loro prezzo è giudicato troppo elevato dagli appassionati di giochi per computer; dall'altro, sebbene la Memotech affermi che la serie 500 è adatta alle piccole attività gestionali, gli utenti orientati verso un tipo di software "serio" hanno ritenuto inaccettabile la mancanza di interfacce incorporate per il collegamento con unità a dischetti. Tali interfacce sono in realtà disponibili come schede opzionali da innestare su un apposito connettore interno.
Tuttavia questa scelta, dovuta alla notevole esperienza della compagnia nella produzione di schede aggiuntive per lo ZX81, non ha incontrato il favore di molti potenziali acquirenti. La Memotech ha quindi deciso di lanciare un nuovo apparecchio, l'RS 128, dotato di interfacce incorporate.

Lo "stile" dell'apparecchio

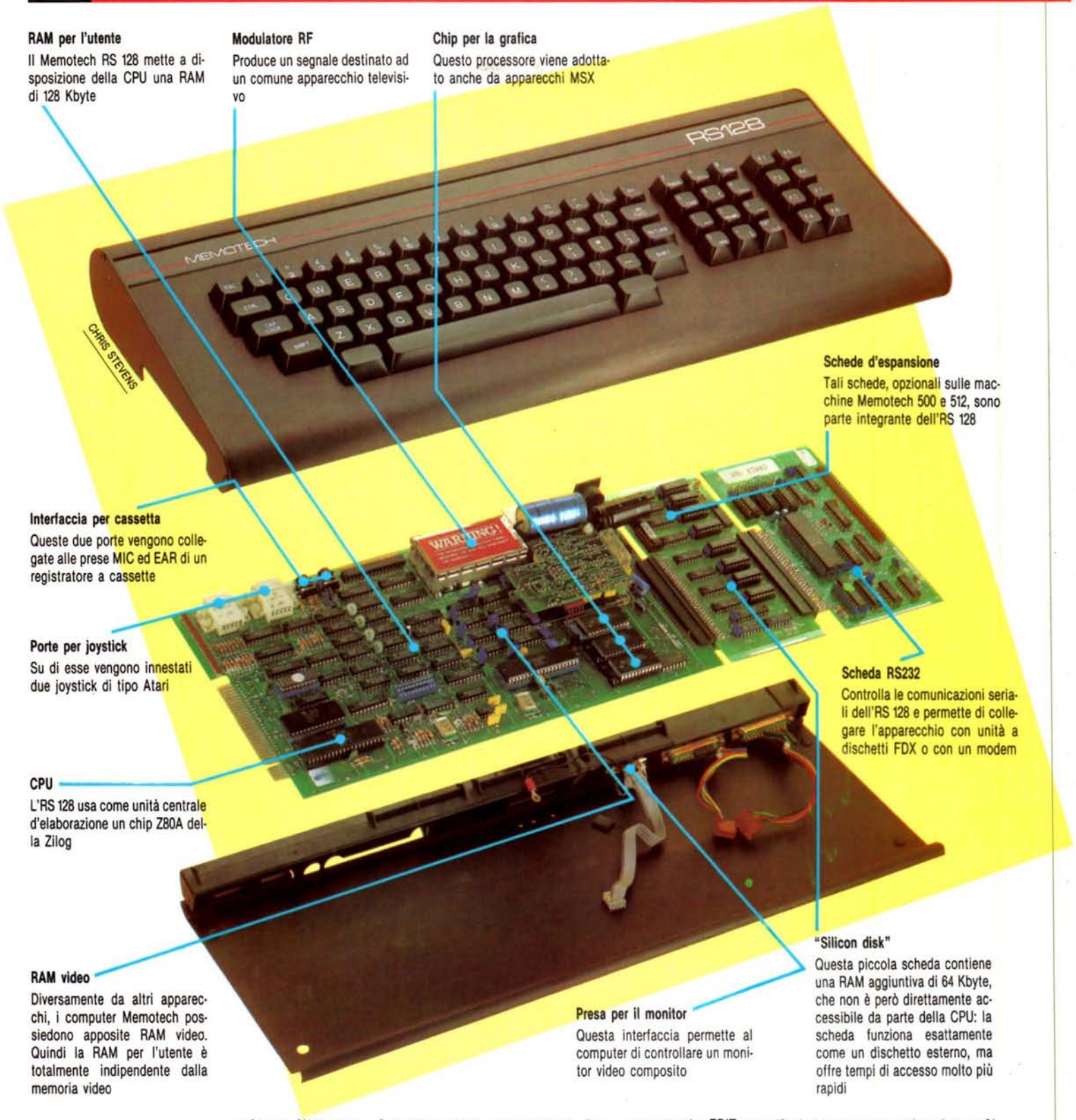
A prima vista l'RS 128 ricorda da vicino, per eleganza e compattezza, gli apparecchi della serie 500. Come i suoi predecessori, ha una custodia in alluminio che lo rende assai più pesante della maggior parte degli home computer. È dotato di una comune tastiera QWERTY e di una tastierina numerica, con tasti programmati per alcuni comandi del linguaggio NODDY. Alla sua destra ci sono 8 tasti di funzione programmabili. L'intera dotazione di tasti è di buona qualità ed offre ottime prestazioni.
La loro disposizione sulla tastiera presenta invece qualche pecca: il tasto Return è insolitamente piccolo e di non facile individuazione, mentre il tasto Delete è stato inserito, incomprensibilmente, nella tastierina numerica. Il tasto Backspace si trova nell'angolo superiore sinistro ma, invece di cancellare il carattere a sinistra come avviene sulla maggior parte dei computer, sul Memotech sposta semplicemente il cursore a sinistra.
Le interfacce si trovano sul retro della macchina: alcune erano già presenti nella serie 500, altre sono nuove. Sulla sinistra ci sono due porte RS232, che permettono di collegare il computer a unità a dischetti FDX o a stampanti parallele, ma possono essere usate anche per collegamenti in rete. Alla loro destra c'è una porta per monitor video composito ed una presa hi-fi per inviare i suoni generati dal computer ad un normale impianto stereo. Seguono quindi la presa d'alimentazione, una porta RF ed una interfaccia per stampante di tipo Centronics. L'interfaccia per cassette è composta da una coppia di prese microjack EAR e MIC, simili a quelle dello Spectrum.
Infine ci sono due porte per joystick a 9 piedini di tipo Atari.
Le porte d'interfaccia, sul retro della macchina, sono contrassegnate da etichette con caratteri bianchi perfettamente leggibili, ma si trovano alloggiate in specie di piccoli affossamenti della custodia che rendono diffi-



CHRIS STEVENS

UN PASSO AVANTI...

Il Memotech RS 128 è un modello perfezionato rispetto alla serie MXT 500. Possiede una coppia di porte RS232 che permettono di collegarlo a unità a dischetti FDX.
Questa caratteristica rende l'apparecchio particolarmente adatto a piccole applicazioni gestionali



RAM per l'utente
Il Memotech RS 128 mette a disposizione della CPU una RAM di 128 Kbyte

Modulatore RF
Produce un segnale destinato ad un comune apparecchio televisivo

Chip per la grafica
Questo processore viene adottato anche da apparecchi MSX

Schede d'espansione
Tali schede, opzionali sulle macchine Memotech 500 e 512, sono parte integrante dell'RS 128

Interfaccia per cassetta
Queste due porte vengono collegate alle prese MIC ed EAR di un registratore a cassette

Porte per joystick
Su di esse vengono innestati due joystick di tipo Atari

Scheda RS232
Controlla le comunicazioni seriali dell'RS 128 e permette di collegare l'apparecchio con unità a dischetti FDX o con un modem

CPU
L'RS 128 usa come unità centrale d'elaborazione un chip Z80A della Zilog

RAM video
Diversamente da altri apparecchi, i computer Memotech possiedono apposite RAM video. Quindi la RAM per l'utente è totalmente indipendente dalla memoria video

Presa per il monitor
Questa interfaccia permette al computer di controllare un monitor video composito

"Silicon disk"
Questa piccola scheda contiene una RAM aggiuntiva di 64 Kbyte, che non è però direttamente accessibile da parte della CPU: la scheda funziona esattamente come un dischetto esterno, ma offre tempi di accesso molto più rapidi

coltoso l'innesto dei rispettivi connettori. Lo schermo BASIC, di 24x40 caratteri, è suddiviso in tre sezioni, chiaramente individuabili al momento dell'accensione. Le prime 19 righe compongono lo schermo principale per la visualizzazione dei listati, seguito dallo schermo di EDIT, sul quale vengono inserite le nuove linee di programma. L'ultima riga è una linea di stato per i messaggi d'errore. Come sugli apparecchi della Sinclair, le linee di programma vengono modificate con il comando EDIT, ma il sistema operativo impedisce l'inserimento di una linea in un programma nel caso che questa contenga un errore di sintassi.

Il dialetto BASIC dell'RS 128, che ricorda da vicino il BASIC MSX, possiede istruzioni come SOUND, PAPER, INK e CIRCLE ma contiene anche comandi per la gestione dello schermo assenti nello standard MSX. Per esempio, il comando CSR x,y posiziona il cursore sul punto indicato dalle coordinate (x,y). CRVS, ancora più

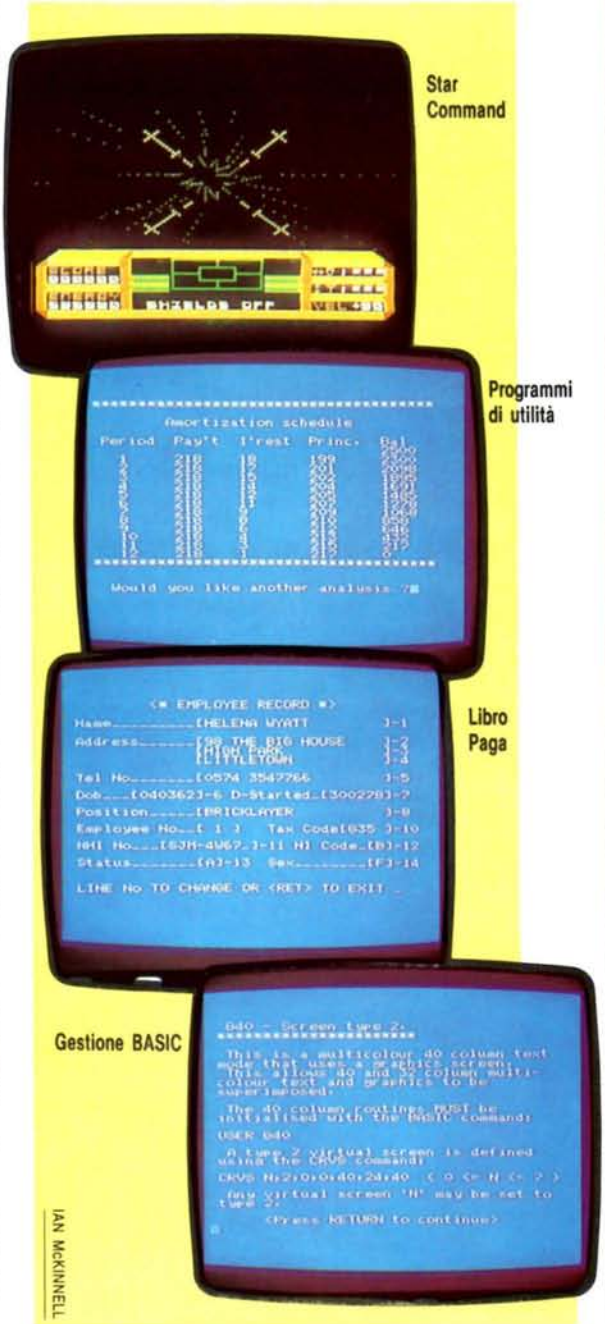
potente, consente all'utente di "aprire" una finestra in qualsiasi settore dello schermo, al cui interno è possibile visualizzare indifferentemente testi o grafica.
Un particolare gruppo di comandi BASIC è dedicato alla gestione degli sprite. GENPAT, ad esempio, permette di definire direttamente uno sprite senza far ricorso alle istruzioni DATA.
La grafica Memotech è gestita dal chip video TMS9929A, un componente tipico degli apparecchi MSX.
Il processore centrale delle macchine Memotech è lo Z80, che consente di utilizzare il sistema operativo CP/M. Lo Z80 viene comunemente adottato da molti piccoli fabbricanti di computer perché è dotato di un vasto corredo di programmi e non implica lo sviluppo di una base software per i nuovi apparecchi. Naturalmente i computer devono possedere uno schermo ad 80 colonne e la Memotech ha risolto il problema in modo originale, introducendo una scheda per ottenere 80 colonne nell'unità a dischetti. Tali unità usano dischetti a doppia faccia e doppia densità, con una velocità di trasmissione di 9200 baud.
L'unità a dischetti è corredata da un pacchetto di software integrato che, oltre al dischetto CP/M di sistema, contiene: il word processor *NewWord*, lo spreadsheet *SuperCalc*, i programmi *Compact* e *TeleVideo* per la lettura di dischetti scritti in altri formati (la Memotech afferma che è possibile leggere anche dischi IBM) ed infine il Contact, che consente di collegare la seconda porta RS232 ad un sistema di comunicazioni in rete.
L'RS 128 possiede una RAM interna di 128 Kbyte. Poiché l'apparecchio usa un processore ad 8 bit, capace di indirizzare solo 64 Kbyte, i rimanenti 64 Kbyte vengono usati sotto forma di "silicon disk", ossia di una serie di chip capaci di immagazzinare file e programmi come un vero e proprio floppy disk, ma dotati di una velocità d'elaborazione 50 volte superiore a quella di un comune dischetto.
Le notevoli dimensioni del manuale di corredo, molto più voluminoso dei comuni manuali per home computer, sono dovute più al tipo d'impaginazione che ad una mole d'informazioni maggiore. Tuttavia, la casa produttrice comunica tutte le caratteristiche tecniche necessarie, compresi gli schemi dei circuiti e delle porte e le chiamate del sistema operativo.
Sviluppando questo nuovo modello della precedente serie 500, la Memotech ha tentato di realizzare un apparecchio con spiccate attitudini gestionali, capace di competere con il QL Sinclair, il Micro BBC ed il Commodore Plus/4.

SOFTWARE DI CORREDO

Queste immagini sono tratte da alcuni dei pacchetti gestionali e di gioco scritti per gli apparecchi Memotech. Se paragonato a quello di altri computer, il software disponibile è piuttosto limitato, ma l'adozione del CP/M consente di accedere ad una gamma di pacchetti software più ampia



Le unità a dischetti "gemelle" FDX
Queste unità a dischetti consentono al computer di funzionare con il sistema operativo CP/M. Ciascuna unità da 5 1/4 pollici può immagazzinare fino a 500 Kbyte d'informazioni



MEMOTECH RS128

DIMENSIONI
mm. 488 x 202 x 56

CPU
Z80A, con frequenza di lavoro a 4 MHz

MEMORIA
64 Kbyte di RAM, 24 Kbyte di ROM

VIDEO
40 caratteri x 24 righe in modo testuale, 32 x 24 in modo misto (testi e grafica) e 256 x 192 pixel con 16 colori in modo grafico. Possibilità di usare fino a 32 sprite controllabili indipendentemente

INTERFACCE
Per cassetta (prese "jack" Mic e Ear); interfaccia di ingresso/uscita; due porte per joystick; due porte seriali RS232; presa audio per impianto hi-fi; uscita video composito e uscita TV; interfaccia parallela

LINGUAGGI FORNITI
BASIC, FORTH e PASCAL

TASTIERA
Tipo QWERTY, con 57 tasti; F e J sono sagomati per facilitarne l'individuazione "a tatto". 12 tasti di funzione sulla tastierina numerica e otto tasti funzione programmabili

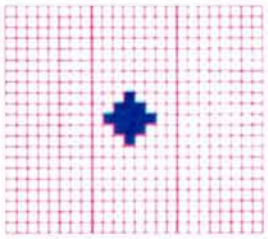
DOCUMENTAZIONE
Il manuale è molto simile a quello fornito assieme alla serie MTX500 e contiene moltissime informazioni, sull'apparecchio anche di carattere altamente tecnico

PRO
L'aggiunta delle porte seriali RS232, rispetto ai modelli precedenti, è senza dubbio un notevole passo avanti

CONTRO
Nonostante la possibilità di impiegare il sistema operativo CP/M offra notevoli potenzialità, c'è una certa carenza di software scritto appositamente per l'apparecchio

Più schermi sul CBM 64

Dopo il BBC e lo Spectrum, è la volta degli schermi grafici sul CBM 64



```
0   16   0
0   56   0
0  124   0
0   56   0
0   16   0
```

PROIETTILE — SPRITE 1

Fuoco!

Sul Commodore 64, la procedura di sparo della porta utente usa uno sprite che rappresenta un proiettile. L'immagine del proiettile viene creata depositando (con POKE) una serie di zeri nell'area di definizione, quindi estraendo (con READ) dalle istruzioni DATA i numeri che definiscono il proiettile e depositandoli (con POKE) nelle giuste locazioni dello sprite

Sprite espanso

I valori per lo sprite "porta joystick" vengono estratti dalle frasi DATA ed inseriti nelle corrette locazioni dello sprite. Poiché l'immagine della porta risulterebbe troppo compressa, lo sprite viene espanso orizzontalmente azzerando il registro di espansione orizzontale

PORTA JOYSTICK — SPRITE 0

La progettazione degli ambienti del gioco *Digitaya*, per il Micro BBC e lo Spectrum, è simile: entrambi gli apparecchi possiedono grafica in alta risoluzione e istruzioni PRINT facili da usare. Le differenze riguardano essenzialmente le dimensioni dello schermo e le parole del dialetto BASIC che controllano la grafica in alta risoluzione. Al contrario, la creazione degli stessi ambienti sul Commodore 64 richiede un'impostazione radicalmente diversa, poiché l'apparecchio manca di comandi BASIC per la grafica in alta risoluzione e tale funzione è concretamente utilizzabile solo in codice macchina. Infatti, l'esecuzione in BASIC dei comandi PEEK e POKE per produrre immagini in alta risoluzione si dimostra troppo lenta per applicazioni di questo genere. Ci si deve quindi limitare alle funzioni grafiche di bassa risoluzione disponibili sul Commodore 64. Tuttavia, i caratteri grafici possono essere combinati per creare lettere grandi o altre immagini e con gli sprite si possono introdurre figure in alta risoluzione sullo schermo Commodore a risoluzione normale. Infine i caratteri PET vengono posizionati sul video con istruzioni PRINT depositando (POKE) il relativo codice di carattere. Vediamo queste due tecniche.

Immagine della porta joystick

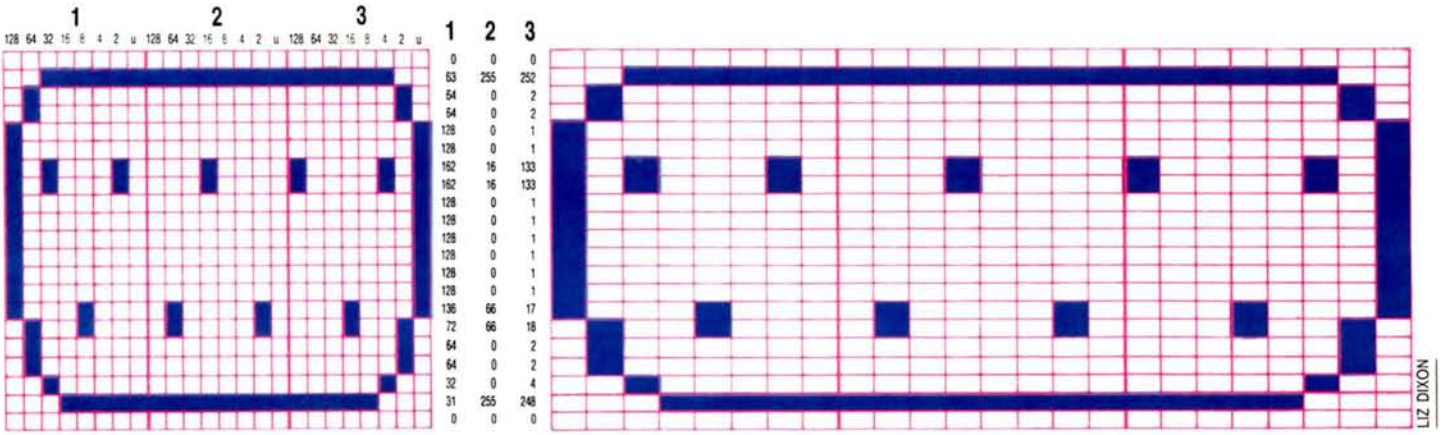
Le linee 8020-8170 del listato *Porta Joystick* riguardano l'estrazione dei dati per i due sprite usati nella routine. Il primo, lo sprite 0, viene definito dai primi 63 numeri in un gruppo d'istruzioni DATA comprese tra le linee 8450 e 8497 e riproduce la porta del joystick (diagramma a fondo pagina). I dati per gli sprite vengono immagazzinati nella parte alta dell'area di programma BASIC ma, se i programmi BASIC sono di grosse dimensioni, corrono il rischio di essere cancellati da sovrascritture. Un'ubicazione alternativa è l'area del buffer per la cassetta (tra le locazioni 832 e 1022) che può contenere solo tre sprite.

Per ottenere un'immagine più realistica della porta, la larghezza dello sprite 0 viene raddoppiata rispetto alle sue dimensioni originali, azzerando il registro di espansione orizzontale, alla linea 8170. Si noti che tutti i registri che controllano gli attributi degli sprite (il colore, la posizione e l'espansione), fanno riferimento all'indirizzo iniziale del chip di controllo dello schermo (VIC). A questo proposito, è più facile ricordare l'indirizzo VIC+29 del registro d'espansione orizzontale che non la sua reale locazione di memoria (53277). Alcuni attributi degli sprite usano un intero registro (ad esempio, i registri per le coordinate x e y), ma quando ciascun bit del registro controlla in modo autonomo una funzione degli otto sprite disponibili, gli attributi possono essere controllati accendendo o spegnendo il relativo bit in un solo registro.

Lo sprite 1 viene definito dai restanti 13 numeri nelle istruzioni DATA e rappresenta il proiettile sparato dalla porta del joystick.

Poiché la parte solida dello sprite 1 è piccola, conviene introdurre i 63 byte di dati che lo definiscono in due fasi successive, inserendo prima 63 zeri nell'area di definizione con il comando POKE, poi i pochi numeri che definiscono il proiettile con le istruzioni READ e POKE. In questo modo si evita l'introduzione di un gran numero di zeri nelle frasi DATA.

Le linee da 8190 a 8220 creano le stringhe usando una serie di caratteri grafici PET (riportati sulla faccia anteriore dei tasti). LE$ forma una linea orizzontale sullo schermo composta da 40 caratteri speciali PET: il carattere usato è quello destro del tasto C. DW$ forma una serie di caratteri grafici col tasto che sposta in basso il cursore. LS$ ed RS$ usano invece i caratteri diagonali sinistro e destro dei tasti M ed N e creano una configura-



zione a lisca di pesce, in primo piano, che dà profondità di campo all'immagine.
La subroutine Sparo, alla linea 8310, sceglie un punto casuale sul margine inferiore dello schermo e muove lo sprite 1 in tale direzione, ripetendo il procedimento tutte le volte che il giocatore batte un tasto. Una seconda subroutine ripristina i colori originari, cancella lo schermo e spegne gli sprite prima che il controllo venga restituito al programma principale. Per usare la subroutine Sparo in *Digitaya* si deve inserire la linea:

```
3845 GOSUB 8000: REM IMMAGINE PORTA DEL JOYSTICK
```

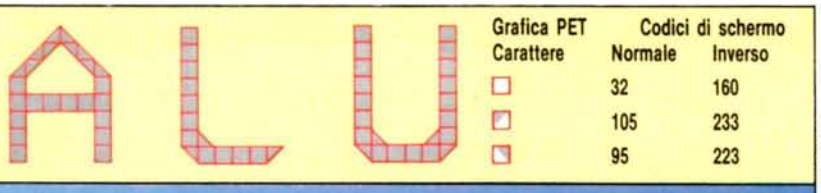
Il secondo listato crea lo scenario per l'ambiente ALU in *Digitaya* usando una diversa tecnica di visualizzazione. Le linee da 7040 a 7090 estraggono i valori da una serie di istruzioni DATA e li inseriscono direttamente nell'area di schermo. Quelle successive inseriscono i codici di colore per i caratteri nelle corrispondenti locazioni dell'area per il colore. Nella figura a destra il codice di colore è 2, quindi i caratteri vengono stampati in rosso.

Per ottenere lo scorrimento verso il basso delle lettere ALU è stata usata una tecnica inconsueta. Con un'istruzione POKE il programma inserisce la prima linea dei codici grafici di carattere che compongono le lettere ALU nella seconda linea. Poi chiama la subroutine alla linea 7680, che muove l'immagine in basso di una linea. Quindi sposta la seconda linea di codici con una POKE nella stessa area di schermo della prima e chiama ancora la subroutine. Ripetendo questa procedura per le otto linee di codice si ottiene lo scorrimento delle lettere ALU a partire dal margine superiore dello schermo.
Infine esistono altri due metodi per visualizzare caratteri: direttamente, come avviene alle linee 7130 e 7140, o estraendoli come stringhe di dati da stampare, come nella subroutine per il punto interrogativo (linee 7170 e 7590-7670). La seconda tecnica facilita la progettazione delle immagini con le istruzioni DATA. Per questa routine aggiungere:

```
4565 GOSUB 7000: REM IMMAGINE ALU
```

Visualizzazione delle lettere

Per creare l'ambiente ALU del gioco "Digitaya" vengono usati tre caratteri grafici PET in bassa risoluzione. Le grandi lettere ottenute attraversano lo schermo a partire dal suo margine superiore, mantenendo le rispettive posizioni



LIZ DIXON

IMMAGINE ALU

```
7000 REM **** S/R IMMAGINE ALU ****
7010 VIC=53248:CS=55296:SC=1024
7020 PRINT CHR$(147): REM CANCELLA LO SCHERMO
7030 POKE VIC+32,0:POKE VIC+33,0: REM SELEZIONA SCHERMO/BORDO
7040 CC=0
7050 FOR J=1 TO 8:GOSUB7680: REM SCORRIMENTO
7060 FOR I=47 TO 72
7070 READ A:CC=CC+A:POKE SC+I,A:POKE CS+I,2
7080 NEXT I,J
7090 READ CS:IF CS<>CC THEN PRINT"ERRORE DI CHECKSUM":STOP
7100 GOSUB 7680: REM SCORRIMENTO
7110 PRINTCHR$(158): REM TESTO GIALLO
7120 FOR I=1 TO 8:PRINT:NEXT I: REM MUOVE IN BASSO
7130 PRINTTAB(9)"AND";SPC(7);"OR";SPC(8);"NOT"
7140 PRINTTAB(10)"o";SPC(9);"o";SPC(9);"o"
7150 PRINTCHR$(28): REM TESTO ROSSO
7160 REM ** PUNTO INTERROGATIVO **
7170 FOR I=1 TO 9:READ Q$:PRINTTAB(16)Q$:NEXT I
7180 REM **** ATTENDE LA PRESSIONE DI UN TASTO E RIPRISTINA ****
7190 GET A$:IF A$="" THEN 7190
7200 POKE VIC+32,14:POKE VIC+33,6: REM SCHERMO/BORDO
7210 PRINT CHR$(154): REM TESTO BLU
7220 PRINT CHR$(147): REM CANCELLA LO SCHERMO
7230 RETURN
7240 REM **** DATI DI SCHERMO PER ALU ****
7250 REM ** RIGA 1 **
7260 DATA 160,32,32,32,32,160,32,32,32,32,
7270 DATA 95,160,160,160,160,105
7280 DATA 32,32,32,32,95,160,160,160,160,10,105
7290 REM ** RIGA 2 **
7300 DATA 160,32,32,32,32,160,32,32,32,32
7310 DATA 160,223,32,32,32,32,32,32,32,32
7320 DATA 160,223,32,32,233,160
7330 REM ** RIGA 3 **
7340 DATA 160,32,32,32,32,160,32,32,32,32
7350 DATA 160,32,32,32,32,32,32,32,32,32
7360 DATA 160,32,32,32,32,160
7370 REM ** RIGA 4 **
7380 DATA 160,160,160,160,160,160,32,32,32,32
7390 DATA 160,32,32,32,32,32,32,32,32,32
7400 DATA 160,32,32,32,32,160
7410 REM ** RIGA 5 **
7420 DATA 160,32,32,32,32,160,32,32,32,32
7430 DATA 160,32,32,32,32,32,32,32,32,32
7440 DATA 160,32,32,32,32,160
7450 REM ** RIGA 6 **
7460 DATA 233,105,95,223,32,32,32,32
7470 DATA 160,32,32,32,32,32,32,32,32,32
7480 DATA 160,32,32,32,32,160
7490 REM ** RIGA 7 **
7500 DATA 32,233,105,95,233,32,32,32,32,32
7510 DATA 160,32,32,32,32,32,32,32,32,32
7520 DATA 160,32,32,32,32,160
7530 REM ** RIGA 8 **
7540 DATA 32,32,233,233,32,32,32,32,32,32
7550 DATA 160,32,32,32,32,32,32,32,32,32
7560 DATA 160,32,32,32,32,160
7570 DATA 14463:REM CHECKSUM
7580 REM ** DATI PER PUNTO INTERROGATIVO **
7590 DATA " ????? "
7600 DATA "?     ?"
7610 DATA "?     ?"
7620 DATA "    ?? "
7630 DATA "   ?   "
7640 DATA "   ?   "
7650 DATA "   ?   "
7660 DATA "       "
7670 DATA "   *   "
7680 REM **** S/R SCORRIMENTO IMMAGINE ****
7690 POKE 218,160
7700 PRINT CHR$(19);CHR$(17);CHR$(157);CHR$(148)
7710 RETURN
```

IMMAGINE PORTA JOYSTICK

```
8000 REM **** S/R IMMAGINE PORTA DEL JOYSTICK ****
8010 PRINTCHR$(147): REM CANCELLA LO SCHERMO
8020 S0=832:S1=S0+64: REM INDIRIZZO INIZIALE DATI SPRITE
8030 CC=0:VIC=53248: REM INDIRIZZO INIZIALE CHIP VIC
8040 FOR I=S0 TO S0+62:READ A:CC=CC+A:POKE I,A:NEXT
8050 FOR I=S1 TO S1+62:POKE I,0:NEXT I
8060 FOR I=S1+25 TO S1+37:READ A:CC=CC+A:POKE I,A:NEXT I
8065 READ CS:IF CS<>CC THEN PRINT"ERRORE DI CHECKSUM":STOP
8070 POKEVIC+33,0:POKEVIC+32,0: REM SELEZIONA SCHERMO/BORDO
8080 REM ** SELEZIONA PUNTATORI ALLO SPRITE **
8090 POKE 2040,S0/64:POKE 2041,S1/64
8100 REM ** SELEZIONA REG. VIC PER CONTROLLO SPRITE
8110 POKE VIC+39,7: REM SELEZIONA COLORE SPRITE 0
8120 POKE VIC+40,7: REM SELEZIONA COLORE SPRITE 1
8130 POKE VIC,65: REM SELEZIONA COORD. X SPRITE 0
8140 POKE VIC+1,70: REM SELEZIONA COORD. Y SPRITE 0
8150 POKE VIC+2,74: REM SELEZIONA COORD. X SPRITE 1
8160 POKE VIC+3,70: REM SELEZIONA COORD. Y SPRITE 1
8170 POKE VIC+29,1: REM ESPANDE ORIZZ. LO SPRITE 0
8190 LE$="":FOR I=1 TO 40:LE$=LE$+CHR$(195):NEXT I
8200 DW$="":FOR I=1 TO 25:DW$=DW$+CHR$(17):NEXT I
8210 LS$="":FOR I=1 TO 11:LS$=LS$+CHR$(206)+" ":NEXT
8220 RS$="":FOR I=1 TO 11:RS$=RS$+CHR$(205)+" ":NEXT
8230 PRINT CHR$(158): REM TESTO GIALLO
8240 PRINT CHR$(19): PRINT TAB (2) "PORTA DEL JOYSTICK"
8250 PRINT CHR$(154): REM TESTO BLU
8260 PRINT CHR$(19);LEFT$(DW$,17);LE$
8270 PRINT CHR$(145);
8280 PT$=LEFT$(LS$,19)+LEFT$(RS$,21)
8290 PT$=PT$+RIGHT$(LS$,19)+RIGHT$(RS$,21)
8300 FOR I=1 TO 3:PRINT PT$;:NEXT I
8305 POKE VIC+21,3: REM ACCENDE GLI SPRITE 0 & 1
8310 REM ** SPARA UN PROIETTILE **
8320 YB=240:Y1=70
8330 X1=74:X=INT(RND(1)*150)+24
8340 G=2*(X-X1)/(YB-Y1)
8350 FOR Y=Y1 TO YB STEP 2
8360 X1=X1+G:POKE VIC+2,X1:POKE VIC+3,Y
8370 NEXT Y
8380 GET A$:IF A$="" THEN 8330
8390 REM ** CANCELLA LO SCHERMO **
8400 POKE VIC+21,0: REM SPEGNE GLI SPRITE
8410 POKE VIC+32,13:POKE VIC+33,6: REM RIPRISTINA SCHERMO
     E BORDO ORIGINALI
8420 PRINT CHR$(147): REM CANCELLA LO SCHERMO
8430 RETURN
8440 REM **** DATA PER GLI SPRITE ****
8450 DATA 0,0,0,63,255,252,64,0,2,64,0,2
8460 DATA 128,0,1,128,0,1,162,16,133,162,16,133
8470 DATA 128,0,1,128,0,1,128,0,1,128,0,1
8480 DATA 128,0,1,128,0,1,136,66,17,72,66,18
8490 DATA 64,0,2,64,0,2,32,0,4,31,255,248
8495 DATA 0,0,0,16,0,0,56,0,0,124,0,0,56,0,0,16
8497 DATA 3701: REM CHECKSUM
```

Fattoriali!

Il LOGO è adatto anche ad esperimenti matematici: proviamo con i fattoriali

In quanti modi possono disporsi quattro persone intorno ad un tavolo con quattro sedie? La prima persona può sedersi su una sedia qualunque, mentre alla seconda restano solo tre scelte, alla terza due ed alla quarta una sola. Quindi il numero delle diverse disposizioni possibili è dato dalla moltiplicazione 4 x 3 x 2 x 1. Questa operazione viene di solito scritta come "4!" e letta come "4 fattoriale". I fattoriali sono abbastanza frequenti nei problemi matematici riguardanti gli ordinamenti, le combinazioni e le probabilità.
Scrivere una procedura ricursiva per calcolare i fattoriali è abbastanza semplice. Innanzi tutto occorre ricordare che il fattoriale 0 viene indicato come 1 mentre quello di qualsiasi numero positivo diverso da 0 (chiamiamolo x) viene definito come fattoriale di x—1 moltiplicato per x. Se traduciamo questa definizione in un programma LOGO otteniamo:

```
TO FATTORIALE: X
   IF :X=0 THEN OUTPUT 1
   OUTPUT (FATTORIALE :X — 1) * :X
END
```

L'istruzione PRINT FATTORIALE 6 permette di eseguire il calcolo che produce come risultato 720.
Questa procedura lavora senza problemi fino a 12!, dopodiché il risultato diviene troppo grande per essere conservato per intero dal computer. Sul Commodore 64, ad esempio, PRINT FATTORIALE 13 dà come risultato 6.22702E9, ossia 622702 x 10^6. Tuttavia il risultato non è completo perché manca delle ultime quattro cifre. Vediamo adesso come ottenere anche le cifre non rappresentate. La prima operazione da compiere è ampliare le capacità aritmetiche del LOGO in modo da poter calcolare con precisione numeri con più di 7 cifre.
Per semplificare il problema, consideriamo solo i numeri interi positivi ed inseriamoli in una lista. Ad esempio, il numero 1234567 viene rappresentato come [1 2 3 4 5 6 7]. Le due procedure che seguono sommano tali numeri. Digitando PRINT SOMMA [1 2 3] [5 6 9] dovremmo ottenere come risultato [692]:

```
TO SOMMA :X :Y
   OUTPUT SOMMA1 :X :Y 0
END

TO SOMMA1 :X :Y :RIPORTO
   IF (ALLOF (EMPTY? :X) (EMPTY? :Y) (:RIPORTO=0)) THEN
      OUTPUT []
   TEST EMPTY? :Y
   IF TRUE IF :RIPORTO=0 THEN OUTPUT :X ELSE OUTPUT
      SOMMA1 :X [1] 0
   TEST EMPTY? :X
   IF TRUE IF :RIPORTO=0 THEN OUTPUT :Y ELSE OUTPUT
      SOMMA1 [1] :Y 0
   MAKE "ADDIZ (LAST :X) + (LAST :Y) + :RIPORTO
   OUTPUT LPUT REMAINDER :ADDIZ 10 SOMMA1
   BUTLAST :X BUTLAST :Y QUOTIENT :ADDIZ 10
END
```

La procedura si sviluppa come una normale addizione eseguita a mano, sommando le cifre a partire da sinistra e aggiungendo il riporto alla colonna successiva.
Ciò vale anche per la sottrazione ma, in questo caso, è necessario includere una routine che cancelli gli zeri non significativi dalla differenza (ad esempio, [00078]).

```
TO SOTTR :X :Y
   OUTPUT CANCELLAZERO SOTTR1 :X :Y 0
END

TO SOTTR1 :X :Y :RESTO
   IF (ALLOF (EMPTY? :X) (EMPTY? :Y) (:RESTO=0)) THEN
      OUTPUT [0]
   TEST EMPTY? :Y
   IF TRUE IF :RESTO=0 THEN OUTPUT :X ELSE OUTPUT
      SOTTR1 :X [1] 0
   IF EMPTY? :X THEN PRINT [SPIACENTE, NON POSSO
      LAVORARE SU UN RISULTATO NEGATIVO] TOPLEVEL
   MAKE "DIFF (LAST :X) — (LAST :Y) — :RESTO
   IF :DIFF<0 THEN OUTPUT LPUT (10+ :DIFF)
   SOTTR1 BUTLAST :X BUTLAST :Y 1
   OUTPUT LPUT :DIFF SOTTR1 BUTLAST :X
   BUTLAST :Y 0
END

TO CANCELLAZERO :X
   IF EMPTY? :X THEN OUTPUT [0]
   IF NOT ((FIRST :X)=0) THEN OUTPUT :X
   OUTPUT CANCELLAZERO BUTFIRST :X
END
```

La procedura della moltiplicazione è leggermente più complessa e segue il procedimento abitualmente insegnato nelle scuole. Se, ad esempio, vogliamo moltiplicare 123 x 338, dobbiamo moltiplicare 123 x 8, poi 123 x 330 ed infine sommare i due prodotti ottenuti. La seconda operazione può essere a sua volta suddivisa in due fasi, moltiplicando 123 x 33 ed aggiungendo uno 0 al prodotto ottenuto. Per moltiplicare un numero per 33 occorre ovviamente usare una procedura ricursiva. L'intera operazione viene eseguita dalla procedura MOLTIP:



IAN McKINNELL

FATTORE DI ESPANSIONE

La tabella evidenzia con quanta rapidità crescano i fattoriali. Poiché i loro valori tendono a divenire molto alti in breve tempo, molti computer e calcolatrici rappresentano i fattoriali dei numeri maggiori di 12 in notazione esponenziale. Quindi il fattoriale di 12 viene espresso come 479E8 o 4.79x108. Ovviamente la precisione dipende dal numero di cifre significative rappresentate

```
TO MOLTIP :X :Y
   IF EMPTY? BUTLAST :Y THEN OUTPUT MOLTIP1 :X LAST :Y 0
   OUTPUT SOMMA (MOLTIP1 :X (LAST :Y)0)
   (LPUT "0 MOLTIP :X BUTLAST :Y)
END
```

I singoli passaggi della moltiplicazione vengono eseguiti dalla procedura MOLTIP1:

```
TO MOLTIP1 :X :NUM :RIPORTO
   TEST EMPTY? :X
   IF TRUE IF :RIPORTO = 0 THEN OUTPUT [] ELSE OUTPUT
      (LIST :RIPORTO)
   MAKE "PROD (LAST :X) * :NUM + :RIPORTO
   OUTPUT LPUT REMAINDER :PROD 10
   MOLTIP1 BUTLAST :X :NUM QUOTIENT :PROD 10
END
```

Il calcolo dei fattoriali non implica l'impiego di divisioni.
Possediamo adesso una serie di primitive per l'esecuzione di calcoli aritmetici con un alto grado di precisione. Tuttavia, la dimensione dei numeri che possiamo trattare dipende dalla quantità di memoria a disposizione del programma.

Modifiche al programma

Modifichiamo adesso il precedente programma per il calcolo dei fattoriali introducendo una nuova procedura di moltiplicazione.

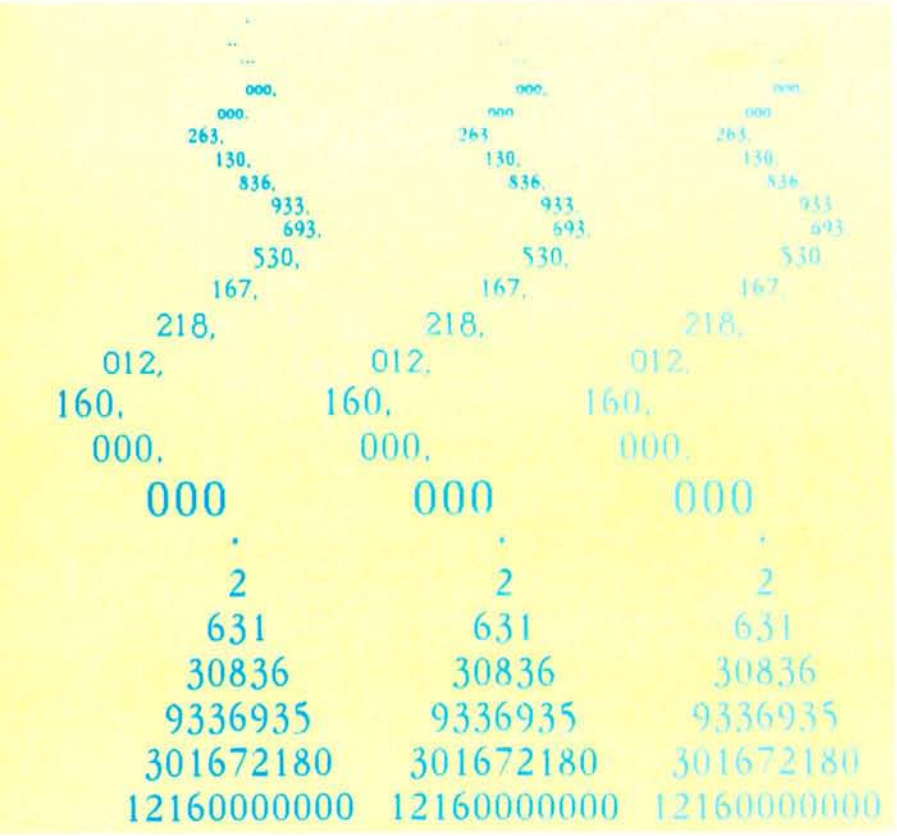
```
TO FATT :X
   IF FIRST :X = 0 THEN OUTPUT [1]
   OUTPUT MOLTIP (FATT SOTTR :X [1]) :X
END
```

Se proviamo a calcolare FATT [13], otteniamo come risultato [6 2 2 7 0 2 0 8 0 0]. Tuttavia sorgono alcuni problemi: sul Commodore 64, il fattoriale più grande calcolabile senza esaurire la memoria è 34!, che dà come risultato un numero a 39 cifre ed impiega un tempo di calcolo abbastanza lungo.
Tuttavia la rappresentazione di grossi numeri sotto forma di liste è piuttosto innaturale: conviene quindi apportare un'ulteriore modifica al programma che consenta di trasformare le liste in numeri espressi nel formato abituale e viceversa.
Le procedure per questa trasformazione sono ESPANDI e COMPATTA.
ESPANDI 123 produce come risultato [1 2 3] mentre COMPATTA [1 2 3] dà 123:

```
TO ESPANDI :X
   IF EMPTY? :X THEN OUTPUT []
   OUTPUT (SENTENCE FIRST :X ESPANDI BUTFIRST :X)
END

TO COMPATTA :X
   IF EMPTY? :X THEN OUTPUT""
   OUTPUT (WORD FIRST :X COMPATTA BUTFIRST :X)
END
```

Tali procedure si basano sul principio che, in LOGO, i numeri vengono trattati come se



Alberi ... fattoriali

Gli alberi fattoriali vengono generati usando la cifra all'estremità sinistra di un numero fattoriale come apice dell'albero. Le cifre successive vengono estratte dall'intero valore, a partire da sinistra, in gruppi di dimensioni crescenti. I gruppi vengono impilati in posizione simmetrica in modo da ottenere il profilo di un albero. Per creare la figura in alto è stato usato il fattoriale di 32. Il valore può essere letto più facilmente quando i numeri vengono disposti a gruppi di tre

fossero parole. Possiamo adesso definire una nuova procedura, chiamata F:

```
TO F :X
   PRINT COMPATTA FATT ESPANDI :X
END
```

che calcola il fattoriale di 13 in risposta al dato in ingresso F13.
Il risultato ottenuto è 6227020800, ossia un numero di difficile lettura. Conviene quindi aggiungere al programma una serie di procedure che suddividano il numero in gruppi di tre cifre e inseriscano un punto decimale.

```
TO PUNTODEC :X
   IF ((COUNT :X) <4) THEN OUTPUT :X
   OUTPUT (WORD PUNTODEC DOPOTRE :X ". ULTIMETRE :X)
END

TO DOPOTRE :X
   OUTPUT BUTLAST BUTLAST BUTLAST :X
END

TO ULTIMETRE :X
   OUTPUT (WORD (LAST BUTLAST BUTLAST :X)
      (LAST BUTLAST :X) (LAST :X))
END
```

Per utilizzare queste procedure è necessario modificare F:

```
TO F :X
   PRINT PUNTODEC COMPATTA FATT ESPANDI :X
END
```

Calcolando con la procedura F i primi 20 fattoriali (vedere tabella alla pagina precedente) possiamo renderci meglio conto della rapidità con cui essi crescono.
Il calcolo dei fattoriali può essere usato anche in modo divertente. Un matematico americano ha avuto l'originale idea di stampare i fattoriali ottenuti da un gruppo di numeri in forma di alberi sui cartoncini di Natale, che ha poi spedito a suoi amici. Poiché solo alcuni fattoriali contengono il numero di cifre necessario per ottenere le sagome degli alberi, egli ha usato le seguenti procedure per ottenere il risultato desiderato:

```
TO ALBERO :L
   ALBERO1 1 :L
END

TO ALBERO1 :NUM :L
   IF EMPTY? :L THEN STOP
   REPEAT ROUND (20 – :NUM/2) [PRINT1 SPAZIO]
   LINEASTAMPA :NUM :L
   ALBERO1 :NUM + 2 RIDUCI :NUM :L
END

TO SPAZIO
   OUTPUT CHAR 32
END

TO LINEASTAMPA :NUM :L
   IF :NUM = 0 THEN PRINT "STOP
   PRINT1 FIRST :L
   LINEASTAMPA :NUM – 1 BUTFIRST :L
END

TO RIDUCI :NUM :L
   IF :NUM = 0 THEN OUTPUT :L
   OUTPUT RIDUCI :NUM –1 BUTFIRST :L
END
```

Anche in questo caso occorre modificare la procedura di controllo F:

```
TO F :X
   ALBERO COMPATTA FATT ESPANDI :X
END
```

Per stampare l'immagine a sinistra è stato impiegato 32!. Un'analisi più attenta di questi alberi fattoriali mostra che esistono solo tre numeri minori di 32 i cui fattoriali permettono di ottenere degli alberi. Il primo fattoriale utile maggiore di 32 è invece 59!

Logo & Logo

In tutte le versioni LCSI, usare:

EMPTYP al posto di **EMPTY?**
AND al posto di **ALLOF**
TYPE al posto di **PRINT1**

IF possiede una diversa sintassi. Ad esempio:

IF:RIPORTO = 0 [OUTPUT []] [OUTPUT (LIST :RIPORTO)]

Sostituire **QUOTIENT :X :Y** con **DIV :X :Y** sullo Spectrum e con **ROUND (:X/:Y)** sull'Atari.

Sull'Atari usare **SE** al posto di **SENTENCE**.

Le chiamate OSBYTE del BBC

Un'altra visita al sistema operativo del Micro BBC

Descrivendo le diverse tecniche di accesso al sistema operativo del BBC, si è accennato brevemente a un gruppo di chiamate OS, dette chiamate OSBYTE, che permettono di modificare il funzionamento di alcune parti del sistema operativo. Per esempio, la chiamata OSBYTE *FX 4,1 consente di modificare la risposta del computer alla pressione dei tasti per il cursore.
La versione 1.2 del SO possiede più di 100 chiamate di questo tipo, che consentono di raggiungere e modificare molte funzioni del Micro BBC. Si è visto che l'uso di chiamate OS indirette protegge il sistema da accidentali modifiche della configurazione hardware e software del computer. Le chiamate OSBYTE si dimostrano quindi il principale strumento per l'impiego delle routine del SO.
Prima di passare ad una loro descrizione dettagliata, ricordiamo che la vecchia versione 0.1 del SO non contempla alcune delle chiamate OSBYTE qui ricordate e riportate dal manuale per l'utente del BBC. Per conoscere la versione installata sul computer che stiamo usando, si usi il comando *HELP <RETURN>.
Vediamo adesso le modalità d'impiego delle chiamate OSBYTE in BASIC e in linguaggio macchina. Come molte altre chiamate OS del BBC, le chiamate OSBYTE usano un vettore (vedere a pag. 1358) con indirizzo &20A e &20B. La seguente tabella illustra i diversi formati di una chiamata OSBYTE. Queste istruzioni eseguono tutte la chiamata equivalente a *FX 4,1:

Con *FX in BASIC	Con USR in BASIC	In Codice Macchina
*FX 4,1	A%=4:X%=1:D%=USR(&FFF4)	LDA #4 LDX #1 JSR &FFF4

I tre esempi mostrano che l'indirizzo di chiamata per le routine OSBYTE è &FFF4 e che i parametri vengono passati alla chiamata OSBYTE tramite i registri A, X ed Y del processore 6502. Usando il BASIC per assegnare un valore a A%, X% ed Y% e per chiamare una routine in codice macchina con i comandi USR o CALL, possiamo lanciare il programma in codice macchina con i registri A, X ed Y del processore che contengono i valori presenti rispettivamente nelle variabili A%, X% e Y%.

L'impiego di USR nel secondo esempio della tabella consente di recuperare un risultato da un programma in codice macchina e di inserirlo in una variabile BASIC. Con questa tecnica, ottenuta tramite alcune chiamate OSBYTE, è possibile ricavare un'informazione per passarla al BASIC.
Il terzo esempio è un breve programma di chiamata OSBYTE in codice macchina: i registri vengono caricati con gli appropriati valori e poi viene chiamata OSBYTE usando il suo indirizzo (&FFF4). Gli esempi descritti mostrano anche che i parametri 4 ed 1, comunicati al SO con una chiamata FX, corrispondono ai parametri forniti nei registri A, X e Y quando una routine OSBYTE viene chiamata in codice macchina o con un comando USR.
Indipendentemente dal sistema di chiamata adottato, il contenuto del registroAindica sempre quale routine OSBYTE si vuole usare, mentre i registri X ed Y vengono impiegati per fornire parametri alla routine. Alcune chiamate OSBYTE non prevedono parametri, mentre altre usano un solo parametro che viene comunicato attraverso il registro X. Altre ancora, meno frequenti, prevedono due parametri attraverso i registri X ed Y. Gli esempi seguenti sono relativi a questi tre tipi di chiamate:

Nessun parametro	Un parametro	Due parametri
*FX0	*FX 4,1	*FX151,96,200

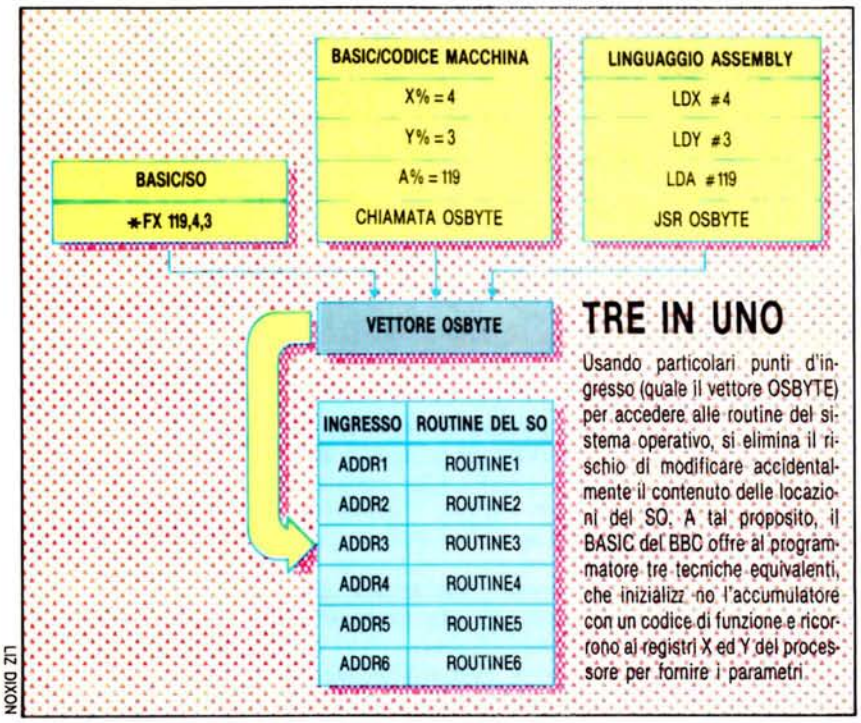
Ci sono due punti da chiarire riguardo alla chiamata OSBYTE con *FX. Il primo è che OSBYTE, come tutti i comandi *, non tratta le variabili BASIC. L'esecuzione della seguente istruzione porta infatti alla comparsa del messaggio d'errore Bad Command:

```
a=4: b=1
*FX a,b
```

Questo ostacolo può essere aggirato comunicando il comando *FX al sistema operativo con OSCLI:

```
10 DIM C 100
20 a=4
30 b=1
40 $C="*FX"+STR$a+","+STR$B
50 X%=C MOD 256
60 Y%=C DIV 256

70 CALL &FFF7
```



LIZ DIXON

Il secondo punto è che il SO non accetta alcuna istruzione posta sulla stessa linea della chiamata FX.
Quindi:

```
*FX 4,1: PRINT "Stampa"
```

produce un altro Bad Command, perché il SO interpreta i due punti e l'istruzione PRINT come parti del comando FX.
Entrambi i problemi dipendono dal fatto che le chiamate *FX non vengono passate all'interprete BASIC, ma direttamente all'interprete della linea di comando (CLI), che non è in grado di elaborare le variabili BASIC né di gestire linee con più istruzioni.
Come si è già detto, è possibile fornire parametri a OSBYTE tramite i registri X e Y oppure recuperare i valori contenuti in alcune variabili di sistema, usate dal SO, per mezzo della chiamata USR o di una routine in codice macchina.
Questo secondo metodo si dimostra più facile da applicare perché i valori recuperati con la chiamata USR devono essere decodificati per poter interpretare l'informazione. I parametri vengono ritrasmessi al BASIC tramite i registri X ed Y e alcune delle chiamate OSBYTE usano il flag di carry per segnalare gli eventuali errori.
Tuttavia la possibilità di recuperare i risultati dipende dal tipo di chiamata eseguita, ossia dipende dal valore passato a OSBYTE nel registro A.
Infatti non tutte le chiamate consentono di ottenere utili informazioni sul sistema operativo ritrasmettendo i risultati al BASIC.
Una chiamata OSBYTE può comunicare due tipi di informazione. Il primo riguarda quelle "lette" da alcuni componenti del sistema, come la porta utente, il generatore vocale e le variabili di sistema. Le chiamate OSBYTE di questo tipo sono dette di sola lettura: ne è un esempio la chiamata OSBYTE con A=129, usata in BASIC per implementare la funzione INKEY().
I registri X ed Y devono essere predisposti per comunicare al sistema operativo il tempo di ritardo necessario.
Essi contengono rispettivamente il byte basso e quello alto del tempo di ritardo, calcolato in centesimi di secondo.
L'uso di questa chiamata, che introduce una pausa di un secondo in attesa che l'utente prema un tasto, implica l'esecuzione della sezione di codice macchina presentata più avanti. I registri X ed Y ritrasmettono i valori letti.
La presenza di uno 0, tanto nel flag di carry quanto nel registro Y, indica che durante la chiamata è stato premuto un tasto il cui codice ASCII si trova nel registro A.
Se Y contiene 255 ed il flag C viene selezionato ad 1, durante la pausa prevista non è stato battuto alcun tasto. C posto a 1 e Y=27 indicano invece che è avvenuta la pressione del tasto Escape.
La seguente subroutine in codice macchina illustra come effettuare questa chiamata OSBYTE.
Se al termine della subroutine il flag vale 1, il flusso di controllo salta ad un'altra routine di gestione, che verifica il valore contenuto nel registro Y per determinare se è stato premuto Escape o un altro tasto.

```
1000    LDA     #129    /seleziona OSBYTE
1010    LDX     #100    /parametri
1020    LDY     #0
1030    JSR     &FFF4   /esegue la chiamata OSBYTE
1040    BCS     errore
1050    RTS
1060    .errore         /codice gestione errore
```

Altre chiamate OSBYTE, soprattutto quelle con valori di A compresi tra 166 e 255, sono dette "di lettura e scrittura" perché consentono tanto la lettura quanto la modifica dei valori contenuti in alcune variabili di sistema. L'ordine di eseguire l'una o l'altra operazione viene impartito alle chiamate OSBYTE in modo abbastanza semplice.
Per ottenere la scrittura di un valore, la procedura che chiama OSBYTE deve inserire in A il valore che OSBYTE scriverà nella variabile e azzerare il registro Y. Al contrario, per ottenere la lettura di un valore contenuto in una variabile di sistema, X deve essere posto a 0 ed Y a 255.
L'eventuale valore ritrasmesso viene caricato nei registri X ed Y.

Uso delle chiamate OSBYTE

Le chiamate OSBYTE sono al servizio del sistema operativo e vengono impiegate in molte sue routine. I sistemi di archiviazione dei file, i tasti Break ed Escape, la tastiera e l'Econet sono tutti sistemi influenzati in vario grado da tali chiamate. Dato il loro elevato numero, è possibile analizzare in questa sede soltanto

quelle di maggior utilità, descritte solo in modo sommario dal manuale per l'utente del Micro BBC.

Tasti funzione: *FX 18 non possiede parametri, ma è abbastanza utile. Cancella dalla memoria la definizione corrente di un tasto funzione e viene usato quando è necessario ridefinire più volte un tasto funzione all'interno di un programma.
Da *FX225 a *FX228: se un tasto funzione non è stato programmato con una stringa, è possibile usare queste chiamate per ottenere un codice ASCII alla pressione di uno dei tasti funzione rossi.
*FX225,n, ad esempio, fa sì che il tasto f0 generi il codice ASCII n, f1 il codice ASCII (n+1) e così via.
*FX226 svolge lo stesso lavoro quando i tasti funzione vengono premuti assieme al tasto Shift.
Ciò vale anche per *FX227 e il tasto Control. Per quanto riguarda *FX228 è invece previsto l'uso combinato dei tasti funzione e di entrambi i tasti Shift e Control.

Funzioni video e di controllo del VDU: sul Micro BBC, la maggior parte delle funzioni di controllo dello schermo vengono realizzate depositando opportuni valori usati dal sistema di controllo del VDU. A questo riguardo esiste una coppia di chiamate OSBYTE di grande interesse:
*FX19: sebbene non possieda parametri, si dimostra molto utile nella programmazione della grafica animata. La sua esecuzione costringe il computer ad una pausa d'attesa finché non viene tracciata l'immagine successiva e consente di produrre sequenze meno "sfarfallanti" del solito. La chiamata deve essere ripetuta tutte le volte che è necessaria una pausa. Poiché l'immagine viene "rinfrescata" 50 volte al secondo, la chiamata può essere usata per generare il tempo di scansione o per fornire le interrupt alla CPU.
*FX218: è una chiamata OSBYTE di lettura/scrittura che comunica all'utente la lunghezza della "coda VDU". Si è già detto che alcuni codici VDU, quando vengono trasmessi al sistema di controllo del VDU, devono essere seguiti da altri codici. Il numero di byte che il sistema di controllo del VDU si aspetta di ricevere in un dato istante è pari al numero dei byte nella coda VDU.
Quindi la chiamata *FX218 può essere usata per leggere questo dato e trasferirlo nel registro X.
*FX20: questa chiamata permette di ridefinire i caratteri con codici ASCII compresi tra 32 e 255, ossia una serie molto più ampia di quella dei comuni caratteri definiti dall'utente. Operando nei MODE da 0 a 6, possiamo modificare tutti i caratteri accessibili da una normale tastiera. Per ridefinire i caratteri che hanno codici compresi tra 32 e 128, occorre riservare una sezione di memoria dell'area di lavoro BASIC selezionando PAGE ad un valore più alto di quello abituale. I caratteri vengono ridefiniti, per mezzo della chiamata VDU 23, in blocchi di 32, ciascuno dei quali occupa 256 byte di memoria. Il solo parametro usato viene caricato nel registro X. Per ulteriori spiegazioni consultare la guida per l'utente del Micro BBC.

Suono: la funzione *FX210,n può essere usata per eliminare gli effetti sonori di alcuni giochi. *FX210,0 abilita l'emissione dei suoni, mentre qualsiasi altro valore in X cancella tutti i suoni.
Da *FX211 a *FX214: queste chiamate di lettura/scrittura controllano i suoni generati con CTRL-G e VDU7.
*FX211,n gestisce il canale del chip di suono sul quale viene emesso il suono VDU7.
*FX 12,n controlla il volume del suono o l'inviluppo usato per generarlo. Il volume viene definito in modo simile al parametro d'ampiezza del comando SOUND del BASIC: −15 produce suoni ad alto volume, 0 suoni inudibili. I numeri positivi sono i numeri d'inviluppo. Il valore fornito nel registro = X nella chiamata FX è (n−1)*8, dove n rappresenta il parametro. *FX213,n controlla l'altezza del tono generato da VDU7; *FX214,n la durata della sua emissione.

Tasto Escape: *FX229,n consente di disattivare (*FX229,1) e riattivare (FX229,0) il tasto Escape.
*FX220,n consente di trasformare un tasto qualsiasi in un tasto Escape: il parametro n deve contenere il codice ASCII del tasto che dovrà fungere da Escape. Il tasto Escape originario va disattivato. Ad esempio, *FX220,65 genera un "evento" Escape quando viene premuto il tasto A.

Buffer: *FX21,n controlla il flusso o svuotamento di un buffer. Lo svuotamento del buffer della tastiera, ad esempio, rimuove tutti i caratteri eventualmente digitati ed accumulati nel buffer durante l'esecuzione di un programma. Se inseriamo un comando GET in presenza di pressioni non ancora elaborate all'interno del buffer, GET estrae il primo in lista, anziché attendere che l'utente prema un tasto. Lo svuotamento di un buffer di suono conclude l'emissione del suono sul relativo canale, anche se ci sono altri comandi che attendono di essere elaborati dal chip di suono. Il buffer da svuotare viene selezionato dal parametro x (vedere tabella a margine di pagina).
*FX138,n,m inserisce un valore m nel buffer indicato da n. I codici numerici dei buffer sono gli stessi di *FX21.
Quindi *FX138,0,65 inserisce la lettera A nel buffer della tastiera.
Naturalmente, tutte queste chiamate OSBYTE richiedono il caricamento dei relativi parametri nei registriA, X e Y e la chiamata alla subroutine come descritto in precedenza.
La panoramica è limitata alle principali funzioni controllate dalle chiamate OSBYTE: le funzioni non menzionate, comunque, implicano tecniche assai simili a quelle descritte. Una tabella completa di tali funzioni si trova sul manuale per l'utente del Micro BBC.

Parametro X	Buffer selezionato
0	Tastiera
3	Stampante
4	Canale di suono 0
5	Canale di suono 1
6	Canale di suono 2
7	Canale di suono 3

Software in vetrina

Una selezione tra i programmi disponibili per i computer più diffusi

I due programmi qui presentati appartengono al genere dei giochi d'abilità ed entrambi prevedono l'uso di joystick.
Il loro contenuto è piuttosto originale e sicuramente divertente.

"Percy the potty pidgeon" per il Commodore 64

Chi abbia mai guidato per le strade di Londra sa quanto siano pericolosi i piccioni "locali", sempre pronti a balzare tra le ruote dei veicoli.
Protagonista di questo gioco è proprio uno di questi piccioni che, abilmente pilotato (si suppone) dal giocatore deve aggirarsi per le strade cittadine o sull'autostrada, per raccogliere pagliuzze per il proprio nido.
Naturalmente, ciò deve avvenire nella totale incolumità del volatile e nel minor tempo possibile.
Ai livelli di difficoltà più alti, il piccione deve gareggiare con altri uccelli che mirano allo stesso scopo, ma anche con avidi rapaci che tentano di catturarlo.
Il gioco è prodotto dalla Gremlin Graphics (GB).





"Caesar the cat" per il Commodore 64

Come nel gioco precedente, anche qui il protagonista è un animale. Stavolta, però, si tratta di un gatto a caccia di topi in una dispensa ben stipata di leccornie di ogni tipo. I topini vi si aggirano "attaccando" forme di formaggio, prosciutti e via dicendo, mentre il gatto (manovrato dal giocatore tramite joystick) deve tentare di fermarli nel minor tempo possibile.
Se Cesare (il gatto) urta una parete o fa cadere un oggetto viene penalizzato di un certo numero di punti. Acquisita pratica al livello più basso, si può tentare con quelli di maggior difficoltà, che risultano molto più divertenti e accattivanti.
Indubbiamente la Mirrorsoft, che ha realizzato il programma, non ha lesinato fantasia per produrre un gioco originale e dal ritmo frenetico.