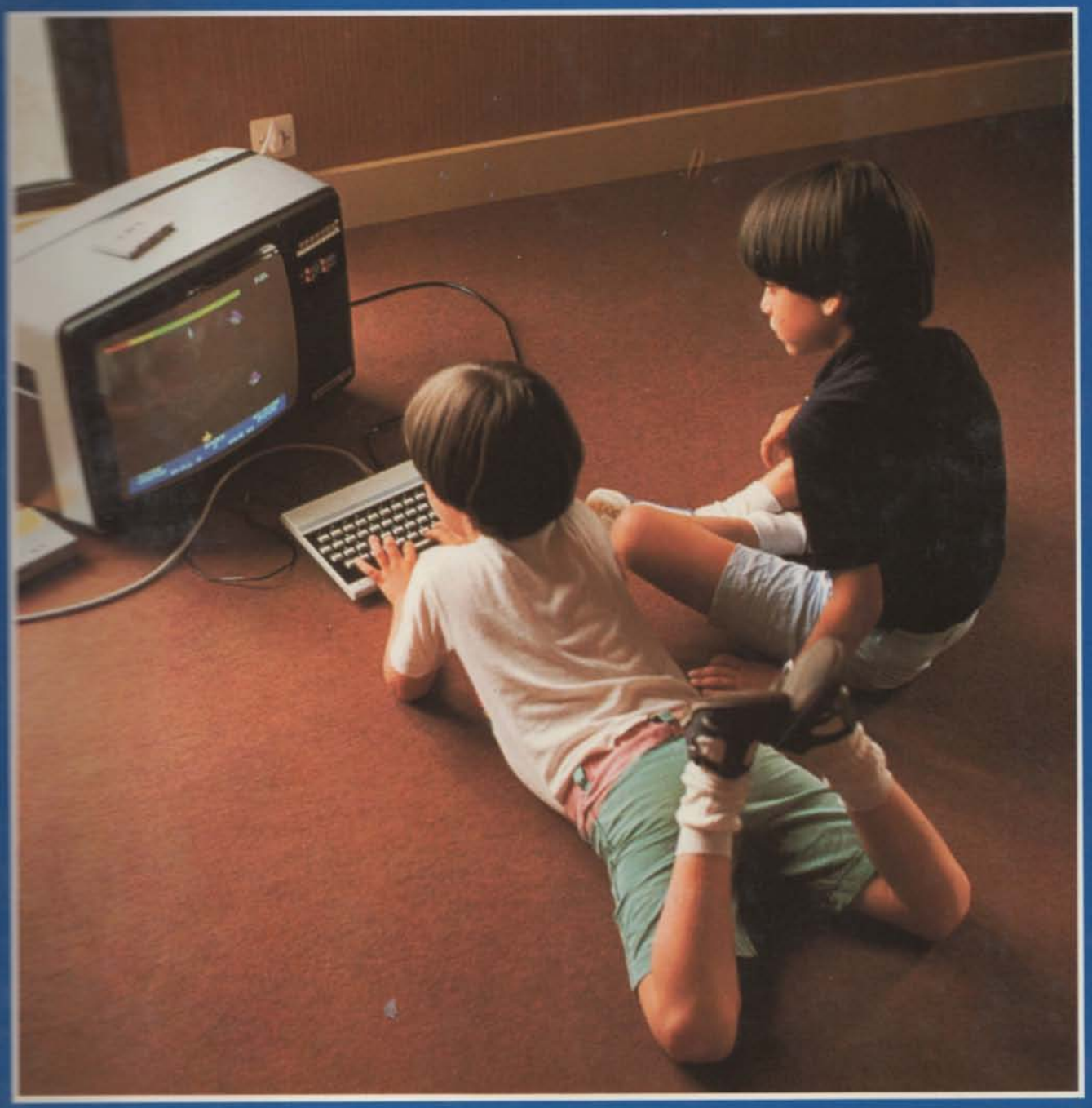
IL MIO COMPUTER

CORSO PRATICO DI INFORMATICA: MICRO, MINI E PERSONAL COMPUTER



ISTITUTO GEOGRAFICO DE AGOSTINI

IL MIO
COMPUTER

IL MIO COMPUTER

CORSO PRATICO DI INFORMATICA: MICRO, MINI E PERSONAL COMPUTER

volume quinto

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, 1984.
Stampato in Italia - IGDA Officine Grafiche, Novara, 1985.

Sommario

961 I creatori di melodie
964 Errori? No, grazie
966 Crimine al computer
968 Software in vetrina
969 L'Einstein
972 Fine alle mine
975 Il Lords of Midnight
976 Colorare col Commodore
980 La Softsel
981 Il QL della Sinclair
984 Il gioco dell'impiccato
986 Introduzione al LOGO
988 Software in vetrina
989 Nuove espressioni musicali
992 L'elemento umano
994 Una porta... da aprire
997 Un po' di magia nera
998 L'Assembly del 6809
1000 La Psion
1001 Confezione "su misura"
1003 Una porta per comandare
1007 I menu di assistenza
1009 Il Sega Sc3000H
1012 LOGO e tartarughe
1014 Suoni in sequenza
1017 Da registro a registro
1020 La Melbourne House
1021 Le tascabili della Casio
1023 Divertirsi con la tartaruga
1026 Collaudiamo la "scatola nera"
1029 Database
1030 Da un'idea al successo
1033 Le interfacce musicali
1036 La scelta... di una scelta!
1038 I campi del 6809
1040 La Casio
1041 Musica... sintetica
1044 Giochi d'incastro
1046 Le "librerie" di routine
1048 L'Apple IIc
1052 Un generatore di caratteri
1054 Dispositivo di controllo
1056 Database
1057 L'Aritmetica del 6809
1060 La Memotech
1061 Gran finale
1065 Controllo di un motore
1068 La creazione di caratteri
1070 Joypad senza fili
1072 Database
1073 Variazioni su... una tartaruga
1076 Software calibrato
1078 Cambio d'indirizzo
1080 Classic racing
1081 Paperi e droidi
1084 La ricursione del LOGO
1086 Il check-up di un programma
1088 Database
1089 Il PX-8
1092 Duplice controllo
1095 Space Invaders
1096 Utili puntatori
1098 Ricerca e abbinamento
1101 Un passo alla volta
1103 "Divide et impera"
1106 La somma delle parti
1108 Database
1109 Il Koala-pad
1112 Una gara... in moto
1114 Controllo del traffico
1117 Salti ad effetto
1120 Pacman
1121 Topi e labirinti
1124 Una sinfonia corale
1126 Fonte d'energia
1129 Database
1130 Un trio da viaggio
1133 Un altro giro in moto
1134 Una tartaruga spaziale
1137 Uso degli stack
1140 Missile Command
1141 Mani e gomiti
1144 Le routine di utilità
1146 Sprite col LOGO
1149 Il Toshiba HX-10
1152 Un maggior controllo
1154 Semplici applicazioni
1156 Database
1157 Ingresso e uscita sul 6809
1160 Star Raiders
1161 I "sensi" dei robot
1165 Un comodo visualizzatore
1168 Database
1169 Un giro tra le vetrine
1172 Una macchina da calcolo
1174 Caccia al drago
1177 Sospendere l'esecuzione
1180 Una utility per il Commodore
1181 Movimenti di precisione
1184 Utility ibride
1186 Il LOGO per gli Atari
1189 Il Plus/4
1192 Mutui e prestiti
1194 Un convertitore D-A
1196 Database
1197 Libero movimento
1200 Psytron

Indice per settore

Panorami



961 I creatori di melodie
966 Crimine al computer
968 Software in vetrina
988 Software in vetrina
989 Nuove espressioni musicali
1001 Confezioni "su misura"
1014 Suoni in sequenza
1033 Le interfacce musicali
1041 Musica... sintetica
1061 Gran finale
1081 Paperi e droidi
1101 Un passo alla volta
1121 Topi e labirinti
1141 Mani e gomiti
1161 I "sensi" dei robot
1181 Movimenti di precisione

Tecniche di programmazione



964 Errori? No, grazie
992 L'elemento umano
1007 I menu di assistenza
1036 La scelta... di una scelta!
1046 Le "librerie" di routine
1076 Software calibrato
1086 Il check-up di un programma

Hardware



969 L'Einstein
981 Il QL della Sinclair
1009 Il Sega Sc3000H
1021 Le tascabili della Casio
1048 L'Apple IIc
1070 Joypad senza fili
1089 Il PX-8
1109 Il Koala-pad
1130 Un trio da viaggio
1149 Il Toshiba HX-10
1169 Un giro tra le vetrine
1189 Il Plus/4

Programmazione BASIC



972 Fine alle mine
984 Il gioco dell'impiccato
1052 Un generatore di caratteri
1068 La creazione di caratteri
1096 Utili puntatori
1112 Una gara... in moto
1133 Un altro giro in moto
1144 Le routine di utilità
1180 Una utility per il Commodore
1184 Utility ibride

Software



975 Il Lords of Midnight
997 Un po' di magia nera
1030 Da un'idea al successo

1080 Classic racing
1095 Space Invaders
1106 La somma delle parti
1120 Pacman
1124 Una sinfonia corale
1140 Missile Command
1152 Un maggior controllo
1160 Star Raiders
1172 Una macchina da calcolo
1192 Mutui e prestiti
1200 Psytron

Linguaggio macchina

976 Colorare col Commodore
998 L'Assembly del 6809
1017 Da registro a registro
1029 Database
1038 I campi del 6809
1056 Database
1057 L'Aritmetica del 6809
1072 Database
1078 Cambio d'indirizzo
1088 Database
1098 Ricerca e abbinamento
1108 Database
1117 Salti ad effetto
1129 Database
1137 Uso degli stack
1156 Database
1157 Ingresso e uscita sul 6809
1168 Database
1177 Sospendere l'esecuzione
1196 Database
1197 Libero movimento

I pionieri dell'Informatica

980 La Softsel
1000 La Psion
1020 La Melbourne House
1040 La Casio
1060 La Memotech

Note d'Informatica

986 Introduzione al LOGO
1012 LOGO e tartarughe
1023 Divertirsi con la tartaruga
1044 Giochi d'incastro
1073 Variazioni su... una tartaruga
1084 La ricursione del LOGO
1103 "Divide et impera"
1134 Una tartaruga spaziale
1146 Sprite col LOGO
1174 Caccia al drago
1186 Il LOGO per gli Atari

Laboratorio

994 Una porta... da aprire
1003 Una porta per comandare
1026 Collaudiamo la "scatola nera"
1054 Dispositivo di controllo
1065 Controllo di un motore
1092 Duplice controllo
1114 Controllo del traffico
1126 Fonte d'energia
1154 Semplici applicazioni
1165 Un comodo visualizzatore
1194 Un convertitore D-A

I creatori di melodie

Strumenti musicali computerizzati producono effetti inimmaginabili

I rapporti armonici che legano musica e matematica sono noti fin dall'antichità. Lo stesso Pitagora pensò di pesare un gruppo di martelli da fabbro, incuriosito dal fatto che sembravano dare suoni "accordati" tra loro quando venivano battuti sull'incudine. Scoprì così che un martello produce un suono di frequenza esattamente doppia, cioè più alto di una ottava rispetto a quello generato da un martello di peso doppio. Questa relazione definisce il primo principio che regola i rapporti tra i vari toni musicali.

Nel Medio Evo, molti compositori scrissero messe e mottetti (composizioni polifoniche corali) le cui strutture ritmiche e le proporzioni numeriche, di straordinaria complessità, erano precise come l'architettura delle cattedrali in cui i brani venivano suonati.

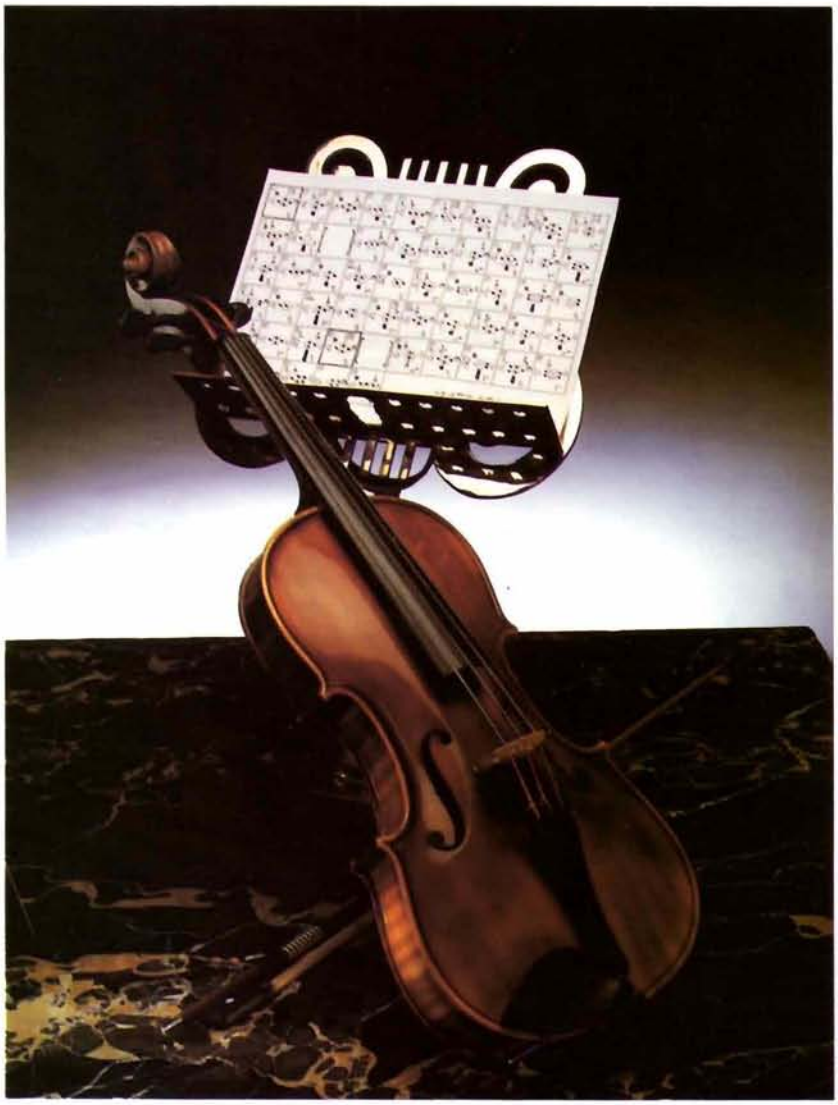
Ai nostri giorni, chi assiste ad un concerto dal vivo noterà che i musicisti, di qualunque brano si tratti, spesso contano le battute, prima di suonare.

Non c'è da stupirsi, dunque, che i mondi della musica e della elaborazione elettronica siano, ad un certo punto, entrati in contatto. Ultimamente, ha suscitato grande interesse una nuova apparecchiatura chiamata MIDI (Musical Instrument Digital Interface) che permette ad un qualsiasi sistema digitale, compresi i microcomputer, di controllare il funzionamento di un sistema per la generazione di suoni.

Poiché la maggior parte degli strumenti musicali sono oggigiorno digitali, questo apparecchio apre ai possessori di home computer una nuova e affascinante gamma di possibilità.

Tuttavia, non si creda che l'interfaccia MIDI sia un apparecchio capace di trasformare magicamente un qualsiasi appassionato di micro in un novello Bach o in uno Stevie Wonder; la composizione di un buon "pezzo" dipende sempre ed esclusivamente dalla fantasia e dalle doti musicali dell'autore, sia che usi un sintetizzatore o una semplice chitarra!

Per farsi un'idea più precisa degli strumenti collegabili alla MIDI e di come sia nata la musica elettronica, occorre tornare indietro di una cinquantina d'anni. Già prima della Seconda Guerra Mondiale, alcuni musicisti avevano fatto esperimenti su semplici generatori di suoni ad onda sinusoidale. Questi erano apparecchi elettrici, usati per far vibrare una lamina metallica che produceva suoni stabili ma di altezza variabile. Queste sonorità vennero spesso impiegate, intorno agli inizi degli anni '50, nelle colonne sonore dei film di fantascienza, per accrescerne il mistero e l'atmosfera futuristica: erano qualcosa di simile al suono emesso dall'altoparlante di un odierno apparecchio televisivo al termine delle trasmissioni.



MARCUS WILSON-SMITH

Musica Oscura

La musica elettronica richiede una notazione musicale totalmente nuova. Gli spartiti di Stockhausen, con le loro rappresentazioni visive dei suoni e le direzioni grafiche di temporizzazione/sincronizzazione, non hanno niente in comune con gli spartiti classici ma, al contrario, somigliano agli schemi di un circuito elettrico

I primi organi Hammond, immessi sul mercato fin dal 1930, erano strumenti elettronici in grado di generare suoni in base a questo principio.

Ma fu il boom dell'elettronica, verificatosi durante la Seconda Guerra Mondiale (in particolar modo l'invenzione dei registratori a nastro in Germania), che permise ai musicisti di creare e manipolare i suoni con tecniche sempre più complesse. I suoni venivano generati utilizzando spezzoni di nastro di registrazione, contenenti suoni analogici, "strumentali" o di altro tipo, registrati in

DOCTOR WHO

Il primo grande successo, nel campo della musica elettronica, è senza dubbio la colonna sonora della serie televisiva "Doctor Who", composta nel 1962 per il BBC Radiophonic Workshop da Ron Grainer (il primo a sinistra nella foto)

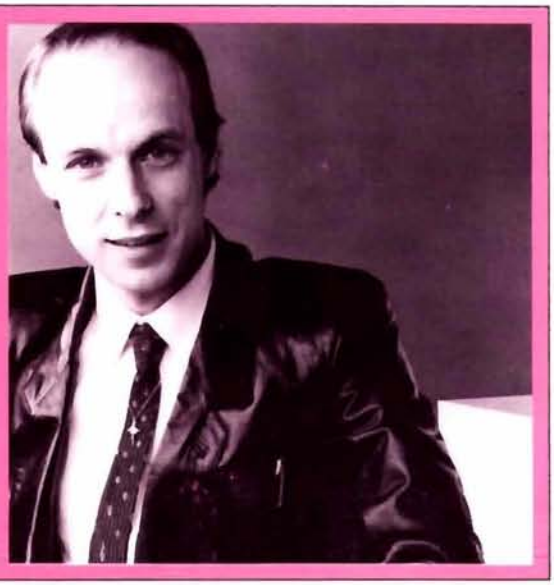


BBC television

precedenza: gli spezzoni venivano poi accuratamente combinati, ottenendo un collage di emissioni sonore.
Questa "new music" trasgrediva tutte le regole teoriche su cui si era fino ad allora basata la composizione musicale e suscitò accesi entusiasmi e critiche amare da parte degli ascoltatori.
Contemporaneamente andarono diffondendosi apparecchiature relativamente sofisticate, in grado di distorcere o di variare i semplici toni emessi dagli oscillatori. Negli anni '50 alcuni compositori, fra i quali Stockhausen, sviluppavano le proprie teorie, componendo musica elettronica "pura" in piccoli studi musicali collegati a stazioni radio locali.
Pierre Schaffer, lavorando assieme ad un gruppo di tecnici del suono della ORFT (l'azienda statale di radiodiffusione francese), si avventurò in quella che fu chiamata *musique concrete*, e cioè una musica composita, creata con suoni registrati dalla vita di tutti i giorni.
In quegli anni, negli Stati Uniti, la Bell Telephone Laboratories costruì quello che può essere considerato il primo sintetizzatore.

MUSICA E RICERCA

Brian Eno, il più noto componente del gruppo Roxy Music, che riscosse grande successo negli anni '70, fu un pioniere nell'uso dei sintetizzatori. Dopo aver lasciato il gruppo è stato uno dei principali rappresentati dell'avanguardia e della musica elettronica "mood". Ha collaborato con famosi musicisti come David Bowie e Robert Fripp. Recentemente si è dedicato alle colonne sonore televisive e cinematografiche e, assieme al fratello, ha scritto per la NASA la musica per un film sull'allunaggio degli astronauti americani



L'apparecchio, che occupava numerose stanze, era stato progettato per studiare la sintesi della voce umana. La Bell si era accorta, infatti, che a causa dei numerosi dialetti e le varie inflessioni, la pronuncia dei numeri telefonici veniva spesso mal interpretata, causando non pochi errori nella selezione delle comunicazioni telefoniche. La Bell Telephone sperava, forse un po' troppo ottimisticamente a quel tempo, di poter risolvere questi problemi usando un sintetizzatore vocale, comprensibile a tutti. Fu grazie a queste ricerche, ed alla cortese disponibilità della Compagnia, che molti musicisti americani mossero i primi passi nel mondo della musica elettronica.
In Gran Bretagna, intanto, si stavano facendo studi dello stesso genere ma su scala meno ambiziosa. Nel 1960, il BBC Radiophonic Workshop (il laboratorio radiofonico della BBC) compose un brano musicale divenuto poi un classico della musica elettronica: la colonna sonora della serie televisiva "Doctor Who".
I primi tentativi di creare musica col computer furono eseguiti da Lejaren Hiller, dell'Università dell'Illinois, agli inizi del 1957. Egli introdusse in un computer Illiac una serie di istruzioni musicali suddivise in quattro gruppi di informazioni di carattere tecnico, che vennero successivamente trascritte in linguaggio musicale.
Il risultato fu un brano, chiamato *Illiac Suite*, composto da quattro "movimenti" per quartetto d'archi. Il pezzo, anche dopo l'arrangiamento per violoncello, viola e due violini, presentava tuttavia sonorità sfuggenti e incerte.
Basta però anche una superficiale conoscenza della musica di quell'epoca, per trovare brani musicali ben peggiori, scritti da veri compositori!
Pochi anni dopo, Hiller compose un altro pezzo, lavorando questa volta su un computer IBM 7090. Egli progettò uno schema di programmazione chiamato MUSICOMP (la sigla deriva da "MUsic Simulator-Interpreter for COMpositional Procedures") che offriva una maggiore flessibilità e una più ampia varietà di passaggi. Il risultato finale fu chiamato *Computer Cantata* e prevedeva la partecipazione di un vocalista accompagnato da una base musicale elettronica registrata su nastro. Anche in questo caso, l'armonia del brano appariva discontinua e altalenante, ciò nonostante Hiller riuscì a dimostrare ai suoi colleghi compositori che un computer poteva essere concretamente usato in modo creativo.
Fatta eccezione per i brani scritti per i film fantascientifici, la musica elettronica si mantenne per molti anni nell'ambito della musica classica ed il pubblico poté accorgersi dei mutamenti e delle innovazioni tecniche che andavano nascendo, solo grazie all'impegno di un ristretto numero di compositori d'avanguardia.
Un tipico concerto di musica elettronica degli anni '60 aveva come protagonisti due tipi di

musicisti: alcuni suonavano strumenti tradizionali, altri "elaboravano suoni", usando quegli stessi strumenti abbinati a filtri e sistemi ad oscillazione di frequenza.
Gli uni e gli altri (come pure i tecnici del suono) seguivano particolari partiture, la cui notazione aveva scarsissimi punti di contatto con quella tradizionale.
Oltre ad adottare nuovi linguaggi tecnici, per descrivere la posizione dei microfoni ed il filtraggio dei suoni, i compositori cercarono di trascrivere su carta i vari effetti. In alcuni casi, i compositori realizzarono spartiti che somigliavano in modo impressionante ai fogli usati dai progettisti grafici. Il problema relativo al linguaggio da usare e a come comunicare e rappresentare graficamente le varie partiture è ancora attuale nei sistemi musicali digitali degli anni '80.
Dal 1960 in poi, i musicisti "pop" della nuova generazione cominciarono a trascorrere sempre più tempo nelle sale d'incisione, dedicandosi con accanimento alla sperimentazione elettronica.
Un esempio classico è quello dei Beatles, che trovarono in George Martin non solo un eccezionale tecnico della registrazione ma, soprattutto, un sensibile musicista con profonde conoscenze in campo classico.
Egli spinse i Beatles ad usare l'intera sala d'incisione come un unico strumento musicale ed in breve tempo il complesso inglese imparò ad usare le tecniche di registrazione "a collage" su nastro, oltre ad una vasta gamma di suoni sintetizzati.
Ma l'esempio non si ferma qui: il chitarrista Jimmy Page ha modellato il proprio stile su quello dei musicisti di colore americani degli anni '50 ma, usando una serie di distorsori, ha ottenuto quelle particolari sonorità che ancor oggi rendono inconfondibili i pezzi dei Led Zeppelin. Il suo sound, assieme a quello del compianto Jimi Hendrix, che si serviva di distorsori d'interferenza e di speciali filtri a scansione rapida (oggi chiamati "pedali whawha"), sta alla base del genere musicale "Heavy Metal".
Intorno al 1970 gli apparecchi per la generazione e l'elaborazione dei suoni, nati negli anni '50, vennero prodotti in versioni a transistor.
Divennero molto meno ingombranti ed il loro impiego non fu più limitato alle sale d'incisione.
I chitarristi, suonando dal vivo, avevano a loro disposizione svariati pedali per effetti speciali. Poco tempo dopo, anche i tastieristi ed i pianisti potevano disporre di sintetizzatori portatili.
Questi sintetizzatori si basavano su una serie di oscillatori di tono, di formatori d'inviluppo (che generano l'Attacco, il Sustain o Prolungamento ed il Decadimento del suono), di filtri variabili, di modulatori di rete (per comporre i segnali in nuove frequenze) e di generatori di rumore. Come Jimi Hendrix era divenuto un modello per i chitarristi, Brian Eno divenne un "classico" per i suonatori di sintetizzatore, soprattutto per i suoi stretti legami musicali con l'avanguardia "pura" della new music.
La consolle di missaggio (o mixer), allora in grado di miscelare suoni su nastri a 16 o 24 piste, era però ancora troppo ingombrante per essere trasportata ed usata dal vivo.
In Giamaica, musicisti e tecnici del suono cominciarono a usare il mixer come un vero e proprio strumento. Registrarono intere canzoni su nastri a più piste e da esse estrassero le varie componenti ritmiche. Le diverse tracce di registrazione vocale e strumentale vennero poi impiegate come materia prima, per ottenere dal mixer sezioni ritmiche diversificate, largamente dipendenti da unità di riverbero e di ritardo del segnale. Questa sofisticatissima tecnica fu chiamata "dub". Intorno alla metà del 1980, le tecniche di "sampling" (campionatura) e di manipolazione dei suoni hanno assunto un ruolo primario nella produzione musicale.
I gruppi musicali di maggior successo, come i "Culture Club", combinano le proprie doti musicali nel comporre le canzoni alle tecniche di creazione digitale di professionisti quali Steve Levine.
La necessità di un'interfaccia che potesse collegare tra loro strumenti digitali diversi o che permettesse di ampliare le capacità di un sintetizzatore, ha spinto i fabbricanti di strumenti musicali a realizzare un progetto comune.
Il progetto di massima del MIDI venne redatto nell'aprile del 1973 e, da allora, sono pochi i sintetizzatori non compatibili con questo sistema.



COMPOSITORE DI SUCCESSO

Oltre ad essere impresario del gruppo inglese "Culture Club", Steve Levine è molto apprezzato per la sua abilità nel "mixare" musica elettronica e performance vocali. Levine è stato uno dei primi compositori inglesi a far uso del Linn (un sintetizzatore di "percussioni" programmabile) ed ha portato le tecniche di registrazione digitale ad un livello di assoluta perfezione. Levine ha fatto grande uso di sintetizzatori e di altre apparecchiature musicali digitali nel brano "Believin", composto assieme a Boy George

Errori? No, grazie

Una breve guida per evitare (o rintracciare) le cause d'errore più frequenti

Nelle diverse fasi dello sviluppo di un programma, può accadere che si commettano errori di vario genere: di impostazione, di progettazione, di codifica e di verifica. Gli errori introdotti nelle fasi di impostazione e di progettazione dipendono spesso da un'insufficiente analisi del problema affrontato o dalla scarsa attenzione prestata nell'assicurarsi che il programma esegua con esattezza ciò che ad esso è richiesto. Per ridurre l'incidenza di questi errori, conviene seguire i metodi di programmazione strutturata, descritti a pag. 956. Altri errori possono verificarsi quando, dalla fase progettuale, si passa alla scrittura del codice (è facile commettere errori di battitura, soprattutto trascrivendo i nomi delle variabili) oppure in fase di verifica e di debugging, quando la correzione di un errore ne provoca altri.
Comunque, la maggior parte degli errori si annida soprattutto nelle "interfacce" tra routine o tra il programma e l'utente. Occorre sempre assicurarsi che i valori trasferiti alle routine appartengano alla corretta gamma di valori e siano del tipo (numerico, alfanumerico, ecc.) richiesto dal programma. La verifica può essere incorporata tanto nella routine che li "invia" quanto in quella che li "riceve": l'importante è che vengano controllati.
Per assicurarsi che l'output dei valori di una routine rientri nella corretta gamma e che sia di giusto tipo, tanto i valori inseriti dall'utente che quelli letti da un file vanno sempre sottoposti a verifica. Infatti, le subroutine si possono progettare affinché generino un ben preciso arco di valori, ma la mente umana non sempre funziona così sistematicamente. Pertanto è bene che ogni routine, destinata a ricevere un dato dall'utente, contenga precise procedure di controllo. La stessa cosa vale per le routine di gestione dei file, poiché le informazioni in essi contenute potrebbero contenere alterazioni o errori di lettura.
Non sempre gli errori provocano un blocco del programma: ciò si verifica solo quando il programma trasgredisce una regola del linguaggio (usando, ad esempio, un operatore illegale come in RISULTATO = PRIMO$ + SECONDO$) oppure una norma del sistema operativo (aprendo troppi file simultaneamente, ad esempio). Il seguente programma può sembrare del tutto corretto:

```
10 FOR CONTATORE = 1 TO 10
20      SOMMA = SOMMA + 1
30      PRINT CONTATORE,SOMMA
40      GOTO 10
50 NEXT CONTATORE
```

ma contiene un ciclo senza fine, che porta alla caduta del sistema, perché trasgredisce una regola del linguaggio. Il BASIC, infatti, usa lo stack per appoggiarvi il conteggio per i cicli FOR...NEXT, incrementando il puntatore allo stack ogni volta che inizia un nuovo ciclo. Nel programma, la linea 50 (la cui istruzione NEXT dovrebbe decrementare lo stack) non viene mai raggiunta e lo stack non viene decrementato. Ad un certo punto, quando lo stack è stracolmo, sullo schermo compare il messaggio "stack overflow" ed il BASIC interrompe il programma. Simili errori sono in genere facilmente individuabili, ma se si trovano in sezioni di programma usate di rado, localizzarli può richiedere enormi sprechi di tempo.
Tuttavia, gli errori più insidiosi sono quelli che permettono di eseguire il programma per intero, ma compromettono la validità dei risultati. Come esempio abbiamo scelto un programma che disegna una figura sullo schermo e poi la colora. Quando viene raggiunto

COME INDIVIDUARE GLI ERRORI

La seguente tabella di analisi degli errori (tratta da "The Art Of Software Testing" di G.J. Myers) propone un metodo logico e strutturato che permette di evitare errori di programmazione e di individuare quelli eventualmente commessi

VARIABILI

1) Ogni variabile ha un proprio specifico nome? (Ricordiamo che molti interpreti riconoscono soltanto i primi due caratteri del nome della variabile)
2) Le variabili (soprattutto quelle usate come contatori di ciclo o parametri di subroutine) sono state nuovamente inizializzate, prima di riusarle?
3) Gli indici delle matrici rispettano la dimensione assegnata con DIM? Gli indici sono numeri interi?
4) Gli indici delle matrici partono da zero o da uno?

CALCOLI

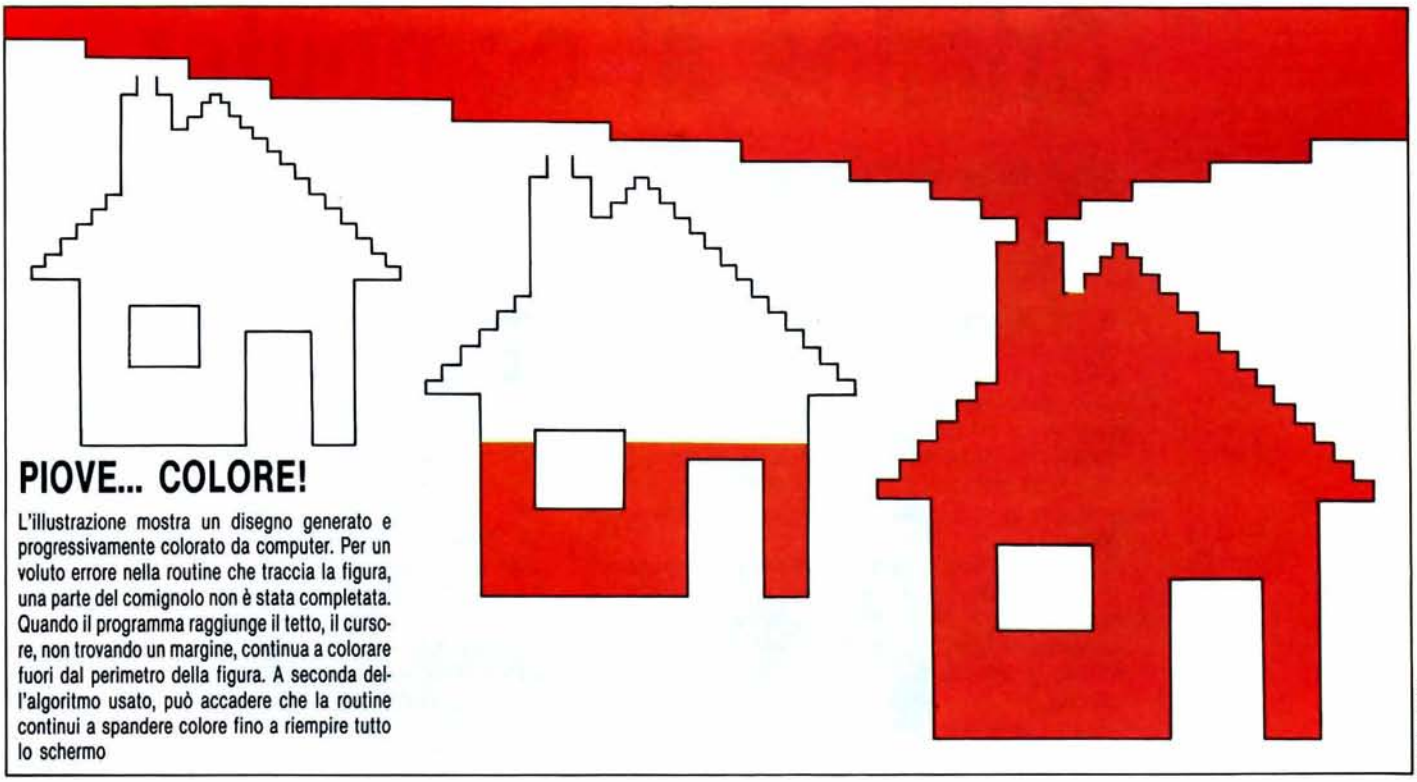
1) I calcoli danno risultati numerici o stringhe? I risultati vengono assegnati, di conseguenza, a variabili numeriche o alfanumeriche?
2) Qualche calcolo genera numeri troppo piccoli o troppo grandi, che il computer non è in grado di trattare? Ciò può causare un errore di "divisione per zero"?
3) Gli errori dovuti agli arrotondamenti operati dall'interprete sono significativi?
4) Tutti gli operatori di un'espressione vengono eseguiti nel corretto ordine logico, anziché nell'ordine imposto dalle norme di precedenza degli operatori aritmetici?

CONFRONTI

1) Si è badato a fare confronti solo tra stringhe o tra numeri?
2) Ha importanza se una stringa è in parte o totalmente in caratteri maiuscoli o minuscoli?
3) Vengono confrontate stringhe di lunghezza diversa? La differenza di lunghezza è più o meno importante della differenza tra i caratteri?
4) Sono stati combinati in modo corretto gli operatori booleani e quelli di confronto?
5) Le norme di precedenza degli operatori booleani e dei confronti influiscono sull'esecuzione degli stessi confronti?

CONTROLLO

1) I cicli e gli algoritmi si concludono regolarmente, indipendentemente dallo stato delle variabili?
2) Ciascun ciclo o routine ha solo un ingresso ed una sola uscita?
3) Quando una condizione IF...THEN fallisce, il controllo passa alla successiva istruzione della linea o alla linea di programma che segue?
4) Cosa accade se non viene soddisfatta alcuna delle condizioni di verifica, in un'istruzione di diramazione multipla?



PIOVE... COLORE!

L'illustrazione mostra un disegno generato e progressivamente colorato da computer. Per un voluto errore nella routine che traccia la figura, una parte del comignolo non è stata completata. Quando il programma raggiunge il tetto, il cursore, non trovando un margine, continua a colorare fuori dal perimetro della figura. A seconda dell'algoritmo usato, può accadere che la routine continui a spandere colore fino a riempire tutto lo schermo

KEVIN JONES

un margine, il computer inverte la direzione del cursore e riprende a colorare fino a raggiungere un altro margine. Perché la routine di colorazione possa funzionare, i margini non devono presentare interruzioni, altrimenti la routine passa a colorare anche zone esterne alla figura.

Le versioni del linguaggio BASIC usate sulla maggior parte degli home computer possiedono procedure di gestione degli errori relativamente semplici, permettendo di riprendere l'esecuzione di un programma interrotto, dopo che l'utente ha corretto tramite la tastiera i valori delle variabili (una funzione assai apprezzabile, durante la messa a punto di un programma). Inoltre, molti dialetti di BASIC possiedono un comando del tipo ON ERROR GOTO, che consente di trasferire il controllo ad una particolare routine di gestione degli errori, scritta dall'utente, per trattare in modo speciale i cosiddetti errori "fatali". La linea che contiene tale comando, ad esempio:

```
30 ON ERROR GOTO 20000: REM Routine di gestione degli errori
```

viene quasi sempre inserita all'inizio del programma, cosicché qualsiasi errore devia il flusso del programma e provoca l'esecuzione del GOTO 20000. All'uso della frase ON ERROR, in genere, si abbina quello di due variabili: la prima immagazzina il codice numerico dell'errore verificatosi, la seconda contiene il numero della linea in cui si è verificato. I nomi assegnati a queste variabili ed i codici d'errore cambiano da macchina a macchina, per cui è necessario consultare il manuale del proprio apparecchio. Ovviamente, la routine che inizia alla linea 20000 deve prendere le opportune contromisure, in base ai valori di queste due variabili.

Un buon programma non deve contenere più di una routine ON ERROR. Questa non sarà forse in grado di gestire gli errori di sintassi, l'eventuale mancanza di memoria o il superamento della capacità dello stack, ma permetterà un ordinato arresto del sistema, assicurerà la chiusura (CLOSE) di tutti i file e darà modo all'utente di capire con esattezza cosa è successo. Altri errori, quali una divisione per zero, vanno gestiti in modo totalmente diverso. Ciò è dovuto ai seguenti motivi:

- Il comando ON ERROR GOTO ed il conseguente ritorno al programma principale costituiscono un'entrata ed un'uscita addizionale alla routine, violando un'importante norma della programmazione strutturata, secondo la quale le routine debbono avere solo un'entrata ed un'uscita.
- La procedura che "protegge" il programma dalle divisioni per zero deve trovarsi all'interno della routine che esegue l'operazione di divisione. Non si dovrebbero mai impiegare algoritmi che possano implicare il blocco del programma. Se la procedura di verifica dell'errore rallenta eccessivamente l'esecuzione del programma, la routine deve essere riprogettata e velocizzata.
- Le routine di gestione degli errori tendono a trasformarsi con facilità in complicate costruzioni di IF...THEN...ELSE, con uscite multiple strettamente collegate alla numerazione del resto del programma. Devono perciò essere riscritte, ogni volta che si modifica la routine da loro chiamata. Le routine di gestione degli errori sono, dunque, proprio le più difficili da programmare e da mettere a punto. Ecco perché i programmatori "pigri" (certo non professionali!) tendono ad ometterle del tutto!

Crimine al computer

L'accesso illegale, da parte di piccoli computer, nelle reti informative

Il film *War Games* ha colpito la fantasia di molti possessori di home computer. Servendosi di un micro e di un modem, il giovane protagonista del film riesce ad accedere illegalmente ad una serie di computer, per cambiare i risultati dei propri esami, per prenotare i biglietti per un volo, o per trasferire sul proprio micro le ultime novità nel software di gioco.

La situazione diventa pericolosa, però, quando il giovane penetra inconsapevolmente nel computer del NORAD, il sistema responsabile della Difesa aerea del Nord America, correndo il rischio di scatenare una vera guerra nucleare. Ovviamente, si tratta di finzione, ma è possibile che una cosa simile accada realmente?

In effetti, la "violazione" di sistemi computerizzati è un evento quasi quotidiano ed i colpevoli sono spesso dei ragazzi, armati di un micro e di un modem. Le "vittime" possono essere grosse unità di elaborazione universitarie o quelle appartenenti a grandi società di servizi. In teoria, qualsiasi computer che consenta un accesso dall'esterno tramite linea telefonica è vulnerabile! Nel 1983, accadde realmente un fatto molto simile a quello narrato nel film: si sospettò che due "violatori" fossero riusciti ad entrare nel sistema di elaborazione NORAD di Omaha, nel Nebrasca.

I due giovani coinvolti chiamavano da Los Angeles ed erano riusciti a penetrare nell'Arpanet, la rete segreta per computer usata dal Dipartimento della Difesa degli Stati Uniti. Usando un Commodore 64 ed un Tandy TRS-80, la coppia aveva esplorato la memoria di molti computer collegati all'Arpanet, che appartenevano ad industrie impegnate nella difesa e ad istituti universitari di ricerca. I due ragazzi non riuscirono ad ottenere informazioni riservate (l'Arpanet viene usato soprattutto per la consultazione di dati scientifici), ma la facilità con cui erano riusciti a "forzare" il sistema mise in grande allarme il Dipartimento della Difesa.

La violazione dipendeva in gran parte dalla pigrizia degli utenti, piuttosto che da un difetto del sistema.

Infatti, ogni utente dell'Arpanet sceglie una propria parola d'accesso che, talvolta, non rispecchia una fervida fantasia. In quel caso, i due ragazzi supposero che l'Università di California, a Berkeley, fosse un utente dell'Arpanet e che avesse scelto come parola d'accesso alla rete la sigla UCB. Servendosi di questa sigla, riuscirono a violare liberamente i diversi computer collegati all'Arpanet, compreso il NORAD. Fortunatamente, il particolare computer del NORAD, direttamente responsabile per la difesa aerea, non fa parte dell'Arpanet, ma si trova ben nascosto nel Colorado e, soprattutto, non è collegato al sistema telefonico pubblico.

In un altro incidente, verificatosi nel luglio dell'83, un gruppo di giovani di Milwaukee violò più di 60 computer appartenenti a college, società ed al "Los Alamos National Laboratory", impegnato in ricerche sugli armamenti.

Anche in questo caso i giovani ebbero accesso ad informazioni non riservate come verbali, bollettini di carattere generale e comunicazioni. La FBI intervenne per scoprire come il gruppo di "violatori", che si era firmato con la sigla "414s", fosse riuscito a compiere una tale impresa, ma risultò che tutti i computer "violati" erano sprovvisti di misure di sicurezza.

Questi sono solo alcuni dei casi di cui si è avuta notizia. Molti altri sono stati taciuti, perché le società coinvolte non gradiscono pubblicizzare le violazioni subite dai loro apparecchi (magari da parte di qualche intraprendente sedicenne, con un computer da quattro soldi!).

In altri casi, le società non si sono neppure accorte delle violazioni, nonostante i "clandestini dei computer" più smaliziati lascino messaggi come "prova a prendermi?" o scritte del tipo "Il Violatore di Sistemi" o "Capitan Zap".

Le tecniche usate sono di diverso genere. Il potenziale violatore deve possedere un home computer, un modem e molta fantasia. Il primo ostacolo è scoprire il numero telefoni-

ILLECITA DISTRIBUZIONE

Sebbene la Pepsi-Cola nord-americana lo neghi, si dice che questa multinazionale sia stata vittima, pochi anni fa, di un clamoroso caso di "violazione". Qualcuno, dagli Stati Uniti, è riuscito ad accedere a computer che la società possiede in Canada. Pare che i violatori abbiano introdotto sostanziose ordinazioni di Pepsi-Cola, accreditandosi (come "provvigione") cospicue somme su conti bancari intestati a nominativi di comodo



IAN McKINNELL

co di un computer. I numeri delle reti pubbliche, come la Telecom Gold in Gran Bretagna o la Source negli Stati Uniti, sono ovviamente noti a tutti. Per le reti private la cosa è più complessa, ma conoscendo la collocazione del computer, il metodo è quello descritto in *War Games*: programmare il micro per chiamare tutti i numeri di quella località. Se risponde un computer, riconoscibile dal caratteristico segnale, il micro registra il numero telefonico chiamato, se invece risponde una persona, la comunicazione viene interrotta per passare al numero successivo. Con un modem a chiamata automatica l'intera operazione può svolgersi senza che si intervenga di persona.

A collegamento effettuato, il computer chiamato chiede sempre una parola d'accesso. Su alcune reti, battendo le parole "GUESS" o "NEWUSER" oppure più semplicemente il tasto RETURN, si ottiene l'accesso, ma limitato ad alcune parti del sistema. Ovviamente, il violatore incallito tenta di scoprire la parola d'accesso. Questa operazione, talvolta, risulta abbastanza facile, perché molti utenti usano, ingenuamente, parole piuttosto ovvie quali "SECRET" o "PASSWORD", mentre altri adottano una sequenza numerica facile da ricordare, come ad esempio la loro data di nascita: "090560". Alcuni computer sono molto pazienti e consentono vari tentativi, prima di interrompere la comunicazione. Ma in quest'ultimo caso, si può chiamare e riprovare senza che il computer centrale sospetti niente.

Dopo aver forzato il sistema, molti violatori si limitano a curiosare tra gli archivi, a cercare programmi di gioco (se ve ne sono) o a "chiamare" altri violatori entrati anch'essi nel sistema. I più violenti cancellano i file, lasciano messaggi osceni e cercano di distruggere l'intero sistema, azione che può comportare ingentissimi danni agli utenti legali.

A rendere più vulnerabili i sistemi, sono spesso i programmatori, che introducono tra i programmi delle unità centrali di elaborazione, delle "scorciatoie di sistema", che consentono, in situazioni di emergenza, di saltare tutte le misure di protezione e di arrivare rapidamente ai programmi. Molte volte la presenza di queste scorciatoie non viene comunicata neppure a chi lavora quotidianamente sull'apparecchio.

Molti dei casi descritti si riferiscono a computer universitari, poiché tali apparecchi, oltre a prevedere un accesso per chiamate esterne, adottano abitualmente una politica di libero accesso al sistema, necessaria per far fronte all'ingente carico di chiamate giornaliere. Sono, quindi, sistemi estremamente vulnerabili agli attacchi dei violatori, che possono poi "saltare" da un computer all'altro comportandosi come utenti legali. Uno studente del campus "San Jose State" ha introdotto nel computer dell'università un programma di chiamata "Talk", che permetteva agli studenti di parlare con tutti gli altri campus della California State University. È poi riuscito ad andare oltre l'ambito locale, collegandosi a computer situati in Svezia, Iran e Cina, oltre che in tutti gli Stati Uniti.

Ciò che spinge questi giovani a compiere le loro imprese è soprattutto l'eccitazione di "violare il sistema".

Alcuni seguono un proprio codice di comportamento, rifiutandosi di cancellare archivi o di lasciare scritte oscene, ma l'uso indebito e gratuito del tempo di un computer è pur sempre un reato!

Ben altra cosa sono i "furti computerizzati", commessi ai danni delle banche, spesso da impiegati disonesti che trasferiscono denaro su conti fasulli, oppure da violatori del sistema protettivo del servizio "Bancomat", come accaduto anche in Italia.

Ovviamente, le banche e le società cercano di dare la minor pubblicità ai furti perpetrati nei loro confronti, e quindi è molto difficile fare una stima esatta delle somme sottratte.

PROVA E RIPROVA...



IAN McKINNELL

I metodi usati dai violatori prevedono molte prove e molti errori e solo raramente hanno successo. Anche se un violatore riesce a mettersi in contatto con un computer, deve poi rispondere ad una serie di domande come queste per poter entrare nel sistema:

```
CONNECT 0X001001 14.32
    12/7/84
>
USER?
>HELP
USER?
>£%OFF
USER?
>UK001
PASSWORD?
>GUEST
PASSWORD?
>NEWUSER
PASSWORD?
>QWERTY
LOGOFF 15.13 CONNECT
    TIME = 0.13 MINS
```

Tasto Return: richiesta d'identificazione (ID)
Nessun aiuto dal sistema
1° tentativo di azzeccare la ID
ID non valida
2° tentativo
ID accettata: richiesta della parola d'accesso
1° tentativo
Non accettato: 2ª richiesta
2° tentativo
Non accettato: 3ª richiesta
3° ed ultimo tentativo. Il collegamento si interrompe dopo che l'utente fallisce il 3° tentativo per la parola d'accesso

In alcuni casi, dopo una lunga ricerca o per un colpo di fortuna, una persona può trovare una parola d'accesso ad alta priorità ed entrare nel sistema. Il seguente dialogo immaginario descrive come un utente "clandestino" possa carpire informazioni confidenziali che riguardano un utente legale:

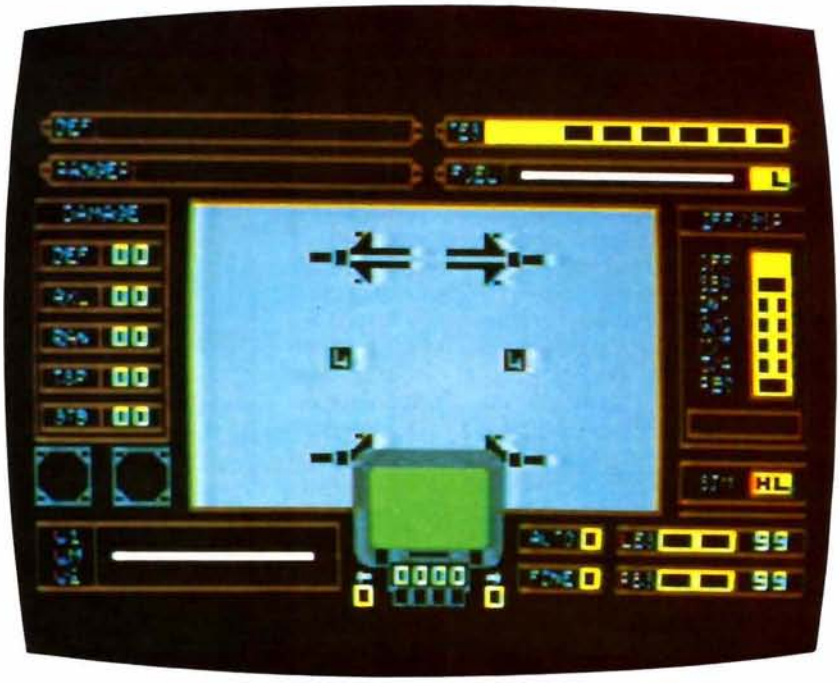
```
CONNECT BYF998
    15.14.02 12/07/84
>
USER?
>UK001
PASSWORD?
>SYSOP
LOGON 15.15.07 12/07/84
HOST: BYF998/SPYLOM
USER: UK001
SERIAL NO: ZA180-7
PRIORITY: SUPERUSER
STATUS: ACTIVE
YOU ARE SYSOP
7 USER(S)
APP01 APP02 BYF7 BTY04
    BZX88 BZX02 SYSOP
>REMOVE ARP01
USER5(S) ARP01
    DISCONNECTED
>WHO IS BTY04
BTY04     CAREY DIMMIT,
    BTY LOPP, 742
    SILICON DV
HERTS 87662-893164
```

Tasto Return
ID valida: richiesta identif.
1° tentativo: operatore di sistema (SYSOP)
Numero di serie del software
Libero accesso a tutti i file
Lo stato di un utente può essere "non attivo" se non accede regolarmente al sistema
Conferma dell'identità
Lista di tutti gli utenti
Comando riservato all'operatore di sistema
Eliminazione (illecita) di un utente autorizzato

Software in vetrina

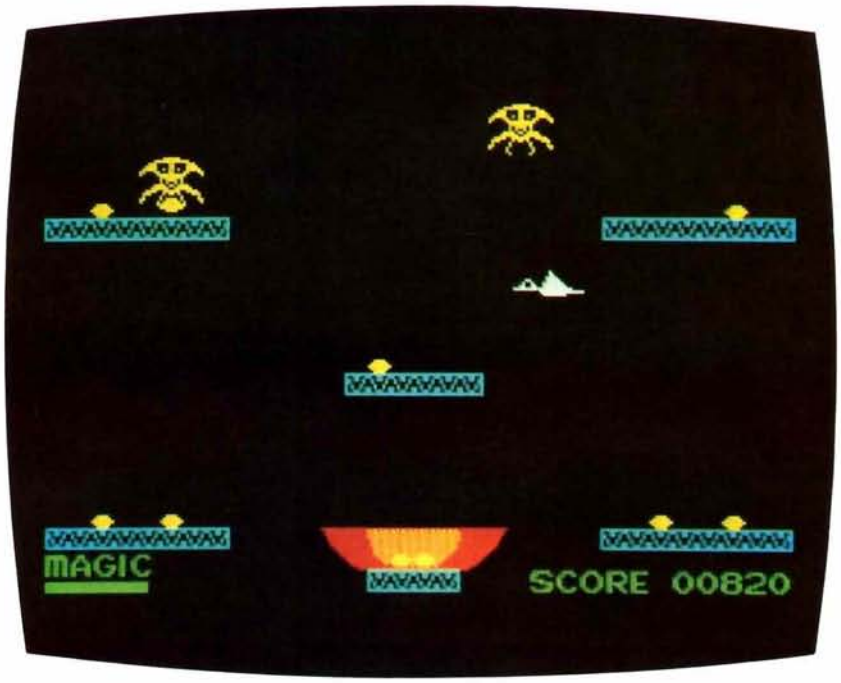
Una selezione tra i numerosi programmi disponibili per i computer più diffusi

Per gli appassionati di volo (simulato!) viene presentato un programma dalle caratteristiche eccezionali, da far dimenticare di essere comodamente seduti di fronte a un micro. Il secondo programma è di un genere più classico, appartenente al filone dei giochi "arcade".



Interdictor Pilot per il Commodore 64

Non si tratta di un semplice simulatore di volo, ma di un gioco estremamente avvincente, scritto da un vero esperto del settore: un pilota della RAF (sì, proprio l'aviazione di Sua Maestà Britannica!).
Sullo schermo è mostrato un complesso pannello di controllo, il cui elemento più importante è forse il "trasponder", ossia lo strumento che trasmette e riceve i codici d'identificazione dei vari oggetti volanti intercettati durante il volo. L'obiettivo del gioco è quello di abbattere quanti più aerei nemici possibile.
Sono ammessi stratagemmi di ogni sorta (anche falsare il proprio codice d'identificazione, per farsi credere un alleato del nemico!), perché la superiorità delle armi avversarie è pericolosissima. Il gioco, prodotto dalla Supersoft, è talmente complesso, che richiede un manuale di ben 50 pagine.



Beaky and the Egg Snatchers per lo Spectrum

Beaky, una femmina di Condor Guerriero, proveniente da Andromeda, ha deciso di deporre le proprie uova sul pianeta Crackit. Una scelta malaugurata, poiché il pianeta è abitato da indigeni ghiottissimi di uova, cosicché il compito del giocatore consiste nell'aiutare Beaky a recuperare le proprie uova, a covarle e ad allevare i piccoli.
La grafica è di ottima qualità ed i movimenti sono ottenuti ricorrendo a numerosi "quadri", in rapida sequenza. Senza dubbio, si tratta di uno spunto piuttosto originale, un po' diverso dai soliti giochi "arcade".

L'Einstein

Anatomia di un computer a diffusione europea: tecnologie a confronto

Il prezzo dell'Einstein fa subito pensare che questo home computer punti essenzialmente ad applicazioni "serie". Ovviamente, può anche essere usato con programmi di gioco ma, in questo caso, ci sono altri apparecchi con caratteristiche simili, dal costo nettamente inferiore. Il suo più accanito concorrente, in prezzo e prestazioni, è il Micro BBC ma, rispetto a questa macchina della Acorn dotata di una memoria RAM di 32 Kbyte, l'Einstein può vantare una capacità di memoria di ben 80 Kbyte.

L'unità a dischetti è inserita in un alloggiamento situato subito al di sopra della tastiera. L'involucro dell'apparecchio è così robusto da poter sostenere il peso del monitor o di un apparecchio televisivo, occupando così uno spazio molto ridotto sulla scrivania di lavoro. La presenza dell'unità a dischetti incorporata accresce notevolmente la disponibilità di software dell'Einstein. Anche altri apparecchi della sua categoria, come il Commodore 64, possono servirsi di dischetti ma il fatto che siano stati progettati per lavorare con registratori a cassette, ha ovviamente portato a privilegiare lo sviluppo di software su cassetta. Questo impedisce a chi ha acquistato un'unità a dischetti, di sfruttare in modo adeguato le maggiori capacità di archiviazione di cui dispone. Nel caso dell'Einstein, al contrario, tutto il software viene prodotto direttamente su dischetti. L'unità a dischetti permette il caricamento rapido ed affidabile di programmi e dati e consente l'impiego di file ad accesso casuale (i computer basati su cassette usano esclusivamente l'accesso sequenziale). I dischetti a tre pollici dell'Einstein hanno una capacità di 190 Kbyte per faccia, ma il sistema prevede l'uso di un solo lato alla volta. L'apparecchio possiede un altro alloggiamento interno, per una seconda unità a dischetti, e un'interfaccia sul retro per altre due.

L'Einstein usa un proprio sistema operativo per dischi (DOS) che ha molti punti di contatto col CP/M standard adottato su molti apparecchi gestionali e la Tatung spera che le Software House si affrettino a modificare i programmi CP/M per poterli utilizzare sul proprio apparecchio. Tanto il sistema operativo, quanto il BASIC Einstein, non sono residenti in ROM, ma vanno letti da dischetto, ogni volta che si riaccende il computer. Questa procedura offre due vantaggi: poiché il BASIC viene caricato solo se necessario, eventuali altri linguaggi di programmazione possono disporre di un maggior spazio RAM. Inoltre, è possibile aggiornare il sistema operativo o il BASIC, semplicemente acquistando un dischetto che contiene versioni più recenti. In futuro, la Tatung dovrebbe corredare l'apparecchio di un dischetto contenente anche il linguaggio DR LOGO, assieme al relativo manuale LOGO (quest'ultimo a pagamento, però!). Dopo il caricamento del BASIC, all'Einstein rimangono ancora 43 Kbyte di RAM per l'utente, molto più di qualsiasi altro apparecchio della stessa categoria. Ciò è possibile perché l'Einstein usa 16 Kbyte di memoria RAM a parte, per la gestione del video.

Il BASIC Einstein sembra essere una "miscela" del BASIC BBC e dell'Extended BASIC della Microsoft (come accade spesso nei computer giapponesi, aderenti allo standard MSX). Esistono comandi per rinumerare i programmi e per generare automaticamente i numeri di linea, che facilitano le operazioni di preparazione dei programmi. Lo "screen editor" permette di apportare qualsiasi modifica ai testi visualizzati sullo schermo. I comandi grafici consentono di tracciare linee, circonferenze ed ellissi e di colorare in modo omogeneo le figure visualizzate. Lo schermo grafico



CHRIS STEVENS

Un sistema... serio

L'Einstein, prodotto dalla Tatung (la "Decca" di molti anni fa), è particolarmente indicato per applicazioni non ricreative e possiede un'unità a dischetti incorporata. L'aspetto è abbastanza elegante e sobrio

ha una risoluzione massima di 256 × 192 pixel ed offre una "tavolozza" di 15 colori, utilizzabili due alla volta in ciascuna riga di 8 pixel: questa limitazione è dovuta alle caratteristiche del chip grafico usato. L'utente può definire fino a 32 sprite, i cui comandi di controllo sono incorporati nel BASIC; ciò permette di scrivere emozionanti programmi di gioco con un rapido movimento. La ROM contiene un programma "monitor", che facilita la programmazione in codice macchina.

Il chip grafico limita a 40 caratteri la larghezza massima dello schermo ma, tra breve, sarà disponibile un chip alternativo, che porterà il video ad una larghezza di 80 caratteri, necessaria per utilizzare sull'Einstein molti programmi adatti al CP/M. Lo schermo a 80 colonne è monocromatico, ma la Tatung ne sta progettando una versione a colori.

La generazione di suoni, sull'Einstein è piuttosto buona: i segnali vengono inviati ad un grosso altoparlante, posto sopra la tastiera, mentre il regolatore di volume si trova sul fianco destro dell'apparecchio. I comandi BASIC per la generazione dei suoni sono esaurienti e facili da usare.

La tastiera è dotata di 8 tasti di funzione, programmabili, ai quali possono essere abbinati comandi o istruzioni di uso frequente. Sopra di essi c'è una targhetta di plastica trasparente, simile a quella del BBC, sulla quale si possono applicare etichette relative alle funzioni. La tastiera è ben dimensionata, ma possiede soltanto due tasti per il cursore (destra/sinistra, alto/basso) invece dei quattro abituali. Per ottenere i restanti movimenti del cursore, i due tasti devono essere usati assieme al tasto Shift. Il set dei caratteri grafici (riportato sui tasti) viene abilitato premendo il tasto per la grafica (Graph), ma il suo uso è soggetto a molte limitazioni.

Solo il Micro BBC può vantare un numero di interfacce pari a quello dell'Einstein: una porta Centronics standard per stampante; una presa RS232 per modem, stampanti o altre periferiche; una porta per monitor a colori RGB; un'uscita per apparecchi televisivi ed una coppia di prese per joystick. Queste possono essere usate anche per altri scopi perché includono la conversione analogico/digitale e possono quindi misurare voltaggi elettrici. Le due prese possiedono 4 canali e vengono integrate da una porta utente a 8 bit, che può trasmettere e ricevere segnali da altre apparecchiature elettriche. La combinazione porta utente e porte A/D rende l'Einstein uno strumento ideale per funzioni di controllo in robotica ed in altre applicazioni scientifiche.

È prevedibile che la porta multifunzionale "Pipe" (simile alla "Tube" del Micro BBC) consenta presto l'impiego di altre espansioni. Già oggi uno zoccolo ROM, interno all'apparecchio, permette di espandere fino a 32 Kbyte gli 8 Kbyte della ROM standard.

L'Einstein viene prodotto in Gran Bretagna e rappresenta un ottimo investimento tuttavia; il prezzo è troppo elevato per l'utente medio di home computer. Un secondo dato negativo è una scarsa disponibilità di software.



Il monitor
Il monitor Tatung accetta segnali RGB e YUV (il sistema di trasmissione adottato dalla Tatung). Poiché l'uscita YUV dell'Einstein possiede una linea video composita, questo computer può essere collegato a qualsiasi tipo di monitor



Unità a dischetti Hitachi
Le unità a dischetti da 3 pollici della Hitachi si vanno sempre più diffondendo sui micro. I dischetti hanno una capacità di 190 Kbyte, ma per usare la seconda facciata, occorre voltarli a mano



Interfacciamento dell'Einstein
L'Einstein è dotato di numerose interfacce, tra le quali la "Pipe", somigliante al "Tube" del Micro BBC. C'è anche un'interfaccia per il collegamento di disk drive aggiuntivi



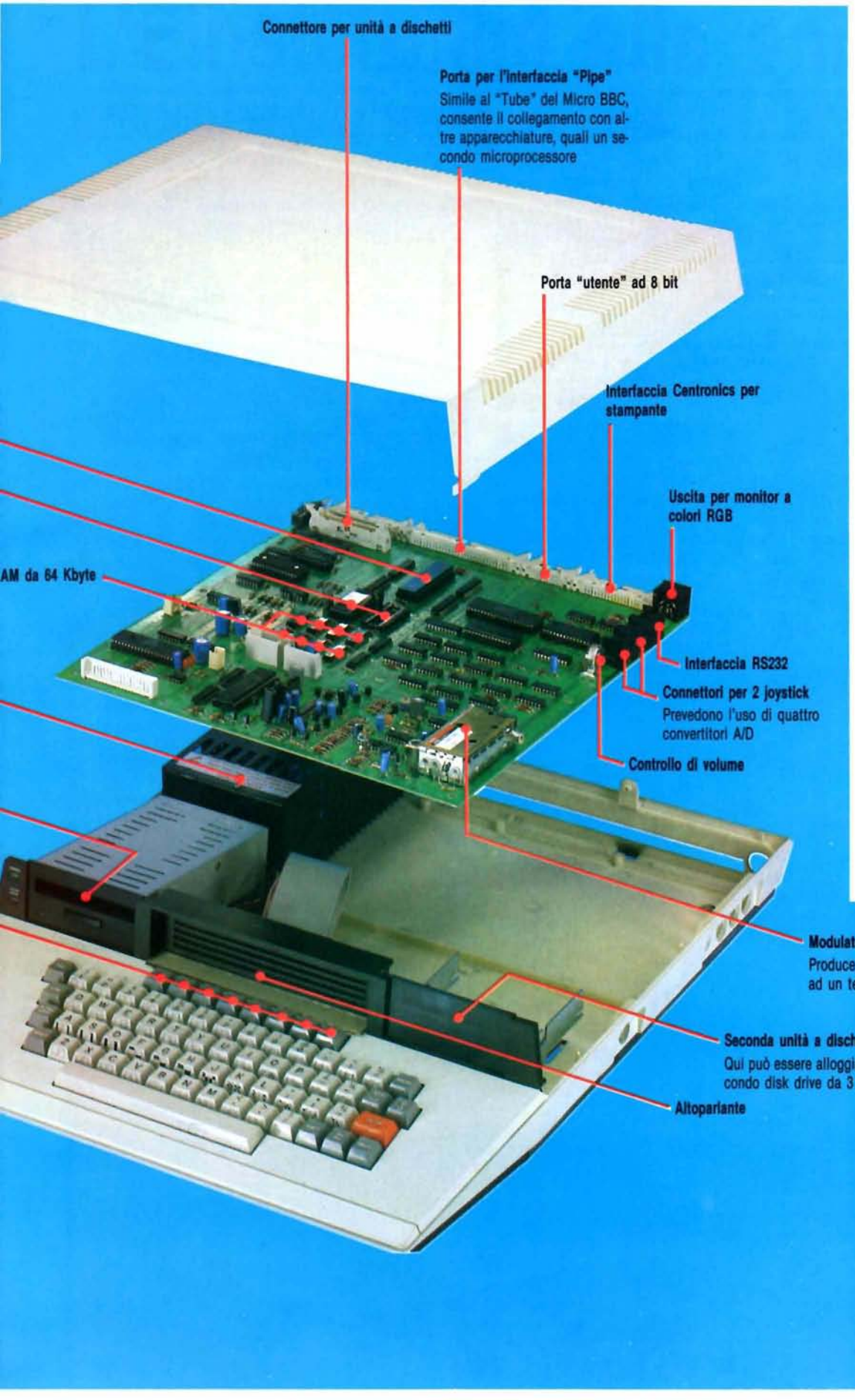
Microprocessore Z80

8 Kbyte di ROM
Questo chip contiene il programma monitor per codice macchina. Accanto, c'è uno zoccolo libero, utilizzabile per l'espansione della ROM

Alimentatore a commutazione
Questo speciale tipo di alimentatore consuma meno di quelli tradizionali ed è meno ingombrante

Unità a dischetti
Nel formato da 3 pollici Hitachi, a singola faccia, con capacità di 190 Kbyte

Tasti di funzione
Sono programmabili da parte dell'utente



EINSTEIN

DIMENSIONI

mm. 510 x 430 x 105

CPU

Z80

MEMORIA

80 Kbyte di RAM (di cui 16 Kbyte dedicati alla gestione del video), 8 Kbyte di ROM (espandibile a 32 Kbyte)

VIDEO

Modo testuale: 24 righe per 32 o 40 colonne. Modo grafico: 256 x 192 pixel, con 15 colori

INTERFACCE

Parallela Centronics, seriale RS232, due porte joystick (convertitori A/D), uscita monitor RGB, "Pipe", porta "utente" e per unità a dischetti esterna

LINGUAGGI DISPONIBILI

BASIC e DR LOGO (forniti), PASCAL e FORTH (opzionali)

TASTIERA

Tipo macchina da scrivere, con 67 tasti, di cui 8 di funzione (programmabili). Formato QWERTY

DOCUMENTAZIONE

I manuali a corredo sono tre (quello del BASIC è un po' scarno)

PRO

Unità a dischetti incorporata, numerose interfacce, ampia memoria RAM, buon BASIC, grafica a sprite

CONTRO

Scarsità di software e prezzo superiore alla media

CHRIS STEVENS

Fine alle mine

I ritocchi finali al programma di gioco per i computer Acorn, fin qui sviluppato

Il MODE 7 grafico del Micro BBC, detto anche "teletext", possiede alcune funzioni assenti negli altri MODE, che permettono di visualizzare informazioni trasmesse da sistemi esterni come il Micronet. Inoltre, le funzioni grafiche offerte dal MODE 7 consentono di creare; con poche e semplici istruzioni, interessanti immagini di testi e titoli; nel nostro programma lo impieghiamo per produrre l'ultima immagine del gioco.
Introducendo i codici di controllo CHR$ nelle istruzioni PRINT, possiamo modificare i colori del testo dello sfondo, creare scritte "lampeggianti" e generare caratteri a doppia altezza. Inoltre possiamo usare la funzione TAB per posizionare i testi sullo schermo, che ha un formato di 40 × 25 caratteri. Il MODE 7 dispone di sette colori selezionabili mediante i seguenti codici:

129	rosso
130	verde
131	giallo
132	blu
133	violetto
134	azzurro
135	bianco

Nel MODE 7, i testi vengono visualizzati in bianco su uno sfondo nero, ma il loro colore può essere modificato inserendo un codice di controllo in qualsiasi punto dell'istruzione PRINT. Ad esempio, le tre parole della frase visualizzata dalla seguente linea d'istruzioni vengono stampate, rispettivamente, in rosso, bianco e blu:

```
PRINT CHR$(129)"GIOCHI";CHR$(135);"SENZA";
      CHR$(132);"FRONTIERE"
```

Si rammenti, però, che il sistema ripristina il colore di partenza, ossia il bianco, ad ogni nuova linea visualizzata. Pertanto, se si vuol continuare nello stesso colore, ogni nuova linea deve contenere gli appropriati codici di controllo.
Dopo aver definito il colore del testo, si può selezionare il colore dello sfondo tramite l'istruzione CHR$(157), seguita dal codice del colore desiderato. Per ottenere lettere blu su sfondo bianco, è necessaria la seguente sequenza di codici di controllo:

```
PRINT CHR$(132);CHR$(157);CHR$(135);"IN MARINA"
```

Il primo codice indica il colore del testo, il secondo ed il terzo definiscono quello dello sfondo. L'uno e l'altro possono essere resi lampeggianti, inserendo il codice CHR$(136) immediatamente prima del codice di colore. Per eliminare questo effetto si usa CHR$(137). In questo esempio vengono fatte lampeggiare le lettere blu della scritta:

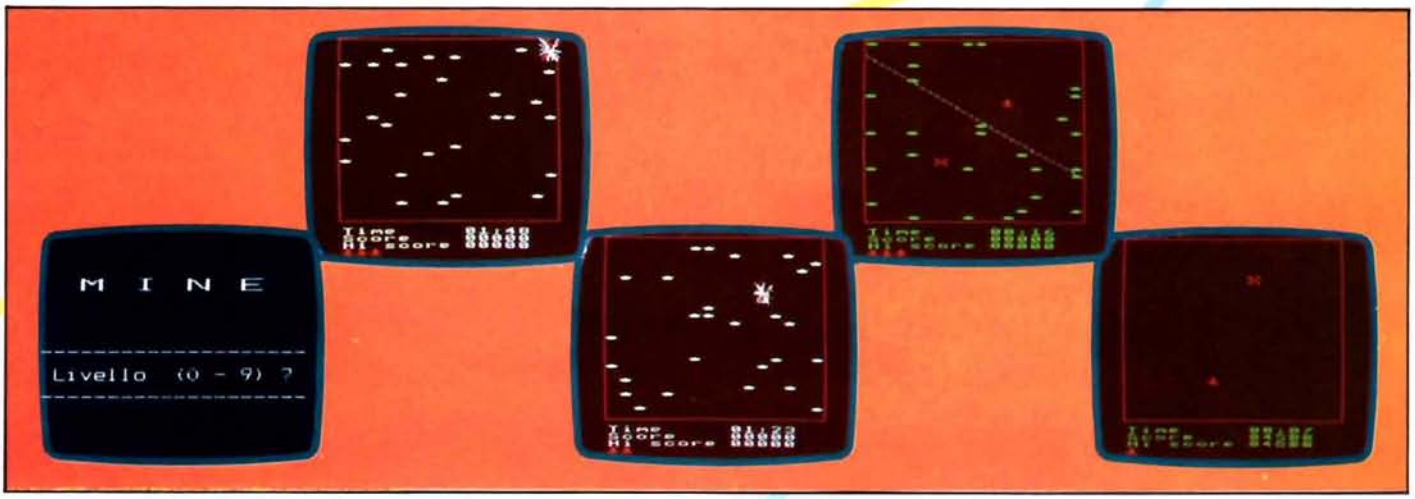
```
PRINT CHR$(136);CHR$(132);CHR$(157)
      CHR$(135);"IN MARINA"
```

Un interessante aspetto del MODE 7 è la capacità di produrre caratteri a doppia altezza. L'istruzione da usare è CHR$(141), ma occorre però visualizzare due volte la stessa linea. Ai caratteri a doppia altezza si possono abbinare tutti gli effetti grafici visti finora. Quindi, per generare caratteri blu, lampeggianti ed a doppia altezza su sfondo bianco, possiamo usare le seguenti linee:

```
10 MODE 7
20 PRINT CHR$(141);CHR$(136);CHR$(132);
   CHR$(157);CHR$(135);"IN MARINA"
30 PRINT CHR$(141);CHR$(136);CHR$(132);
   CHR$(157);CHR$(135);"IN MARINA"
```

Sequenze del gioco

Queste sequenze, tratte da alcune fasi del gioco, mostrano le mine, l'assistente, il colpo sparato dal cecchino, un'esplosione e le immagini iniziali e finali del gioco



Il "Lords of Midnight"

Un'insolita combinazione tra un gioco di strategia ed uno d'avventura

I "war games" per computer, che derivano dai giochi da tavolo come *Blitzkrieg* e *Diplomacy*, sono notevolmente diversi dai giochi "adventure", la maggior parte dei quali si richiamano, anche se vagamente, al classico *Dungeon* e *Dragons*. Un war game richiede l'applicazione di piani tattici e strategici, che tengano conto di tutta una serie di elementi, ossia gli eserciti, i rifornimenti, le armi ecc., visualizzati su una mappa del campo di battaglia. I giochi stile "adventure" puntano, invece, sulla sorpresa e sull'ingegnosità: il giocatore deve risolvere una serie di problemi, via via che il gioco si sviluppa. In *Lords of Midnight*, scritto per lo Spectrum con 48 Kbyte, la Beyond Software ha cercato di fondere i due tipi di gioco per crearne un terzo del tutto nuovo.

Un opuscolo di una trentina di pagine contiene una mappa dello scenario di gioco (la Land of Midnight); il giocatore interpreta il ruolo di Luxor, principe della Luna (Moonprince) e Lord dei Liberi (Lord of Free). Luxor possiede l'Anello della Luna (Moon Ring), un talismano che gli permette di guidare, osservando attraverso i loro occhi, i suoi quattro compagni d'armi o qualsiasi altro Lord dei Liberi da lui reclutato. I Liberi controllano la maggior parte dei territori del Sud, da difendere dagli attacchi lanciati da Doomdark, il Re degli Stregoni di Mezzanotte (Witchking of Midnight). Gli armati di Doomdark sono controllati dal computer e la loro roccaforte si trova nel Nord: è da questa che si muovono per conquistare i territori meridionali.

Doomdark possiede l'Ice Crown, una corona magica che lancia Spire di Ghiaccio contro i suoi nemici, sottraendo energia ai Lord dei Liberi, quando si avvicinano ai confini settentrionali. Tuttavia Morkin, uno dei compagni di Luxor, è immune da questa limitazione ed ha il compito di muoversi furtivamente verso il Nord per distruggere la corona, mentre gli altri armati effettuano attacchi diversivi, in campo aperto, contro le forze di Doomdark.

I giochi di guerra tradizionali si svolgono su una mappa che permette ai giocatori di conoscere la posizione delle forze schierate contro di loro. In *Lords of Midnight*, invece, il giocatore non può prendere visione della mappa aggiornata del campo di battaglia, perché questa è conservata nella memoria dello Spectrum. Lo svolgimento dell'azione viene visto attraverso lo sguardo di un Lord dei Liberi, che può volgersi in una qualsiasi delle otto direzioni selezionabili da tastiera. Quindi, se c'è un'armata nemica appostata dietro un gruppo di colline che si trovano davanti al Lord, il giocatore non potrà vederla finché non aggira le colline e guarda nella giusta direzione. Questa impostazione propone una nuova dimensione ai giochi di guerra.

La caratteristica più interessante del *Lords of Midnight* è la sua grafica curatissima. La Beyond Software afferma che il gioco è in grado di visualizzare fino a 32.000 scenari e che il giocatore può imbattersi in 32 personaggi diversi. Naturalmente, la memoria dello Spectrum non è in grado di contenere questo immenso numero di immagini, perciò la Beyond ha sviluppato una tecnica chiamata "landscaping", che permette di costruire scenari differenti partendo da 15 diverse figure base, ciascuna delle quali è disponibile in quattro dimensioni o prospettive. Queste figure vengono combinate per creare più scene di gioco: colline, foreste, montagne, armate, villaggi, rocche e così via. La memoria dello Spectrum contiene una mappa, composta da 4000 locazioni, che indica la posizione dei vari elementi dell'immagine. Questa informazione consente al computer di calcolare la prospettiva da un qualsiasi punto.

Lords of Midnight è un gioco concepito e presentato con accuratezza. In armonia con lo spirito del gioco, la Beyond ha ridefinito il set di caratteri dello Spectrum, per dare un aspetto "gotico" ai messaggi. A differenza di molti giochi, nei quali si tenta di mascherare la povertà del disegno con una ricca grafica, questo avvince l'attenzione del giocatore con sempre nuove sorprese.

Lords of Midnight: per lo Spectrum a 48 Kbyte
Editori: Beyond Software, (GB)
Autore: Mike Singleton
Joystick: Non richiesto
Formato: Su cassetta



LIZ HEANEY





Effetto gotico

La Beyond Software afferma che il gioco può visualizzare 32.000 scenari diversi, anche se è difficile credere che qualcuno li abbia veramente contati! Interessante è l'uso delle lettere "gotiche" per i messaggi di testo che la Beyond ha ottenuto ridefinendo il set di caratteri dello Spectrum

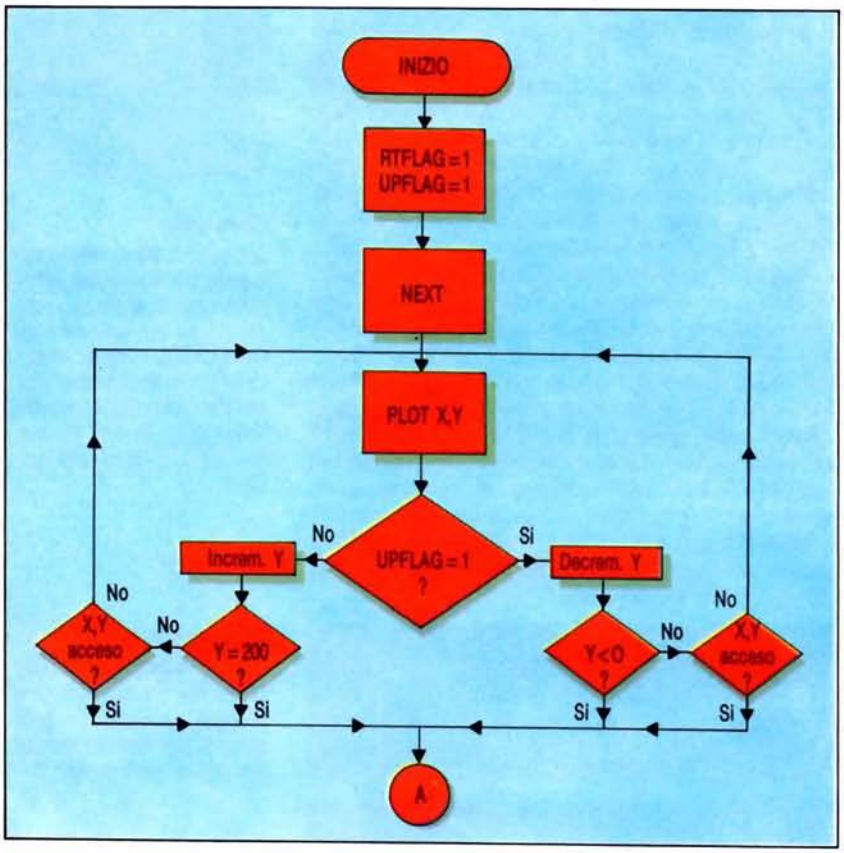
Mascherina dei comandi

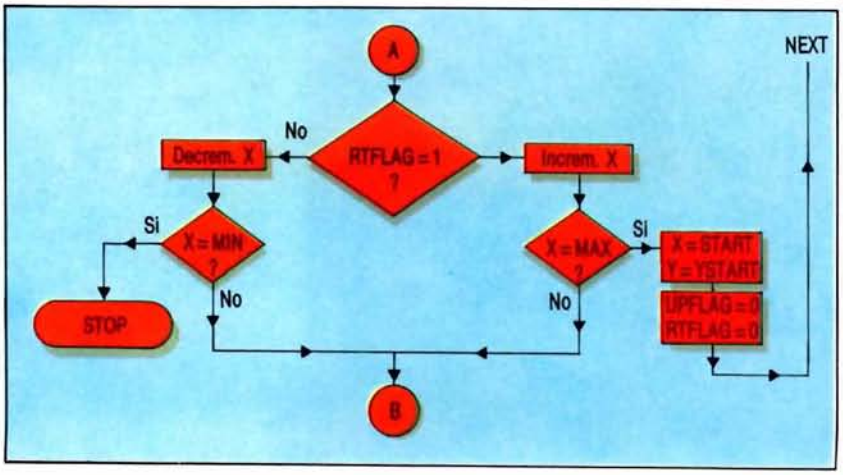
"Lords of Midnight" è corredato di una speciale mascherina, da applicare sulla tastiera dello Spectrum, che facilita l'impostazione dei comandi di gioco

Colorare col Commodore

Una routine grafica che permette di colorare figure sullo schermo

Esistono molti modi per formulare un algoritmo che colori figure sullo schermo ma, pur sembrando un lavoro semplice, in realtà si dimostra più complicato del previsto. Infatti, la colorazione di figure dai contorni non regolari presenta più di un problema; è possibile, infatti, creare una routine capace di risolvere alcune di queste difficoltà, ma non tutte, perché il computer non riesce a "capire" come è fatta una figura chiusa.



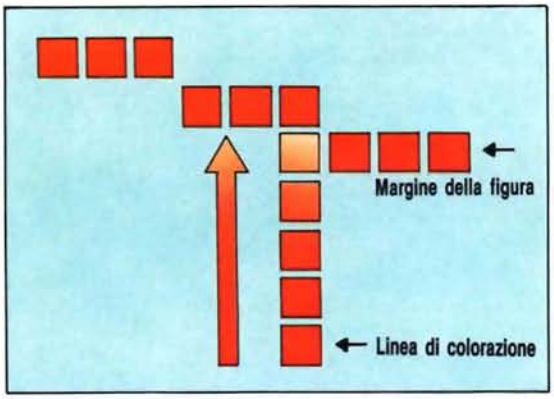


Col metodo adottato nella nostra routine, la colorazione inizia da un punto interno alla stessa figura, definito dal "disegnatore". Inizialmente la routine procede verso l'alto tracciando punti colorati fino a raggiungere il margine della figura; quindi si sposta di un pixel a destra e riprende a tracciare in direzione contraria (verso il basso) fino ad incontrare un altro margine. Qui si sposta nuovamente a destra di un pixel e riprende a muoversi verso l'alto. La procedura prosegue fino a colorare completamente tutta la parte destra della figura. Un procedimento identico si ripete poi per la parte sinistra, sempre iniziando dal punto di partenza definito dall'utente.

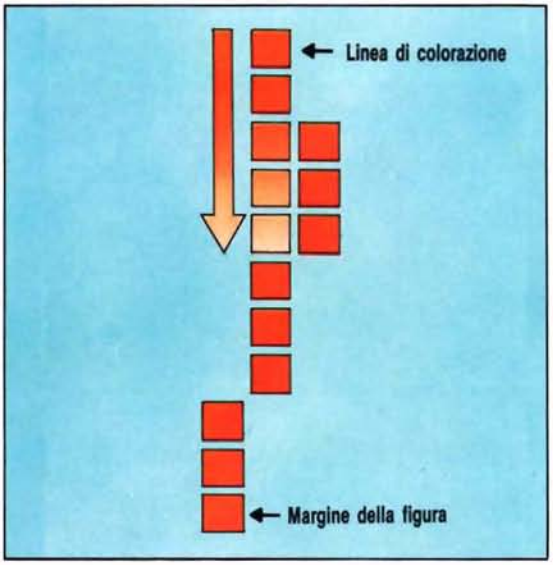
La prima parte della routine è abbastanza chiara. I due flag UPFLAG e RTFLAG vengono usati per indicare la direzione seguita dalla routine di colorazione in ogni stadio del programma. La prima sezione del diagramma di flusso descrive le fasi di tracciamento, incremento e verifica. Il ciclo principale di questa sezione incrementa o decrementa il valore della coordinata Y, a seconda dello stato di UPFLAG. Per prima cosa, la routine controlla di non aver raggiunto il margine dello schermo, verificando che il pixel successivo non sia già acceso, poi passa a colorare un nuovo punto. Quando si imbatte nel margine dello schermo o in un pixel acceso, il flusso di controllo salta alla fase successiva. La routine si muove verso destra o verso sinistra a seconda dello stato di RTFLAG. Poiché l'individuazione dei margini destro e sinistro della figura sarebbe molto complessa, la routine chiede all'utente di indicare i valori minimo e massimo per la coordinata X. Questa scelta permette inoltre di colorare una figura a strisce alternate.

Il secondo diagramma di flusso mostra che, se X viene incrementato (ossia il movimento è verso destra) fino a raggiungere il valore massimo, allora le coordinate X e Y vengono riportate al loro valore iniziale ed entrambi i flag di direzione vengono azzerati, per prepararli alla colorazione della parte sinistra della figura; quando X viene decrementato fino a raggiungere il valore di XMIN, la routine termina. Ne consegue, che la routine continua a tracciare linee finché non vengono raggiunti i valori massimo o minimo di X.

L'algoritmo capace di lavorare con linee di vario tipo, più o meno inclinate oppure più o meno ravvicinate, sembra piuttosto complicato, ma i principi su cui poggia sono relativamente semplici. Vediamo cosa accade quando la routine, muovendosi verso l'alto, incontra una linea leggermente inclinata.

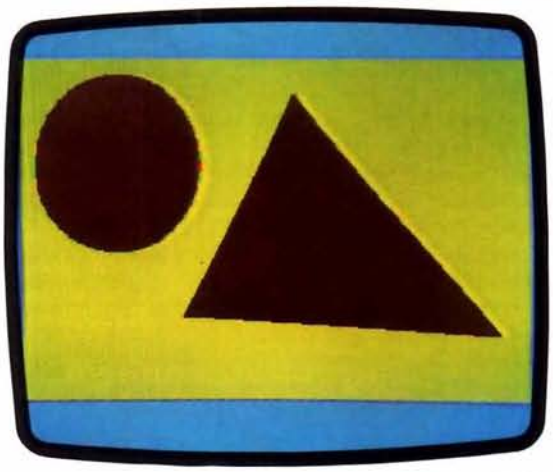


Per prima cosa l'algoritmo provvede a tornare indietro di un pixel. Se la colorazione sta procedendo verso destra, la routine deve compiere un ulteriore spostamento, affinché la linea successiva di colorazione possa partire da un pixel ancora spento. Le cose cambiano se la routine, durante il movimento verso il basso, incontra una linea notevolmente inclinata.



Se la colorazione procede verso destra, allora la routine deve tornare indietro di tre pixel e spostarsi di una posizione a destra per trovare un pixel spento, dal quale ripartire verso l'alto. Se invece la colorazione si sviluppa verso sinistra, la routine deve spostarsi di altri tre pixel verso il basso e di uno a sinistra, per riprendere la colorazione verso l'alto dal primo pixel spento, situato sopra la linea di margine. Per ciascuna direzione di colorazione occorrono dunque due cicli, che muovano nell'una o nell'altra direzione, fino a trovare il primo pixel spento dal quale ripartire. Si esamini attentamente il diagramma di flusso riportato qui accanto.

Il corpo principale del programma Fillsub segue il diagramma appena visto. Per favorire la comprensione del programma, le etichette citate nel listato del codice sorgente compaiono, nelle posizioni corrispondenti, sul diagramma di flusso. L'algoritmo di colorazione esegue numerose verifiche, in particolari punti, per verificare se un pixel è acceso o no. Per questa operazione il programma usa una subroutine chiamata POINT, nella quale l'indirizzo del punto è calcolato mediante le sue coordinate X, Y, con lo stesso criterio adottato nella procedura Plotsub. In questo caso, però, anziché usare una OR logica per porre un particolare bit a uno, POINT si serve di una AND logica, semplicemente per verificare se il bit in questione vale uno o zero. Se il risultato dell'operazione AND è "non zero", allora il bit vale 1 e questo risultato viene depositato in PTFLAG per una successiva verifica da parte del programma principale. Gran parte della subroutine POINT lavora in modo simile a Plotsub e chi abbia esperienza del codice macchina del 6502 può modificare Plotsub in modo da poter verificare il valore di un bit, oltre ad azzerarlo o porlo ad uno.

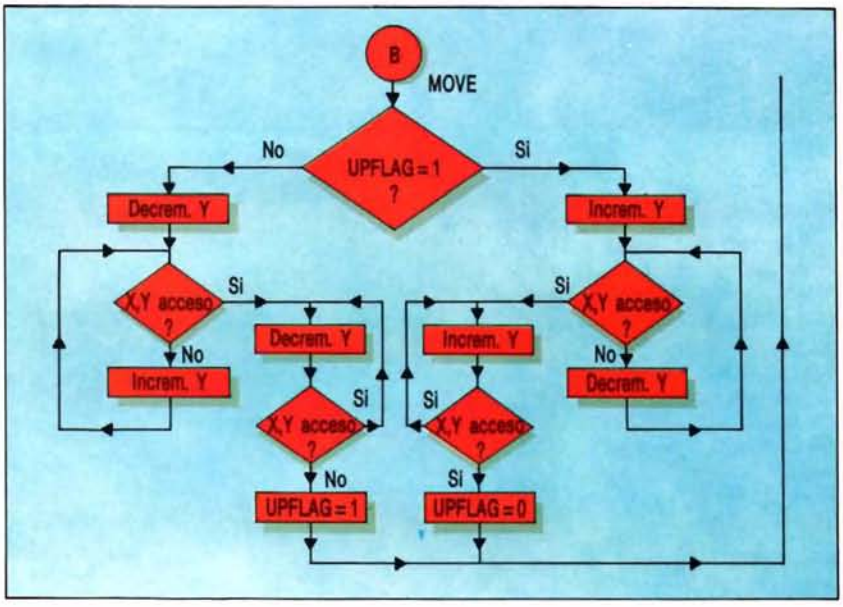


Finalmente, la grafica!

Dopo la descrizione e lo studio di numerosi programmi, siamo finalmente riusciti ad ottenere sul Commodore 64 alcune elementari funzioni grafiche in alta risoluzione previste dal BASIC standard di quasi tutti gli altri micro sul mercato. L'unica alternativa, volendo evitare un simile lavoro di programmazione, è l'acquisto di una cartuccia contenente un BASIC esteso

Osserviamo, infine, come si usa la routine Fillsub. Innanzi tutto, Fillsub deve ricevere dall'esterno un certo numero di parametri. Essi sono:

- Le coordinate del punto di partenza (che deve trovarsi all'interno della figura da colorare).
- I valori minimo e massimo della coordinata X. Quando si vogliono colorare figure con angoli molto acuti, occorre stare attenti che Fillsub non finisca fuori della figura, se raggiunge il margine di uno spazio



Diagrammi di KEVIN JONES

interno prima di aver raggiunto uno dei valori limite selezionati (massimo e minimo). Il problema si risolve specificando valori limite, leggermente più ampi del necessario.

- Si ricordi che la coordinata X del punto di partenza e i valori limite di X devono essere espressi in forma byte alto/byte basso, come nel programma dimostrativo.
- Sebbene Fillsub non sia direttamente collegato a nessuna delle altre routine sviluppate per il Commodore 64, le tre routine (Plotsub, Linesub e Circsub) vengono caricate dal programma dimostrativo, per poter disegnare le figure da colorare.

Un'ultima considerazione: la versione originale di questa routine colorava le figure procedendo in senso orizzontale e non verticale, ma si è visto che la procedura di colorazione verticale velocizza notevolmente l'esecuzione del programma.

PROGRAMMA DI CARICAMENTO PLOTSUB/II

Questa è una versione modificata della routine Plotsub pubblicata a pag. 819. La creazione di un nuovo file "oggetto" (su cassetta o su nastro), chiamato "PLOTSUB.HEX", avviene in modo analogo a quanto spiegato a pag. 819

```
 10 FORI=49408TO49408+314
 20 READA:POKEI,A:S=S+A:NEXT
 30 READCC:IFCC<>STHENPRINT"CHECKSUM ERROR"
100 DATA1,0,3,6,0,5,0,0,6,5,5,69,38,2
110 DATA72,138,72,152,72,173,0,193,240
120 DATA83,169,0,133,251,169,4,133,252
130 DATA162,3,160,0,173,2,193,145,251
140 DATA136,208,251,230,252,202,48,8
150 DATA208,244,145,251,160,231,208
160 DATA238,173,1,193,240,24,169,0,133
170 DATA251,169,32,133,252,162,32,160
180 DATA0,169,0,145,251,136,208,251
190 DATA230,252,202,208,246,173,24,208
200 DATA41,240,9,8,141,24,208,173,17
210 DATA208,9,32,141,17,208,76,125,193
220 DATA173,24,208,41,240,9,4,141,24
230 DATA208,173,17,208,41,223,141,17
240 DATA208,104,168,104,170,104,96,72
250 DATA138,72,152,72,173,4,193,141,7
260 DATA193,173,3,193,41,248,141,6,193
270 DATA173,3,193,41,7,141,8,193,173,5
280 DATA193,41,7,141,10,193,162,3,78,5
290 DATA193,202,208,250,173,5,193,141
300 DATA9,193,169,0,141,11,193,141,12
310 DATA193,162,5,173,11,193,24,109,9
320 DATA193,141,11,193,202,208,243,162
330 DATA6,14,12,193,14,11,193,144,3
340 DATA238,12,193,202,208,242,173,11
350 DATA193,24,109,6,193,141,11,193
360 DATA173,12,193,109,7,193,141,12
370 DATA193,173,11,193,24,105,0,141,11
380 DATA193,173,12,193,105,32,141,12
390 DATA193,173,11,193,24,109,10,193
400 DATA141,11,193,173,12,193,105,0
410 DATA141,12,193,173,11,193,133,251
420 DATA173,12,193,133,252,169,1,141
430 DATA13,193,56,169,7,237,8,193,240
440 DATA7,170,14,13,193,202,208,248
450 DATA160,0,177,251,13,13,193,145
460 DATA251,76,125,193
470 DATA37523:REM*CHECKSUM*
```

FILLSUB DEMO

```
 10 REM *** PROGRAMMA DIMOSTRATIVO FILLSUB ***
 15 DN=0: REM PER CASSETTA DN=1
 20 IFA=0THENA=1:LOAD"PLOTSUB.HEX",DN,1
 25 IFA=1THENA=2:LOAD"LINESUB.HEX",DN,1
 27 IFA=2THENA=3:LOAD"CIRCSUB.HEX",DN,1
 30 IFA=3THENA=4:LOAD"FILLSUB.HEX",DN,1
 40 GOSUB1000:REM ABILITA L'ALTA RISOLUZIONE
 50 REM *** TRACCIA UN TRIANGOLO ***
 60 XA=100:YA=150:XB=300:YB=160:XC=170:YC=20
 80 X1=XA:Y1=YA:X2=XB:Y2=YB:GOSUB2000
 90 X1=XC:Y1=YC:GOSUB2000
100 X2=XA:Y2=YA:GOSUB2000
102 REM *** TRACCIA UN CERCHIO ***
103 XC=60:YC=60:R=50:GOSUB4000
120 REM *** COLORA UN TRIANGOLO ***
130 XS=170:YS=130:REM PUNTI DI PARTENZA
140 MIN=100:MAX=299:REM VALORI LIMITE
150 GOSUB3000
161 REM *** COLORA UN CERCHIO ***
162 XS=60:YS=60:REM PUNTO DI PARTENZA
163 MIN=10:MAX=109
164 GOSUB3000
```

FILLSUB DEMO (CONT.)

```
 200 GETA$:IFA$=""THEN200:REM ATTENDE PRESS.TASTO
 210 POKE49408,0:SYS49422:REM PREDISPONE SCHERMO
 220 PRINTCHR$(147):REM CANCELLA LO SCHERMO
 225 PRINT"FINE DELLA ROUTINE"
 230 END
1000 REM *** ABILITA L'ALTA RISOLUZIONE ***
1010 POKE49408,1:POKE49409,1
1020 POKE49410,7
1030 SYS49422
1040 RETURN
2000 REM *** LINESUB ***
2010 MHI=INT(X1/256):MLO=X1-256*MHI
2020 NHI=INT(X2/256):NLO=X2-256*NHI
2030 POKE49920,MLO:POKE49921,MHI
2040 POKE49922,NLO:POKE49923,NHI
2050 POKE49924,Y1:POKE49925,Y2
2060 SYS49934
2070 RETURN
3000 REM *** FILLSUB ***
3010 SH=INT(XS/256):SL=XS-SH*256
3020 HAX=INT(MAX/256):LAX=MAX-256*HAX
3030 HIN=INT(MIN/256):LIN=MIN-256*HIN
3040 POKE50955,SL:POKE50956,SH
3050 POKE50957,YS
3060 POKE50958,LIN:POKE50959,HIN
3070 POKE50960,LAX:POKE50961,HAX
3080 SYS50967
3090 RETURN
4000 REM *** CIRCSUB ***
4010 CHI=INT(XXC/256):CLO=XC-256*CHI
4020 POKE50497,CLO:POKE50498,CHI
4030 POKE50499,YC:POKE50500,R
4040 SYS 50521
4060 RETURN
```

FILLSUB LOADER

```
 10 REM *** CARICATORE BASIC PER FILLSUB ***
 20 FORI=50944 TO 51375
 30 READA:POKEI,A:CC=CC+A:NEXT
 40 READA:IFCC<>A THEN PRINT"CHECKSUM ERROR":END
100 DATA11,0,6,8,0,3,6,5,136,39,16,60
110 DATA0,60,10,0,109,0,0,1,10,0,16
120 DATA173,11,199,141,20,199,173,12
130 DATA199,141,21,199,172,13,199,169
140 DATA1,141,18,199,141,19,199,140,5
150 DATA193,173,20,199,141,3,193,173
160 DATA21,199,141,4,193,32,131,193
170 DATA173,19,199,208,8,200,192,200
180 DATA240,19,76,82,199,136,192,0,144
190 DATA11,32,246,199,173,22,199,208,3
200 DATA76,46,199,173,18,199,208,31
210 DATA173,20,199,56,233,1,141,20,199
220 DATA173,21,199,233,0,141,21,199
230 DATA205,15,199,208,65,173,20,199
240 DATA205,14,199,208,57,96,173,20
250 DATA199,24,105,1,141,20,199,173,21
260 DATA199,105,0,141,21,199,205,17
270 DATA199,208,34,173,20,199,205,16
280 DATA199,208,26,173,11,199,141,20
290 DATA199,173,12,199,141,21,199,172
300 DATA13,199,169,0,141,19,199,141,18
310 DATA199,76,46,199,173,19,199,208
320 DATA28,136,32,246,199,173,22,199
330 DATA208,4,200,76,191,199,238,19
340 DATA199,136,32,246,199,173,22,199
350 DATA208,247,76,46,199,200,32,246
360 DATA199,173,22,199,208,4,136,76
370 DATA219,199,206,19,199,200,32,246
380 DATA199,173,22,199,208,247,76,46
390 DATA199,72,138,72,152,72,140,2,199
400 DATA173,20,199,141,0,199,173,21
410 DATA199,141,1,199,173,1,199,141,4
420 DATA199,173,0,199,41,248,141,3,199
430 DATA173,0,199,41,7,141,5,199,173,2
440 DATA199,41,7,141,7,199,162,3,78,2
450 DATA199,202,208,250,173,2,199,141
460 DATA6,199,169,0,141,8,199,141,9
470 DATA199,162,5,173,8,199,24,109,6
480 DATA199,141,8,199,202,208,243,162
490 DATA6,14,8,199,46,9,199,202,208
500 DATA247,173,8,199,24,109,3,199,141
510 DATA8,199,173,9,199,109,4,199,141
520 DATA9,199,173,8,199,24,105,0,141,8
530 DATA199,173,9,199,105,32,141,9,199
540 DATA173,8,199,24,109,7,199,133,251
550 DATA173,9,199,105,0,133,252,169,1
560 DATA141,10,199,56,169,7,237,5,199
570 DATA240,7,170,14,10,199,202,208
580 DATA250,160,0,177,251,45,10,199
590 DATA141,22,199,104,168,104,170,104
600 DATA96
610 DATA50785:REM *CHECKSUM*
```

Dischetto o cassetta?

La linea 15 DN=8 indica che i file oggetto (Plotsub.Hex, ecc.) devono essere caricati da dischetto. Per il caricamento da registratore si deve usare DN=1. In entrambi i casi, i file oggetto devono essere registrati nell'ordine indicato dalle linee 20-30. Se i file si trovano su cassette diverse, è necessario aggiungere le linee 22, 26 e 28, tutte con il seguente codice:

```
INPUT "CAMBIA CASSETTA & BATTI RETURN"; A$
```

Tutti questi codici di controllo occupano un discreto spazio nel programma e richiedono un certo tempo di elaborazione per essere eseguiti. Un metodo alternativo consiste nel collegare tra loro più codici all'interno di un'unica stringa, utilizzata in seguito assieme ad un'istruzione PRINT. Se, ad esempio, desideriamo visualizzare caratteri rossi su uno sfondo giallo, possiamo creare un'apposita stringa (rosso$), ed usarla all'occorrenza nelle istruzioni PRINT. Ad esempio:

```
10 MODE 7
20 rosso$ = CHR$(129) + CHR$(157) + CHR$(131)
30 PRINT rosso$;"FINE GIOCO"
40 PRINT rosso$;"PUNTEGGIO"
```

LA PROCEDURA DI FINE GIOCO

Esaminiamo ora come adoperare questi effetti nella procedura che riguarda l'immagine finale del gioco:

```
2110 DEF PROCfinegioco
2120 IF punti$>maxpunti$ THEN maxpunti$=punti$
2130 rosso$=CHR$(129)+CHR$(157)+CHR$(131)
2140 gioco$="F I N E   G I O C O "
2150 PRINTTAB(0,5) rosso$;CHR$(141);CHR$(136);TAB(12)
     ;gioco$
2160 PRINT rosso$;CHR$(141);CHR$(136);TAB(12);gioco$
2170 PRINT:PRINT rosso$;"Punteggio";TAB(30);punti$
2180 PRINT:PRINT rosso$;"Massimo";TAB(30);maxpunti$
2190 PRINT:PRINT rosso$;"Tempo";TAB(30);time$
2200 blu$=CHR$(132)+CHR$(157)+CHR$(134)
2210 go$="UN'ALTRA PARTITA S/N?"
2220 PRINT:PRINT
2230 PRINT blu$;CHR$(141);CHR$(136);TAB(5);go$
2240 PRINT blu$;CHR$(141);CHR$(136);TAB(5);go$
2250 REM ** RISPOSTA ? **
2260 *FX 15,1
2270 risposta$=INKEY$(0)
2280 IF GET$="N" THEN flagfinale=1
2290 ENDPROC
```

La linea 2120 verifica se il punteggio ottenuto nel gioco appena concluso è superiore a quello del punteggio massimo e, in caso affermativo, aggiorna il punteggio massimo. Quindi la procedura visualizza "FINE GIOCO" (linee 2130-2160), i punteggi ed il tempo (linee 2170-2190), in caratteri rossi, a doppia altezza e lampeggianti su uno sfondo giallo. Poi il programma chiede al giocatore se vuol fare un'altra partita e, se la risposta è "N", pone a 1 la variabile flagfinale.

Il MODE 7 non viene selezionato, durante la procedura di fine gioco, perché sul Micro BBC non è consentito variare il modo grafico dall'interno di una procedura. In risposta ad un simile tentativo, il sistema visualizza un messaggio d'errore (BAD MODE). Al contrario, il MODE 7 va selezionato nel breve programma principale, che richiama le varie procedure. Si tratta di aggiungere al programma principale le linee seguenti. Notare che l'intero programma di chiamata si trova ora inserito in un ciclo REPEAT...UNTIL, che verrà ripetuto finché la variabile flagfinale non sarà posta ad 1.

```
1100 REPEAT

1200 MODE7
1210 REM ** CANCELLA IL CURSORE **
1220 VDU23,1,0;0;0;0;
1230 PROCfinegioco
1240 UNTIL flagfinale=1
1250 CLS
1260 END
```

I possessori dell'Electron saranno rimasti un po' interdetti durante la trattazione del MODE 7, non disponibile sul loro apparecchio. Per ovviare a questa mancanza, è stata sviluppata una procedura alternativa, che ricorre al MODE 5 per completare l'immagine del gioco.

Nella versione per l'Electron si elimina la linea 1200 dal programma di chiamata, inserendo la seguente procedura:

```
2100 DEF PROCfinegioco
2110 IF punti$=maxpunti$ THEN maxpunti$=punti$
2120 REM ** SELEZIONA SFONDO GIALLO **
2130 VDU19,130,3,0,0,0
2140 GCOL0,130: CLG: REM COLORE SCHERMO
2150 COLOUR1: COLOUR130: REM SELEZIONA COLORI TESTO
2160 gioco$="F I N E   G I O C O "
2170 PRINTTAB(2,4);gioco$
2180 COLOUR0
2190 PRINTTAB(0,8);"Punteggio";TAB(15);punti$
2200 PRINT:PRINT"Massimo";TAB(15);maxpunti$
2210 PRINT:PRINT"Tempo";TAB(15);time$
2220 go$="UN'ALTRA PARTITA S/N?"
2230 REM ** CAMBIA COL3 IN LAMPEGGIO GIALLO/BLU **
2240 VDU19,3,11,0,0,0
2250 COLOUR3
2260 PRINT:PRINT
2265 PRINTTAB(2)go$
2270 REM ** RISPOSTA ? **
2275 *FX 15,1
2280 answer$=INKEY$(0)
2285 IF GET$="N" THEN flagfinale=1
2290 VDU20: REM RIPRISTINA I COLORI DI PARTENZA
2300 ENDPROC
```

IL LISTATO FINALE

```
1000 REM ********************
1010 REM **                  **
1020 REM ** CAMPO MINATO     **
1030 REM **                  **
1040 REM ********************
1050 :
1060 maxpunti$="00000"
1070 flagfinale=0
1080 :
1090 REM ** PROGRAMMA PRINCIPALE **
1100 REPEAT
1110 MODE5
1120 REM ** CANCELLA IL CURSORE **
1130 VDU23;8202;0;0;0;
1140 PROCtitolo
1150 CLS
1160 PROCpreparazione
1170 :
1180 PROCciclo
1190 :
1200 MODE7
1210 REM ** CANCELLA IL CURSORE **
1220 VDU23,1,0;0;0;0;
1230 PROCfinegioco
1240 UNTIL flagfinale=1
1250 CLS
1260 END
1270 :
1280 :
1290 REM ** DEFINIZIONE PROCEDURA **
1300 DEF PROCtitolo
1310 GCOL 0,129
1320 CLG
1330 GCOL 3,3
1340 PROCmusica
1350 Y=100: X=0
1360 REPEAT
1370 X=X+20: Y=Y+50
1380 FOR I=1 TO 2
1390 PROCmine
1400 NEXT I
1410 UNTIL Y>700
1420 :
1430 PROCmine
1440 PRINTTAB (0,20)"Livello (0-9)?"
1450 PROCmusica
1460 REPEAT
1470 livello=GET-48
1480 UNTIL livello>-1 AND livello<10
1490 ENDPROC
1500 :
1510 DEF PROCmine
1520 PLOT 4,X,Y
1530 REM **LETTERA M**
1540 PLOT 1,0,200
1550 PLOT 1,80,-100
1560 PLOT 1,80,100
1570 PLOT 1,0,-200
1580 REM **LETTERA I**
1590 PLOT 0,40,0
1600 PLOT 1,80,0
1610 PLOT 0,-40,0
1620 PLOT 1,0,200
1630 PLOT 0,-40,0
1640 PLOT 1,80,0
1650 REM **LETTERA N**
1660 PLOT 0,40,-200
1670 PLOT 1,0,200
1680 PLOT 1,120,-200
1690 PLOT 1,0,200
1700 REM **LETTERA E**
1710 PLOT 0,160,0
1720 PLOT 1,-120,0
```

Listato finale

Nel listato è necessario fare attenzione alle variabili ed ai nomi di procedura

```
1730 PLOT 1,0,-200
1740 PLOT 1,120,0
1750 PLOT 0,-40,100
1760 PLOT 1,-80,0
1770 ENDPROC
1870 :
1880 DEF PROCpreparazione
1890 COLOUR 2
1900 flagfine=0
1910 PROCinizializza
1920 PROCcreafig
1930 fattore=livello*3+30
1940 PROCmettimine(fattore)
1950 PROCdisegnabordo
1960 PROCiniziatempo
1970 PROCazzerapunteggio
1980 PROCassistente
1990 PROCmettifigure
2000 ENDPROC
2010 :
2020 DEF PROCciclo
2030 REPEAT
2040 PROCaggiornatempo
2050 PROCcontrollotasti
2060 rand=RND(50-livello)
2070 IF rand=1 THEN PROCcecchino
2080 UNTIL TIME>12099 OR flagfine=1
2090 ENDPROC
2100 :
2110 DEF PROCfinegioco
2120 IF punti$>maxpunti$ THEN maxpunti$=punti$
2130 rosso$=CHR$(129)+CHR$(157)+CHR$(131)
2140 gioco$="G A M E  O V E R"
2150 PRINTTAB(0,5)rosso$;CHR$(141);CHR$(136);TAB(12);gioco$
2160 PRINTrosso$;CHR$(141);CHR$(136);TAB(12);gioco$
2170 PRINT:PRINTrosso$;"Punteggio ";TAB(30);punti$
2180 PRINT:PRINTrosso$;"Massimo ";TAB(30);maxpunti$
2190 PRINT:PRINTrosso$;"Tempo";TAB(30);tempo$
2200 blu$=CHR$(132)+CHR$(157)+CHR$(134)
2210 go$="UN'ALTRA PARTITA  S/N?"
2220 PRINT:PRINT
2230 PRINTblu$;CHR$(141);CHR$(136);TAB(5);go$
2240 PRINTblu$;CHR$(141);CHR$(136);TAB(5);go$
2250 REM ** RISPOSTA ?**
2260 *FX 15,1
2270 risposta$=INKEY$(0)
2280 IF GET$="N" THEN flagfinale=1
2290 ENDPROC
2300 :
2310 REM ** PROCEDURE - SECONDO LIVELLO  **
2320 DEF PROCinizializza
2330 xrivel=2: yrivel=25: xomino=17: yomino=1
2340 xinizio=120: xfine=1144
2350 zero$="000000"
2360 ENDPROC
2370 :
2380 DEF PROCcreafig
2390 REM ** MINE **
2400 VDU23,224,0,0,56,254,254,124,0,0
2410 REM **RIVELATORE DI MINE **
2420 VDU23,225,231,195,189,36,36,189,195,231
2430 REM ** ASSISTENTE **
2440 VDU23,226,56,56,16,124,186,170,40,108
2450 ENDPROC
2460 :
2470 DEF PROCdisegnabordo
2480 GCOL 0,1
2490 MOVE 120,188
2500 DRAW 120,992
2510 DRAW 1152,992
2520 DRAW 1152,188
2530 DRAW 120,188
2540 ENDPROC
2550 :
2560 DEF PROCmettimine(numeromine)
2570 REM ** ASSEGNA IL COLORE 2 AL VERDE **
2580 VDU19,2,2,0,0,0
2590 FOR I=1 TO numeromine
2600 PRINT TAB(RND(16)+1,RND(25));CHR$(224)
2610 NEXT I
2620 ENDPROC
2630 :
2640 DEF PROCiniziatempo
2650 PRINT TAB(2,27)"Tempo      02:00"
2660 TIME=0
2670 ENDPROC
2680 :
2690 DEF PROCassistente
2700 omino$=CHR$(226)+CHR$(226)+CHR$(226)
2710 conto=1
2720 COLOUR 1
2730 PRINT TAB(2,30);omino$
2740 COLOUR 2
2750 ENDPROC
2760 :
2770 DEF PROCazzerapunteggio
2780 punti=0: punti$="00000"
2790 PRINT TAB (2,28)  "Punteggio   00000"
2800 PRINT TAB (2,29)  "Massimo "; maxpunti$
2810 ENDPROC
2820 :
2830 DEF PROCmettifigure
2840 COLOUR 1
2850 PRINT TAB(xrivel,yrivel); CHR$(255)
2860 PRINT TAB(xomino,yomino); CHR$(226)
2870 COLOUR 2
2880 ENDPROC
2890 :
2900 DEF PROCaggiornatempo
2910 sec$=STR$(((12100-TIME)DIV 100)MOD 60)
2920 min$=STR$(((12100-TIME)DIV 6000)MOD 60)
2930 REM ** AGGIUNGE ZERI DAVANTI AL NUMERO **
2940 sec$=LEFT$(zero$,2-LEN(sec$))+sec$
2950 min$=LEFT$(zero$,2-LEN(min$))+min$
2960 tempo$=min$+":"+sec$
2970 PRINT TAB (11,27);tempo$
2980 ENDPROC
2990 :
3000 DEF PROCcontrollotasti
3010 REM ** SU ? **
3020 IF INKEY (-58)=-1 THEN PROCmovimento(0,-1)
3030 REM ** GIU' ? **
3040 IF INKEY (-42)=-1 THEN PROCmovimento (0,1)
3050 REM ** A DESTRA ? **
3060 IF INKEY (-122)=-1 THEN PROCmovimento (1,0)
3070 REM ** A SINISTRA ? **
3080 IF INKEY (-26)=-1 THEN PROCmovimento (-1,0)
3090 ENDPROC
3100 :
3110 DEF PROCcecchino
3120 viniziale=RND(750)+220
3130 vfinale=RND(750)+220
3140 dx=32: dy=(vfinale-viniziale)/32
3150 GCOL 3,3
3160 PROClinea
3170 IF POINT(x,y)=1 THEN PROCesplosione(x,y) ELSE PROClinea
3180 ENDPROC
3190 :
3200 REM ** PROCEDURE - TERZO LIVELLO **
3210 :
3220 DEF PROCmovimento (deltax, deltay)
3230 REM ** CANCELLA LE VECCHIE POSIZIONI **
3240 COLOUR 1
3250 PRINT TAB(xrivel,yrivel);" "
3260 PRINT TAB(xomino,yomino);" "
3270 REM ** MUOVE IL RIVELATORE **
3280 xrivel=xrivel+deltax
3290 yrivel=yrivel+deltay
3300 REM ** VERIFICA PERIMETRO DI GIOCO **
3310 IF xrivel>17 THEN xrivel=17
3320 IF yrivel>25 THEN yrivel=25
3330 IF xrivel<2 THEN xrivel=2
3340 IF yrivel<1 THEN yrivel=1
3350 REM ** CALCOLA COORDINATE DELL'ASSISTENTE **
3360 xomino=19-xrivel
3370 yomino=26-yrivel
3380 PROCconversione (xomino,yomino)
3390 IF POINT(xgraf,ygraf)=2 THEN PROCesplosione
     (xgraf,ygraf)
3400 PROCconversione (xrivel,yrivel)
3410 IF POINT (xgraf,ygraf)=2 THEN PROCminatrovata
3420 PROCmettifigure
3430 ENDPROC
3440 :
3450 DEF PROClinea
3460 SOUND 0,-8,4,5
3470 x=xiniziale: y=yviniziale
3480 MOVE x,y
3490 REPEAT
3500 DRAW x,y
3510 x=x+dx: y=y+dy
3520 UNTIL x>xfinale OR POINT(x,y)=1
3530 ENDPROC
3540 :
3550 DEF PROCesplosione(xesplos,yesplos)
3560 REM ** EFFETTO SONORO **
3570 SOUND 0,-15,6,50
3580 REM ** SELEZIONA RITMO LAMPEGGIAMENTO **
3590 *FX9,20
3600 *FX10,50
3610 FOR I=1 TO 100
3620 MOVE xesplos,yesplos
3630 VDU19,2,RND(15),0,0,0
3640 GCOL 0,RND(3)
3650 PLOT 1,RND(100)-50,RND(100)-50
3660 NEXT I
3670 PROCripristino
3680 ENDPROC
3690 :
3700 REM ** PROCEDURE - QUARTO LIVELLO **
3710 :
3720 DEF PROCconversione(xcar,ycar)
3730 xgraf=64*xcar+32
3740 ygraf=1023-(32*ycar+16)
3750 ENDPROC
3760 :
3770 DEF PROCminatrovata
3780 REM ** SEGNALE SONORO **
3790 SOUND 2,-15,170,3
3800 REM ** INCREMENTO DEL PUNTEGGIO **
3810 COLOUR 2
3820 punti=punti+150
3830 punti$=STR$(punti)
3840 punti$=LEFT$(zero$,5-LEN(punti$))+punti$
3850 PRINT TAB(11,28);punti$
3860 ENDPROC
3870 :
3880 DEF PROCripristino
3890 conto=conto+1
3900 IF conto>4 THEN flagfine=1: ENDPROC
3910 CLS
3920 VDU19,2,2,0,0,0
3930 COLOUR 2
3940 PROCinizializza
3950 minerimaste=fattore-punti/150
3960 PROCmettimine(minerimaste)
3970 PROCdisegnabordo
3980 PRINT TAB(2,27);"TEMPO"
3990 PRINT TAB(2,28)"PUNTEGGIO"
4000 PRINT TAB(11,28)punti$
4010 PRINT TAB(2,29)"MASSIMO"
4020 PRINT TAB(18,29)maxpunti$
4030 ominirimasti$=LEFT$(omino$,4-conto)
4040 COLOUR 1
4050 PRINT TAB(2,30);ominirimasti$;"   "
4060 COLOUR 2
4070 PROCmettifigure
4080 ENDPROC
4090 DEF PROCmusica
4100 REM **PRIMA BATTUTA**
4110 SOUND 1,-8,213,5
4120 SOUND 1,-8,209,5
4130 SOUND 1,-8,213,5
4140 SOUND 1,-8,209,5
4150 SOUND 1,-8,213,5
4160 SOUND 1,-8,193,5
4170 SOUND 1,-8,205,5
4180 SOUND 1,-8,197,5
4190 REM **SECONDA BATTUTA**
4200 SOUND 1,-8,185,20
4210 SOUND 1,-8,165,5
4220 SOUND 1,-8,185,5
4230 SOUND 1,-8,193,20
4240 REM **TERZA BATTUTA**
4250 SOUND 1,-8,165,5
4260 SOUND 1,-8,193,5
4270 SOUND 1,-8,197,20
4280 ENDPROC
```

LISTATO ASSEMBLY

```
                ;
                ;********************
                ;********************
                ;**              **
                ;**  FILLSUB &4  **
                ;**              **
                ;********************
                ;********************
                ;
                ;*** VALORI PLOTSUB ***
                ;
                PLOTSUB = $C183
                XLO     = $C103
                XHI     = $C104
                YLO     = $C105
                MPBLO   = $00
                MPBHI   = $20
                PTR     = $FB
                *       = $C700
                ;
                ;*** VARIABILI FILLSUB ***
                ;
                ;
                PXLO    ****1
                PXHI    ****1
                PYLO    ****1
                PHBLO   ****1
                PHBHI   ****1
                PREMX   ****1
                PVBYTE  ****1
                PREMY   ****1
                PROMLO  ****1
                PROMHI  ****!
                PBPOS   ****1
                ;       ****1
                XSTLO   ****1
                XSTHI   ****1
                YST     ****1
                XMINLO  ****1
                XMINHI  ****1
                XMAXLO  ****1
                XMAXHI  ****1
                RTFLAG  ****1
                UPFLAG  ****1
                FXLO    ****1
                FXHI    ****1
                PTFLAG  ****1
                ;
                ;*** TRASFERIMENTO PUNTO INIZIO IN FX,FY ***
                ;
AD 0B C7                LDA XSTLO
8D 14 C7                STA FXLO
AD 0C C7                LDA XSTHI
8D 15 C7                STA FXHI
AC 0D C7                LDY YST
                ;
                ;*** SELEZIONA I FLAG ***
                ;
A9 01                   LDA #$01
8D 12 C7                STA RTFLAG
8D 13 C7                STA UPFLAG
                ;
                ;*** TRACCIA UN PUNTO ***
                ;
                NEXT
8C 05 C1                STY YLO
AD 14 C7                LDA FXLO
8D 03 C1                STA XLO
AD 15 C7                LDA FXHI
8D 04 C1                STA XHI
20 83 C1                JSR PLTSUB
                ;
                ;*** INCR/DECR COORDINATA Y ***
                ;
AD 13 C7                LDA UPFLAG
D0 08                   BNE DECRY
C8                      INY
C0 C8                   CPY #$C8        ;Y HA RAGGIUNTO MAX?
F0 13                   BEQ NEXLIN
4C 52 C7                JMP TESTPT
                DECRY
88                      DEY
C0 00                   CPY #$00        ;Y HA RAGGIUNTO MIN?
90 0B                   BCC NEXLIN
                TESTPT
20 F6 C7                JSR POINT       ;IL PUNTO E' ACCESO?
AD 16 C7                LDA PTFLAG
D0 03                   BNE NEXLIN      ;SE SI SALTA
4C 2E C7                JMP NEXT
                ;
                ;*** INIZIO LINEA SUCCESSIVA ***
                ;
                NEXLIN
AD 12 C7                LDA RTFLAG
D0 1F                   BNE INCRX       ;VERIFICA IL GIUSTO FLAG
AD 14 C7                LDA FXLO
38                      SEC
E9 01                   SBC #$01
8D 14 C7                 STA FXLO       ;DECREMENTA IL BYTE BASSO DI X
AD 15 C7                LDA FXHI
E9 00                   SBC #$00
8D 15 C7                 STA FXHI       ;DECREMENTA IL BYTE  ALTO DI X

CD 0F C7                CMP XMINHI
D0 41                   BNE MOVE
AD 14 C7                LDA FXLO        ;X HA RAGGIUNTO XMIN ?
CD 0E C7                CMP XMINLO
D0 39                   BNE MOVE
60                      RTS             ;FINE DELLA ROUTINE
                ;
                INCRX
AD 14 C7                LDA FXLO
18                      CLC
69 01                   ADC #$01        ;INCREMENTA IL BYTE  BASSO DI X
8D 14 C7                STA FXLO
AD 14 C7                LDA FXHI
A9 00                    ADC #$00       ;INCREMENTA IL BYTE  ALTO DI X
8D 15 C7                STA FXHI
                ;
CD 11 C7                CMP XMAXHI
D0 22                   BNE MOVE
AD 14 C7                LDA FXLO        ;X HA RAGGIUNTO XMAX ?
CD 10 C7                CMP XMAXLO
D0 1A                   BNE MOVE
                ;
AD 0B C7                LDA XSTLO
8D 14 C7                STA FXLO
AD 0C C7                LDA XSTHI       ;RESETTA IL PUNTO D'INIZIO
8D 15 C7                STA FXHI
AC 0D C7                LDY YST
                ;
A9 00                   LDA #$00
8D 13 C7                STA UPFLAG
8D 12 C7                STA RTFLAG      ;AZZERA I FLAG
4C 2E C7                JMP NEXT
                ;
                ;*** TROVA L'INIZIO DELLA LINEA SUCCESSIVA ***
                ;
                MOVE
AD 13 C7                LDA UPFLAG
D0 1C                   BNE DOWN        ;VERIFICA UPFLAG
88                      DEY             ;Y=Y-1
                AGAIN1
20 F6 C7                JSR POINT
AD 16 C7                LDA PTFLAG
D0 04                   BNE CONT1
C8                      INY
4C BF C7                JMP AGAIN1
                CONT1
EE 13 C7                INC UPFLAG      ;PONE IL BIT A UNO
                AGAIN2
88                       DEY
20 F6 C7                JSR POINT
AD 16 C7                LDA PTFLAG
D0 F7                   BNE AGAIN 2
4C 2E C7                JMP NEXT
                ;
                DOWN
C8                      INY             ;Y=Y1
                AGAIN3
20 F6 C7                JSR POINT
AD 16 C7                LDA PTFLAG
D0 04                   BNE CONT2
88                      DEY
4C DB C7                JMP AGAIN3
                CONT2
CE 13 C7                DEC UPFLAG      ;PONE IL BIT A ZERO
                AGAIN4
C8                      INY
20 F6 C7                JSR POINT
AD 16 C7                LDA PTFLAG
D0 F7                   BNE AGAIN4
4C 2E C7                JMP NEXT
                ;*** FINE DEL PROGRAMMA PRINCIPALE ***
                ;
                ;*** SUBROUTINE PER VERIFICA DI UN PUNTO ***
                ;
                POINT
48                      PHA
8A                      TXA
48                      PHA             ;APPOGGIA I REGISTRI SULLO STACK
98                      TYA
48                      PHA
                ;
8C 02 C7                STY PYLO
AD 14 C7                LDA FXLO
8D 00 C7                STA PXLO        ;TRASFERISCE LE COORDINATE
AD 15 C7                LDA FXHI
8D 01 C7                STA PXHI
                ;
                ;*** CALCOLA L'INDIRIZZO DEL PUNTO ***
                ;
AD 01 C7                LDA PXHI
8D 04 C7                STA PHBHI
AD 00 C7                LDA PXLO
29 F8                   AND #$F8
8D 03 C7                STA PHBLO
AD 00 C7                LDA PXLO
29 07                   AND #$07
8D 05 C7                STA PREMX
                ;
AD 02 C7                LDA PYLO
29 07                   AND #$07
8D 07 C7                STA PREMY
                ;
A2 03                   LDX #$03
                SHIFT
4E 02 C7                LSR PYLO
CA                      DEX
D0 FA                   BNE SHIFT
AD 02 C7                LDA PYLO
8D 06 C7                STA PVBYTE
                ;
A9 00                   LDA #$00
8D 08 C7                STA PROMLO
8D 09 C7                STA PROMHI
A2 05                   LDX #$05
                FIVE
AD 08 C7                LDA PROMLO
18                      CLC
6D 06 C7                ADC PVBYTE
8D 08 C7                STA PROMLO
CA                      DEX
D0 F3                   BNE FIVE
A2 06                   LDX #$06
                MULT
0E 08 C7                ASL PROMLO
2E 09 C7                ROL PROMHI
CA                      DEX
D0 F7                   BNE MULT
                ;
AD 08 C7                LDA PROMLO
18                      CLC
6D 03 C7                ADC PHBLO
8D 08 C7                STA PROMLO
AD 09 C7                LDA PROMHI
6D 04 C7                ADC PHBHI
8D 09 C7                STA PROMHI
                ;
AD 08 C7                LDA PROMLO
18                      CLC
69 00                   ADC #MPBLO
8D 08 C7                STA PROMLO
AD 09 C7                LDA PROMHI
69 20                   ADC #MPBHI
8D 09 C7                STA PROMHI
                ;
AD 08 C7                LDA PROMLO
18                      CLC
6D 07 C7                ADC JPREMY
85 FB                   STA PTR
AD 09 C7                LDA PROMHI
69 00                   ADC #$00
85 FC                   STA PTR+1
                ;
A9 01                   LDA #$01
8D 0A C7                STA PBPOS
38                      SEC
A9 07                   LDA #$07
ED 05 C7                SBC PREMX
F0 07                   BEQ BITON
AA                      TAX
                POWER
0E 0A C7                ASL PBPOS
CA                      DEX
D0 FA                   BNE POWER
                ;
                ;*** VERIFICA PER BIT POSTO A UNO ***
                ;
                BITON
A0 00                   LDY #$00
B1  FB                  LDA  (PTR),Y    ;CARICA  IL  CONTENUTO DELL'INDIRIZZO
2D 0A C7                AND PBPOS
8D 16 C7                STA PTFLAG      ;IMMAGAZZINA IL RISULTATO
                ;
68                      PLA
A8                      TAY
68                      PLA             ;RECUPERA I REGISTRI DALLO  STACK
AA                      TAX
68                      PLA
                ;
60                      RTS
```

La Softsel

Una delle più grandi ditte distributrici di software e di hardware

La Softsel fornisce ai rivenditori di software per computer un servizio estremamente prezioso. Nei primi anni del boom dell'informatica, i pacchetti software (alcuni dei quali molto costosi) venivano immessi sul mercato ad un ritmo incalzante. I rivenditori si trovavano di fronte a ben poche alternative: o esaminare personalmente i vari prodotti per valutarne la qualità (con enormi perdite di tempo e aumento dei costi), oppure fidarsi del giudizio di altri, correndo però il rischio di acquistare software costoso, ma di scarso successo commerciale. Il servizio fornito dalla Softsel libera i rivenditori di tutti questi problemi, poiché la ditta effettua un esame completo e dettagliato di tutti i pacchetti, prima di inserirli nel proprio listino software. Simon Rhodes, il direttore commerciale della Softsel inglese, spiega come si svolge questo lavoro: «Il package viene innanzi tutto esaminato da uno staff tecnico, che ne dà una valutazione tenendo conto della qualità della programmazione, della completezza della documentazione e del livello grafico. Quindi viene passato alla sezione marketing e vendite, che decide se e come impostarne il lancio pubblicitario». La Softsel afferma che meno del 10% dei prodotti presi in esame entrano a far parte del catalogo di vendita.
Questo controllo di qualità, abbinato a sconti promozionali, a vantaggiosi accordi per le rimanenze ed alla possibilità di poter trattare con una sola ditta, ha indotto molti rivenditori a scegliere la Softsel come loro unica fornitrice di prodotti software. Probabilmente, un rivenditore potrebbe acquistare i pacchetti ad un prezzo inferiore (ad esempio ordinandoli direttamente al fabbricante), ma gli sconti che la Softsel può permettersi, grazie al suo colossale giro commerciale, rendono la differenza di prezzo quasi insignificante.

Herb Blumstein
Direttore Amministrativo





IAN McKINNELL

Simon Rhodes
Direttore dei Servizi Commerciali

Origini americane

La Softsel è stata fondata nel 1980 da Robert Leff e David Wagman, che avevano lavorato assieme per il Dipartimento elaborazione dati della Transaction Technology, una società controllata dalla potente finanziaria Citicorp. La fiducia che Leff e Wagman riposero nella Softsel fu ripagata da un rapido successo. A quattro anni dalla nascita, la ditta aveva già un organico di 350 persone e, nell'anno seguente, il suo fatturato superò i 100 milioni di dollari. Solo negli Stati Uniti, la Softsel possiede 4 grandi centri di deposito e smistamento dei prodotti (ad Atlanta, a Chicago, a Los Angeles e a New York) in grado di fornire 4.500 diversi pacchetti software ai rivenditori di tutto il paese.
L'ingresso della Compagnia sul mercato europeo è iniziato dalla Gran Bretagna nel settembre dell'82. Nell'aprile dell'anno successivo è stata fondata la consociata inglese Softsel Computer Product, la cui sede è a Feltham (vicino all'aereoporto di Heathrow). Essa fornisce 2.500 differenti prodotti ai rivenditori di tutta Europa e perfino del medio oriente.
La consociata inglese della Softsel intende sviluppare soprattutto il software gestionale, che già oggi occupa più del 50% dei prodotti in catalogo, senza per questo tralasciare un altro grande settore della produzione software: quello dei programmi di gioco.
Per quanto riguarda gli altri mercati europei, la Softsel ha già dato vita ad una nuova società in Germania, con sede a Monaco, e pensa di fondarne altre in Francia ed in Italia.

Il QL della Sinclair

Anatomia di un microcomputer a diffusione europea: tecnologie a confronto

Non c'è da stupirsi se molta gente ha idee piuttosto confuse sul QL della Sinclair e sull'effettivo valore di questo apparecchio. Nella massiccia campagna pubblicitaria in Gran Bretagna, paese d'origine, si afferma che il QL è il miglior home computer sul mercato, fornito di una CPU a 32 bit (il 68008), di quattro programmi applicativi integrati e di un BASIC "residente", la cui qualità è superiore a quella di ogni altro BASIC.
Tutto l'entusiasmo suscitato inizialmente, però, si è presto affievolito soprattutto a causa delle esasperanti attese per la consegna. Il QL, inizialmente messo in vendita soltanto su ordinazione postale diretta alla Sinclair, ha addirittura provocato uno scandalo a livello nazionale in Inghilterra, perché si è scoperto che l'apparecchio era stato "messo in vendita" addirittura molto prima che se ne fosse terminato il progetto! Si può immaginare l'indignazione di chi, attirato dall'allettante pubblicità, aveva già versato un congruo deposito per l'acquisto!
La Sinclair non sembra nuova a questo genere di esperienze: era già accaduto un fatto simile anche per lo Spectrum. Tuttavia c'era una differenza significativa: lo Spectrum, tutto sommato, funzionava! Nel caso del QL, invece, la stessa Sinclair ha dovuto rimangiarsi alcune ambiziose promesse: il preannunciato "Superbasic", per esempio, non è stato implementato sull'apparecchio, "semplicemente" perché i 32 Kbyte di ROM a disposizione non bastavano (né era previsto nel circuito, uno zoccolo per ROM aggiuntive!).
Poiché il costo per riprogettare l'intera scheda sarebbe stato troppo alto, i tecnici della Sinclair hanno pensato di rimediare realizzando un modulo aggiuntivo, da collegare ad una porta del QL, contenente le parti "mancanti" del BASIC e del sistema operativo.
Questo espediente ha permesso alla Sinclair di evadere i primi ordini, con la promessa di una più elegante soluzione nelle versioni successive.
La reazione del pubblico, a questo genere di soluzione "posticcia", non è stata certo positiva; cosicché, per eliminare la scomoda "scatoletta nera", è stato adottato un nuovo stratagemma che ha fatto rabbrividire più di un ingegnere elettronico: il chip aggiuntivo di ROM è stato collocato "a cavallo" di quello già esistente, saldando ciascun piedino dell'u-



IAN McKINNELL

Una foto di famiglia
Il marchio Sinclair è apparso su una vasta gamma di prodotti elettronici: dagli amplificatori audio, al famoso orologio digitale Black Watch, dalle calcolatrici ai computer, fino al tanto sospirato televisore a schermo piatto. Caratteristiche tipiche della Sinclair sono sempre state un'ingegnosità tecnica, un design molto attraente e una strategia di marketing molto ambiziosa. Quest'ultima, però, è stata spesso contrastata da enormi ritardi nella produzione e nella commercializzazione degli apparecchi

no a quello corrispondente dell'altro (salvo uno, necessario alla selezione del chip, collegato con un cavetto "volante" ad un'altra parte della scheda).
Da allora, anche il sistema operativo ha subito alcune vicissitudini e, dopo numerose versioni e revisioni, pare essersi stabilizzato (la versione "AH" è stata la prima ad essere distribuita in quantità industriali).
Ciò che ha fatto risparmiare tempo nella produzione, ma che deve aver lasciato alla Sinclair un margine di guadagno davvero esiguo, è stato l'impiego delle EPROM al posto delle più economiche ROM, adottate solo in un secondo momento, più o meno corrispondente a quando è uscita la versione AH del sistema operativo.
In una situazione simile, valutare il QL è veramente difficile. Si potrebbe dire che è qualcosa di più di un home computer, ma un po' meno di un microcomputer per applicazioni commerciali. Come tutti gli home computer è dotato di un BASIC residente, di una buona grafica a colori e di una porta per joystick. Inoltre, è collegabile ad un normale televisore e le sue dimensioni sono veramente contenute. Le due caratteristiche che lo avvicinano ad un apparecchio per usi commerciali sono: una discreta capacità delle memorie di massa (i Microdrive incorporati, certamente più veloci dei normali nastri a cassetta) e la presenza di un package di quattro programmi di buon livello: un word processor, una lavagna elettronica, un database e un programma di gestione grafica.
Considerato come home computer, il QL offre senz'altro molte attrattive: per un costo relativamente basso è dotato di un BASIC molto sofisticato (si deve riconoscere che il Superbasic è, in effetti, uno dei migliori dialetti in commercio) e di Microdrive incorporati, anche se parte del software può non interessare l'utente medio di home computer. Indubbiamente, un fatto che gioca a sfavore del QL è l'incompatibilità tra i nuovi Microdrive e quelli impiegati con lo Spectrum. Oltre al sistema operativo, anche la gestione dello schermo è diversa e ciò ha ostacolato la produzione e la diffusione su larga scala di versioni "adattate" dei vecchi programmi Spectrum.
Pare che nel Superbasic vi siano ancora alcuni bug da correggere. Inoltre il suo editor per redigere i programmi in BASIC, assomiglia molto a quello dello Spectrum; non si può certo dire che appartenga alla cosiddetta Quinta Generazione! Comunque, l'aver introdotto, in questo dialetto BASIC, l'uso di procedure simili a quelle del Micro BBC e di strutture SELECT, paragonabili alla CASE del PASCAL, costituisce una grande innovazione.
La grafica del QL è molto buona, mentre il generatore audio lascia un poco a desiderare.
Considerato come apparecchio professionale, invece, il QL è un po' meno convincente: vada per il pacchetto di software fornito di corredo, ma all'utente potrebbe anche servire un programma di contabilità che per il momento non è reperibile in commercio. La stessa lavagna elettronica (spreadsheet) lascia liberi appena 15 Kbyte di RAM, non sempre sufficienti per modelli finanziari di qualche complessità. Inoltre manca la possibilità di collegarsi a qualcosa di più consistente dei piccoli Microdrive, mentre spesso è indispensabile usare vere unità a dischetti.
Tutto sommato, il QL, come tanti altri apparecchi della gamma Sinclair, è innovativo, discutibile, controverso e talvolta frustrante, ma costituisce senz'altro una pietra di paragone per le case produttrici di altri computer.

LA CONCORRENZA

Il QL ed il Micro BBC

Per accaparrarsi una buona fetta del mercato degli home computer, il QL deve gareggiare contro un concorrente molto affermato: il Micro BBC. Quest'ultimo non è dotato di tutto il software venduto assieme al QL, non possiede un microprocessore altrettanto potente, né una memoria altrettanto espandibile. Tuttavia, il QL non sembra destinato ad usi didattici (dove invece il Micro BBC è uno specialista), il software disponibile è ancora molto scarso e l'apparecchio non offre, nemmeno lontanamente, le stesse possibilità del Micro BBC di collegare periferiche



Il QL ed il Macintosh

È significativo il fatto che Sir Clive Sinclair abbia scelto, per il lancio pubblicitario del QL, il momento in cui usciva il Macintosh. Evidentemente, egli ha volutamente contrapposto il proprio apparecchio a quello della Apple che è basato sul medesimo chip e costa circa cinque volte di più del QL. Nonostante molte somiglianze (la stessa CPU, la stessa frequenza di funzionamento, la stessa espandibilità di memoria e il software integrato), i due apparecchi differiscono notevolmente nella filosofia progettuale. Il QL, pur essendo un microcomputer "brillante", non ha niente di veramente innovativo, mentre il Macintosh è dotato di un sistema operativo sofisticatissimo, molto utile ad utenti inesperti di programmazione. Inoltre, il Macintosh è fornito di un mouse, che pone questo apparecchio ad un livello superiore, rispetto alla maggioranza di altri computer più o meno similari





Il software integrato

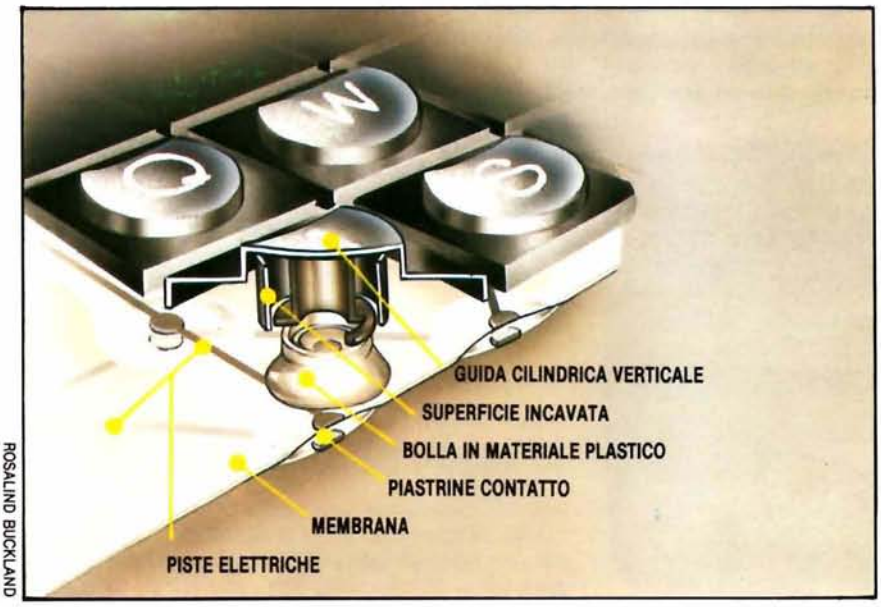
I quattro programmi hanno tutti la stessa impostazione grafica e comandi molto simili. I dati possono essere scambiati tra i vari programmi, tramite il Microdrive. Malauguratamente, le versioni attualmente distribuite soffrono ancora di qualche fastidioso bug, ma le cose dovrebbero presto aggiustarsi. Il word processor "Quill" consente la redazione di testi nei formati a 40, 64 ed 80 colonne; c'è tuttavia un fastidioso ritardo tra la pressione dei tasti e la comparsa, sullo schermo, dei caratteri corrispondenti. "Abacus" è una lavagna elettronica molto sofisticata, peccato che i soli 15 Kbyte di RAM che rimangono liberi non consentano lo sviluppo di elaborati modelli. "Archive" è il programma per la gestione di archivi: oltre ad un uso "diretto" è previsto anche un linguaggio interno, simile al Superbasic, ma lento in esecuzione, dato che ricorre all'uso del Microdrive. "Easel" è un programma per la creazione di grafici lineari, a barre o a torta: sia il suo uso che il passaggio da un formato all'altro è abbastanza semplice



IAN McKINNELL



IAN McKINNELL



ROSALIND BUCKLAND

QL SINCLAIR

DIMENSIONI

mm. 472 x 138 x 46 (tastiera)

CPU

68008 della Motorola, funzionante a 7,5 MHz

MEMORIA

128 Kbyte di RAM (espandibile fino a 640 Kbyte), 48 Kbyte di ROM

VIDEO

25 righe di 85 caratteri (su monitor). Risoluzione grafica di 512 x 256 pixel (4 colori), oppure di 256 x 256 pixel (8 colori)

INTERFACCE

Seriale RS232 (due), per 2 joystick, per Microdrive, per rete locale (LAN), per TV, per monitor RGB

LINGUAGGI FORNITI

SuperBASIC

TASTIERA

Dispozione QWERTY, tipo macchina da scrivere, con 65 tasti "a membrana". A differenza degli altri apparecchi Sinclair, esiste una vera e propria barra spaziatrice. Esistono 5 tasti di funzione, ma non quello di DELETE (per cancellare un carattere appena digitato)

DOCUMENTAZIONE

Il manuale operativo è di ottima qualità e spiega ampiamente sia il funzionamento del SuperBASIC, sia dei quattro programmi applicativi in dotazione

PRO

Il 68008 è un chip a 32 bit molto veloce. La qualità del software incluso è ottima: addirittura il SuperBASIC è uno dei migliori dialetti oggi disponibili. Anche la grafica è piuttosto buona

CONTRO

I Microdrive, alla base del funzionamento del QL, hanno un funzionamento lento e non sono compatibili con quelli adoperati con lo Spectrum. C'è una notevole scarsità di software ed il sistema operativo non è del tutto privo di inconvenienti

SCRITTURA CON LE... BOLLE

Il QI è dotato di una tastiera "a membrana". Ogni nodo della matrice di connessioni, corrispodente ad un tasto, è fatto in modo che il contatto elettrico sia racchiuso in una "bolla" di materiale plastico. Premendo un tasto, la bolla si comprime chiudendo il contatto. È la stessa bolla a produrre la forza "di richiamo", ottenuta generalmente con l'impiego di una piccola molla. La tastiera del QL è senza dubbio di gran lunga migliore rispetto a quella dello Spectrum, sia per la forma dei tasti che per la loro disposizione QWERTY. Tuttavia, non c'è paragone, ad esempio, con la tastiera del Vic 20, per non citare poi quelle ottime del Micro BBC o del Macintosh



KEVIN JONES

Il gioco dell'impiccato

Giocare con le parole è divertente, ma può essere anche molto istruttivo

L'impiccato è un gioco piuttosto antico, frequentemente giocato d'estate sulla spiaggia. Uno dei giocatori pensa una parola e traccia sulla sabbia tanti trattini quante sono le lettere che la compongono; l'altro giocatore deve indovinare la parola una lettera alla volta. Se la lettera fa parte della parola, essa viene sostituita ai trattini corrispondenti; in caso contrario, viene disegnato un elemento del disegno che rappresenta, appunto, l'impiccato. Nel programma qui presentato, la figura è composta da 10 elementi, e compare nella parte destra dello schermo: se si falliscono più di dieci tentativi, il gioco termina... con l'"esecuzione" della figurina.

Il funzionamento del programma è abbastanza semplice: il computer seleziona una parola a caso tra le 11 contenute nelle frasi DATA. Ad ogni lettera immessa dal giocatore, il programma verifica se e dove compare la lettera nella parola scelta, visualizzandola nelle corrette posizioni. Se la lettera non c'è, essa viene visualizzata a parte (per ricordare al giocatore quali lettere ha già provato) e viene tracciato un pezzetto di disegno.

In entrambe le versioni del programma (per il BBC e per lo Spectrum) l'elenco di parole è contenuto nelle frasi DATA alle linee 1020—1030. Ovviamente, nulla vieta di aggiungere nuove parole, purché il numero di lettere che le compongono non sia maggiore di dieci. Volendo superare questo limite, occorre modificare le linee 30 e 50. Inoltre, va cambiato il numero totale di parole (la variabile N alla linea 20).

Inizialmente, il programma legge (con una READ) tutte le parole trasferendole in una matrice. Poi sceglie a caso uno degli elementi di questa matrice e, sullo schermo, vengono visualizzati tanti punti quante sono le lettere della parola data; il resto del gioco è formato da un ciclo iterativo. Per prima cosa ogni carattere immesso da tastiera viene "filtrato": se il carattere immesso non corrisponde ad una lettera o se si è digitato più di un carattere, il computer emette un bip, tornando a richiedere l'immissione. La lettera digitata viene poi confrontata con quelle immesse precedentemente e, se è già stata adoperata, appare un messaggio e il programma chiede l'immissione di una nuova lettera.

Se la lettera digitata ha superato queste verifiche, essa viene accodata alla lista di quelle "usate" e poi confrontata con i caratteri che compongono la parola da indovinare. Ogni volta che la lettera digitata è contenuta nella parola da indovinare il corrispondente punto sullo schermo viene sostituito dalla lettera in questione. Se, invece, la lettera non compare, il programma richiama la routine per disegnare il successivo elemento dell'omino.

Al decimo tentativo errato, l'omino viene "impiccato", viene suonato un breve motivetto ed il gioco ricomincia con una nuova parola.

Anche se il programma viene qui presentato in sole due versioni, se ne possono estrarre le basi per redigerne altre adatte agli altri micro. Ciò che richiede un numero maggiore di modifiche è la routine per il disegno dell'omino, in quanto le istruzioni grafiche cambiano da un BASIC all'altro. Inoltre questa parte del programma può essere arricchita, creando immagini grafiche o sequenze musicali più elaborate.

Anche il resto del programma può essere migliorato; ad esempio, si potrebbe aggiungere una verifica, durante la selezione della parola, per far sì che non venga scelta una parola già "usata". Nella versione attuale, infatti, la scelta è puramente casuale, quindi la stessa parola potrebbe comparire due volte di seguito.

Inoltre, si potrebbe aggiungere una piccola routine che trasformi automaticamente i caratteri digitati in maiuscole, evitando così di premere continuamente il tasto SHIFT.



LIZ HEANEY

VERIFICA IMMEDIATA

Ecco cosa appare sullo schermo durante il gioco: sulla destra, viene tracciato un elemento dell'omino ad ogni lettera "sbagliata". Se si digita una lettera già usata in precedenza, compare un messaggio appropriato

I LISTATI DEL PROGRAMMA

Ecco le versioni per il BBC e lo Spectrum. Si noti che il giocatore va avvertito del fatto che vanno usate le maiuscole o le minuscole (a meno che non si modifichi il programma per accettare solo le maiuscole)

MICRO BBC

```
 10 REM        L'IMPICCATO
 20 N=11
 30 DIM WS(N)
 40 REM Inizializzazione
 50 D$="----------"
 60 FOR I=1 TO N
 70   READ W$
 80   W$(I)=W$
 90   NEXT I
100 MODE 1
110 REM scelta di una nuova parola
120 CLS
130 PRINT TAB (5,1);"L'IMPICCATO"
140 W$=W$(RND(N))
150 C=0:W=0:D=0
160 G$=""
170 PRINT TAB (5,23);"Lettere adoperate :"
180 L=LEN(W$)
190 PRINT TAB(5,20);LEFT$(D$,L)
200 REM Immissione lettera "campione"
210 PRINT TAB(5,30);STRING$(34,"")
220 INPUT TAB(5,30);L$
230 REM verifica sulla lettera digitata
240 IF L$="" THEN SOUND 1,-15,50,5:GOTO 210
250 IF LEN(L$)>1 THEN SOUND 1,-15,50,5:GOTO 210
260 F=0
270 FOR I=1 TO LEN(G$)
280   IF L$<>MID$(G$,I,1) THEN GOTO 330
290   PRINT TAB(5,30)"HAI GIA' PROVATO QUESTA LETTERA"
300   SOUND 1,-15,50,5
310   T%=TIME:REPEAT UNTIL TIME>T%+50
320   F=1
330   NEXT I
340 IF F=1 THEN 210
350 G$=G$+L$
360 PRINT TAB(5,25);G$
370 REM Confronto con la parola segreta
380 F=0
390 FOR I=1 TO L
400   IF L$=MID$(W$,I,1) THEN GOSUB 480
410   NEXT I
420 IF F<>1 THEN GOSUB 670
430 IF D=1 THEN GOSUB 590:GOTO 120
440 IF C=L THEN GOSUB 530:GOTO 120
450 GOTO 210
460 END
470 REM La lettera corrisponde
480 PRINT TAB(4+I,20);L$
490 C=C+1
500 F=1
510 SOUND 1,-15,200,5:SOUND 1,0,0,2
520 RETURN
530 REM Successo!
540 FOR I=1 TO 200 STEP 10
550   SOUND 1,-15,I,2
560   NEXT I
570 RETURN
580 REM Fiasco completo!
590 FOR I=200 TO 1 STEP -10
600   SOUND 1,-15,I,2
610   NEXT I
620 CLS
630 PRINT TAB(5,20);"LA PAROLA ERA : ";W$
640 T%=TIME:REPEAT UNTIL TIME>T%+200
650 RETURN
660 REM La parola non contiene la lettera
670 W=W+1
680 ON W GOTO 690,760,780,810,830,870,890,910,930,96
690 VDU29,800;800;
700 MOVE 50,0
710 FOR A=0 TO 7 STEP 0.4
720   DRAW 50*COS(A),50*SIN(A)
730   NEXT A
740 VDU29,0;0;
750 RETURN
760 MOVE 800,750:DRAW 800,550
770 RETURN
780 MOVE 740,450:DRAW 750,450:DRAW 800,550
790 DRAW 850,450:DRAW 860,450
800 RETURN
810 MOVE 750, 630:DRAW 800,730:DRAW 850,630
820 RETURN
830 PLOT69,800,800
840 PLOT69,820,820
850 PLOT69,780,820
860 RETURN
870 MOVE 780,790:DRAW 800,770:DRAW 820,790
880 RETURN
890 MOVE 600,400:DRAW 1100,400
900 RETURN
910 MOVE 1000,400:DRAW 1000,900
920 RETURN
930 MOVE 1000,900:DRAW 800,900
940 MOVE 1000,850:DRAW 950,900
950 RETURN
960 MOVE 800,900:DRAW 800,850
970 MOVE 780,790:PLOT 14,800,770:PLOT 6,820,790
980 MOVE 780,770:DRAW 800,780:DRAW 820,770
990 D=1
1000 RETURN
1010 REM Elenco delle parole
1020 DATA "MUTAZIONE","BATTITO","CHERUBINO","TRESCA","AGO"
1030 DATA "SPETTRO","GELATO","CARTONE","POSIZIONE",
     "PIANOFORTE","PAPERINO"
```

SPECTRUM

```
 10 REM              L'IMPICCATO
 20 LET N=11
 30 DIM X$(N,10)
 35 DIM Y(N)
 40 REM inizializzazione
 50 LET D$="----------"
 60 FOR I=1 TO N
 70 READ W$
 75 LET Y(I)=LEN W$
 80 LET X$(I)=W$
 90 NEXT I
110 REM scelta di una nuova parola
120 CLS
130 PRINT AT 1,1;"L'IMPICCATO"
140 LET I=(1+INT (RND*N))
145 LET W$=X$(I)
150 LET C=0: LET W=0: LET D=0
160 LET G$=""
170 PRINT AT 20,1;"USATE :"
180 LET L=Y(I)
190 PRINT AT 16,1;D$(1 TO L)
200 REM immissione lettera "campione"
220 INPUT L$
230 REM verifica sulla lettera digitata
240 IF L$="" THEN BEEP 0.25,-10: GO TO 210
250 IF LEN L$>1 THEN BEEP 0.25,-10: GO TO 210
260 LET F=0
270 FOR I=1 TO LEN G$
280 IF L$<>G$(I) THEN GO TO 330
290 PRINT AT 21,1;"HAI GIA' PROVATO QUESTA LETTERA"
300 BEEP 0.25,-10
310 PAUSE 25: PRINT AT 21,1;
320 LET F=1
330 NEXT I
340 IF F=1 THEN GO TO 210
350 LET G$=G$+L$
360 PRINT AT 20,8;G$
370 REM confronto con la parola segreta
380 LET F=0
390 FOR I=1 TO L
400 IF L$=W$(I) THEN GO SUB 480
410 NEXT I
420 IF F<>1 THEN GO SUB 670
430 IF D=1 THEN GO SUB 590: GO TO 120
440 IF C=L THEN GO SUB 530: GO TO 120
450 GO TO 210
460 STOP
470 REM la lettera corrisponde
480 PRINT AT 16,I;L$
490 LET C=C+1
500 LET F=1
510 BEEP 0.25,+16
520 RETURN
530 REM successo!
540 FOR I=-10 TO 10
550 BEEP 0.1,I
560 NEXT I
570 RETURN
580 REM fiasco completo!
590 FOR I=10 TO -10 STEP -1
600 BEEP 0.1,I
610 NEXT I
620 CLS
630 PRINT AT 10,3; "LA PAROLA ERA : ";W$
640 PAUSE 50
650 RETURN
660 REM la parola non contiene la lettera
670 LET W=W+1
680 IF W=1 THEN GO TO 690
681 IF W=2 THEN GO TO 760
682 IF W=3 THEN GO TO 780
683 IF W=4 THEN GO TO 810
684 IF W=5 THEN GO TO 830
685 IF W=6 THEN GO TO 870
686 IF W=7 THEN GO TO 890
687 IF W=8 THEN GO TO 910
688 IF W=9 THEN GO TO 930
689 IF W=10 THEN GO TO 960
690 CIRCLE 200,150,10
700 RETURN
760 PLOT 200,140: DRAW 0,-40
770 RETURN
780 DRAW -10,-20:DRAW -4,0
790 PLOT 200,100:DRAW 10,-20:DRAW 4,0
800 RETURN
810 PLOT 190,110: DRAW 10,25: DRAW 10,-25
820 RETURN
830 PLOT 200,150
840 PLOT 196,155
850 PLOT 204,155
860 RETURN
870 PLOT 196,148: DRAW 4,-4: DRAW 4,4
880 RETURN
890 PLOT 170,60: DRAW 80,0
900 RETURN
910 PLOT 240,60: DRAW 0,115
920 RETURN
930 DRAW -40,0
940 PLOT 240,165: DRAW -10,10
950 RETURN
960 PLOT 200,175: DRAW 0,-15
970 OVER 1: PLOT 196,148: DRAW 4,-4: DRAW 4,4: OVER 0
980 PLOT 196,144: DRAW 4,4: DRAW 4,-4
990 LET D=1
1000 RETURN
1010 REM Elenco delle parole
1020 DATA "MUTAZIONE","BATTITO","CHERUBINO","TRESCA","AGO"
1030 DATA "SPETTRO","GELATO","CARTONE","POSIZIONE",
     "PIANOFORTE","PAPERINO"
```

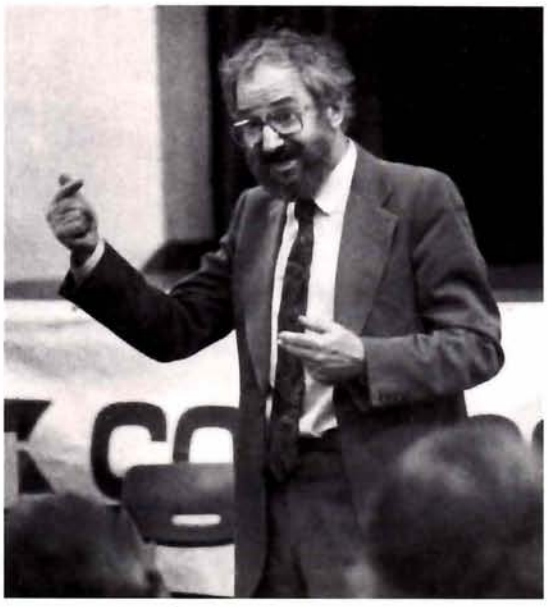
Introduzione al LOGO

La prima parte di una serie dedicata a questo interessante linguaggio di programmazione

Le principali caratteristiche del linguaggio BASIC e della programmazione in codice macchina sono state esaminate in dettaglio in altre sezioni de IL MIO COMPUTER; tuttavia, questi non sono gli unici linguaggi disponibili e soprattutto, non sono particolarmente adatti alla didattica.
Al contrario, il LOGO è uno dei migliori linguaggi attualmente esistenti per avvicinare alla programmazione anche i meno esperti, addirittura anche i bambini. Il LOGO serve anche a chi conosce altri linguaggi (il BASIC, ad esempio) nei quali la stesura di programmi strutturati non è affatto facilitata dall'impostazione stessa del linguaggio. Il LOGO è reperibile su cassetta o cartuccia, per la maggior parte degli home computer (una delle migliori versioni di LOGO è quella scritta per lo Spectrum della Sinclair). Infine, tale linguaggio, è un vero e proprio "sistema didattico"; il fatto più curioso è che non è facile da apprendere, ma estremamente semplice da usare.
La nascita del LOGO è strettamente legata agli studi sull'Intelligenza Artificiale (AI) e al linguaggio LISP (realizzato, quest'ultimo, alla fine degli anni '60 per facilitare l'elaborazione su computer di complesse strutture di dati). Il nome LISP deriva dalle parole inglesi LISt Processing (processore di liste), il che significa che la struttura basilare dei dati è appunto, una "lista", anziché una matrice di caratteri o numeri (come accade nel BASIC). Le principali funzioni del LISP riguardano il trattamento degli elementi di tali liste, che possono essere semplici simboli, oppure, a loro volta, anche intere liste. Il vantaggio di questo principio è che si riesce ad elaborare con maggiore facilità i dati non numerici (gli elementi di una frase in un discorso, ad esempio).
Il LISP fonda gran parte del proprio funzionamento sul principio della "ricursione", per cui una funzione o una procedura è definita in base a se stessa (vedere a pag. 954). Nel caso del LISP, l'elemento definito è sempre una "lista". Questa caratteristica non è casuale, ma il frutto di annose indagini, effettuate su computer, sul linguaggio "naturale" e sull'intelligenza umana. Il LISP è un linguaggio particolarmente complesso, perciò un gruppo di studiosi del Massachusetts Institute of Technology (MIT) decise, nel 1968, di creare un nuovo linguaggio, basato sul LISP, ma adatto ad esser usato anche dai bambini nelle scuole.
Il capo carismatico di questo gruppo era Seymour Papert, che aveva dedicato numerosi anni allo studio dei processi d'apprendimento nei bambini, assieme a Jean Piaget (1896—1980), il più insigne psicologo educazio-

CONOSCERE IL LOGO

Il LOGO possiede due caratteristiche che lo rendono uno strumento didattico potenzialmente fenomenale. La prima è che si tratta di un linguaggio "interattivo": quando si impartisce un comando se ne vedono immediatamente gli effetti sullo schermo. Ciò significa che è facile apprezzare, specialmente da parte di un bambino, i progressi che si fanno, potendo verificarli ad ogni passo.
La seconda è l'espandibilità: raggruppare un certo numero di comandi elementari, si possono eseguire operazioni anche molto complesse. Questi gruppi di comandi vengono chiamati "procedure" e, una volta definitane una, la si può richiamare col nome ad essa associato, come se si trattasse di un nuovo comando del LOGO (vedere a pag. 164). Si può così creare, con estrema semplicità, una vasta gamma di nuovi "comandi".
Rispetto ad altri linguaggi, col LOGO si nota una maggior tendenza all'"esplorazione" nei metodi e nei criteri da adoperare: talvolta il procedimento seguito è diretto e ben strutturato, ma in altri casi si parte da un nucleo, attorno al quale viene letteralmente costruito un intero programma. Ciò è dovuto al fatto che, nel LOGO, esistono più percorsi per arrivare al medesimo risultato



Il "padre" del LOGO

Seymour Papert, qui raffigurato durante una conferenza promossa dalla Commodore, è considerato il "padre" del LOGO. Egli fa parte della LOGO Computer Systems Inc., che sviluppa e fornisce versioni di LOGO adatte alla maggior parte degli home computer

nale degli ultimi anni. Papert iniziò a lavorare al MIT, affiancato da un esperto in intelligenza artificiale, Marvin Minsky, e con lui tentò di formulare un nuovo linguaggio, il LOGO, in cui riversò le proprie conoscenze e quelle acquisite dai suoi colleghi, sia sull'intelligenza artificiale che sui processi d'apprendimento.
Lo sviluppo del LOGO, su computer di enormi dimensioni e soprattutto nell'ambito universitario, si è protratto fino agli anni '70. Il suo successo, però, è principalmente dovuto alla comparsa dei primi home computer, che lo hanno diffuso su larga scala.
Il LOGO necessita di una notevole quantità di memoria, sia per il programma in se sia per i dati da elaborare: un interprete LOGO tipico occupa sui 30 Kbyte, più un'altra decina di Kbyte, se si adoperano funzioni grafiche; a ciò va poi aggiunta l'occupazione del programma!
Ecco perché, inizialmente, i fabbricanti di home computer hanno optato per il BASIC, molto più economico in termini di memoria. Solo in seguito, quando sono comparsi i primi apparecchi con almeno 48Kbyte di RAM, si è avuta la possibilità di usare il LOGO.
Comunque, il merito di aver portato tale linguaggio fuori dagli ambienti di ricerca, proponendolo al grande pubblico, è stato di Papert, col suo libro intitolato *Mindstorms*. In questo scritto egli sviluppa una precisa teoria su come adoperare il computer nell'insegnamento, basandosi sui frutti delle sue ricerche fatte sull'intelligenza artificiale, sui processi d'apprendimento e sui più moderni criteri della pedagogia. Il pensiero di Papert si può riassumere in una sua semplice, ma efficace, frase: «Voglio vedere i bambini che programmano i computer e non viceversa, come accade invece con gli attuali sistemi didattici».
In sostanza si tratta di formulare una nuova "cultura informatica", in cui il bambino vive da protagonista, esplorando personalmente le proprie capacità logiche, sfruttando il LOGO come semplice mezzo e non come fine, piuttosto che subire passivamente quanto preparato ed imposto da altri.
Le teorie di Papert, benché convincenti, lasciano ancora qualche perplessità. In primo luogo, nonostante siano stati fatti molti studi in proposito, non esistono ancora sufficienti dati sperimentali per confermarne le basi. In secondo luogo, le teorie di Piaget sono forse state interpretate in modo diverso da quanto lo stesso Piaget avrebbe voluto. Infine, esistono aree dell'insegnamento (perfino in matematica!), in cui il LOGO non è applicabile.
Spesso, all'atto pratico, gli insegnanti che adoperano il LOGO nelle scuole si accorgono che i risultati ottenuti non sono quelli sperati né quelli teorizzati da Papert. Esiste dunque il rischio di una caduta d'interesse nei confronti del LOGO, provocata da un'eccessiva declamazione dei suoi meriti.
Forse, una volta ridimensionate le capacità di questo linguaggio, lo si potrà impiegare con più profitto specialmente quando le caratteristiche hardware dei piccoli computer didattici saranno tali da rendere il LOGO più veloce e maneggevole.

A CHI SERVE IL LOGO?

Certamente non solo ai principianti, ma anche a chi sia solito programmare in altri linguaggi in particolare:

- a chiunque si avvicini per la prima volta alla programmazione
- a chiunque piaccia "giocare" coi computer, pensando che questi apparecchi devono essere divertenti da usare
- a chiunque consideri frustranti le limitazioni espressive tipiche di altri linguaggi
- a chiunque abbia interesse ad esplorare i campi del pensiero, dell'apprendimento o dell'insegnamento
- a chiunque desideri approfondire le proprie conoscenze nel settore dell'intelligenza artificiale

Tuttavia, si rammenti che il LOGO non è facile da apprendere quanto il BASIC, quindi non risulta particolarmente adatto:

- a chi pensa che "computer" sia sinonimo di "lavoro". Alcuni linguaggi di programmazione sono progettati per il settore commerciale: il LOGO è ben diverso!
- a chi si aspetta una grande rapidità d'esecuzione nei programmi. Il LOGO occupa una vasta quantità di memoria e, attualmente, il funzionamento è piuttosto lento (addirittura il doppio del tempo richiesto in BASIC, per alcuni programmi).

Comunque, una conoscenza della struttura logica del LOGO è utile a tutti; c'è chi redige un primo abbozzo di programma in LOGO, per poi produrne una versione definitva in un altro linguaggio.



VERSIONI AUTORIZZATE

Queste sono le tre versioni di LOGO più complete e meglio documentate (per il Commodore 64, per lo Spectrum e per gli Atari). Benche' siano relativamente costose, sono versioni di LOGO "approvate", che più si avvicinano a quella formulata dal MIT.
Il LOGO è disponibile su dischetto per il Commodore 64, su cartuccia per gli Atari e su cassetta per lo Spectrum

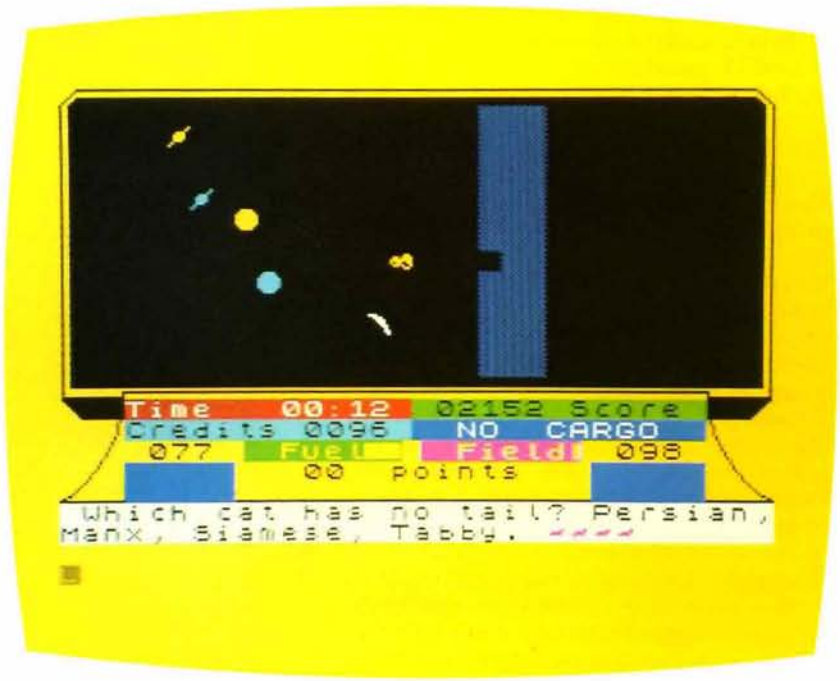
Software in vetrina

Una selezione tra i numerosi programmi disponibili per i computer più diffusi

I due programmi presentati stavolta appartengono a generi alquanto diversi. Il primo è, una volta tanto, un gioco adatto ai più piccoli (purché sappiano un pò d'inglese!). Il secondo, invece, è di carattere sportivo, anzi addirittura olimpionico.

Clever Clogs Startrucker per lo Spectrum

Questo gioco è particolarmente orientato ai più giovani (la casa produttrice afferma che è adatto a bambini di 9 anni): lo svolgimento del gioco vero e proprio, un viaggio galattico, è preceduto da un breve "esamino". La maggiore o minore quantità di carburante assegnata dipende dall'esito del test preliminare (per rispondere, tuttavia, occorre conoscere qualche parolina d'inglese). Le finalità didattiche di questo programma sono piuttosto discutibili, tuttavia la grafica è piacevole, anche se l'azione non è molto rapida.





Decathlon per il Commodore 64

L'abilità di spostare rapidamente il joystick può far meritare al giocatore una bella medaglia, in questo gioco olimpionico del Decathlon. Come molti appassionati di sport sanno, il Decathlon è una gara (quella vera, s'intende!) molto impegnativa, poiché l'atleta deve affrontare ben dieci prove. In questo programma, il realismo è incredibilmente efficace: la "folla" si acquieta mentre stiamo per spiccare un salto con l'asta. È consigliabile, a chi intenda acquistare questo gioco, allenarsi tenacemente all'uso dei joystick: vincere la corsa dei 1.500 metri implica manovrarli a velocita supersonica, per ben quattro minuti! La grafica è eccellente e, tra le varie simulazioni attualmente disponibili sul mercato, il *Decathlon* è senz'altro una delle migliori.

Nuove espressioni musicali

La codifica digitale dei suoni ed i musicisti contemporanei



Gemelli musicali
Questi sono Alannah Currey, Tom Bailey e Joe Leeway, ossia i "Thompson Twins". Inizialmente un complesso di sette elementi, è oggi un trio che suona accompagnato da una base ritmica "sequenziata", pre-registrata

Nonostante i notevoli progressi compiuti in questo settore negli ultimi 60 anni, vale la pena di ricordare il funzionamento di un semplice apparato: l'oscillatore. Quando un qualsiasi oggetto (sia esso l'ala di un'ape o una corda vocale umana) vibra l'aria che lo circonda si comprime e si espande molto rapidamente, producendo una particolare forma d'onda, ossia un suono, riconoscibile dall'orecchio e dal cervello. Similmente, una lamella metallica vibra se ad essa viene applicato, per mezzo di un modulatore, una corrente elettrica e la forma d'onda prodotta è di tipo sinusoidale. La frequenza della vibrazione dipende dal voltaggio applicato e, in minor misura, dalla densità della sostanza metallica. Questo piccolo generatore di suoni viene chiamato "oscillatore". Per decine d'anni, l'unica forma di sintetizzazione dei suoni si è basata su oscillatori di questo tipo.

L'interfaccia MIDI, introdotta sul mercato nel 1983, è un'unità progettata per permettere ad un sistema digitale (un computer) di controllarne un altro (un sintetizzatore, ad esempio). La realizzazione dell'interfaccia è il frutto dei notevoli progressi nella tecnologia musicale di quest'ultimo decennio.

Fino a pochi anni fa, negli studi di registrazione si trovavano numerosi apparecchi di riverbero e di filtraggio elettronico dei suoni. Analogamente, i primi suonatori di "synth", negli anni '70, erano circondati da innumerevoli tastiere ed apparecchiature di controllo.

Quello che accade in uno studio di registrazione è paragonabile ad un vecchio gioco per ragazzi, tipo *Passa la parola*, dove una parola o un'intera frase viene sussurrata da un partecipante all'altro. L'ultimo della fila ripete ad alta voce ciò che gli è giunto alle orecchie e, spesso, il risultato è qualcosa di molto diverso dalla parola o frase iniziale. Negli studi di registrazione, anziché parole si adoperano suoni e, al posto dei giocatori, c'è una serie di apparecchi elettronici, ognuno dei quali ha il compito di modificare o filtrare i suoni originali. Per coordinare la qualità e la quantità dell'intervento di ciascun apparecchio occorre una mano ben addestrata; spesso, si tratta di quella di un ingegnere.

Il suonatore di tastiere degli anni '70 aveva problemi ben diversi: doveva spostarsi da una tastiera ad un'altra per ottenere i più disparati effetti musicali (per quanto, suonare simultaneamente su due tastiere non sia poi così facile!). Se il brano, anziché dal vivo, era eseguito in sede di registrazione le cose erano



Controllo Centralizzato

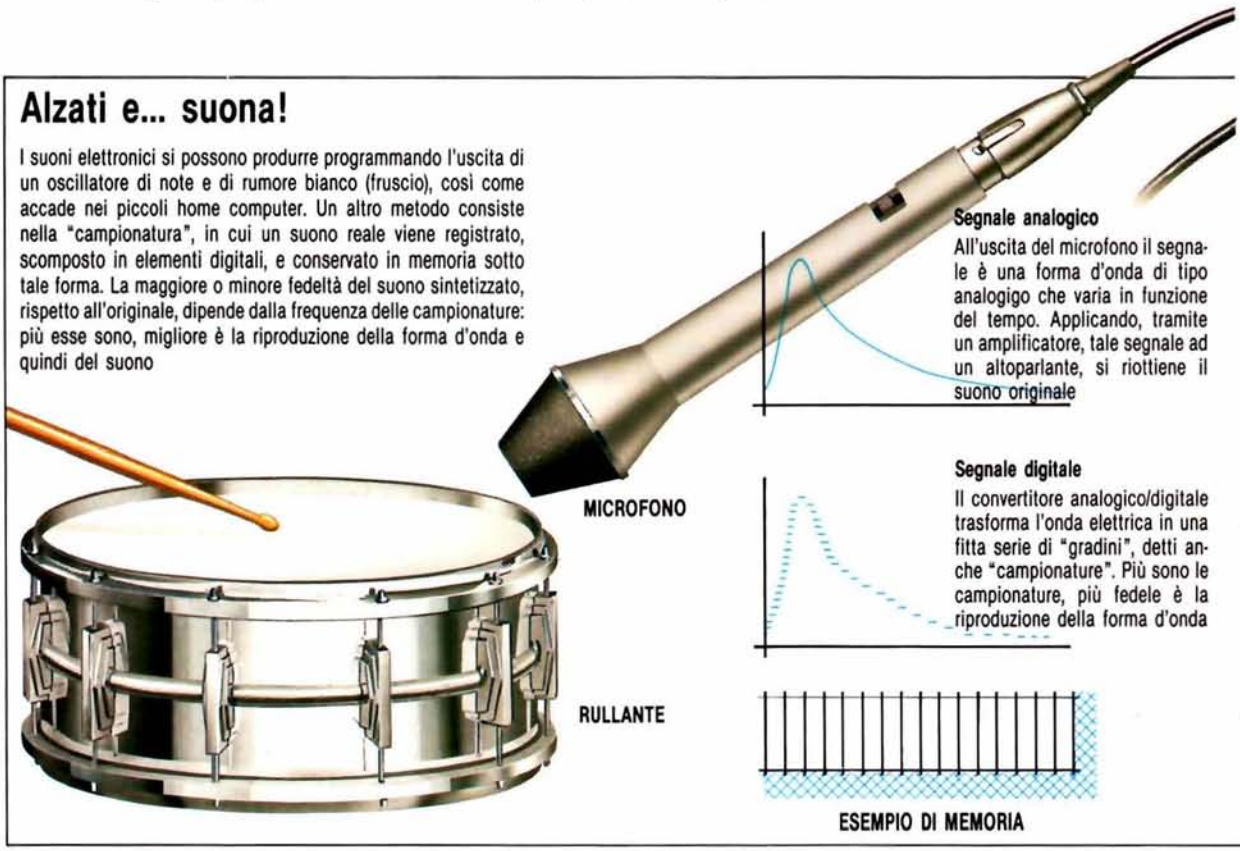
L'interfaccia MIDI MP401 della Roland è un'unità molto sofisticata, usata per collegare un microcomputer a uno o più sintetizzatori digitali. L'unità controlla tutte le funzioni esterne di sincronizzazione, le temporizzazioni interne ed esterne, l'ingresso/uscita del registratore e l'uscita del sintetizzatore. Ciò significa che il computer "ospite" è dedicato unicamente all'invio e alla ricezione di comandi, oltre che al controllo della memoria. La MP401 è progettata per lavorare assieme al PC della IBM, all'Apple II e IIe. Una versione adatta al Commodore 64 è in fase di progetto

ancor più facili: bastava registrare i passaggi, uno alla volta, per montare solo in seguito le varie "piste", sincronizzate tra loro, ed ottenere così il "pezzo" per intero.
Ma l'innovazione più importante, di questo periodo, è stata l'aver incorporato negli stessi sintetizzatori i generatori di suono, rivoluzionando l'uso degli strumenti tradizionali. Un buon esempio è fornito dal sintetizzatore di contrabbasso, dotato di tastiera. Negli anni '60 il genere musicale "Mowtown" era basato su di un largo impiego del basso elettrico e, nei dieci anni successivi, si sono potuti apprezzare alcuni bassisti capaci di virtuosismi degni di una chitarra solista. Di più non si poteva fare, quindi è stato "inventato" il sintetizzatore di basso controllato da tastiera. Ma ciò ha aggravato la situazione del tastierista che, oltre alle proprie funzioni, si è ritrovato a controllare l'andamento del basso, da sincronizzare con l'altro elemento musicale importante: la batteria. Quest'ultima, però, era uno strumento ancora tradizionale e quindi una perfetta coordinazione tra i due strumenti non era sempre possibile. Ecco dunque che, ben presto, sono apparsi nuovi sintetizzatori, completi di una sezione ritmica: non solo trombe, sassofoni e violini, ma anche le percussioni. Per coordinare efficacemente tutti questi strumenti, venne introdotto un nuovo dispositivo: il "sequencer", o sequenziatore, che controlla luscita degli osclillatori.
Questo apparecchio gestisce completamente la riproduzione di intere sequenze musicali in cui compaiono svariati strumenti (tutti simulati dal sintetizzatore, naturalmente). Il principio di funzionamento è abbastanza semplice, in quanto si tratta di usare particolari voltaggi in speciali oscillatori per produrre i suoni voluti. Più alto è il voltaggio più rapide sono le vibrazioni e più alta è la frequenza del suono.
La musica è raramente composta da suoni continui; al contrario, occorrono brevi pause o lunghi silenzi per ritmare l'esecuzione musicale. Queste pause si ottengono facendo sì che l'unità di controllo invii un voltaggio zero all'oscillatore e, ad assicurarsi che ciò avvenga nel preciso momento in cui è richiesto, ci pensa il sequenziatore.

STEVE CROSS

Alzati e... suona!

I suoni elettronici si possono produrre programmando l'uscita di un oscillatore di note e di rumore bianco (fruscio), così come accade nei piccoli home computer. Un altro metodo consiste nella "campionatura", in cui un suono reale viene registrato, scomposto in elementi digitali, e conservato in memoria sotto tale forma. La maggiore o minore fedeltà del suono sintetizzato, rispetto all'originale, dipende dalla frequenza delle campionature: più esse sono, migliore è la riproduzione della forma d'onda e quindi del suono



Nel decennio scorso erano pochi i sintetizzatori dotati di sequenziatori, ma i musicisti seppero sfruttare appieno quanto disponibile. La pulsante "disco music" prodotta da Giorgio Moroder, con Donna Summer, è un valido esempio dell'impiego di un sequenziatore musicale, e anche molti altri complessi hanno sviluppato ed adattato il proprio genere a questa nuova forma di musica. Non è più necessario enfatizzare gestualmente ogni singola nota, stile Rick Wakeman; al contrario, un'intera sequenza (detta anche "riff") può essere avviata o fermata con un semplice comando. Nel frattempo, il musicista può allontanarsi dalla tastiera, ballare, cantare o suonare un altro strumento, per intervenire soltanto quando è necessario cambiare sequenza. Ci sono alcuni complessi (i Thompson Twins ad esempio) il cui stile è fondato radicalmente sull'impiego del sequenziatore.

Ultimamente, sono stati introdotti sul mercato anche i sintetizzatori digitali che hanno riscosso un successo enorme, ampliando maggiormente la gamma di suoni a disposizione del musicista. Uno tra i tanti apparecchi digitali in commercio è la batteria elettronica Linn, la prima ad impiegare suoni veri ma scomposti e ricomposti digitalmente (in questo particolare caso, i suoni originali sono stati registrati da un batterista d'eccezione: Steve Gadd). Nella batteria Linn, i suoni sono memorizzati, né più né meno, come le informazioni su un dischetto da computer: interminabili serie di 0 e 1, conservati su chip di ROM. L'esecutore, tramite alcuni tasti, seleziona il suono desiderato che viene riprodotto con una fedeltà assoluta, come se si avesse di fronte una batteria "vera", ma col vantaggio di poter variare il ritmo e le combinazioni a piacimento, sicuri che non "sgarrerà" un colpo!





per concessione della RECORD MIRROR

Un dinamico duetto

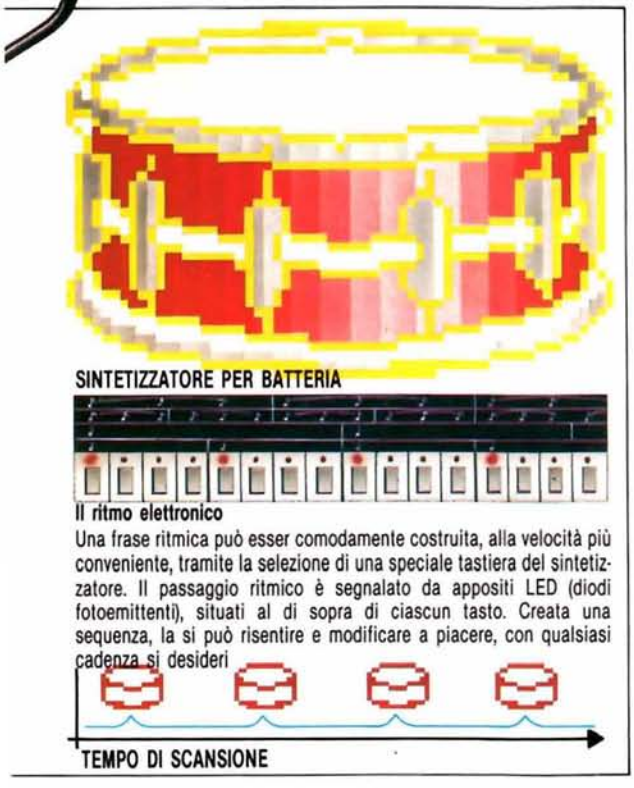
Donna Summer, col suo produttore Giorgio Moroder, è stata una delle prime "vedette" musicali ad avvalersi di ritmi creati al sintetizzatore e controllati da sequenziatori

A questo punto, la differenza tra quanto accade in uno studio di registrazione e sulla pedana di un concerto dal vivo diventa minima assottiglia. Oggi comunque, il lavoro svolto in uno studio di registrazione non è più limitato alla musica, ma interessa anche produzioni video. Un brano musicale senza il relativo "video", ormai è diventato raro.

Se la musica è prodotta con strumenti tradizionali, sincronizzare l'audio al video è un'operazione abbastanza comune, usata anche nei film. Ma se, al contrario, la colonna audio è composta da singoli suoni o sequenze prodotte da un sintetizzatore digitale, i problemi aumentano considerevolmente. Supponiamo che, in una particolare scena di un video, si veda cadere un vaso di fiori. Il musicista ha preparato una sequenza in "crescendo", che culmina in un accordo nel preciso momento in cui il vaso tocca terra, ma pur avviando simultaneamente audio e video di questo apparecchio, si può essere certi che, ai primi tentativi, uno dei due terminerà prima dell'altro. Non resta che, come il buon vecchio Galileo, provare e riprovare! Ciò che servirebbe è un apparecchio in grado di sincronizzare automaticamente il tutto, ossia un'interfaccia digitale.

L'interfaccia è utile anche al suonatore dal vivo, per collegare assieme magari due sintetizzatori di marca diversa ed una batteria Linn, in modo che la combinazione di tutti gli strumenti proceda, ritmicamente, di pari passo (anche il più lieve ritardo di uno rispetto agli altri produce effetti disastrosi).

L'interfaccia digitale MIDI è arrivata come manna dal cielo per chi, dal vivo o negli studi di registrazione, fa largo uso dei sintetizzatori. Questo apparecchio verrà descritto dettagliatamente più avanti in questa stessa sezione.



Il ritmo elettronico

Una frase ritmica può esser comodamente costruita, alla velocità più conveniente, tramite la selezione di una speciale tastiera del sintetizzatore. Il passaggio ritmico è segnalato da appositi LED (diodi fotoemittenti), situati al di sopra di ciascun tasto. Creata una sequenza, la si può risentire e modificare a piacere, con qualsiasi cadenza si desideri

L'elemento umano

L'interfaccia software "uomo-macchina" è di capitale importanza

La programmazione dei computer è stata, per lungo tempo, una materia quasi "misteriosa", alla quale i professionisti si dedicavano con molto impegno e fatica. Precedentemente all'avvento dei microcomputer, con le loro confortevoli tastiere tipo macchina da scrivere, i programmi venivano solitamente immessi, un byte alla volta, tramite l'azionamento di una serie di interruttori situati sul pannello frontale del computer. In alternativa, le istruzioni e i dati venivano immessi tramite schede perforate.

Oggi le cose sono ben diverse: i fabbricanti di computer fanno di tutto per facilitare le operazioni di programmazione, tanto che è stato coniato un apposito appellativo, per indicare la possibilità, anche per i non esperti, di programmare un computer. Si parla, così, di un programma o di un computer "user friendly", ossia dal "comportamento amichevole". Nel 1982, la Commissione Alvey preparò una relazione (A Programme for Advanced Information Technology) in cui l'interfaccia "uomo–macchina" (abbreviata in MMI) veniva indicata, assieme all'ingegneria software, al progetto di circuiti integrati a larghissima scala (VLSI) ed ai sistemi computerizzati basati sullo sviluppo della conoscenza, come una delle quattro maggiori aree per la ricerca e lo sviluppo futuri.

In molte applicazioni, l'interazione tra utente e computer (ossia lo scambio di informazioni tra loro) è di capitale importanza. Il "dialogo" avviene tramite i dispositivi di I/O (Input/Output) del computer, generalmente: la tastiera (in input) e lo schermo (in output). Altre forme di immissione sono i joystick, i mouse e gli schermi "a sfioramento". Canali alternativi di emissione sono: stampanti, generatori di suoni (o di parole) o, addirittura, robot.

Oltre alle eventuali limitazioni imposte dagli elementi hardware del sistema, il dialogo tra utente e macchina è fortemente influenzato dalla qualità del software. Il sistema operativo, ad esempio, controlla molte delle funzioni connesse all'uso della tastiera e dello schermo. Il ritmo della ripetizione automatica dei tasti (e l'intervallo tra una ripetizione e l'altra) viene di solito regolato dal sistema operativo che, talvolta, offre anche la possibilità di usare un "buffer", per accumulare caratteri digitati ad una velocità superiore a quella di visualizzazione. Quest'ultimo rappresenta un discreto vantaggio, poiché sveltisce le operazioni di immissione dati. L'utilizzatore del computer dovrebbe conoscere l'ampiezza del buffer di cui dispone: quello del CP/M, ad esempio, è di un solo carattere.

Esistono occasioni in cui la lunghezza del buffer della tastiera provoca inconvenienti. Supponiamo che l'operatore di un sistema adoperi frequentemente un programma pilotato da menu. Ricordandosi la sequenza delle opzioni da scegliere da ciascun menu, egli le digita rapidamente una dopo l'altra, ad esempio: 2,3,7 e 6. Se il buffer ha una lunghezza adeguata, l'operatore giunge effettivamente all'opzione desiderata, ma se il sistema operativo è capace di "ricordare" un solo carattere alla volta, ci sono forti probabilità che egli si ritrovi nell'opzione 6 (l'ultima digitata) del primo menu!

D'altro canto, anche i buffer troppo lunghi causano problemi: se il tempo necessario ad esaminarli è relativamente lungo, l'operatore può pensare che il sistema si sia bloccato ed è tentato di ripetere il comando o di provare qualche altro tasto, per vedere se il computer reagisce. Ovviamente, a tempo debito, tutti

CP/M



questi caratteri estranei vengono elaborati dal sistema operativo, con risultati talvolta imprevedibili.

Un'altra fonte di perplessità, per l'utente, è la procedura, adottata da alcuni sistemi operativi e da certi interpreti BASIC, con la quale vengono liberate zone di memoria non più utilizzate dal programma in esecuzione. Talvolta, questa operazione (chiamata in gergo "Garbage collection", ossia "Raccolta dei rifiuti") richiede molto tempo, specialmente se il programma elabora molte stringhe, tanto da far credere che esso si sia bloccato. Alcune versioni di BASIC lasciano al programmatore la decisione su quando effettuare questa ope-

razione ed è buona norma che ciò avvenga spesso, purché una scritta appropriata avverta l'operatore di "Attendere un momento".
Le possibilità offerte da alcuni linguaggi di programmazione facilitano le operazioni di input/output. Ad esempio, le funzioni per l'elaborazione delle stringhe, in BASIC, sono molto più sofisticate di quelle in PASCAL. Di conseguenza, l'interfaccia software può raggiungere livelli molto apprezzabili dal punto di vista dell'utilizzatore del programma. I BASIC che prevedono istruzioni per il posizionamento del cursore e per gli effetti grafici consentono di costruire "videate" ordinate e ben disposte, più facili da comprendere. A partire dalle versatili INPUT e PRINT, si possono sperimentare GET$, INKEY$, INPUT$() e via dicendo. Anche PRINT USING è un'istruzione utilissima, anzi spesso indispensabile, per incolonnare valori numerici sullo schermo o sulla stampante.

Micronet 800 "Log-on"



La psicologia dell'utente

L'utilizzatore è l'elemento più imponderabile di tutto un sistema computerizzato. Tuttavia, come accade per le altre componenti, esistono alcune caratteristiche di comportamento abbastanza regolari, di cui va tenuto conto nel progetto di un'interfaccia software.
L'uomo e i computer hanno una qualità in comune: sono entrambi degli "elaboratori di informazioni", ma i primi hanno difficoltà soggettive nell'accumulare nuove informazioni nella loro "memoria di lavoro". È stato calcolato che per la maggior parte delle informazioni elaborate dall'uomo, occorre tenere a mente almeno sette diversi elementi. Le dimensioni di questi elementi dipendono dal loro significato e dalla loro struttura. Ad esempio, se un'informazione consiste in una sequenza di caratteri casuali, ogni elemento corrisponde ad un singolo carattere. Ma se, invece, la sequenza è raggruppabile in combinazioni sensate (una serie di cognomi, per esempio), allora ogni elemento può corrispondere a ciascun gruppo (o nominativo, nel nostro caso). Esistono diversi metodi per aiutare l'utente di un computer a strutturare le informazioni. Uno di questi consiste nell'associazione delle informazioni tramite l'impiego di strutture ben radicate e familiari: è questo il metodo adottato nel Lisa della Apple, in cui lo schermo mostra una serie di oggetti su di una scrivania. Per analogia, una lavagna elettronica può essere arricchita graficamente, facendola sembrare un quaderno quadrettato, con tanto di pagine numerate. Un altro metodo consiste nell'abituare l'utente a strutture

Apple Macintosh



non familiari, di cui, però, vengono spesso presentati esempi e spiegati i dettagli. Lo stesso programma può insegnare come strutturare le informazioni, per ottenere i migliori risultati. Lo svantaggio di questo secondo metodo è che implica notevoli sforzi ed investimenti nella programmazione. Si può dotare un programma di speciali menu esplicativi o "di assistenza", ma questo rappresenta un compromesso non sempre efficace.
Anche la presentazione dei dati, se fatta ricorrendo a strutture ben note all'utente, aiuta a servirsi di un programma con maggior profitto. Spesso, per ottenere ciò, basta un astuto impiego del colore, evidenziando sullo schermo le parti più importanti durante una particolare fase. Un esempio di questa tecnica è dato dai servizi Televideo e Videotel, in cui l'uso di colori abbonda: il titolo e le note a pie' di pagina sono nello stesso colore, lo sfondo in un altro e le scritte, a seconda della loro importanza, in altri colori ancora. Anche particolari parole "chiave" o comandi possono essere evidenziati, colorandoli. Lo scopo di tutto ciò è di facilitare la selezione delle informazioni, consentendo di ignorare intere parti che rivestono un interesse meno immediato. Tuttavia non conviene abusare della colorazione, altrimenti si ottiene l'effetto opposto: l'operatore si ferma a leggere informazioni di scarsa rilevanza, soltanto perché evidenziate. Una buona regola da seguire è di non usare mai più di quattro colori contemporaneamente.

Tre livelli

Queste foto illustrano alcune differenze tra vari sistemi operativi, a seconda del loro livello di "facilità d'uso" (la cosiddetta "user friendliness"). Nella prima immagine, l'utente tenta di comunicare col sistema operativo CP/M, ma in quest'ultimo non è incorporata alcuna funzione di assistenza, cosicché è necessaria una conoscenza piuttosto approfondita dei vari comandi. Il secondo esempio riguarda un sistema pilotato da menu. In questo caso si tratta del menu principale, presentato quando si accede alla rete Micronet 800: le opzioni sono opportunamente evidenziate, e la loro selezione avviene semplicemente con la pressione del tasto numerico corrispondente. Le varie voci del menu non offrono un'approfondita descrizione, quindi occorre una certa dimestichezza col sistema per orientarsi. Infine, un esempio di ciò che appare sullo schermo del Macintosh: una precisa serie di indicazioni grafiche,assieme a chiari e semplici menu

Una porta... da aprire

Come usare la porta del computer per controllare apparecchiature esterne

La maggior parte degli home e personal computer possiede almeno una porta "utente" tramite la quale scambiare informazioni con la memoria e con l'unità centrale dell'apparecchio. Il vantaggio di trattare con segnali digitali è che le condizioni "0" (off) e "1" (on) si possono comodamente rappresentare con impulsi elettrici, rispettivamente di 0 e di +5 volt.
Ogni locazione di memoria, in effetti, contiene otto cellette, ciascuna delle quali ha un livello di 0 o di +5 volt, ed è proprio la configurazione nel suo insieme che stabilisce il numero binario corrispondente. Per convenzione, si dice che una celletta è a livello "alto" se essa è posta a +5 volt, a livello "basso" se si trova a 0 volt. Poiché le porte di input/output si possono considerare come locazioni di memoria speciali, in cui vengono depositati i valori immessi nel computer o quelli da inviare all'esterno, tramite esse siamo in grado di controllare sistemi elettrici esterni al computer.

Esistono due tipi di connessioni per le porte. Nel primo caso, la porta possiede due distinte serie di piedini (otto d'ingresso e otto di uscita), corrispondenti a due locazioni di memoria. Nel secondo caso, vengono usati soltanto otto piedini, sia in ingresso che in uscita. Poiché questa seconda configurazione è quella adottata dal Commodore 64 e dal Micro BBC, essa viene esaminata per prima.

I registri indicatori di direzione

Oltre ad associare una locazione di memoria agli otto piedini della porta, i micro che possiedono porte bi-direzionali di questo tipo sfruttando una seconda locazione, chiamata Registro per la Direzione dei Dati (Data Direction Register, o DDR). Il valore di tale registro, per ciascuna delle otto linee, determina se i dati vengono trasmessi o ricevuti. Vale a dire che, se un particolare bit del DDR vale 1, la corrispondente linea è adoperata in

IL COLLEGAMENTO ELETTRICO

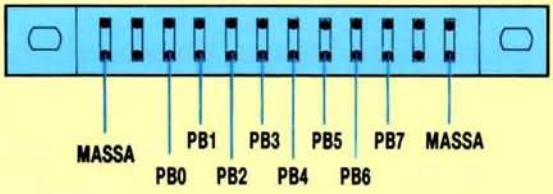
Per prima cosa realizziamo i cavetti, necessari allo scambio di dati col computer, per il Commodore 64 e il BBC. Ecco ciò che occorre:

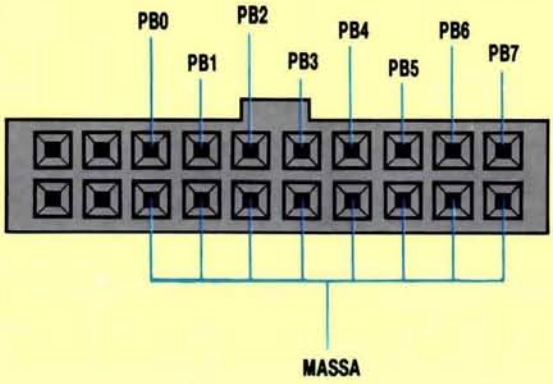
Micro BBC
- Connettore IDC a 20 poli
- 1 metro (circa) di cavo IDC piatto a 20 poli
- Saldatore e stagno

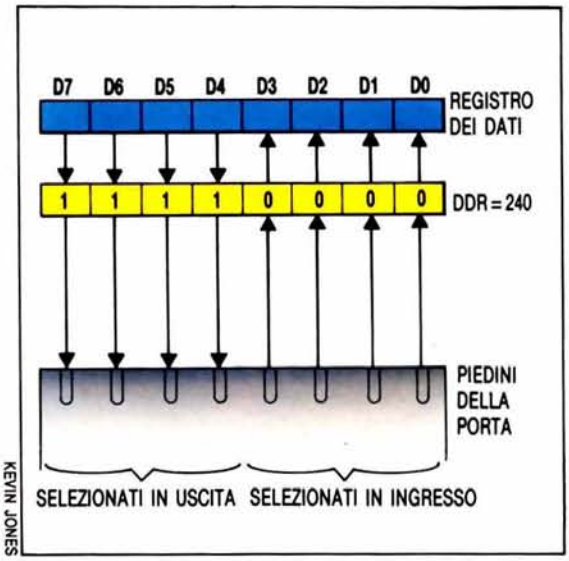
Commodore 64
- Connettore a 24 poli per scheda a circuito stampato
- 1 metro (circa) di cavo piatto a 10 poli
- Saldatore e stagno

I diagrammi mostrano la disposizione delle 10 linee che ci occorrono, sulla presa della porta utente. Il connettore per il BBC qui raffigurato è del tipo a pressione: il cavo piatto va inserito per un paio di centimetri, senza spellarne i capi, come mostrato nella figura alla pagina seguente. Si rispetti la posizione del filo colorato, che serve da riferimento. A questo punto, con moderata pressione, si chiuda il connettore, in modo da fissare saldamente il cavo. All'estremità opposta del cavo si separi ciascun filo, spellando solo i 10 indicati in figura (vedere a tal proposito la pag. 524).
Per il Commodore 64, occorre marcare chiaramente la parte superiore del connettore ("TOP", nella figura alla pagina seguente) per evitare di inserirlo in modo errato. Dopo aver spellato e saldato i 10 capi del cavo come illustrato, si spellino anche le altre 10 estremità di esso.

Poi si usi un tester (vedere a pag. 567) o, se non lo si possiede, direttamente il programma di prova, per verificare il funzionamento del cavo.
Attenzione: le correnti in gioco non sono affatto pericolose, né si rischia di danneggiare il computer, purché si operi con la dovuta attenzione.







uscita; se vale 0, la linea è in ingresso. Per abilitare tutte le linee in uscita, quindi, il DDR va posto a 255 (ossia 11111111 in binario). Analogamente, volendo ottenere otto linee in ingresso, il DDR va posto a zero, mentre tutte le configurazioni intermedie si ottengono con opportuni valori. Un ultimo esempio: volendo usare in uscita le quattro linee più significative della porta e in ingresso le altre, il DDR va posto a 240 (ossia 11110000 in binario).
La locazione riservata al registro dei dati ed il DDR hanno i seguenti indirizzi:

Microcomputer	Registro dati	Registro DDR
Micro BBC	$FE60 (65120 dec.)	$FE62 (65122 dec.)
Commodore 64	$DD01 (56577 dec.)	$DD03 (56579 dec.)

Il seguente programma seleziona tutte le linee della porta in ingresso, e visualizza il contenuto del registro dei dati:

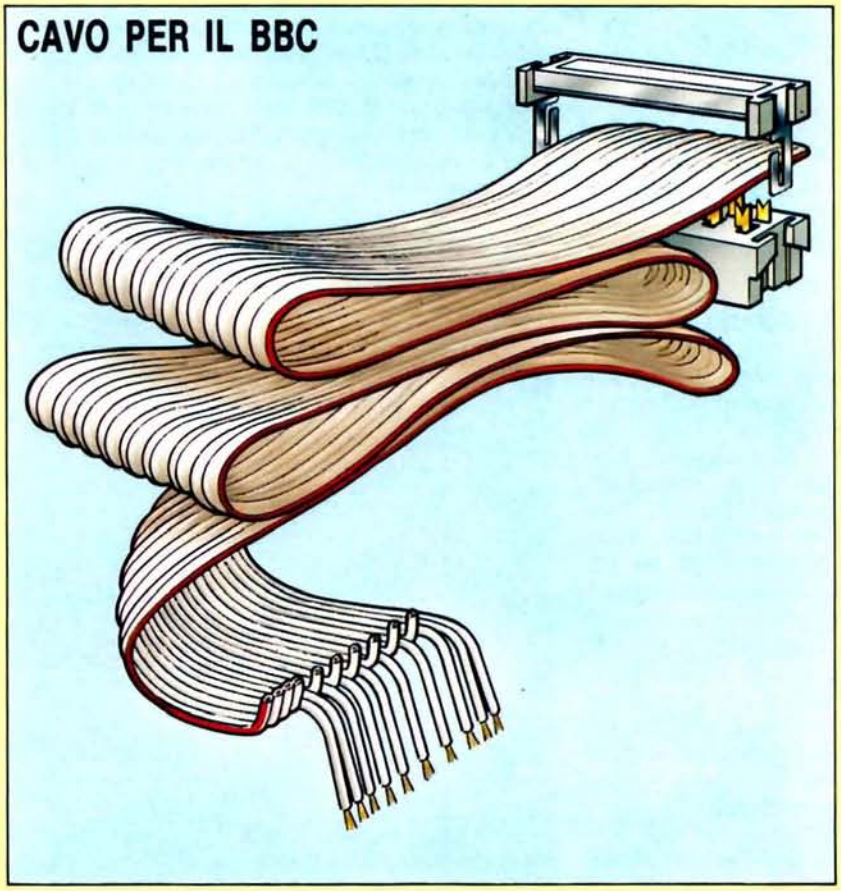
```
4 REM ********** C64 **********************
5 REM *   VISUALIZZA REG. DATI             *
6 REM ********** C64 **********************
10 DIM A$(10):A$(0)="B":A$(1)="A"
20 REGDAT=56577:DDR=56579
30 POKE DDR,0: REM SOLTANTO IN INGRESSO
50 :
100 PE=PEEK(REGDAT):GOSUB 500
150 PRINT"REGDAT =";PE;"=";B$
200 GOTO 100
300 :
499 REM ************************************
500 REM *       CONVERSIONE BINARIA        *
501 REM ************************************
550 B$="":N=PE
600 FOR D=1 TO 8
650 N1=INT(N/2):R=N-2*N1
700 B$=A$(R)+B$:N=N1
750 NEXT D:RETURN
```

Sul Micro BBC, apportare le seguenti modifiche:

```
20 REGDAT=&FE60:DDR=&FE62
30 ?DDR=0
100 PE=?(PE):GOSUB 500
```

Eseguendo il programma, viene visualizzato il normale contenuto del registro dei dati, ossia 255 (sullo schermo rappresentato da AAAAAAAA), a significare che tutte le linee di REGDAT si trovano in condizione "alta", ovvero

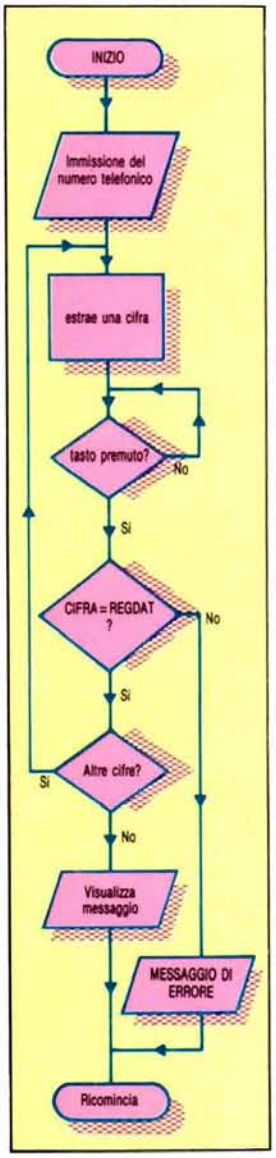
CAVO PER IL BBC







I connettori adatti
Ecco i connettori da usare per il BBC e per il Commodore 64

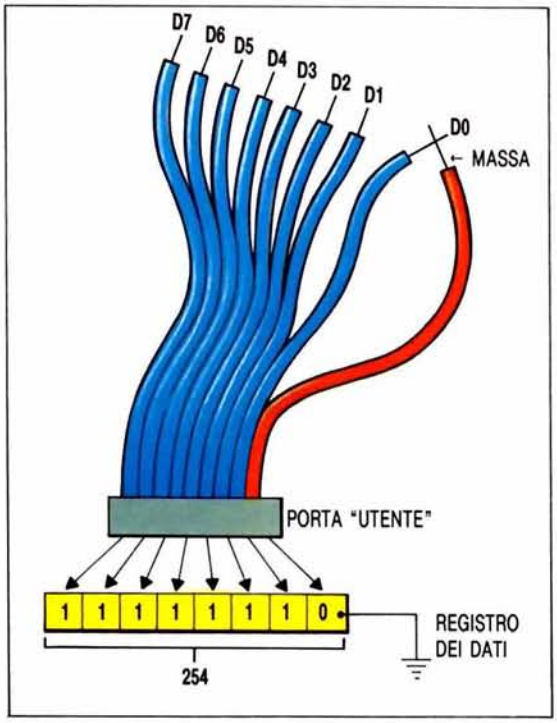


Linee incrociate

Il programma per i numeri telefonici accetta un numero, digitato tramite tastiera, di qualsiasi lunghezza. Quindi, una cifra alla volta, lo confronta con il contenuto del registro associato alla porta "utente". La presenza di eventuali spazi, nel numero, viene ignorata. Prima di esaminare una cifra, il programma attende la pressione di un qualsiasi tasto, per dare il tempo di operare la connessione degli appropriati cavetti

a +5 volt. Se adesso colleghiamo un cavetto alla porta del computer, possiamo modificare lo stato di ciascun canale, e quindi il contenuto di REGDAT.

Nel nostro caso, alla porta sono collegati 10 fili, otto dei quali si riferiscono ai dati, mentre i restanti due sono collegati alla "massa" del circuito elettrico. Mettendo in contatto una delle linee di massa con una di quelle dei dati, la D0 ad esempio, si otterrà che quest'ultima viene posta a livello "basso". Se il programma è in corso d'esecuzione, si vedrà che REGDAT contiene 254 (sullo schermo compare AAAAAAAA). Ciò significa che la linea meno significativa (la D0, appunto) si trova a livello basso, mentre le altre a livello alto. A questo punto si possono provare, di volta in volta, le varie configurazioni, facendo così variare REGDAT da 0 a 255.



Un programma di controllo

La base delle capacità decisionali di un computer è fondata sulla possibilità di confrontare valori numerici o condizioni logiche, deviando, di conseguenza, il percorso del programma. Scrivere un programma in grado di controllare un'apparecchiatura esterna, tramite la porta del computer, è dunque molto facile: basta verificare il contenuto di un registro ed intraprendere la corretta azione in base al risultato. L'esempio di applicazione, riportato più avanti, concerne una verifica sulla corretta selezione di un numero telefonico. La selezione del numero avviene semplicemente ponendo a massa le opportune linee della porta utente, depositando così il corretto valore numerico nel registro dati, per ciascuna cifra del numero telefonico.

Siccome occorre un po' di tempo per collegare a massa i vari fili che provengono dalla porta, per poter selezionare il numero desiderato, il programma attende la pressione di un tasto prima di operare la lettura del registro dei dati. Il diagramma di flusso del programma, molto facile da seguire, è riportato a fianco.

```
100 REM ************ C64 *****************
101 REM *        NUMERI TELEFONICI          *
102 REM ************ C64 *****************
120 :
130 REGDAT=56577:DDR=56579
140 POKE DDR,0:REM SOLTANTO IN INGRESSO
150 :
160 INPUT "NUMERO TELEFONICO";TN$
170 :
180 FOR K=1 TO LEN(TN$)
190 DG$=MID$(TN$,K,1):REM ESTRAE UNA CIFRA
200 IF DG$<>" " THEN GOSUB 500
210 NEXT K
250 :
300 PRINT "-------- MI SPIACE --------"
310 PRINT "  NON RISPONDE NESSUNO AL ";TN$
320 PRINT "  RIPROVARE PIU' TARDI"
350 PRINT : PRINT : RUN
400 :
500 REM *** CONVERSIONE E VERIFICA ***
550 DG=VAL(DG$)
600 PRINT "SELEZIONARE IL NUMERO SULLE LINEE"
650 PRINT "POI PREMERE QUALSIASI TASTO"
700 GET GT$:IF GT$="" THEN 700
750 PE=PEEK(REGDAT):IF PE=DG THEN PRINT
    " OK":RETURN
800 :
850 PRINT "???NUMERO SBAGLIATO???"
900 PRINT TN$ "<>" LEFT$(TN$,K-1);PE
950 PRINT "??? RIPROVARE ???"
999 PRINT : PRINT : RUN
```

Sul Micro BBC, apportare le seguenti modifiche:

```
130 REGDAT=&FE60:DDR=&FE62
140 ?DDR=0
700 A=GET
750 PE=?(REGDAT):IF PE=DG THEN PRINT DG
    " OK":RETURN
```

In questa prima parte, abbiamo visto come immettere dati nel computer tramite la porta utente e come controllare i risultati con un programma. Nelle parti successive, viene spiegato come emettere dati tramite la stessa porta, progettando una semplice unità di controllo.

ESERCIZI DI PROGRAMMAZIONE

1) Se ne scriva uno per simulare il funzionamento di un allarme antifurto
2) Se ne scriva uno per contare il numero d'impulsi ricevuti dalla porta in un determinato lasso di tempo
3) Si modifichi il precedente programma, affinché gli impulsi su ciascuna linea vengano contati separatamente
4) Si scriva un programma per simulare il funzionamento di una serratura a combinazione
5) Se ne scriva uno per cambiare il colore dello schermo, in base agli impulsi immessi nella porta "utente"

Le soluzioni degli esercizi a pag. 1006

Un po' di magia nera

Una commedia computerizzata: un nuovo filone letterario?

Lo sviluppo del *Necromancer* (il Negromante), un programma realizzato dalla Synapse Software, è paragonabile a quello di una commedia teatrale: suddiviso in tre "atti", rappresenta l'eterna lotta tra il Bene e il Male.

L'Overture
Il caricamento del programma, sia sull'Atari che sul Commodore 64, non presenta problemi. Su quest'ultimo, i rumori prodotti durante lo stesso caricamento fanno pensare ad un'orchestra che stia accordando gli strumenti. In entrambe le versioni, la pagina introduttiva è commentata da un motivo musicale, in cui vengono sfruttate al massimo le capacità sonore dei due apparecchi.

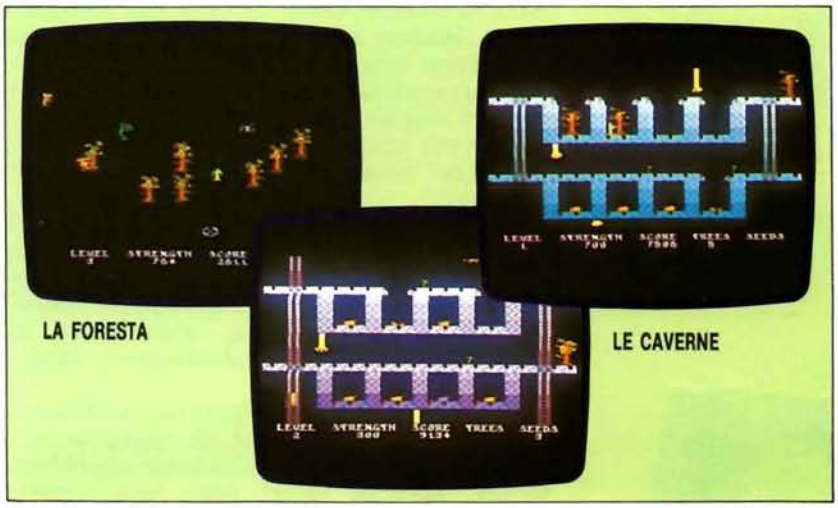
Primo Atto: La Foresta
Cronologicamente il dramma inizia durante i "secoli bui", quando Tetragon (il mago del male) regna incontrastato. Il giocatore interpreta il ruolo dell'Illuminato, ossia di un druido, descritto come "difensore della verità e protettore della razza umana". Si può arguire che il compito non è per niente facile.
La scena iniziale è notturna: un ambiente all'aperto, nel quale appare il druido. È circondato da un alone di stelle che lo proteggono dall'attacco di centinaia di orchi i quali, brandendo lunghi coltellacci, marciano senza sosta attraverso lo schermo. L'azione è ritmata da un azzeccato brano musicale. Col joystick, il giocatore manovra sullo schermo una sorta di bacchetta magica, capace di distruggere gli orchi, accumulando così punti a favore.
La bacchetta magica ha una seconda funzione: posizionandola e premendo il pulsante di fuoco, si "pianta" un nuovo albero. Lo scopo è quello di creare una foresta protettiva, utile in seguito. Naturalmente, la "semina" va protetta dai coltellacci degli orchi e dai tremendi ragni di Tetragon, tenendo sempre d'occhio il livello della propria energia: ogni morso di ragno ne succhia un po', ma è possibile reintegrarla uccidendo i ragni. Dopo cinque fasi di gioco, il primo atto culmina con un pericoloso attacco in massa da parte dei ragni. Al termine, il programma "congela" l'azione, valuta le perdite e trasporta il druido nello scenario successivo.

Secondo Atto: Le Caverne
È in questo luogo che i ragni depositano le proprie uova.
Vi sono cinque diversi livelli, ognuno dei quali contiene due strati di quattro caverne. Negli antri delle grotte, le uova dei malefici ragni sono pronte a schiudersi, come si può intuire dalla loro mutante colorazione. Il giocatore, manovrando abilmente il joystick, deve trapiantare gli alberi piantati durante il primo atto, sopra l'antro attraversato: in questo modo, le radici penetrano nel terreno, facendo crollare la volta della grotta e distruggendo le uova prima che esse si schiudano.
Ad aggiungere un po' di imprevisti, provvedono le "mani del fato": esse calano dall'alto, afferrando qualsiasi cosa riescano ad arraffare, compresi alberi, lo stesso druido o i punti interrogativi. Questi ultimi rappresentano i premi misteriosi: occorre accaparrarsene uno per poter calare una scala e raggiungere il livello inferiore.

Terzo Atto: La Tana del Negromante
L'azione non può che svolgersi in quel camposanto, dove le lapidi indicano le molte tombe del Negromante, da distruggere affinché egli non si reincarni. La lotta tra il bene e il male è molto ardua ed occorre aver accumulato sufficiente energia.
Delle due versioni, quella per l'Atari è di maggior effetto: la grafica è più dettagliata e l'azione più incalzante.

Necromancer: per Atari 400/800 (32K)
per Commodore 64
Editori: Synapse Software (USA)
Autori: Bill Williams (Atari)
Scott e Steve Coleman (Commodore 64)
Joystick: Standard Atari / Commodore
Formato: Disco o cassetta

Una grafica d'effetto
Nel "Necromancer" viene sapientemente combinato l'uso di grafica ad alta risoluzione e quello di sprite, con risultati a dir poco eccellenti. I movimenti, dalla crescita degli alberi alle mosse dei ragni, sono tutti ben curati. Tutto sommato, si tratta di un gioco molto appassionante



LIZ HEANEY

L'Assembly del 6809

La prima parte di una serie dedicata al codice macchina usato da Dragon e Tandy

Un microprocessore è suddiviso in tre parti principali: i registri (che fanno parte della memoria interna al processore), l'unità aritmetico/logica (ALU) che svolge alcune importanti funzioni matematiche sulle locazioni di memoria, ed un'unità di controllo, necessaria affinché tutte le operazioni siano svolte nel corretto ordine e nel momento giusto.
A livello elementare, il microprocessore non fa che reagire ai segnali elettrici applicati alle sue connessioni con l'esterno, modificando il suo stato interno (ossia il contenuto dei registri) oppure rispondendo con l'emissione di altri segnali. I livelli dei segnali elettrici che attraversano il chip e la memoria esterna sono soltanto due ("alto" e "basso"): essi sono pertanto assimilabili a valori numerici binari. Ne consegue che, per programmare il microprocessore, è sufficiente applicarvi una serie di numeri binari (ossia, le istruzioni in codice macchina). Ciò implica una ferrea conoscenza della notazione binaria ed esadecimale (hex), oltre a quella delle istruzioni del particolare microprocessore.
Una CPU ad 8 bit, quale il chip 6809, è in grado di lavorare soltanto con istruzioni formate, appunto, da 8 bit. La gamma di valori possibili, per le istruzioni, va quindi da 0 a 255. Tuttavia, molte istruzioni prevedono l'accesso alla memoria esterna e per individuare una tra le sue 65535 locazioni occorrono "indirizzi" formati da 16 bit (due byte). Ovviamente, il microprocessore elabora tali indirizzi un byte per volta.

I registri del 6809

I registri di un microprocessore possono assumere diverse forme, in base alla loro funzione. Alcuni, riservati esclusivamente ad operazioni interne al microprocessore, non sono accessibili al programmatore. Nel 6809 i registri maggiormente usati sono quattro.
Quello più adoperato, in assoluto, è l'"accumulatore", dove i dati vengono depositati per effettuarvi calcoli o confronti. Ad esempio, una delle istruzioni in codice macchina del 6809, la ADD, somma il contenuto dell'accumulatore a quello di una locazione di memoria specificata nella stessa istruzione. Il risultato viene, per così dire, "accumulato" in questo speciale registro chiamato di conseguenza "accumulatore".
Il "registro indice" è impiegato per modificare il valore di un indirizzo di memoria: ciò consente di esaminare ed elaborare facilmente una serie di valori, ossia una tabella. Quando un'istruzione si riferisce al registro indice, il contenuto di quest'ultimo viene sommato all'indirizzo specificato, ottenendo quello relativo alla particolare locazione che si intende esaminare. Per scorrere una tabella, ad esempio, basta specificarne l'indirizzo "base" e poi incrementare il registro indice. Poiché il contenuto dei registri indice è formato da indirizzi, è normale che la loro dimensione sia di 16 bit, anziché di 8.
Lo "stack pointer" è il registro che contiene l'indirizzo della locazione "in cima" allo stack (una particolare zona di memoria che serve da deposito temporaneo). Questo metodo di accedere ai dati è particolarmente veloce ed efficiente. Lo stack è usato per depositarvi il contenuto dei registri del microprocessore quando, ad esempio, viene fatta una chiamata ad una subroutine. Tale operazione, indicata col termine "push" (letteralmente "spingi") è complementare a quella di "pop" (estrai), che serve invece a recuperare i valori "appoggiati" sullo stack. In sostanza, lo stack pointer o "puntatore dello stack", contiene sempre l'indirizzo dell'ultima informazione depositata nello stack. Anche questo registro, dovendo contenere un indirizzo, è a 16 bit.
Il quarto registro è di grandissima importanza, ma il suo funzionamento è automatico: si tratta del contatore di programma (Program Counter o, brevemente, PC). Esso contiene l'indirizzo della "prossima" istruzione da eseguire nel programma in codice macchina. Il microprocessore preleva l'istruzione dall'indirizzo specificato da questo puntatore ed esegue l'istruzione. Normalmente, il program counter viene automaticamente incrementato per puntare all'istruzione successiva. Modificare il valore di questo registro, significa alterare il normale accesso sequenziale alle istruzioni, il che equivale ad alterare il percorso del programma. Nel codice macchina esistono particolari istruzioni corrispondenti alla GOTO del BASIC: la JMP (salto) e la BRA (diramazione), che depositano nel program counter l'indirizzo specificato assieme alla stessa istruzione.
Esiste un quinto tipo di registro, il cui funzionamento, però, differisce da quello di tutti gli altri: il registro "di condizione" del processore. Conviene considerarlo come una sequenza di singoli bit, ognuno dei quali corrisponde ad una particolare condizione del microprocessore. Per esempio, uno dei bit è usato per indicare se il risultato di una somma o di un confronto sull'accumulatore vale 0. Un'applicazione tipica di questo bit (chiamato, in effetti, "flag di zero") è quando si esaminano i



IAN McKINNELL

valori di una tabella alla ricerca di un valore zero: ogni byte della tabella viene copiato nell'accumulatore. Poi esso viene confrontato col valore 0 e se la condizione è soddisfatta, allora il "flag di zero" è automaticamente modificato. L'istruzione successiva al confronto verifica lo stato di tale flag, consentendo di terminare la ricerca non appena si è trovato il valore cercato. Ciò permette di costruire strutture decisionali (le IF...THEN del BASIC) o cicli iterativi (FOR...NEXT, WHILE...WEND, REPEAT...UNTIL del BASIC).

Molti microprocessori prevedono uno speciale uso della "pagina zero", che di norma contiene le prime 256 locazioni di memoria (da 0000 a 00FF in hex). Per accedere ai valori qui depositati basta un indirizzo formato da un solo byte (il byte più significativo è sempre 00), quindi le operazioni sono decisamente più rapide rispetto a quelle che implicano un indirizzo completo a 16 bit.

Nel 6809 questo principio è ampliato dalla presenza di un "registro di pagina diretta", in cui viene depositato il byte più significativo dell'indirizzo di una pagina di memoria. Alterando opportunamente tale valore si può usare qualsiasi pagina di memoria (ossia 256 byte consecutivi) come se si trattasse della pagina zero, con tutti i vantaggi che ne conseguono.

Un programma in codice macchina è una sequenza di byte: alcuni rappresentano istruzioni, ma altri costituiscono i dati o gli indirizzi utilizzati dalle stesse istruzioni. Alcuni programmatori sono capaci di maneggiare con destrezza anche questi codici numerici, ma i più preferiscono adoperare uno speciale linguaggio, l'Assembly, nel quale ad ogni istruzione è associato un codice letterale, molto più facile da ricordare, e ad ogni indirizzo è abbinata una "etichetta", anch'essa molto più facile da ricordare e da individuare in un listato. In Assembly, ad esempio, per copiare il contenuto di una locazione di memoria nell'accumulatore basta scrivere:

```
LOCAZ FCB 0
```

per riservare ed individuare il byte di memoria sul quale intendiamo operare. LOCAZ è l'etichetta che noi abbiniamo alla locazione (l'indirizzo effettivo viene calcolato in seguito dal programma assemblatore, che converte l'Assembly in codice macchina) mentre FCB è una "direttiva" (ossia non una vera e propria istruzione), con la quale si comunica all'assemblatore che l'indirizzo di questa locazione va sostituito ogni volta che l'etichetta ricorre nel programma. Per inciso, qui la locazione viene posta inizialmente a zero. Più oltre, nel listato, basta usare il formato:

```
LDA LOCAZ
```

per copiare il contenuto di LOCAZ nell'accumulatore.

Come accennato, per convertire automaticamente il linguaggio Assembly in effettivo codice macchina, occorre uno speciale programma chiamato Assembler. La conversione non è molto complicata, considerata la corrispondenza univoca tra le istruzioni espresse in Assembly e quelle in codice macchina.

Nella prossima parte dedicata a questo argomento, verrà esaminata la struttura interna del processore 6809 e l'impiego di altre istruzioni.

Chi usa il 6809

I due microcomputer più diffusi, che adoperano il 6809 come unità centrale, sono il Dragon (da 32 o 64 Kbyte) ed il Tandy Color Computer. Tuttavia esistono numerosi apparecchi sperimentali, abbastanza diffusi nelle scuole e nelle università, che hanno questo microprocessore. Internamente, i circuiti del Dragon e del Tandy Color Computer sono pressoché identici

La Psion

Questa software house ha prodotto una grossa quantità di programmi per la Sinclair

Un sistema integrato

L'"Xchange" della Psion consiste in una serie di programmi applicativi, per uso commerciale, perfettamente integrati l'uno con l'altro. Essi sono stati sviluppati in modo analogo a quelli prodotti per il QL della Sinclair. Il package include un word processor, una lavagna elettronica, un database ed un eccellente programma grafico. I programmi, acquistabili tutti assieme o separatamente, sono stati prodotti in più versioni: per il PC ed il PC XT della IBM, per l'Apricot e l'Apricot XI della Act e per il Sirius I. Sono previste versioni anche per il Macintosh della Apple e per il Rainbow della Digital



IAN McKINNELL

La Psion è stata fondata nel 1981 dal Prof. David Potter, un docente dell'Imperial College di Londra. Il primo successo commerciale della Psion fu un package di quattro programmi per l'allora nuovissimo ZX81: un simulatore di volo, un gioco di "backgammon", una lavagna elettronica (il Vu-Calc) ed un database (il Vu-File), tutti quanti sviluppati da Charles Davies e Colly Myers.

Questa piccola selezione di programmi, di ottima fattura, contribuì immediatamente all'affermazione della Psion. Quando, nel 1982, la Sinclair Research cercò una software house in grado di produrre una serie di programmi dimostrativi per lo Spectrum, la scelta cadde inevitabilmente sulla Psion.

In effetti, molta della popolarità goduta dai computer della Sinclair deriva probabilmente dalla grande diffusione dei programmi sviluppati da questa ditta. Ad esempio, del solo simulatore di volo (in versione ZX81 e Spectrum) sono state vendute oltre 500.000 copie. A livello mondiale, si calcola che la Psion abbia venduto oltre tre milioni di cassette. Ultimamente, la Psion si è dedicata ad una "linea" di programmi chiamata *Xchange*, manifestando una forte intenzione di diversificare la propria produzione.

Questa software house è sempre stata all'avanguardia: è stata una delle prime ad adottare il metodo della "compilazione incrociata" (ossia la stesura e la compilazione di software, tramite speciali programmi, su di un apparecchio diverso da quello al quale il software è destinato).

Il programma introduttivo *Horizons*, per lo Spectrum, è stato in effetti preparato su di un TRS-80 della Tandy. Attualmente, per lo sviluppo del proprio software, la Psion impiega due minicomputer VAX 750. Tutti i programmi che accompagnano il QL della Sinclair sono stati compilati su questi potenti apparecchi.

Inoltre, la Psion ha già iniziato a produrre programmi anche per il PC della IBM, per l'Apricot, per il Sirius, per il Rainbow della Digital e per il Macintosh della Apple. Similmente al pakage per il QL, l'*Xchange* prevede una lavagna elettronica, un database, un programma grafico ed un word processor. Ciò che distingue questi programmi da altri dello stesso genere, stando alla Psion, è l'enorme facilità con la quale si possono trasferire i dati da un programma all'altro.

Le attività della Psion non si limitano al software: un esempio è costituito dall'Organiser, un piccolo gioiello di computer, per di più tascabile (vedere a pag. 923). In questo caso, la Psion ha ribaltato il concetto tradizionale di progetto: è partita da un'applicazione software per poi produrre un apparecchio che si adattasse agli scopi del programma.

Attualmente, esistono soltanto tre cartucce applicative per l'Organiser (una scientifica, una matematica ed una finanziaria). Ma i progetti, in merito, sono molti.

La Psion sta estendendo, con successo, la propria fascia di mercato anche ai paesi d'oltre oceano. Inoltre, considerando che la Sinclair ha promosso una campagna di vendite perfino nei paesi dell'Est (con una prima mandata di 400 ZX81 in Cecoslovacchia), si può esser certi che la Psion seguirà a ruota, non essendo nuova alla conversione dei propri programmi in altre lingue, diverse dall'inglese.



Il fondatore della Psion

Il Prof. David Potter, ex-docente di Fisica presso l'Imperial College di Londra è il fondatore della Psion

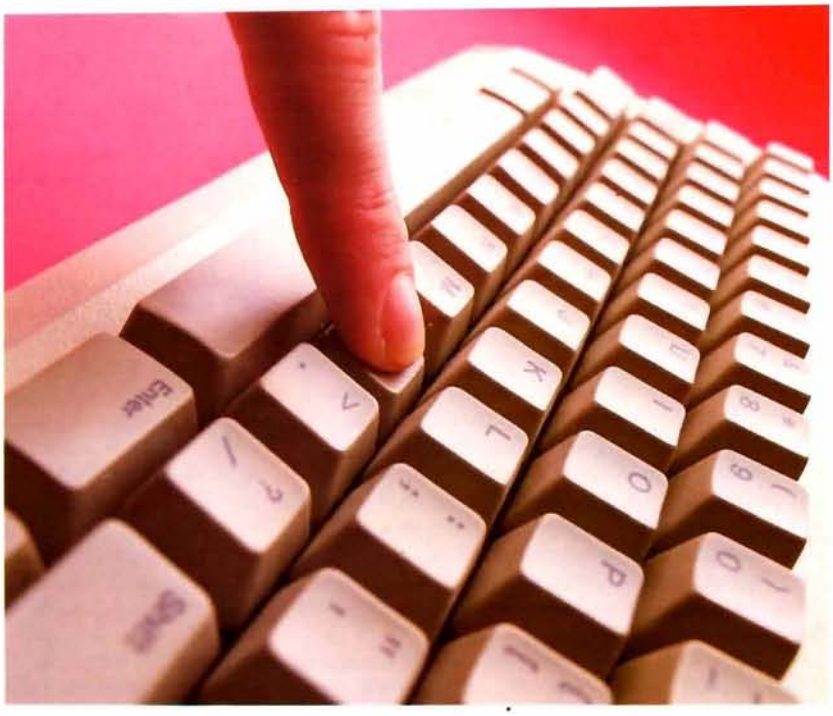
Confezione "su misura"

La forma fisica di uno strumento di lavoro influisce notevolmente sul rendimento

La crescente diffusione di home e personal computer, in casa e al lavoro, ha spinto ad approfondire i problemi di carattere ergonomico connessi al loro uso. È noto come colori e suoni possano influenzare le prestazioni lavorative degli individui. In alcune nazioni, ad esempio, i centralini telefonici trasmettono musica all'utente collegato, per rendere l'attesa piacevole e rilassante. Anche le differenti combinazioni di colore sembrano provocare reazioni diverse: il colore di una stanza può influire sulle capacità lavorative di una persona: il giallo viene considerato "stimolante", il blu ed il verde "rilassanti" mentre la combinazione marrone-grigio, tipica di molti uffici, è spesso considerata "deprimente". Si è visto che anche gli utenti di home e personal computer giudicano gli schermi verdi più gradevoli e riposanti dei comuni schermi in bianco e nero, considerati faticosi e stressanti per la vista; anche gli schermi blu e giallo sono molto apprezzati, mentre quelli azzurro verde hanno scarso successo.

Queste considerazioni possono sembrare trascurabili e di scarsa importanza a chi lavora su un home computer, ma gli studiosi di ergonomia hanno più volte riscontrato che se un individuo non è "in sintonia" col proprio ambiente di lavoro è spesso soggetto ad un maggior affaticamento della vista e della schiena (sebbene spesso questi malesseri non vengano associati ai colori delle pareti o allo sfondo di uno schermo). Chi lavora coi computer è particolarmente sensibile all'ambiente in cui opera ed alla disposizione degli apparecchi. Un interessante studio ergonomico, eseguito in un ufficio dove gli operatori di computer lavoravano su terminali disposti lungo le pareti della stanza, ha rivelato che la loro scarsa efficienza e l'alta incidenza degli errori dipendevano dall'isolamento e dallo stress causato dalle condizioni ambientali. Si è visto che, collocando invece gli operatori al centro della stanza e facendoli sedere l'uno di fronte all'altro davanti a terminali a basso profilo, l'atmosfera dell'ufficio diveniva molto più gradevole e gli standard lavorativi risultavano superiori.

Spesso, l'introduzione di procedure computerizzate provoca un calo nell'impegno individuale, rendendo il lavoro privo di interesse, ripetitivo e scarsamente stimolante. Oggi, gran parte della ricerca degli analisti di sistema mira a sviluppare tecniche e procedure che stimolino l'interesse e le motivazioni individuali di chi opera quotidianamente coi computer. Un grosso contributo a questo lavoro di ricerca proviene dall'ergonomia.



IAN McKINNELL

Nel progettare un sistema di elaborazione che tenga realmente conto degli aspetti ergonomici, occorre considerare l'operatore come un'importante componente del sistema stesso, con proprie caratteristiche operative e ben precisi livelli di tolleranza. Quindi le necessità fisiche, percettive ed ambientali dell'elemento umano devono avere la stessa importanza di quelle caratteristiche economiche o tecniche del sistema, come la capacità di un bus d'indirizzi o la velocità di clock.

Anche i programmatori possono trarre vantaggi dagli accurati studi e dalle ricerche rivolte al loro specifico settore di lavoro. La programmazione richiede infatti notevoli capacità di concentrazione, una mente logica ed una minuziosa cura dei particolari, doti che non tutti hanno.

Per anni gli psicologi hanno esplorato la mente umana, considerata come un dispositivo d'informazione ed elaborazione, sviluppando teorie ben precise (sebbene ancora incomplete) sulla struttura mnemonica del cervello e sulla sua capacità e velocità di elaborazione. Questi studi vengono ora impiegati nell'industria, per progettare sistemi di controllo e nuove tecniche per l'immagazzinamento delle informazioni e per facilitare il lavoro di chi opera sugli elaboratori elettronici. Grazie a questi studi, si sta cercando di sviluppare sistemi "a misura d'uomo" e non

Tra passato e futuro

La principale interfaccia tra utente e computer è la tastiera che, nel tempo, si è evoluta secondo le indicazioni degli studi ergonomici (dai tasti "scolpiti" delle tastiere a sezione concava, alle tastierine numeriche ed ai visualizzatori a cristalli liquidi). Ma il difetto più grosso della tastiera, cioè la disposizione QWERTY dei tasti, ha resistito ad ogni modifica, poiché la maggior parte degli utenti rifiuta di adeguarsi ad una nuova configurazione. Simili atteggiamenti emotivi ed irrazionali sono spesso il maggior ostacolo all'introduzione di nuove apparecchiature e tecniche più avanzate

strutture nelle quali l'uomo sia costretto ad adattarsi all'hardware e al software usato sugli apparecchi.
Negli ultimi anni gli studiosi di ergonomia hanno anche indagato su come l'individuo reagisca a particolari aspetti del software, le cosiddette "interfacce utente". Solo fino a poco tempo fa, i futuri informatici venivano preparati emotivamente e professionalmente ad accettare le difficoltà e i disagi derivanti dalla propria attività ed avevano bisogno di un'elevata conoscenza tecnica per poter dominare la "potenza" dei computer. Oggi, invece, gli utenti dei computer possono essere persone comuni, con scarsa disposizione a lavorare su macchine complesse o a studiare difficili linguaggi di programmazione. I programmi per la gestione di archivi (database) permettono di registrare milioni di informazioni, ma spesso le loro capacità potenziali non vengono sfruttate a fondo, in quanto il recupero delle informazioni implica la familiarità con complicati "dialetti", quali il liguaggio SQL. È una speranza comune, tanto ai ricercatori quanto agli utenti, che la prossima generazione di sistemi basati su linguaggi naturali permetta di comunicare con le normali espressioni della lingua parlata. Con tali sistemi, dovrebbero esistere appositi programmi per tradurre le richieste dell'utente nel linguaggio di comando del sistema, magari chiedendo automaticamente all'operatore di precisare o articolare meglio le domande e permettendo allo stesso sistema di ampliare le proprie risposte.
Il software può offrire diversi tipi di assistenza all'utente; le ricerche nel campo dell'intelligenza artificiale hanno portato allo sviluppo di programmi capaci di spiegare come il sistema elabori le proprie decisioni, e già si pensa di adoperare tali tecniche per creare "moduli informativi" che consentano un vero e proprio dialogo tra l'uomo e la macchina. Il software "intelligente" pone però un quesito sul senso della parola "spiegazione": ad un programmatore può servire una spiegazione sull'organizzazione dei dati o sulle tecniche di elaborazione adottate nel sistema, ma a chi si occupi di gestione aziendale possono occorrere chiarimenti di carattere commerciale. Queste diverse esigenze creano difficoltà di genere semantico e linguistico agli studiosi di ergonomia e di informatica.
Un'altra importante branca nello studio delle "interfacce utente" è nota come "indagine sui modelli". Ogni individuo segue modelli o schemi mentali propri, che poggiano sull'esperienza personale o su stereotipi di carattere razziale, sociale o politico; l'interpretazione della realtà dipende molto da questi canoni. Gli studiosi hanno riscontrato che molta gente si avvicina al mondo dei computer con un'idea preconcetta e fuorviante della capacità e del funzionamento di questi apparecchi: i computer vengono spesso considerati come macchine estremamente intelligenti e capaci di eseguire qualsiasi tipo di lavoro meglio di un essere umano. Di fronte ad un elaboratore, le risposte impersonali della macchina (specialmente se vengono commessi errori nell'immissione di dati o richieste) provocano in queste persone un certo disagio, e fanno sì che il computer venga considerato un' "entità" fredda e quasi ostile. Naturalmente simili valutazioni sono prive di senso se riferite ad una macchina. Tuttavia, un rapporto così personalizzato influisce sull'intero sistema interattivo e spesso porta ad una vera e propria incomunicabilità tra l'uomo e la macchina.
I programmi che cercano di rendere meno aspro questo conflitto rischiano talvolta di aggravare il problema, facendo apparire il sistema più intelligente di quanto in realtà non sia. Per esempio, un messaggio di questo tipo: «SALVE MARIO, SONO IL "GENIO" DEL TUO DATABASE», può indurre l'utente a rispondere sullo stesso tono e in modo superficiale, magari con conseguenze catastrofiche, seguite da un forte senso di frustrazione.
Affinché il rapporto tra il computer e chi lo adopera sia veramente interattivo, il software deve rispondere alle esigenze psicologiche e di lavoro dell'utente e metterlo in condizione di eseguire il proprio compito con naturalezza: qualcosa di più della semplice inclinazione ergonomica di uno schermo o del giusto colore per le unità a dischetti!



Ambienti di lavoro
Le capacità lavorative di ogni individuo vengono notevolmente influenzate da alcuni importanti fattori fisici come l'altezza del piano di lavoro, la temperatura e la rumorosità ambientale. Tuttavia, anche fattori una volta ritenuti secondari come il colore, la luce e la disposizione spaziale degli oggetti giocano un ruolo fondamentale nell'ambiente di lavoro, soprattutto dove si fa uso di computer. Quando l'analista di sistema progetta un nuovo centro di elaborazione, oltre a scegliere l'hardware ed il software, deve tener conto di tutti questi fattori

FLESSIBILITÀ DEL FORTH

Il FORTH, un linguaggio che molti considerano il più probabile sostituto del BASIC in un prossimo futuro, è stato creato da Charles Moore, un astronomo del Kitts Peak Observatory, in Arizona. Moore, lavorando sulla programmazione in FORTRAN delle apparecchiature che controllano i movimenti del telescopio, si accorse che questo linguaggio influenzava in modo negativo l'elaborazione dei dati e si addiceva al controllo di apparecchiature esterne al sistema. Decise allora di sviluppare un nuovo linguaggio che non presentasse questi problemi. Il FORTH è in effetti molto diverso dai linguaggi rigidamente strutturati come il FORTRAN: la sua flessibilità permette infatti di impiegarlo nelle più svariate applicazioni, come se si usasse un dialetto sempre nuovo. Queste apprezzabili doti vengono in parte bilanciate dalla notevole complessità e difficoltà di interazione

Una porta per comandare

Come utilizzare i segnali emessi dalla porta di ingresso/uscita del computer

Nella maggioranza delle applicazioni coi microprocessori, il termine "buffer" ha il significato di memoria di accumulo temporanea, nella quale vengono depositate le informazioni da trasferire da una parte all'altra del sistema di elaborazione. Nell'elettronica analogica invece, questo termine indica un circuito che protegge un apparecchio dai possibili danni arrecati da un secondo apparecchio ad esso collegato. Volendo pilotare, attraverso la porta "utente" del computer, motori "passo passo" o altre apparecchiature elettriche occorre proteggere i delicati circuiti interni del micro dalle apparecchiature poste sotto il suo controllo.

Il chip di input/output situato all'interno del microcomputer lavora a tensioni di 0 e +5 volt ed usa correnti di pochi milliAmpere (mA). È quindi necessario assicurarsi che, tramite le linee della porta utente, non giungano al processore tensioni superiori a +5 volt, o correnti non comprese tra 30 e 40 mA.

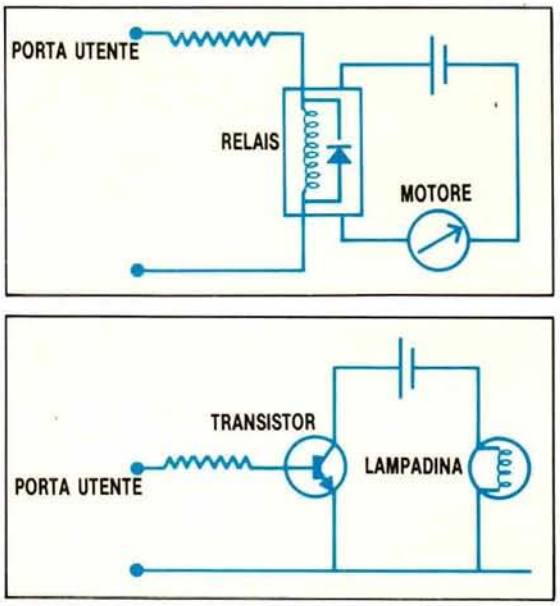
In precedenza (vedere a pag. 994) abbiamo visto come cambiare il contenuto del registro dei dati, mettendo in contatto i fili provenienti dalla porta utente; nel collegare a massa le celle del registro prima di eseguire questa operazione, le correnti in gioco erano talmente basse da non richiedere la presenza di un circuito di protezione. Desiderando collegare altri tipi di apparecchi, il micro deve essere sempre provvisto di un circuito di protezione. Tali circuiti possono essere di diverso tipo. Supponiamo di voler proteggere il sistema, limitando la corrente assorbita da uno qualunque dei piedini della porta utente ad un massimo di 20 mA. In questo caso si può usare un relais, in grado di accendere e spegnere l'apparecchio collegato, inserendo nel circuito anche una resistenza di alimentazione del relais. Se il circuito funziona a +5 volt ed usa una corrente non superiore a 20 mA, il valore della resistenza si calcola applicando la Legge di Ohm (tensione = corrente x resistenza):

V = I x R
R = V / I
R = 5 / 0,02
R = 250 Ohm

V = *tensione in Volt*
R = *resistenza in Ohm*
I = *corrente in Ampere*

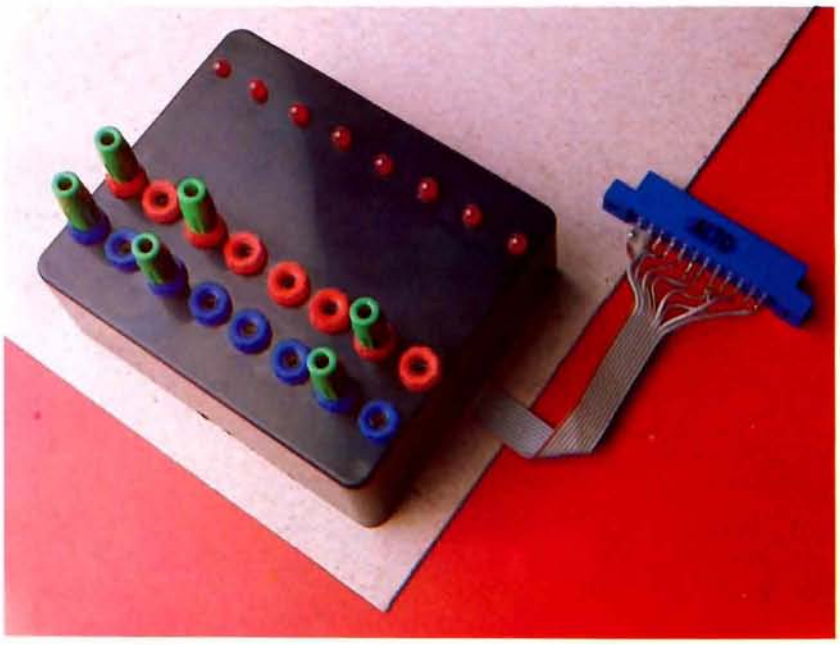
Come alternativa, possiamo applicare la corrente proveniente dal piedino della porta utente ad un transistor, che chiuderà un circuito esterno collegato a una lampadina.

Il dispositivo di protezione (buffer box) che intendiamo realizzare sfrutta questa capacità di commutazione, caratteristica dei transistor, per proteggere la porta utente. Nel nostro caso questo è il sistema più conveniente, in quanto i circuiti per tutte e 8 le linee sono disponibili su un unico chip.

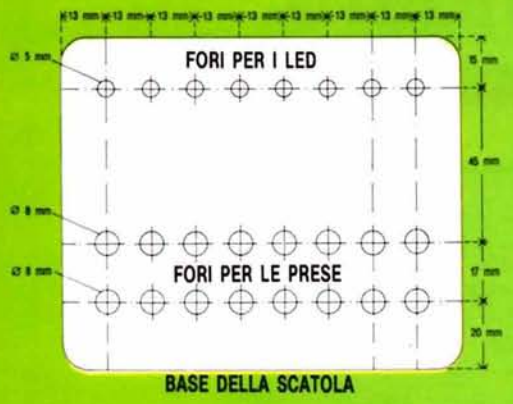


Dopo aver protetto la porta utente, possiamo connettere all'interfaccia una serie di moduli che consentano di collegarla ad altri apparecchi. Questi moduli saranno in grado di controllare l'accensione di LED, di motori a basso ed alto voltaggio e di relais, collegati ad appa-

Interfaccia di protezione
L'apparecchio si collega alla porta utente del Commodore 64 o del Micro BBC, usando il cavetto descritto in precedenza (vedere a pag. 996). L'interfaccia protegge il computer da pericolosi sbalzi di tensione nella corrente di input/output. I LED indicano lo stato delle linee in uscita dalla porta utente, mentre gli spinotti e le prese funzionano come interruttori sulle linee in ingresso



IL QUADRO DI COMANDO



I LED e le prese nere e rosse vengono montate sul quadro secondo lo schema qui rappresentato. Spesso la plastica dei contenitori in commercio è piuttosto morbida; per evitare rotture o scheggiature, nel praticare i fori con un trapano, conviene applicare sulla superficie da forare una striscia di nastro adesivo ed incidere la plastica con un punteruolo o un trapano a punta sottile. Il diametro dei fori è di 5 mm. per i LED e di 8 mm. per le prese

IL MATERIALE OCCORRENTE

L'elenco dei pezzi necessari per costruire l'interfaccia riporta i valori e le caratteristiche di ciascun componente, facilmente reperibile presso qualsiasi rivenditore di materiale elettronico. Le dimensioni della scatola usata come contenitore possono variare, ma il disegno del nostro progetto si adatta con precisione alle misure sopra riportate

Quantità	Descrizione
8	Resistenze 4700 Ohm da 0,5 watt
8	Resistenze 240 Ohm da 0,5 watt
1	Condensatore elettrolitico da 1 μF
1	Condensatore da 0,1 μF
8	LED (diodi fotoemittenti) rossi
8	Diodi 1N4148
1	Raddrizzatore a ponte W005
3	Integrati 7407 (buffer esadecimali)
1	Regolatore di tensione μA7805UC
1	Scheda prestampata Veroboard a 36 piste x 50 fori
3	Zoccoli per chip DIL a 14 piedini
1	Rocchetto di stagno per saldature
1	Metro di cavo a nastro a 12 linee
8	Prese nere (diametro 4 mm.)
8	Prese rosse (diametro 4 mm.)
8	Spinotti neri (diametro 4 mm.)
8	Spinotti rossi (diametro 4 mm.)
1	Presa d'alimentazione da 2,1 mm. per circuiti stampati
1	Connettore minicon a 12 canali per porta utente
1	Scatola di plastica da 115 x 95 x 37 mm.

Nuovi componenti

L'elenco contiene alcuni componenti già descritti in precedenza (vedere alle pagg. 595 e 618) ed altri che incontriamo per la prima volta. La presa d'alimentazione, il raddrizzatore a ponte e il regolatore di tensione fanno parte del dispositivo di alimentazione. L'interfaccia accetta in ingresso qualsiasi tipo di trasformatore, purché la sua tensione di output sia compresa tra 7 e 25 volt (CA o CC).
La presa a 12 canali è per la porta di input/output dell'interfaccia e viene attraversata dalle informazioni scambiate con le periferiche collegate



Raddrizzatore a ponte
Trasforma una qualsiasi tensione in ingresso (CA o CC) in una corrente continua (CC) di polarità nota



Regolatore di tensione
Stabilizza la tensione applicata al suo ingresso e proveniente dal raddrizzatore a ponte, trasformandola in una tensione continua a 5 volt



Presa d'alimentazione
Accoglie uno spinotto da 2 mm., simile a quelli usati per l'alimentazione esterna di molti piccoli registratori e computer (lo Spectrum, ad esempio). Essa viene montata direttamente sulla scheda del circuito

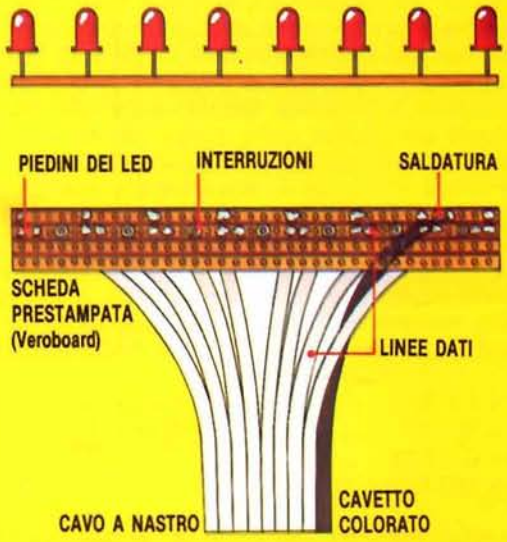


Connettore minicon
Viene fissato alla scheda e facilita il fissaggio dei cavetti provenienti dall'esterno

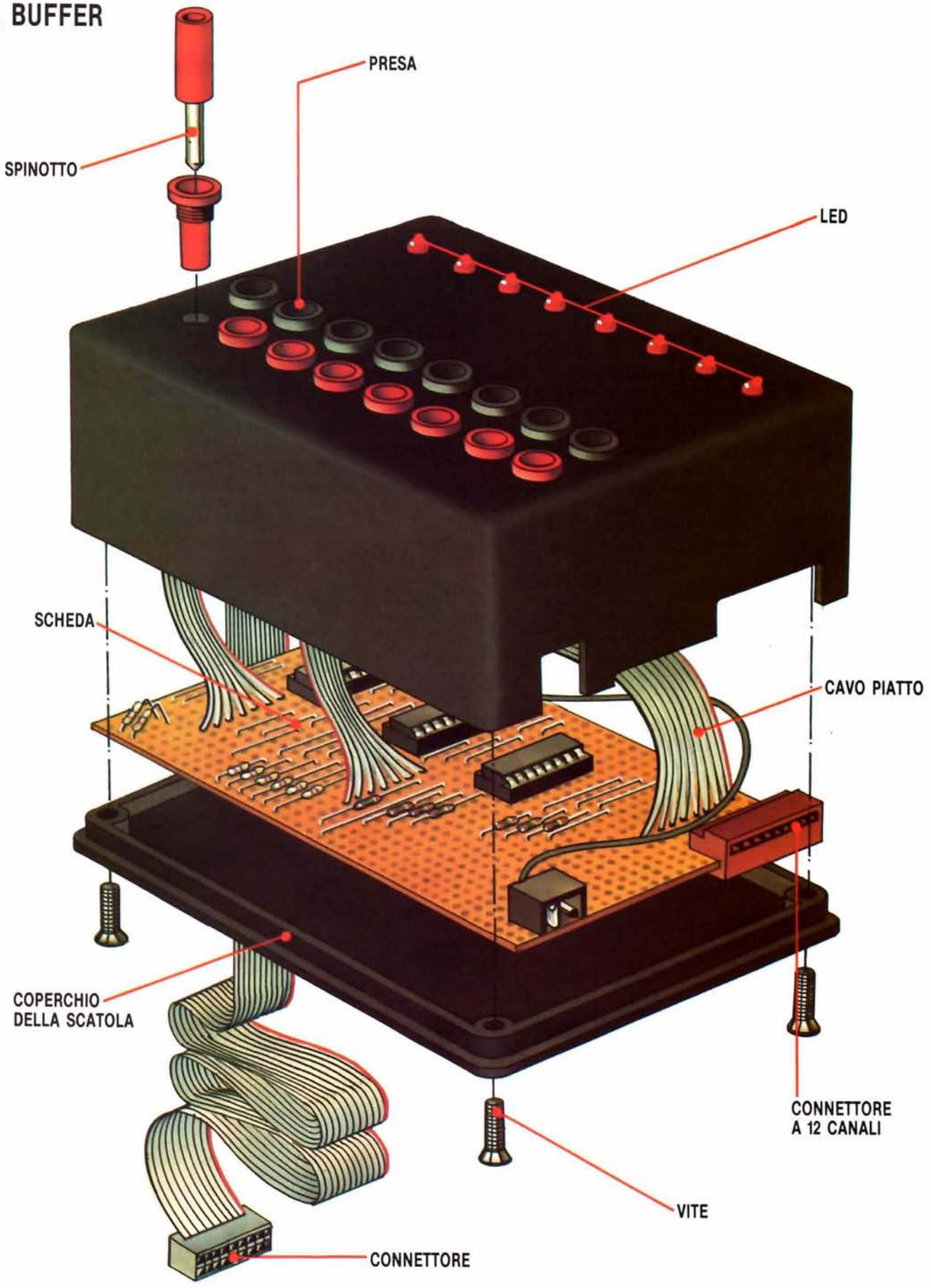
recchi di uso domestico quali televisori, registratori, impianti luminosi, e via dicendo. Usando un commutatore D/A (da Digitale ad Analogico) potremo anche pilotare visualizzatori luminosi a sette segmenti (quelli adoperati per gli orologi digitali). Per supplire alla bassa tensione di corrente in uscita dalla porta utente è necessario applicare una corrente esterna di 9 volt. I vari moduli devono poi essere collegati ad un "bus" comune, al quale giungono anche le 8 linee per i dati, la terra e la linea di alimentazione a 9 volt. È quindi possibile realizzare collegamenti multipli tra i moduli e il sistema.

COSTRUZIONE DEL VISUALIZZATORE A LED

Disporre gli 8 LED con i piedini più lunghi sulla più esterna delle quattro tracce presenti sulla scheda preforata. Per corrispondere ai fori sull'involucro, i LED devono distare 4 fori l'uno dall'altro. Saldare i piedini alle piste di rame, facendo attenzione a non creare contatti accidentali tra due piste adiacenti. Usare un ohmmetro (o meglio un multimetro) per controllare la resistenza tra le due tracce: se c'è contatto, il quadrante indica un valore 0.
Tagliare circa 20 cm. del cavo a nastro a 12 linee. Tre di queste non vengono usate e si possono eliminare (non quella colorata, però), ottenendo così un cavo a 9 linee. Dopo averne spellato un breve tratto (basta qualche millimetro), si saldi un'estremità del cavetto colorato alla pista di rame più esterna della scheda. Saldare adesso gli altri cavetti sulla seconda traccia, in corrispondenza del piedino più corto di ciascun Led. Tagliare la seconda pista di rame in 8 segmenti, in modo che ciascuna coppia piedino-cavetto si trovi isolata dalle altre. Con il multimetro, controllare di nuovo che gli 8 segmenti non presentino contatti tra loro o con la traccia più esterna.
Infine, inserire con delicatezza i Led nei fori del contenitore ed avvitare le prese rosse e nere nelle apposite aperture. Nella prossima lezione si spiegherà come realizzare il resto del circuito



IL BUFFER



ROSALIND BUCKLAND

Risposte agli esercizi di pag. 996

Gli esercizi di programmazione proposti a pag. 996 ricalcano tutti il programma per i numeri telefonici fornito come esempio Vengono qui fornite alcune, ma non certo le uniche, soluzioni funzionanti

1) ALLARME ANTIFURTO

```
 10 REM VERSIONE BBC 1.1
 20 REGDAT=&FE60:DDR=&FE 62
 30 ?DDR=0:REGDAT=127
 40 REPEAT
 50 PRINT "TUTTO BENE"
 60 ALLARME=?REGDAT
 70 UNTIL ALLARME<>225
 80 PRINT "ALLARME"
 90 FOR PIEDINO=0 TO 7
100 IF (ALLARME AND 2^PIEDINO)=0 THEN PRINT
    "ALLARME ALL'INTERRUTTORE NO."PIEDINO
110 NEXT
120 END
```

```
 10 REM VERSIONE COMMODORE 64 1.1
 20 DDR=56579: REGDAT=56577
 30 POKE DDR,0: POKE REGDAT=56577
 40 FOR C=0 TO 1 STEP 0
 50 PRINT "TUTTO BENE"
 60 ALLARME=PEEK (REGDAT)
 70 IF ALLARME <>255 THEN C=1
 80 NEXT C
 90 PRINT "SEGNALE D'ALLARME"
100 FOR PIEDINO=0 TO 7
110 IF (ALLARME AND 2^PIEDINO)=0 THEN PRINT "ALLARME
     ALL'INTERRUTTORE #";PIEDINO
120 NEXT:END
```

2) CONTO DEGLI IMPULSI (PRIMA VERSIONE)

```
 10 REM VERSIONE BBC 1.2
 20 REGDAT=&FE60:DDR&FE62
 30 ?DDR=0:?REGDAT=127
 40 CONTO=0: REM INIZIO CONTO
 50 TIME=0: REM INIZIO TEMPO
 60 REPEAT
 70 UNTIL ?REGDAT<>255
 90 CONTO=CONTO+1
100 UNTIL TIME >6000: REM 1 MIN
110 PRINT "IMPULSI="CONTO
120 END
```

```
 10 REM VERSIONE COMMODORE 64 1.2
 20 DDR=56579:REGDAT=56579
 30 POKE DDR,0: REM IMMISSIONE DA TUTTI I CANALI
 40 T=TI: REM INIZIO TEMPO
 50 IF PEEK (REGDAT)=225 THEN 50
 60 CONTO=CONTO+1
 70 IF (TI-T)<3600 THEN 50
 80 PRINT "IMPULSI=":CONTO
 90 END
```

3) CONTO DEGLI IMPULSI (SECONDA VERSIONE)

```
 10 REM VERSIONE BBC 1.3
 20 REGDAT=&FE60:DDR=&FE62
 30 DIM C(8)
 40 ?DDR=0: TIME=0
 50 REPEAT
 60 FOR N=0 TO 7
 70 C(N)+NOT(?REGDAT AND(2^N))
 80 NEXT N
 90 UNTIL TIME>6000
100 FOR N=0 TO 7
110 PRINT "LINEA"N"="C(N)
120 NEXT N
130 END
```

```
 10 REM VERSIONE COMMODORE 64 1.3
 20 DDR=56579:REGDAT=56579
 30 POKE DDR,0:T=TI
 40 FOR K=0 TO 7
 50 C(K)=C(K)+NOT(PEEK(REGDAT)AND(2^K)
 60 NEXT K
 70 IF (TI-T)<3600 THEN 40
 80 FOR K=0 TO 7
 90 PRINT "LINEA";K;"="C(K)
100 NEXT K
110 END
```

4) SERRATURA A COMBINAZIONE

```
 10 REM VERSIONE BBC 1.4
 20 REGDAT=&FE60:DDR=&FE62:FLAG=1
 30 ?DDR=15: REM LINEE 0-3 IMMISSIONE
 40 DIM C(3)
 50 REM CODICE=359
 60 C(1)=3:C(2)=5:C(3)=9
 70 FOR N=1 TO 3
 80 A$=GET$: REM ATTENDE PRESSIONE TASTO
 90 NEXT N
100 IF FLAG=1 THEN PRINT "APERTA" ELSE PRINT
    "COMBINAZIONE SBAGLIATA"
110 END
```

```
 10 REM VERSIONE COMMODORE 64 1.4
 20 DDR=56579:REGDAT56577
 30 POKE DDR,15:REM LINEE 0-3 IMMISSIONE
 40 C(1)=1:C(2)=2:C(3)=4: REM CODICE 124
 50 FOR K=1 TO 3
 55 PRINT "IMMETTI UNA CIFRA"
 60 GET A$: IF A$="" THEN 60: REM ASPETTA TASTO
 65 PRINT A$
 70 IF PEEK(REGDAT)<>C(K) THEN FL=1
 80 NEXT K
 90 IF FL=1 THEN PRINT "COMBINAZIONE SBAGLIATA": END
100 PRINT "APERTA"
110 END
```

5) CAMBIO DI COLORE

```
 10 REM VERSIONE BBC 1.5
 20 MODE 5
 30 REGDAT=&FE60:DDR=&FE62
 40 ?DDR=128: REM IMMISSIONE ECCETTO CANALE 7
 55 ?REGDAT=255
 50 A$=GET$: REM ASPETTA PRESSIONE TASTO
 60 SCOL=?REGDAT: REM SFONDI>127
 70 GCOL0,SCOL: REM CAMBIA COLORE ALLO SCHERMO
 80 CLG: REM RIPULISCE LO SCHERMO GRAFICO
 90 END
```

```
10 REM VERSIONE COMMODORE 64 1.5
20 DDR=56579: REGDAT=56577
30 POKE DDR,0: REM IMMISSIONE DA TUTTI I CANALI
40 GET A$: IF A$="" THEN 40
50 SCOL=PEEK(REGDAT)
60 POKE 53281,SCOL
70 END
```

I menu di assistenza

Le caratteristiche più importanti dell'interfaccia software tra uomo e macchina

Il costo della memoria dei computer cala di giorno in giorno. Probabilmente, la prossima generazione di home computer conterrà memorie con un minimo di 128 Kbyte ed offrirà molto più spazio di quello oggi richiesto dai più sofisticati programmi. In tutta la storia dell'informatica, la scarsità di memoria è stato il pretesto principale, accampato dai fabbricanti di computer, per non fornire agli utenti un numero sufficiente di comandi e di adeguati messaggi d'errore o di assistenza "on-line". Oggi questa scusa non regge più.
All'interno di un programma, le funzioni di assistenza a disposizione dell'utente possono assumere tre diverse forme: i testi di spiegazione, le pagine di "soccorso" ed i menu. I testi di spiegazione sono di due tipi: possono visualizzare un unico grosso messaggio all'inizio del programma o essere richiamati, quando è necessario, da qualsiasi punto del programmma. L'ideale sarebbe poter disporre sempre di entrambe le funzioni.
Nella loro forma più semplice, i testi di spiegazione sono costituiti da una o più pagine che illustrano in modo chiaro e conciso il funzionamento del programma. Il testo, contenuto in una stringa o in una serie di frasi DATA, viene visualizzato chiamando una subroutine appositamente progettata. Quando il programma viene avviato, esso chiede



all'utente se durante l'esecuzione intenda o meno avvalersi dei testi di spiegazione. Se la risposta è affermativa, viene richiamata un'apposita subroutine ed il sistema predispone tutte le altre routine richiamabili da tastiera. A questo punto, con la semplice pressione di un tasto (solitamente "?" oppure "H", per "HELP"), l'utente ottiene sullo schermo le spiegazioni. Conviene definire un comando standard per visualizzare le spiegazioni e ad esso uniformare tutte le routine di accesso al repertorio di "pagine". Ovviamente, tutti i messaggi del tipo «Battere un tasto qualsiasi per » vanno cambiati in «Battere un tasto qualsiasi per ... (oppure il tasto "H" per le istruzioni)». Questo messaggio standard può essere poi usato in tutti i programmi scritti dall'utente.
Oltre al testo vero e proprio, le spiegazioni possono contenere una serie di diagrammi e prevedere la chiamata a routine "didattiche", che permettano ai principianti di far pratica e imparare ad usare il programma. Simili routine sono molto comuni nel software di simulazione scientifica, nel quale l'utente



deve eseguire una serie di operazioni a vari livelli di difficoltà, prima di affrontare il programma principale. La loro progettazione è abbastanza complessa, poiché le routine devono simulare il funzionamento di tutto il programma e, allo stesso tempo, valutare le prestazioni dell'utente. Questo lavoro di programmazione permette di rilevare ed approfondire i problemi incontrati dall'utente nel programma principale.
In genere, le pagine di "soccorso" contengono informazioni più limitate e sono utili per chiarire il funzionamento di particolari sezioni del programma. Spesso vengono usate nei sistemi in cui occorre spiegare l'uso dei comandi: nel sistema operativo Unix, ad esempio, l'intero manuale per l'utente può essere visualizzato in forma di messaggi "on-line". Come già accennato per i testi di spiegazione, è possibile richiamare le pagine di soccorso anche dall'interno dei programmi scritti

Assistenza "on-line"
Il Wordstar della Micropro è un ottimo esempio di software pilotato mediante comandi e dotato di routine di assistenza "on-line". Il menu di soccorso presente sullo schermo può essere ridotto o totalmente eliminato dall'utente mentre resta sempre disponibile un file Help estremamente dettagliato che viene richiamato sul video battendo un solo tasto

dall'utente: basta battere sulla tastiera una particolare richiesta d'aiuto. In risposta, il programma chiama la routine di soccorso e ne visualizza il contenuto. Poiché per i programmi più complessi sono necessarie molte pagine di soccorso, conviene creare una routine generica, che contenga un indice delle pagine: digitando un numero, l'utente richiama la pagina desiderata. Nei sistemi a dischetti, le pagine di soccorso possono essere registrate direttamente su disco come file indipendenti. In tal caso, la routine generale, ricevuta la richiesta dell'utente, cerca il nome del file corrispondente, lo legge e ne visualizza il contenuto sullo schermo.



Se le routine di spiegazione o quelle di soccorso contengono più di una pagina di informazioni, la routine di visualizzazione deve consentire all'utente di passare da una pagina all'altra. Quando l'utente chiede di lasciare la routine di soccorso per tornare al punto del programma principale da cui era partito, la routine di soccorso modifica il valore di un flag, che provoca il ritorno all'ultima operazione eseguita, prima della chiamata di soccorso. Ricordiamo che il flag dovrà essere azzerato prima di tornare al programma principale.

I programmi più complessi vengono spesso paragonati ad un'aggrovigliata rete di sentieri logici, che facilmente disorientano chi li usa per la prima volta e non ne conosce la struttura. In questi casi, la presenza di menu permette all'utente di muoversi con una certa sicurezza all'interno di un programma. I menu servono quindi da cartelli indicatori che, collocati in posizioni strategiche, mostrano le possibili direzioni o scelte a disposizione dell'utente. Su alcuni computer (il Lisa ed il Macintosh della Apple, ad esempio) questo criterio è facilitato dall'impiego di simboli o "icone", al posto delle comuni opzioni espresse in forma di testo scritto.

In un sistema basato su speciali comandi, le opzioni possono essere molteplici, ma di solito alcune sono più importanti di altre e non tutte possono essere selezionate da uno stesso punto del programma. Se il menu offre un numero limitato di scelte, può esser utile introdurre una o due righe di spiegazione. Alcuni comandi di uso comune, come QUIT, UNDO e SAVE, devono essere disponibili in qual-



IAN McKINNELL

siasi momento ed è consigliabile inserirli in un menu costantemente visualizzato sullo schermo, del quale dovrà far parte anche l'opzione di uscita dal programma: oltre a instaurare un senso di sicurezza, soprattutto nei principianti, ciò è di grande utilità in caso d'emergenza. Spesso, i comandi appena citati vengono abbinati ad alcuni tasti funzione, mentre sullo schermo viene visualizzata una linea che descrive la funzione di ciascun tasto.

Ultimamente sono stati sperimentati sistemi in grado di analizzare il comportamento dell'utente e di modificare il livello di aiuto offerto dal sistema in base alla bravura dell'operatore. L'applicazione di tali sistemi ai programmi commerciali è ancora lontana, ma già oggi è possibile raggiungere parte di questi obiettivi usando tecniche abbastanza semplici. Se all'utente viene chiesto di comunicare il proprio nome, tutte le volte che usa un dato programma, il sistema può registrare in un apposito file il nome e il livello di abilità. Tale livello può essere calcolato considerando la frequenza d'uso del programma o il massimo punteggio ottenuto (se il programma è un gioco). Al termine di ogni sessione col programma, il livello viene aggiornato. Man mano che l'abilità dell'utente cresce, gli aiuti ed i menu a sua disposizione cambiano, divenendo sempre più brevi e meno ingombranti.

Nel pacchetto di word processing "Wordstar", ad esempio, l'utente è libero di scegliere il livello di aiuto che ritiene più adatto alle proprie capacità.

Le routine di assistenza possono rivelarsi un ottimo strumento per analizzare e migliorare il funzionamento di un programma. Con una semplice modifica al loro progetto originale è possibile tenere conto della frequenza d'uso delle pagine di soccorso e ciò può fornire una precisa indicazione dei punti meno comprensibili del programma.

UNA BUONA GESTIONE

Il software del package Apricot Manager, prodotto dalla ACT, guida l'utente, con un sistema gerarchico di menu, attraverso una serie di rigidi programmi di utilità. Ogni menu possiede l'opzione Help e presenta una breve descrizione dei menu di gerarchia inferiore. Questo è un tipico esempio di software controllato da menu e corredato da esaurienti file di assistenza

Il Sega SC3000H

Anatomia di un microcomputer a diffusione europea: tecnologie a confronto

L'SC3000H, un home computer della nipponica Sega, ha riscosso giudizi molto lusinghieri. Sebbene il progetto non presenti innovazioni rivoluzionarie (usa infatti il diffusissimo processore Z80A), è un apparecchio ben concepito, facilmente espandibile e può contare su una discreta quantità di software.

Il Sega pesa poco più di un chilo ed il colore dell'involucro, in plastica nera, contrasta piacevolmente col bianco dei tasti alfanumerici e col verde dei tasti di funzione (Control, Shift, Return, ecc.). La tastiera è del tipo macchina da scrivere ed i tasti, la cui corsa è di circa un centimetro, hanno una specie di scatto (probabile eredità della versione giapponese della tastiera a tasti in gomma morbida). Nel complesso, tuttavia, la qualità della tastiera è più che buona. Oltre alla serie di tasti standard, l'SC3000H possiede un tasto di funzione (non programmabile) usato per inserire le "parole chiave" del BASIC, un tasto per abilitare i caratteri grafici, due tasti rispettivamente per ripulire lo schermo e per le funzioni di inserimento e cancellazione. Per il movimento del cursore c'è una tastierina separata, a quattro tasti, particolarmente utile nei programmi di gioco. Purtroppo, i simboli grafici non sono riprodotti sui tasti, come lo sono invece nella versione SC3000 a tasti "morbidi". Questa mancanza genera qualche problema quando l'apparecchio viene usato per creare immagini in BASIC.

L'SC3000H è dotato di numerose interfacce: sulla destra ci sono due porte per joystick di tipo Atari e, sulla sinistra, un'interfaccia per cartucce. Sul retro della macchina sono visibili: un'uscita per il televisore, una porta per monitor video composito a colori, un connettore di tipo DIN per la stampante, un'interfaccia per cassette e la presa di alimentazione per l'adattatore di rete a 9 volt. Ci sono inoltre una scatola di commutazione (da abbinare al televisore) e una cartuccia BASIC corredata da un breve manuale d'istruzioni. L'installazione del Sega è semplice e non richiede particolare esperienza. Un opuscolo di due paginette spiega come realizzare i collegamenti tra il computer e l'apparecchio televisivo e quelli per l'alimentazione. Un LED verde indica quando la macchina è



CHRIS STEVENS

Novità dal Giappone
Il Sega SC3000H è un diretto concorrente dello Spectrum Sinclair e del Commodore 64. L'apparecchio possiede numerose periferiche, tra cui un registratore a cassetta, 2 joystick e una stampante/plotter a colori

accesa, ma l'opuscolo tralascia di ricordare che il Sega non funziona se nella porta ROM non viene inserita una cartuccia contenente il BASIC o un programma di gioco. Il sistema non possiede un interprete BASIC incorporato e, una volta caricata la cartuccia BASIC, restano solo 515 byte di RAM utente: meno memoria di quella presente sullo ZX81 non espanso! Quindi, volendo usare il computer per programmare, piuttosto che per semplici cartucce di gioco, occorre acquistare un modulo d'espansione.

Nel manuale originale c'è un po' di confusione sulla grafica, poiché esso fa riferimento alla tastiera "morbida" dell'SC3000 e si è così costretti a consultare il diagramma della tastiera per capire l'esatta posizione dei caratteri grafici o delle parole chiave del BASIC, che vengono immesse premendo contemporaneamente il tasto di funzione e quello della parola richiesta.

La grafica dell'SC3000H dà ottimi risultati: il modo testuale presenta uno schermo a due colori di 24 righe per 40 colonne, mentre in modo grafico si hanno 6 colori, con una risoluzione di 256x192 pixel. Il sistema permette di scegliere tra 210 livelli di luminosità e consente di creare fino a 32 sprite. Il BASIC del Sega, simile al Microsoft Extended BASIC, possiede comandi grafici, quali DRAW, COLOR , PAINT e SPRITE.

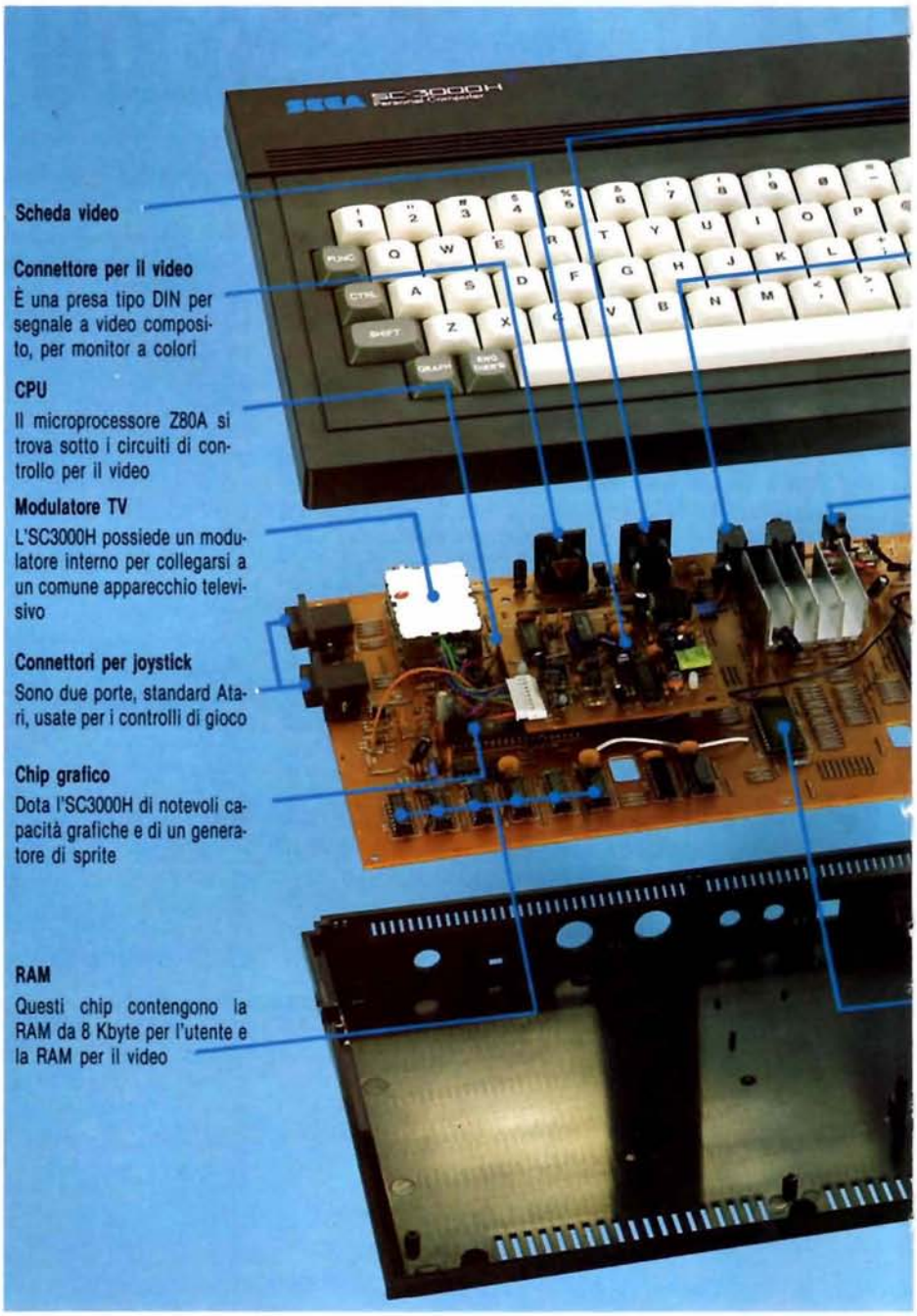
L'SC3000H può inoltre contare su sei canali di suono indirizzabili tramite POKE o con le apposite istruzioni BASIC di suono, BEEP e SOUND. Una cartuccia musicale facilita la composizione di melodie, ma il paragone fatto nel manuale tra i suoni ottenibili sull'apparecchio e quelli "tipo sintetizzatore" è un po' azzardato.

L' esperienza che la Sega ha nel settore del software per sale giochi è rilevata dal discreto livello dei programmi ricreativi. La grafica è buona, ma l'SC3000H ha una certa difficoltà a visualizzare contemporaneamente grafica e testi. Lo svolgimento dell'azione viene accompagnato da piacevoli "colonne sonore". Il tasto di Reset ha una funzione tutta particolare: anziché riavviare il gioco, consente di sospenderne momentaneamente l'esecuzione.

La Sega fabbrica due modelli di joystick, entrambi però hanno una risposta piuttosto lenta e poco precisa. Fortunatamente, la tastierina per il cursore dà risultati migliori e consente un più immediato e facile controllo del software di gioco.

Tra le numerose periferiche prodotte per l'SC3000H vale la pena di ricordare la stampante/plotter a colori, il registratore a cassette e l'unità di espansione che, per un prezzo ragionevole, offre una RAM extra di 64 Kbyte e una compatta unità a dischetti.

Se confrontato con altri apparecchi della stessa categoria, il Sega SC3000H sorprende per la facilità d'uso e per le sue buone prestazioni. La Sega dovrebbe però migliorarne la documentazione e la tastiera, riproducendo sui tasti le parole chiave ed i simboli grafici. L'inconveniente principale resta co-

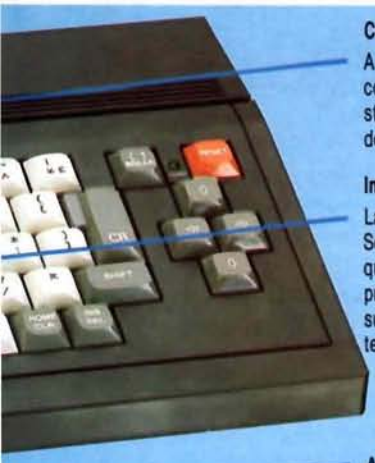




I Joystick del Sega SC3000H

La Sega produce due joystick diversi, ciascuno adatto ad un particolare tipo di gioco. Entrambi possiedono un meccanismo di funzionamento ad 8 posizioni di cloche e sono compatibili con il connettore standard a 8 piedini dell'Atari

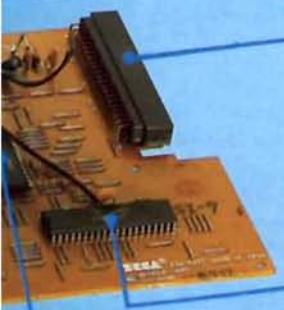
CHRIS STEVENS



Connettore per la stampante
A questa presa tipo DIN viene collegata la stampante o la stampante/plotter a colori prodotta dalla SEGA

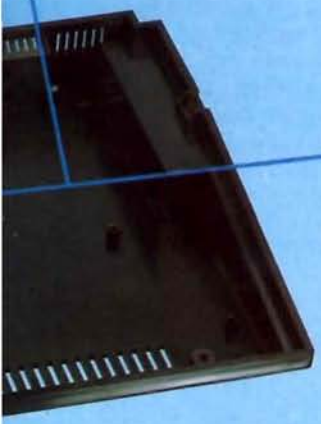
Interfaccia per cassetta
La maggior parte del software Sega è su cartuccia, quindi questa porta viene usata soprattutto per immagazzinare sui nastri i programmi dell'utente

Adattatore di tensione



Porta per cartucce
Gran parte del software Sega si trova su cartucce ROM, che vengono collegate all'SC3000H tramite questa porta

Controllo di I/O
Questo chip controlla le numerose funzioni di input e output del computer



ROM
Poiché il Sega non possiede un linguaggio BASIC residente, la ROM viene usata principalmente per la gestione delle operazioni interne e delle immagini video



"Safari Hunting"
Il giocatore, armato di un fucile a proiettili soporiferi, va a caccia di animali feroci in una giungla simile ad un labirinto



"Congo Bongo"
In questo pazzo gioco stile Arcade, il protagonista scala pericolosi dirupi all'inseguimento di un gorilla, cerca di schivare le noci di cocco che cadono dall'alto ed è perseguitato da pestifere scimmiette



"Baseball"
Il giocatore si misura contro una serie di avversari, raffigurati con grande realismo



"Sinbad Mistery"
Nasce da una fusione dei due stili di gioco "adventure" e "labirinto"

Software Sega
La Sega è anche un' eccellente creatrice di giochi elettronici e le ottime qualità grafiche e sonore del software prodotto per il Sega SC3000H non ne smentiscono la fama. Alcuni "games" presentano immagini "3-D" di discreta fattura e sono un'originale rielaborazione di molti giochi di successo

SEGA SC3000H

DIMENSIONI
mm. 353 x 210 x 46

PESO
Kg. 1,1

CPU
Z80A con frequenza di lavoro a 4 MHz

MEMORIA
8 Kbyte di RAM, espandibile a 48 Kbyte. 18 Kbyte di ROM, espandibile a 32 Kbyte, 16 Kbyte di RAM per il video

VIDEO
In modo testuale: 24 righe di 40 caratteri. In modo grafico, la risoluzione è di 256 x 192 pixel, con grafica a sprite e 16 colori in 15 tonalità

INTERFACCE
Porta per cartucce ROM, porte per joystick, per TV, per monitor a colori, per registratore a cassette e per stampante. Uscita audio.

LINGUAGGI FORNITI
BASIC, LOGO

TASTIERA
Tipo macchina da scrivere, con 64 tasti, tasti di controllo per il cursore e tasti di funzione

DOCUMENTAZIONE
Viene fornito un opuscolo di due pagine ed un manuale, assieme alla cartuccia del BASIC. La documentazione non è molto esauriente ed ha diverse lacune. Tutto ciò che riguarda la tastiera si riferisce alla versione con tasti in gomma, anziché alla tastiera della versione europea

PRO
L'apparecchio è buono, dotato di una piacevole grafica, di un apprezzabile generatore audio e di un dialetto BASIC di ottimo livello. Si tratta di un computer adatto ai principianti ed agli appassionati di giochi

CONTRO
La documentazione lascia a desiderare e la ridottissima capacità di memoria (degli 8 Kbyte resta ben poco dopo aver caricato il BASIC) non offre molte possibilità. Sulla tastiera non sono riprodotti i simboli grafici né le parole chiave del BASIC

munque la scarsità della memoria utente: chi vuol scrivere i propri programmi deve ampliare la memoria interna o acquistare l'unità d'espansione.

LOGO e tartarughe

Disegnare figure complesse con la tartaruga, tramite il LOGO

ABBREVIAZIONI

Molti comandi LOGO possiedono delle abbreviazioni; sono qui riportate quelle dei comandi incontrati nel testo:

FORWARD	FD
BACK	BK
RIGHT	RT
LEFT	LT
PENUP	PU
PENDOWN	PD
PRINT	PR
FULLSCREEN	FS
TEXTSCREEN	TS
SPLITSCREEN	SS

La prima versione del linguaggio LOGO adottata sui microcomputer fu il MIT LOGO, che oggi è considerato il dialetto "standard" e viene prodotto dalla Terrapin Inc. per gli apparecchi Apple e per il Commodore 64. La Logo Computer Sistems Inc. (LCSI) offre una seconda versione per i computer Apple, Atari e Spectrum (e presto anche per il Micro BBC). Esistono anche altre versioni, ma queste sono le più usate. I programmi che ci apprestiamo a creare usano tutti il MIT LOGO. Le differenze tra i dialetti MIT e LCSI vengono evidenziate nella rubrica "LOGO & LOGO".

Per imparare il LOGO non c'è che un modo: provare a scrivere dei programmi e tentare di risolvere da soli (con i suggerimenti qui presentati) i problemi incontrati!

Una volta caricato in memoria, il LOGO è immediatamente utilizzabile ed è pronto a ricevere ed eseguire le istruzioni immesse dal programmatore. Digitando il comando DRAW, si abilita il modo chiamato "splitscreen", che suddivide lo schermo in due aree: la superiore, che occupa la maggior parte del video, è per la grafica e al suo centro è visualizzato un triangolo (che rappresenta la "tartaruga"). La parte inferiore viene usata per i messaggi di testo ed inizialmente contiene un semplice punto interrogativo.

Consideriamo la tartaruga come un immaginario robot, che l'utente deve programmare usando una serie di comandi. I principali parametri da considerare sono: la posizione della tartaruga, la direzione o "rotta" del suo movimento e lo stato della "penna" che essa trasporta: se questa è abbassata, la tartaruga, spostandosi, traccia una linea sullo schermo; se è sollevata, la tartaruga si muove senza lasciare alcuna traccia.

Il comando DRAW posiziona la tartaruga al centro dello schermo con la parte anteriore rivolta verso l'alto e la penna abbassata.

Se battiamo l'istruzione:

```
FORWARD 40
```

la tartaruga si muove di quaranta unità verso l'alto, tracciando una linea. FORWARD è il comando per la tartaruga e 40 è il suo "parametro". Altri comandi (DRAW, ad esempio) non richiedono alcun parametro.

Un secondo comando è BACK. BACK 10 ordina alla tartaruga di retrocedere di 10 unità. Quindi, tanto FORWARD quanto BACK modificano la sua posizione sullo schermo. I comandi RIGHT e LEFT non cambiano la posizione della tartaruga, ma la fanno ruotare, modificando la sua direzione. Entrambi impiegano, come parametro, un angolo compreso tra 0 e 360 gradi. Si provino questi comandi per tracciare alcune semplici figure e si veda cosa accade quando si ordina alla tartaruga di eseguire uno spostamento che la porta fuori dallo schermo o se si usano numeri negativi come parametri. Iniziamo con lo schermo ripulito ed impartiamo il comando DRAW.

Sperimentando i vari comandi, può accadere che la tartaruga finisca sotto la parte dello schermo dedicata ai messaggi di testo e non

COS'È UNA TARTARUGA

Una tartaruga è una specie di robot che traccia delle linee. Essa poggia su ruote controllate da motori "passo passo" e possiede una penna retrattile. La tartaruga può essere programmata per muoversi avanti, indietro, a destra e a sinistra e la penna può sollevarsi ed abbassarsi: quando è abbassata, essa traccia una linea sulla superficie su cui la tartaruga si muove. Le prime tartarughe avevano una struttura a cupola, come la tartaruga di Edinburgo mostrata in figura, e venivano controllate tramite la tastiera del computer al quale erano collegate per mezzo di un cavetto parallelo. Le tartarughe più recenti possiedono invece un controllo a distanza. Oggi vengono prodotti una versione radiocomandata della tartaruga di Edinburgo ed un robot a forma di tartaruga, chiamato Valiant, che viene guidato dal computer mediante raggi infrarossi. Nel LOGO, il termine 'tartaruga' viene usato anche per indicare il cursore grafico dello schermo. La maggior parte di questi cursori/tartaruga è a forma di triangolo, ma il LOGO Atari produce un cursore assomigliante ad una piccola tartaruga



PAUL CHAVE



sia più visibile. In questo caso possono essere d'aiuto i seguenti comandi:

FULLSCREEN permette di usare l'intero schermo come area grafica
TEXTSCREEN elimina l'area grafica e lascia solo quella di testo
SPLITSCREEN suddivide nuovamente lo schermo in due aree (modo splitscreen)
PENUP permette alla tartaruga di muoversi senza tracciare linee
PENDOWN la tartaruga in movimento traccia sempre una linea

Tutte le istruzioni finora descritte rappresentano comandi impartiti alla tartaruga, ma il LOGO può essere usato anche per ottenere informazioni dalla tartaruga. La direzione del suo movimento viene misurata in gradi d'angolo (con ampiezza massima di 360) ed in senso orario: il parametro 0 indica che la tartaruga è rivolta perpendicolarmente verso l'alto. Per conoscerne la direzione si deve usare il comando:

```
PRINT HEADING
```

La posizione della tartaruga sullo schermo viene definita da un sistema di coordinate con origine nel centro dello schermo (quindi le coordinate x e y della sua posizione iniziale hanno entrambe valore 0). Inserendo il comando:

```
PRINT XCOR PRINT YCOR
```

possiamo conoscere in qualsiasi momento la sua posizione. Proviamo adesso a disegnare una figura e, successivamente, ad accertare la posizione e la direzione verso la quale è rivolta la tartaruga. Usiamo poi queste informazioni per riportare la tartaruga alla posizione di partenza. Durante questi esercizi con i comandi LOGO, può darsi che nell'area riservata al testo compaiano dei messaggi d'errore. Se questa ipotesi non si verifica, inseriamo deliberatamente un errore per vedere che cosa accade. Battendo, ad esempio:

```
FORWARD50
```

comparirà il messaggio:

```
THERE IS NO PROCEDURE NAMED FORWARD50
```
(non esiste una procedura chiamata FORWARD50)

Ciò avviene perché nel LOGO è obbligatorio inserire uno spazio tra l'istruzione (FORWARD) ed il parametro (50), evitando così di confondersi con un eventuale comando FORWARD50. Se scriviamo un comando in lettere minuscole, sullo schermo compare un altro messaggio d'errore.
Il LOGO possiede una "linea di edit", per correggere gli errori commessi durante la battitura di una linea di programma. La correzione deve precedere la pressione del tasto Return e viene effettuata spostando il cursore, fino a raggiungere l'errore. A questo punto, per inserire caratteri, basta digitarli sulla tastiera: il testo che si trova a destra del cursore scorre automaticamente. Per cancellare caratteri si usa il tasto di cancellazione, che permette di eliminare i caratteri che si trovano a sinistra del cursore. Quando la linea è corretta, si preme il tasto Return ed il LOGO accetta la nuova linea di programma. Se, dopo aver premuto il tasto Return, si scopre un nuovo errore, la combinazione di tasti Control-P consente di richiamare e correggere l'ultima linea redatta. Questa procedura viene usata anche per ripetere più volte uno stesso comando, senza doverlo riscrivere per intero.
Proviamo adesso a disegnare una figura geometrica, per esempio un quadrato. Ovviamente, il quadrato ha 4 lati uguali e 4 angoli retti (90 gradi) e la procedura appena vista si adatta perfettamente a questo caso:

```
FD 50 RT 90
FD 50 RT 90
FD 50 RT 90
FD 50 RT 90
```

Si noti che è possibile abbreviare i comandi ed inserire più istruzioni su una stessa linea. A questo punto sorge un problema di carattere tecnico: poiché il rapporto tra le unità di passo delle due dimensioni dello schermo (orizzontale e verticale) non è di 1:1, sul video compare un rettangolo e non un quadrato. Per ovviare a questo inconveniente, il LOGO offre il comando ASPECT, seguito da un numero (se non specificato il valore è di 0.8), che permette di modificare il rapporto tra le due unità di passo fino ad ottenere una figura perfettamente quadrata. Proviamo ora a tracciare le seguenti figure: un triangolo equilatero, un pentagono, un esagono, vari rettangoli, un rombo ed un parallelogramma.
Il comando REPEAT consente di semplificare alcuni comandi e di ridurne il numero:

```
REPEAT 4 [FD 50 RT 90]
```

REPEAT è un comando con due parametri. Il primo indica quante volte il LOGO deve ripetere una data sequenza di comandi, mentre il secondo contiene la sequenza di comandi in questione. Quest'ultima deve sempre trovarsi tra parentesi quadre. Nel caso del quadrato, il LOGO deve ripetere per quattro volte la sequenza FORWARD 50 RIGHT 90. Usiamo adesso il comando REPEAT per semplificare i programmi delle figure già realizzate e vediamo se riusciamo a creare le figure "a stella" illustrate a fianco.

Logo & Logo

Sull'Atari, il cursore non è un semplice triangolo, ma raffigura una vera e propria tartaruga!
Tutte le versioni LCSI usano:

* l'istruzione CLEARSCREEN (abbreviata in CS) prima d'iniziare a disegnare
* Control-Y per richiamare l'ultima linea redatta (solo lo Spectrum è privo di questa funzione)

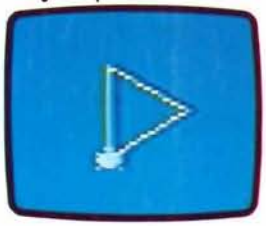
Per modificare il rapporto tra le unità di passo:

* l'Apple LCSI usa SETSCRUNCH seguito dal nuovo rapporto
* l'Atari usa .SETSCR seguito dal nuovo rapporto
* lo Spectrum usa SETSCRUNCH seguito dai valori delle due coordinate [xy] (di solito [100 100])

Esercizi di LOGO

Gli esempi qui riprodotti sono stati ottenuti col LOGO della LCSI su di un Atari 600XL

Triangolo equilatero



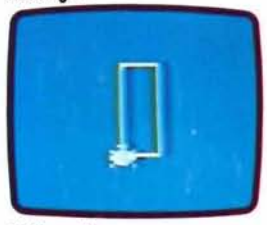
Pentagono



Esagono



Rettangolo



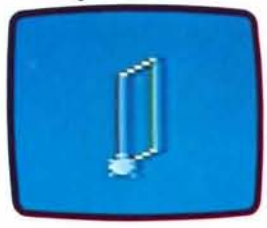
Stella a cinque punte



Stella a dieci punte



Parallelogramma



IAN McKINNELL

Suoni in sequenza

Cos'è l'interfaccia MIDI e come servirsene

L'uso del sequenziatore ha provocato una vera e propria rivoluzione in campo musicale, tanto nei concerti dal vivo quanto nelle sale d'incisione; presenta tuttavia un inconveniente: quello di poter controllare una serie di variabili, ma all'interno di un solo sintetizzatore. Se un tastierista usa due sintetizzatori, l'uno dotato di sequenziatore e l'altro con discrete capacità di suono, non ha alcuna possibilità di collegarli ad un'unica unità di coordinamento. Il problema diventa ancora più grave nelle sale d'incisione, dove occorre coordinare suoni prodotti da strumenti diversi e da una serie di attrezzature. L'obiettivo di un'interfaccia digitale è appunto quello di realizzare un simile collegamento e di passare il controllo dell'intero sistema ad un unico, potente apparecchio. Il MIDI è uno strumento col quale si tenta di dare un formato standard a collegamenti di questo genere.

Il MIDI (Musical Instrument Digital Interface) è stato sviluppato dopo una serie di riunioni tra i più importanti fabbricanti di strumenti musicali americani e giapponesi. Il progetto attuale, definito nell'agosto del 1983, prevede la realizzazione di un circuito di controllo per la trasmissione di informazioni da uno strumento a un altro o da un microprocessore a uno strumento.

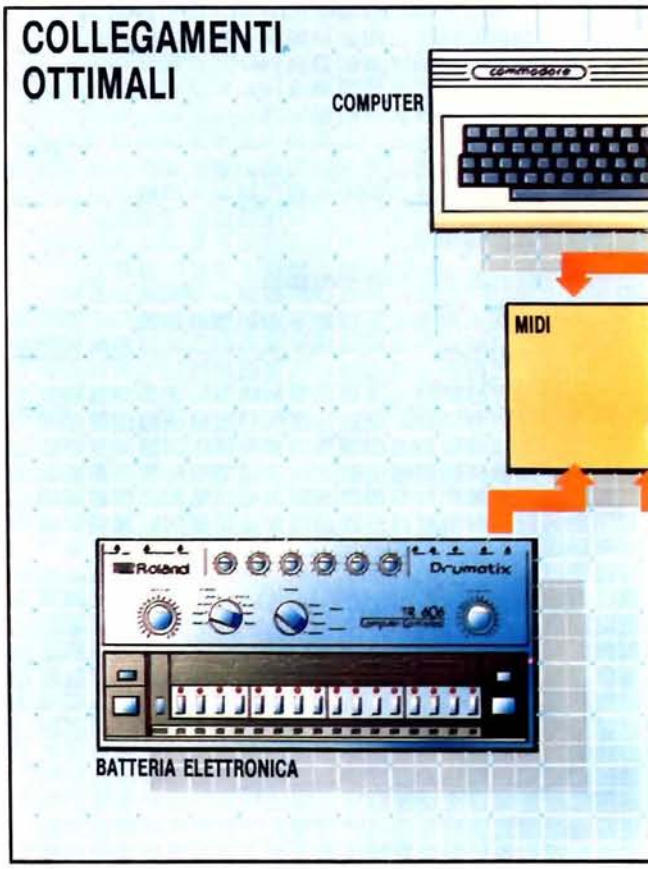
Il MIDI opera con un microprocessore ad 8 bit. I suoi progettisti hanno dovuto scegliere tra due possibili modi di trasmissione: parallela o seriale. La trasmissione parallela usa 8 canali individuali e consente la trasmissione simultanea degli 8 bit di un byte d'informazione. Comporta però costi molto elevati e l'inconveniente di dover collegare almeno otto cavetti ad un connettore di tipo D, a 25 piedini. La trasmissione seriale, invece, usa solo due canali: uno per inviare in sequenza i bit d'informazione, l'altro per consentire allo strumento ricevente di segnalare all'apparecchio di controllo o al microcomputer errori di parità nel flusso dei dati (vedere a pag. 658). Quindi la trasmissione seriale è più lenta della parallela, ma ha il vantaggio di essere meno costosa e di usare un connettore più semplice.

I progettisti del MIDI si proponevano di creare un circuito di controllo che, oltre ad offrire una serie di moduli di controllo del suono, permettesse alla musica sintetizzata di esplorare nuove vie espressive. Il costo doveva essere adeguato alle possibilità dei possessori di home computer e dei "solisti" che abitualmente usano sintetizzatori e batterie elettroniche (come la Linn). Questi motivi hanno portato alla scelta della trasmissione seriale.

Il MIDI trasmette in modo asincrono sugli stessi canali dell'interfaccia standard RS232, comunemente usata per collegare molti home computer ai modem o alle stampanti seriali. Nella trasmissione asincrona, ciascun byte d'informazione deve essere indirizzato con precisione allo strumento ricevente. Sul MIDI, questa operazione viene eseguita dal chip Motorola 6850 ACIA (Asynchronous Communications Interface Adaptor) che aggiunge due bit ad ogni byte trasmesso dall'apparecchio di controllo. Il primo, o "bit di inizio" (start bit), è 0 e ad esso seguono gli 8 bit dell'informazione seriale. Il secondo, o "bit di stop", è 1. Questa parola seriale di 10 bit viene trasmessa allo strumento, dove un secondo chip ACIA la trasforma nuovamente nell'informazione ad 8 bit.

I chip ACIA sono piuttosto costosi e vengono protetti all'interno del circuito con un accoppiatore ottico, ossia uno strumento dotato di cellule fotoelettriche che permette a due circuiti elettrici indipendenti di scambiarsi segnali pur restando elettricamente isolati. In questo modo, il chip viene protetto da eventuali sbalzi di tensione.

Tuttavia esiste una differenza fondamentale tra il MIDI e l'interfaccia RS232. La RS232 ha una velocità di trasmissione di 1920 parole seriali al secondo, cioè 19,2 Kbaud, mentre sul





L'effetto "glissando"
Il "glissando" (cioè la flessione del tono musicale da una nota ad un'altra) si può ottenere sugli strumenti a corda facendo scorrere le dita avanti o indietro sul manico dello strumento. Un sintetizzatore invece riproduce questo suono come un effetto di "slittamento"

MIDI la velocità di trasmissione è di circa la metà. Questa differenza non influisce sulla compatibilità con i microcomputer, poiché i circuiti logici interni al MIDI possono selezionare una nuova velocità di clock di 3125 parole seriali per secondo, ovvero 31,25 Kbaud. Questa è una velocità di trasmissione seriale relativamente alta, ma probabilmente non ancora sufficiente.



In un sistema musicale, il computer può svolgere due diverse funzioni, a seconda del tipo di MIDI usato: innanzi tutto può selezionare il MIDI nella configurazione "attiva" e mettere a sua disposizione una certa area di memoria per la registrazione di brani musicali. Oppure può caricare il software MIDI non residente, nel MIDI stesso. Inoltre, il MIDI può funzionare come interfaccia digitale tra lo strumento musicale ed il computer o, sempre come interfaccia, svolgere attività di controllo sui dati trasmessi.
Esistono due modi di trasmissione: registrazione e playback. Nel modo registrazione, la musica prodotta da uno strumento viene trasmessa attraverso il MIDI al buffer del computer o del MIDI e, successivamente, alla memoria. Nel modo playback, questa informazione digitale viene elaborata dal MIDI e trasmessa agli strumenti: i parametri di temporizzazione, sincronizzazione e controllo vengono assegnati ai vari strumenti in base alle specificazioni indicate protocollo stabilito dall'utente

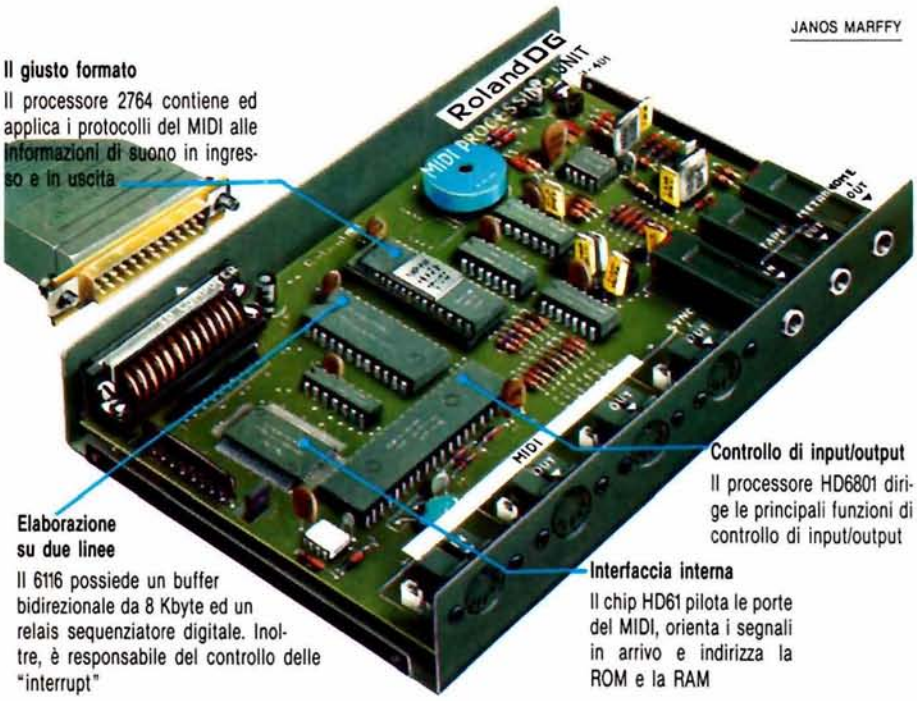
KEVIN JONES

Il MIDI può essere interfacciato con più strumenti contemporaneamente. Occorre però accertarsi che ogni strumento riceva solo le informazioni che gli competono, altrimenti può accadere che uno strumento a percussione suoni per errore un pezzo spiccatamente melodico oppure che un sintetizzatore polifonico finisca per generare sul suo DO centrale una sequenza destinata ad un tamburo basso. Gli strumenti collegati devono quindi possedere un proprio codice d'identificazione (chiamato in gergo 'ID') corrispondente ad uno dei sedici canali disponibili sul MIDI: ciascun canale trasmette solo i segnali destinati ad un particolare strumento. La prima parte di una trasmissione del MIDI contiene infatti un byte che indica il codice d'identificazione, mentre i dati successivi chiariscono come il comando debba essere interpretato.
L'interfaccia, le cui dimensioni sono mm. 100 x 120 x 45, possiede due prese DIN a 5 piedini, chiamate "MIDI IN" e "MIDI OUT". La "MIDI IN" riceve tutte le istruzioni provenienti da un microcomputer o dal sintetizzatore principale, mentre la "MIDI OUT" trasmette agli strumenti riceventi la sequenza di bit modificati. Alcuni modelli hanno una seconda porta di output, detta "MIDI THRU", per trasmettere ad una seconda interfaccia la sequenza originale ricevuta dalla porta "MIDI IN". Il cavo di collegamento ha una lunghezza massima di 15 metri ed è dotato di due spinotti a 5 piedini, da innestare sul retro di un microcomputer o del sintetizzatore principale.

Il DO centrale

Immaginiamo di usare per la prima volta un MIDI e di voler suonare una breve melodia che inizi dal DO centrale, passando al MI superiore, quindi al SOL e così via. Le istruzioni impiegate per suonare questo pezzo dipendono dal tipo di software musicale usato. Possiamo servirci di una penna luminosa per indicare le note su un pentagramma visualizzato sullo schermo oppure, grazie ad un particolare linguaggio detto MCL (Music Composition Language), usare la tastiera del nostro microcomputer per inserire direttamente sullo schermo le note musicali. In alternativa, possiamo suonare le note della melodia su una tastiera musicale collegata al computer: questa può non essere in grado di produrre suoni, ma può ricorrere ad una delle due tecniche appena descritte per visualizzare le note sullo schermo. In tutti i casi, la trasmissione MIDI è sempre la stessa: per iniziare a suonare, il primo byte, trasmesso come una parola seriale contenente i due bit extra generati dal chip ACIA, deve contenere l'istruzione SUONA / SUL CANALE 6. Il secondo byte, contiene il dato DO CENTRALE.
Questa breve istruzione fa suonare allo strumento ricevente la nota DO centrale, che viene mantenuta finché non giunge un'istruzione che ne definisce la durata, ad esempio: FINE DEL SUONO / SUL CANALE 6 (primo byte); DO CENTRALE (secondo byte); CONTENENDO UNA DURATA X (terzo

JANOS MARFFY



Il giusto formato
Il processore 2764 contiene ed applica i protocolli del MIDI alle informazioni di suono in ingresso e in uscita

Elaborazione su due linee
Il 6116 possiede un buffer bidirezionale da 8 Kbyte ed un relais sequenziatore digitale. Inoltre, è responsabile del controllo delle "interrupt"

Controllo di input/output
Il processore HD6801 dirige le principali funzioni di controllo di input/output

Interfaccia interna
Il chip HD61 pilota le porte del MIDI, orienta i segnali in arrivo e indirizza la ROM e la RAM

Funzioni del MIDI
L'interfaccia MIDI armonizza i protocolli di input/output del computer con gli strumenti musicali collegati, consentendo a questi ultimi di usare la memoria del computer. Inoltre, elabora i suoni "digitalizzati" che lo attraversano ed aggiunge ai segnali per il sintetizzatore una serie di parametri di controllo, temporizzazione e sincronizzazione

byte). Se quest'ultimo comando manca, e il resto della melodia viene trasmesso senza che vengano definiti i parametri di durata, tutte le note continueranno ad essere suonate. Si otterrà così un accordo prolungato composto da tutte le note del pezzo musicale.
Fortunatamente, basta una minima conoscenza della musica per accorgersi che, sul pentagramma, il pezzo si è trasformato da melodia in accordo. Se si usa il software MCL, l'errore può essere facilmente corretto inviando al sintetizzatore l'istruzione MONOPHONIC ossia "Monofonia". Ciò significa che le note verranno suonate l'una dopo l'altra e non contemporaneamente ("polifonia"). Dopo questo accorgimento, il sintetizzatore suonerà il pezzo come una melodia monofonica e non come un singolo accordo.
Supponiamo di aver immesso il nostro programma musicale senza errori e che il sintetizzatore collegato lo stia suonando. La melodia segue il tempo ed il ritmo, scandito dalle istruzioni di durata. Va notato, tuttavia, che fino a questo momento abbiamo usato solo due tipi d'istruzione, cioè i parametri musicali della tonalità (DO centrale, MI, SOL ecc.) e della durata.
Dopo aver ascoltato la melodia, ci si accorge infatti che il pezzo suonato è piuttosto "freddo": per migliorarlo si può modificare la durata delle note e prolungare la durata del DO oltre l'attacco del MI. Questo tipo di movimento è chiamato "glissando" o flessione di tonalità e permette di aggiungere un tocco di brio alla musica generata dal sintetizzatore. Inseriamo dunque questa istruzione per il DO centrale al posto della precedente, aggiungendo un ulteriore byte.
Questa modifica solleva un nuovo problema per il collegamento del MIDI: se il sintetizzatore ricevente non possiede la funzione "glissando", non è in grado di eseguire l'ultima istruzione e suona il DO centrale come se continuasse a ricevere l'istruzione precedente, oppure genera un suono completamente diverso. Se le istruzioni introdotte fanno parte di un brano di musica polifonica e il sintetizzatore ricevente è uno strumento esclusivamente monofonico, l'apparecchio sceglierà a caso tra le note del pezzo polifonico e le eseguirà l'una dopo l'altra e non come una serie di accordi. Quindi i risultati che si possono ottenere interfacciando un MIDI con un microcomputer e con un sintetizzatore di ridotte capacità dipendono esclusivamente dalle potenzialità musicali di quest'ultimo.
D'altro canto, se il programma musicale progettato è di bassa qualità e i moduli di controllo del sintetizzatore non vengono impiegati nel modo migliore, anche usando come strumento ricevente un costosissimo sintetizzatore, la qualità della musica non sarà migliore di quella ottenibile su una comune calcolatrice tascabile.
Indubbiamente, questa seconda situazione può esser migliorata: i moduli di controllo del sintetizzatore consentono di definire come "costanti" numerosi parametri musicali e di usare le istruzioni del MIDI per elaborarli. Questa è la tecnica preferita dai suonatori di sintetizzatore.
Finora abbiamo parlato dell'altezza del tono e della durata ma il MIDI offre ben 128 funzioni di controllo che riguardano il filtraggio, la distorsione, il "rumore bianco" (tutte le frequenze possibili), il "rumore rosa" (solo le medie frequenze), ciascuna con valori compresi tra 0 e 128. Queste funzioni sono più che sufficienti per controllare i parametri disponibili sulla maggior parte dei sintetizzatori e sono senz'altro i parametri che interessano soprattutto ai possessori di microcomputer.
Ma è proprio riguardo a queste funzioni che sorge il problema della velocità di trasmissione. In precedenza abbiamo visto come un semplice comando, per suonare una nota definita da due soli parametri, occupi tre parole seriali. Con una velocità di 31,25 Kilobaud, la trasmissione di questo comando dura un millisecondo. Per accordi a sei note, molto comuni in alcuni generi musicali, il tempo di trasmissione è di circa 5,76 millisecondi. Se un simile accordo viene programmato usando i controlli del MIDI, il tempo di trasmissione diventa eccessivamente lungo e l'orecchio umano comincia a percepire dei cambiamenti nelle caratteristiche del suono. Queste modificazioni divengono chiaramente udibili solo quando le note (soprattutto quelle con frequenze simili) vengono suonate assieme. Purtroppo, la musica si basa quasi esclusivamente su emissioni parallele (come gli accordi).
È quindi abbastanza naturale che la trasmissione seriale del MIDI sia stata molto criticata: la trasmissione parallela avrebbe dato risultati di gran lunga migliori. Rimane da vedere se gli utenti del MIDI saranno disposti ad accettare questi limiti. Attualmente, il progetto dell'interfaccia sembra essere il risultato di un compromesso tra costi ed efficienza, ma potrà essere modificato nelle versioni future.

Da registro a registro

Il trasferimento di dati all'interno del microprocessore 6809

Come già spiegato, un registro è una particolare locazione di memoria, interna al microprocessore, e la programmazione del linguaggio Assembly implica la manipolazione dei valori immagazzinati nei registri come pure il loro trasferimento da e verso la memoria principale. Abbiamo osservato che alcuni registri, soprattutto quelli che memorizzano ed elaborano gli indirizzi, hanno una lunghezza di 16 bit, mentre altri sono ad 8 bit, e che differenti registri svolgono funzioni diverse.

- **Registri indice**: vengono usati per modificare gli indirizzi utilizzati in un programma.
- **Puntatori allo Stack**: vengono usati dal processore per indirizzare la memoria di lavoro (RAM) e possono essere impiegati nello stesso modo dal programmatore per velocizzare le operazioni di memorizzazione e di estrazione.
- **Contatore di programma**: contiene l'indirizzo della successiva istruzione e può essere modificato per deviare il percorso del programma; offre al linguaggio Assembly un equivalente del comando GOTO.
- **Accumulatori**: sono i più usati e si impiegano per eseguire le funzioni aritmetiche.
- **Registro di condizione del processore**: contiene un certo numero di flag, indicanti lo stato del processore (per esempio, se l'ultima operazione ha dato risultato zero). Questi flag si possono sottoporre a verifica all'interno di procedure di selezione o di cicli iterativi e forniscono al linguaggio Assembly un equivalente dell'istruzione IF...THEN.

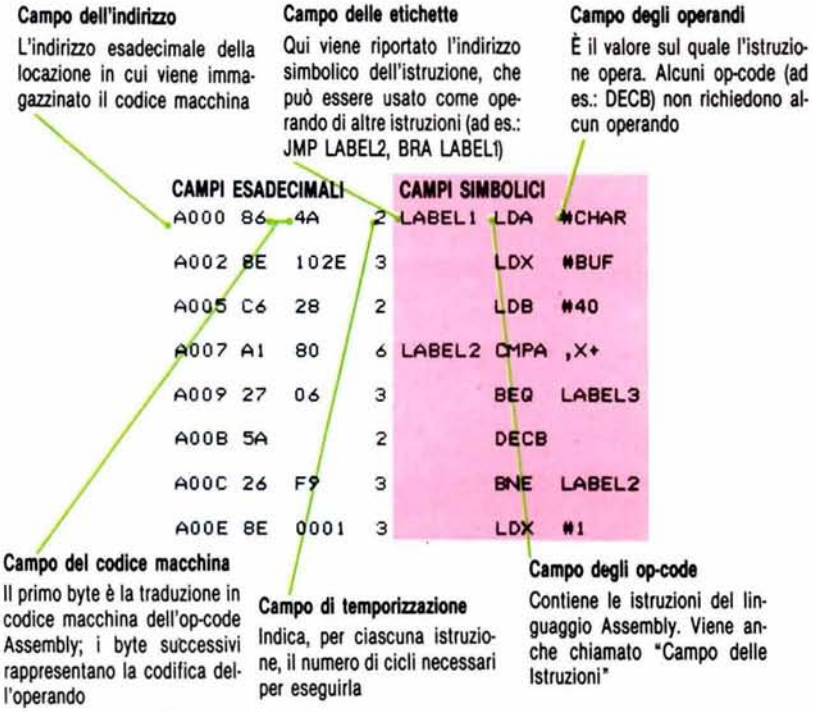
Il processore 6809 possiede tutti questi registri. Poiché si tratta di una variante del processore Motorola 6800 (come lo è il 6502, adottato sul Micro BBC e su molti altri computer), i linguaggi Assembly usati sui due processori si somigliano molto sebbene non siano compatibili: il codice macchina di uno non funziona sull'altro. Tuttavia, molti programmi del 6800 sono compatibili per quanto riguarda il codice sorgente, quindi un programma in Assembly scritto per il 6800 e riassemblato per il 6809, ha qualche possibilità di funzionare. Purtroppo questa remota compatibilità non vale per il 6502 (né per il 6510, suo "discendente", usato sul Commodore 64). Resta il fatto che la somiglianza tra i due microprocessori rende meno laboriosa la trasposizione di un programma in Assembly nella versione 6809: questo può essere un vantaggio per chi già conosce il 6502. Il 6809 contiene i seguenti registri:

♦ Due accumulatori ad 8 bit, chiamati A e B. La presenza di due accumulatori permette di utilizzarne uno per conservare i valori mentre il lavoro viene svolto sull'altro. In alternativa, i due accumulatori possono essere trattati come un unico potente accumulatore a 16 bit. Per questo motivo il 6809 viene spesso definito come uno pseudo-processore a 16 bit, anche se in realtà non lo è ed è sempre meglio considerarlo un processore ad 8 bit "potenziato". Quando i due accumulatori vengono usati assieme, si è soliti chamarli "registro D a 16 bit".

♦ Due registri indice, detti X e Y. Anche questi funzionano in modo simile tra loro e sono quindi utilizzabili indifferentemente. L'unica piccola differenza operativa è che alcune istruzioni relative al registro Y hanno un codice di due byte, mentre le corrispondenti, per il registro X, sono formate da un solo byte. Quando si ha bisogno di un solo registro d'indice è meglio usare X.

♦ Due puntatori allo stack, indicati con S e U. Il processore usa S per tutte le operazioni di stack. Il programmatore è libero di impiegare S dove e quando vuole, ma deve sempre assicurarsi di non interferire col lavoro del processore: l'uso di U comporta, dunque, meno rischi.

♦ I registri X, Y, S e U possono essere usati anche in modo diverso da quello previsto dal loro progetto: S e U possono, ad esempio, essere adoperati come registri indice ed è previsto l'impiego di tutti e quattro i registri



Campi di studio

I listati dei programmi in linguaggio Assembly sono molto diversi da quelli in BASIC e, quando stampati dal programma Assembler (che traduce l'Assembly nel codice macchina), appaiono di difficile comprensione. Inizialmente, conviene riferirsi solo a quelle colonne o "campi" che interessano (campi delle etichette, degli op-code e degli operandi), ignorando il resto

per l'immagazzinamento e l'elaborazione di numeri a 16 bit.

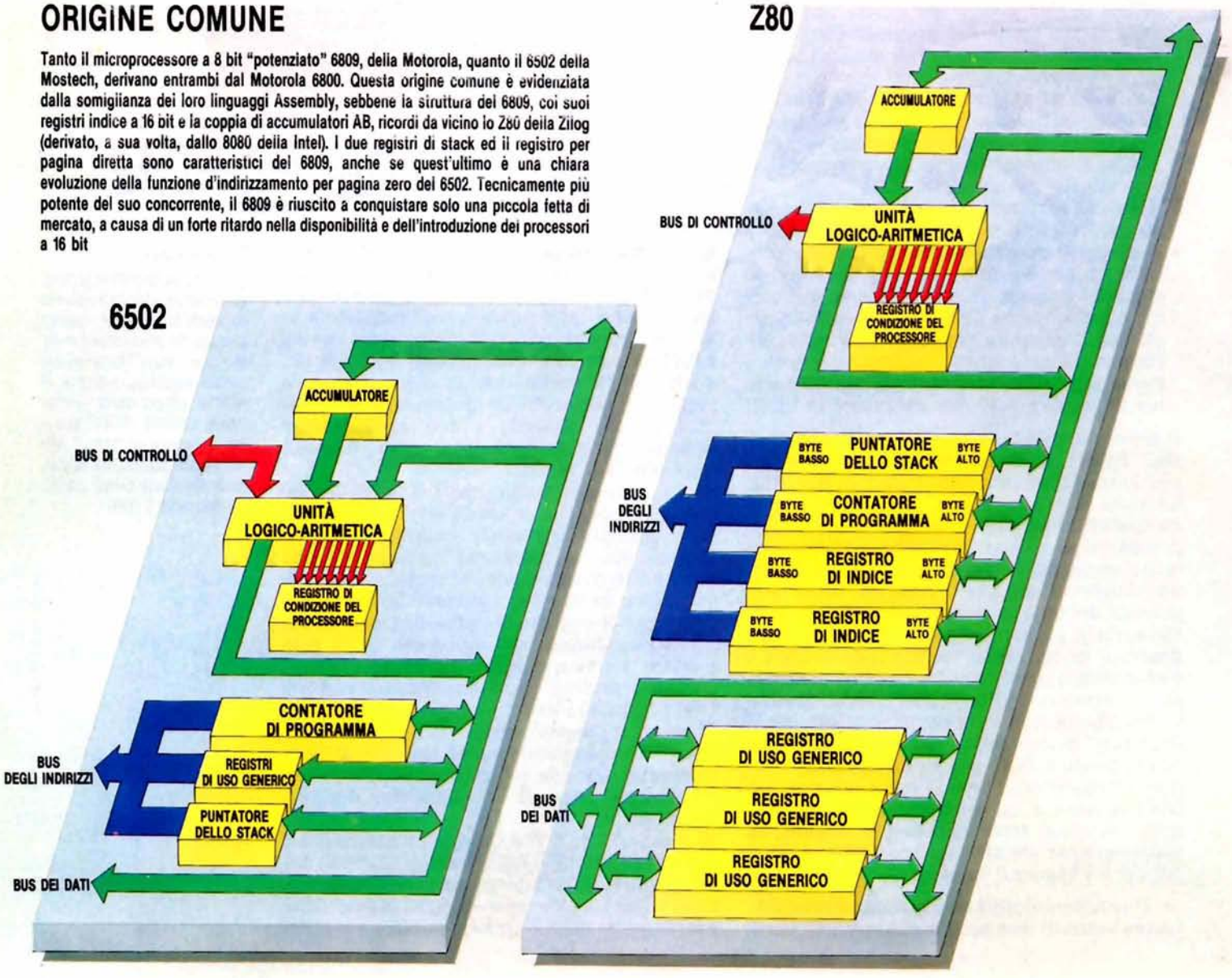
◆ Il contatore di programma (PC) è un registro a 16 bit che il processore modifica automaticamente per puntare all'istruzione successiva. Le istruzioni di salto (JMP) e di diramazione (BRA) alterano il contenuto del contatore di programma. Nel 6809, questo registro si può usare anche come una specie di registro indice. Esso verrà esaminato più dettagliatamente in seguito: per adesso basti sapere che usare indirizzi "relativi", rispetto a quello contenuto nel PC, anziché specificarne di "assoluti" (memorizzati in X o Y), permette di ottenere programmi veramente indipendenti dalla posizione in cui si trovano nella RAM. In altre parole, i programmi scritti con questa tecnica possono essere eseguiti, senza alcuna modifica, in qualsiasi locazione di memoria.

◆ Il registro di condizione del processore contiene 8 bit indipendenti, ciascuno usato come flag per indicare lo stato del processore. Le istruzioni come BNE ("Branch Not Equal", ossia "diramazione se non uguale") eseguono una verifica su un flag in particolare, provocando un cambiamento nel flusso di controllo a seconda del suo valore (1 o 0).

Esistono numerose istruzioni per scambiare informazioni tra i vari registri o tra i registri e la memoria. Negli esempi che seguono, supponiamo di aver usato le direttive Assembly FCB e FDB (che non sono vere e proprie istruzioni del processore) per riservare un certo numero di locazioni di memoria. Durante il processo di conversione da codice "sorgente" a codice "oggetto", l'Assembler esegue direttamente queste direttive, accantonando le locazioni di memoria specificate per usarle come depositi dei dati. Le direttive non generano alcun codice macchina, poiché non riguardano direttamente il processore. Siccome assomigliano ai codici operativi del linguaggio Assembly (come BNE o JMP), vengono anche chiamate "pseudo-op". Se a tali direttive viene assegnata un'etichetta, l'Assembler associa alle locazioni di memoria così riservate un preciso indirizzo. Possiamo quindi avere:

ORIGINE COMUNE

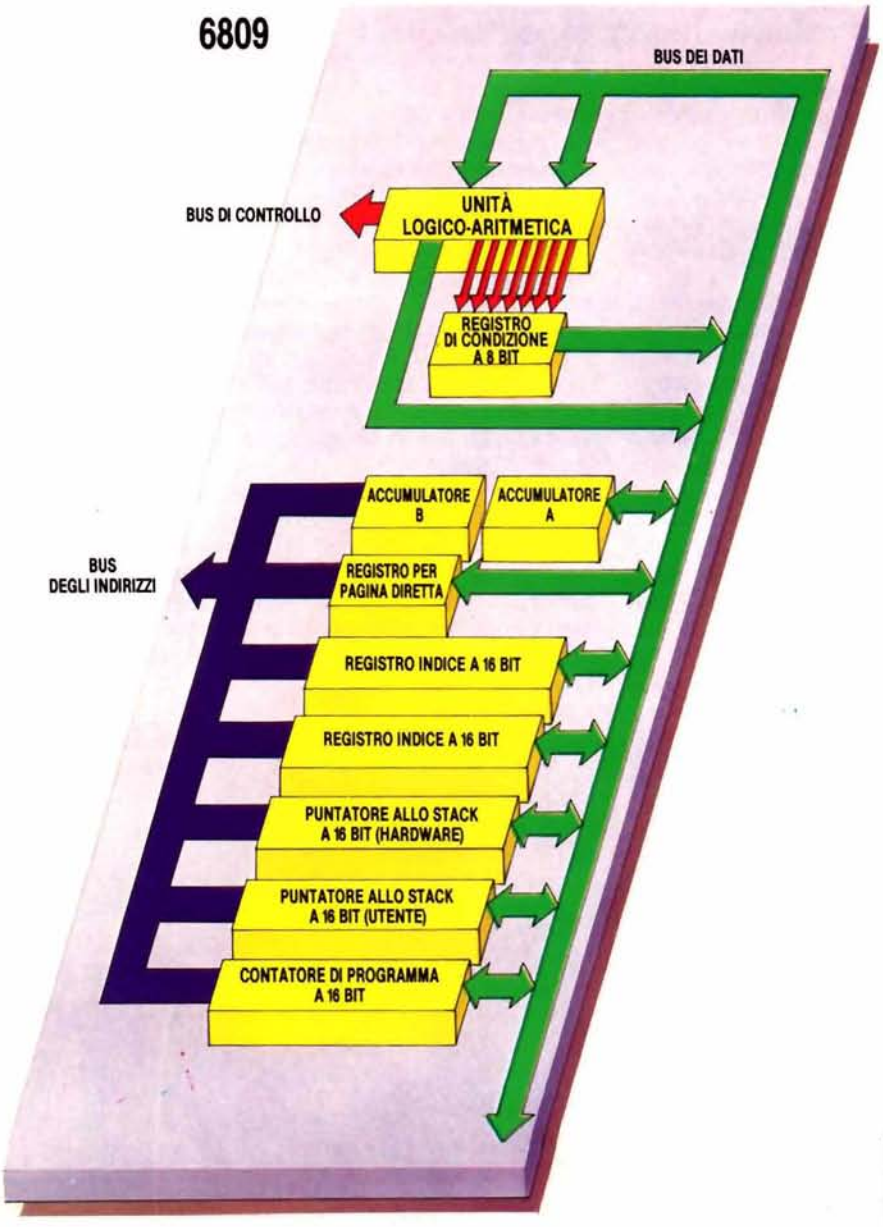
Tanto il microprocessore a 8 bit "potenziato" 6809, della Motorola, quanto il 6502 della Mostech, derivano entrambi dal Motorola 6800. Questa origine comune è evidenziata dalla somiglianza dei loro linguaggi Assembly, sebbene la struttura del 6809, coi suoi registri indice a 16 bit e la coppia di accumulatori AB, ricordi da vicino lo Z80 della Zilog (derivato, a sua volta, dallo 8080 della Intel). I due registri di stack ed il registro per pagina diretta sono caratteristici del 6809, anche se quest'ultimo è una chiara evoluzione della funzione d'indirizzamento per pagina zero del 6502. Tecnicamente più potente del suo concorrente, il 6809 è riuscito a conquistare solo una piccola fetta di mercato, a causa di un forte ritardo nella disponibilità e dell'introduzione dei processori a 16 bit



NUM1 FCB 0	riserva un solo byte, il cui indirizzo è NUM1, e il valore iniziale è 0
NUM2 FCB 0	simile alla precedente
NUM3 FDB #A93B	riserva due byte per il numero #A93B a 16 bit (il segno # viene spesso usato dagli Assembler del 6809 per indicare che il numero è in notazione esadecimale)

Le seguenti istruzioni caricano i valori memorizzati in queste locazioni in registri diversi:

LDA NUM1	carica nell'accumulatore A il numero ad 8 bit registrato nella locazione di memoria corrispondente a NUM1
LDB NUM2	come la precedente, ma carica il contenuto di NUM2 nell'accumulatore B
LDX NUM3 LDY NUM3 LDS NUM3 LDU NUM3 LDD NUM3	queste istruzioni caricano il numero a 16 bit immagazzinato in NUM3 nei rispettivi registri X, Y, S, U e D



KEVIN JONES

Analogamente, i contenuti a 8 o 16 bit dei registri possono essere depositati in una locazione di memoria usando una delle seguenti istruzioni:

```
STA NUM1
STB NUM2
STX NUM3
STY NUM3
STS NUM3
STU NUM3
STD NUM3
```

Si rammenti che, nel caricare nell'accumulatore NUM1, in realtà si copia il contenuto di NUM1 nell'accumulatore, senza alterare il valore nella locazione. Le operazioni di immagazzinamento funzionano in modo analogo.
L'istruzione EXG permette di scambiare i contenuti di due registri (purché entrambi siano a 8 oppure a 16 bit).
Per esempio:

```
EXG A,B scambia i contenuti dei registri A e B
EXG X,S scambia i contenuti dei registri X e S
```

L'istruzione TFR trasferisce invece il contenuto di un registro nell'altro: ad esempio, TFR Y,U copia il contenuto di Y in U. Anche in questo caso i due registri devono essere entrambi a 8 o a 16 bit.
Un'altra istruzione fondamentale è ADD, che somma il contenuto di una locazione di memoria a quello di uno degli accumulatori. Essa ha il seguente formato:

ADDA NUM1	che significa "somma il contenuto della locazione di memoria NUM1 al contenuto del registro A e deposita il risultato dell'addizione nello stesso registro A"

Adesso proviamo a sommare i due numeri ad 8 bit contenuti in NUM1 e NUM2, riportando il risultato in NUM1. L'eventuale overflow (superamento della capacità del registro) qualora il risultato sia un numero più grande di 8 bit, viene ignorato. Poi sommiamo di nuovo i contenuti delle due locazioni, ma stavolta immagazziniamo il risultato ottenuto in NUM3.
Prima fase:

LDA NUM1	copia il primo numero in A
ADDA NUM2	somma il secondo numero
STA NUM1	immagazzina il risultato in NUM1

Seconda fase:

LDB NUM1	copia il primo numero in B
SEX	trasforma il numero ad 8 bit contenuto in B in un numero a 16 bit e lo immagazzina in D
STD NUM3	copia D in NUM3
LDB NUM2	copia il secondo numero in B
SEX	lo trasforma in un numero a 16 bit e lo immagazzina in D
ADDD NUM3	somma a D il primo numero a 16 bit contenuto in NUM3
STD NUM3	registra il risultato in NUM3

La Melbourne House

Produttrice di giochi quali "Hobbit" e "Mugsy", questa ditta è presente anche nell'editoria

La Melbourne House è stata fondata nel 1977 dall'australiano Fred Milgrom. Il lancio dello ZX80 fece intuire a Milgrom le notevoli possibilità offerte dal mercato editoriale delle pubblicazioni per home computer e, nel 1980, la Melbourne House pubblicò il volumetto *30 Programs for the ZX80*. Il successo ottenuto portò ad una serie di pubblicazioni dedicate all'apparecchio della Sinclair e, poco tempo dopo, al lancio del primo pacchetto software della compagnia: il gioco *Space Invaders* per lo ZX80.

Nell'anno successivo comparve lo ZX81 ed il crollo delle vendite delle pubblicazioni dedicate allo ZX80 portò la Melbourne House sull'orlo del fallimento.

Dopo questa dura esperienza, la compagnia decise di diversificare la propria produzione. I libri ed il software pubblicati dalla Casa per i nuovi apparecchi che comparivano sul mercato ebbero un grandissimo successo, perché i manuali che corredavano i vari computer di solito erano poco chiari e di bassa qualità.

Per lo Spectrum Sinclair, la Melbourne House creò una serie di programmi di gioco che sfruttavano brillantemente la grafica ad alta risoluzione e le funzioni audio dell'apparecchio. Il gioco *Penetrator* ha riscosso un buon successo, ma il fiore all'occhiello della ditta è *The Hobbit* (un gioco grafico stile Adventure, tratto dall'omonimo romanzo di Tolkien), che ha vinto il premio Golden Joystick per il miglior gioco di strategia. La cassetta del gioco è venduta abbinata al libro di Tolkien, secondo una clausola del contratto che legava la casa all'autore del romanzo, e, nonostante il prezzo pittosto elevato, *The Hobbit* ha ottenuto un eccezionale successo di pubblico. Oggi il gioco viene prodotto in varie versioni. comprese quelle per il Micro BBC e l'Oric/Atmos.

La sede principale della ditta si trova a Melbourne, in Australia, e suoi programmatori lavorano, a gruppi di quattro, sui diversi aspetti dei giochi che vengono prodotti. Questa impostazione comporta tempi decisamente lunghi nello sviluppo dei programmi, ma è confortata da ottimi risultati di mercato, che a loro volta favoriscono la vendita dei libri pubblicati dalla casa. La Melbourne House pubblica libri per tre livelli di difficoltà (per lettori principianti, di medio livello ed esperti) con una copertina simile a quella del software cui si riferiscono: questa politica editoriale consente di trarre tutti i vantaggi possibili dal successo dei propri programmi di gioco. Inoltre, ad ogni pacchetto è allegata una scheda che il compratore è invitato a riempire esprimendo giudizi e consigli sul prodotto acquistato.

L'ultimo gioco prodotto si chiama *Mugsy* (vedere a pag. 828) e la Melbourne House lo ha definito "il primo fumetto comico del mondo che dialoga col lettore". Il giocatore viene posto al comando di una banda di teppisti che si muovono nella Chicago degli anni '20. La grafica è ben disegnata e ricca di particolari, ma lo sviluppo del gioco è piuttosto complesso.

La Melbourne House sta per lanciare un gioco chiamato *Sherlock Holmes*, un Adventure simile a *Hobbit*, che ha richiesto oltre 15 mesi di lavoro. Dell'argomento del gioco si sa ancora molto poco: l'unica notizia certa è che, per la maggior parte del tempo, il giocatore dovrà "viaggiare" su mezzi di trasporto dell'epoca vittoriana!



Alfred Milgrom, Condirettore della Melbourne House



Philip Mitchell, autore dei giochi "The Hobbit" e "Sherlock Holmes"

Dal libro al gioco

Per il lancio di "The Hobbit", la Melbourne House ha ideato una nuova formula di vendita: abbinare alla cassetta software il libro di Tolkein a cui si ispira la trama del gioco



IAN McKINNELL

Programmatori della Melbourne House



Le tascabili della Casio

Una panoramica sui computer da tasca, prodotti dalla nota casa giapponese

Tra le calcolatrici programmabili ed i computer tascabili esistono alcune differenze sostanziali: le prime possono essere usate per numerose applicazioni simili tra loro, adottano propri linguaggi speciali (spesso somiglianti al codice macchina) ed eseguono solo lavori di carattere numerico. I secondi, invece, offrono il vantaggio di usare il BASIC e di poter anche elaborare testi. I computer tascabili sono quindi lo strumento ideale per le attività che comportano il calcolo ripetitivo di una data formula o procedura consentendo, inoltre, la stesura di specifici programmi per la gestione di tali calcoli.

Il più economico tra i computer tascabili prodotti dalla Casio è l'FX-720P, che pesa soltanto 210 grammi e misura appena mm. 165 x 85 x 15. La tastiera, di tipo QWERTY, misura circa la metà di una normale tastiera per computer, ma i tasti sono incolonnati uno sotto l'altro, anziché essere sfalsati. Usati insieme ai tasti Shift e Function, i tasti alfabetici generano, rispettivamente, un secondo carattere (ad esempio, i segni d'interpunzione) e le parole chiave del BASIC. Diversamente dal solito però, i tasti Shift e Function vanno premuti prima dei tasti alfanumerici e non simultaneamente. Questa particolarità permette l'impiego della tastiera con una sola mano, tenendo l'apparecchio nell'altra.

L'FX-720P possiede un display a cristalli liquidi (LCD) con 12 caratteri, su di una sola riga. Due tasti per il cursore permettono di visualizzare righe di lunghezza maggiore, facendole scorrere verso destra o verso sinistra. Il contrasto del visualizzatore LCD può essere variato, a seconda dell'angolo di visuale e dell'illuminazione ambientale, tramite una piccola manopola, situata sul fianco sinistro. Per risparmiare energia, il computer si spegne automaticamente quando non viene usato per più di sei minuti.

L'FX-720P viene venduto con 2 soli Kbyte di memoria, ma sono disponibili cartucce ad innesto aggiuntive da 2 e da 4 Kbyte. Queste memorie sono di tipo non volatile, vale a dire che, una volta rimosse, i programmi o i dati registrati non vengono perduti. Ciò consente di distribuire convenientemente programmi e dati su più cartucce, da usare all'occorrenza. La loro limitata capacità però, costringe a frequenti sostituzioni, soprattutto se si considera che il computer può simultaneamente contenere in memoria fino a 10 programmi BASIC. Infine, per l'immissione e l'esecuzione dei programmi BASIC, occorre selezionare uno speciale modo di funzionamento. Il dialetto BASIC dell'FX-720P è sorprendentemente completo, per una macchina di dimensioni così ridotte: possiede quasi tutte le istruzioni e le funzioni standard, ad eccezione dei comandi CHR$ e ASC. Il comando BEEP genera due diversi suoni.

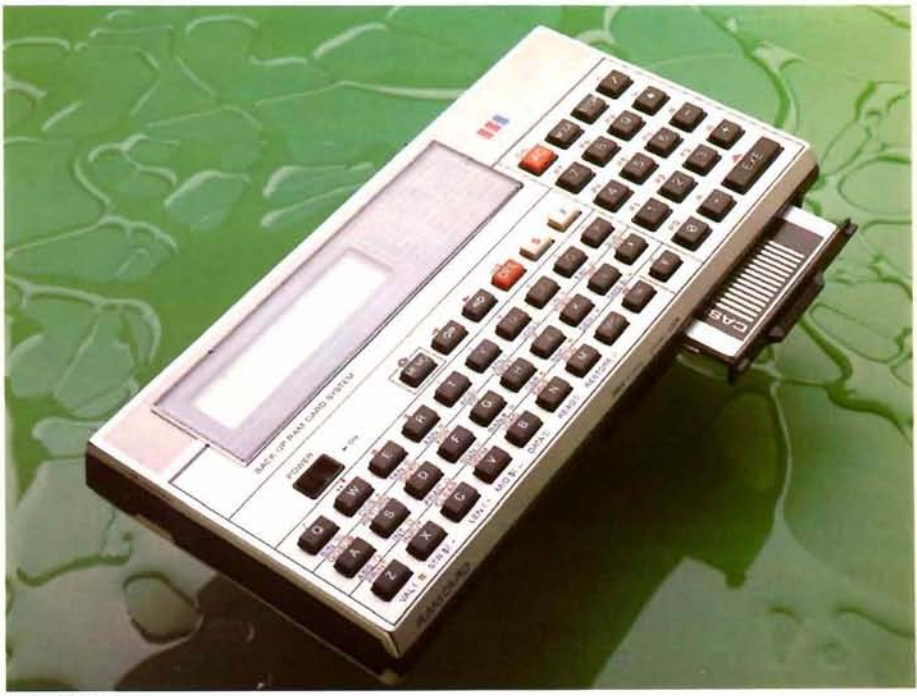
L'FX-720P è il più piccolo della gamma di computer tascabili della Casio, quindi molte delle sue funzioni sono state migliorate nei modelli successivi. Tuttavia, questo apparecchio è l'unico della "famiglia" ad offrire il *Data Bank*, un piccolo programma per la registrazione di nomi e indirizzi, note di spesa ed appuntamenti. Il programma è in grado di eseguire ricerche sulle informazioni immagazzinate, anche secondo criteri selettivi. È quindi un diretto concorrente dell'Organiser della Psion, ma costa meno ed è più versatile.

I Casio FX-750P e PB-700

Mentre l'FX-720P è destinato soprattutto agli uomini d'affari, il Casio FX-750P tiene particolarmente conto delle esigenze di scienziati e tecnici. L'apparecchio, che ha una larghezza di 185 mm. ed un peso di 250 grammi, possiede un display da 24 caratteri ed una tastiera leggermente migliore del modello precedente. Viene venduto completo di una

Bilancio ... in tasca

Il computer tascabile FX-720P, che possiede 2 Kbyte di memoria RAM, ha un display a cristalli liquidi con una sola linea ed un set completo di tasti alfabetici e numerici. La memoria può essere espansa, innestando sull'apparecchio cartucce aggiuntive da 2 Kbyte



CHRIS STEVENS



L'interfaccia FA-20
Tutti i computer tascabili della Casio possono essere collegati ad interfacce di questo tipo. L'FX-750P viene accolto nell'apposito alloggiamento dell'interfaccia FA-20, venduta con un set di batterie ricaricabili ed un adattatore di corrente

CHRIS STEVENS

cartuccia di memoria da 2 Kbyte, identica a quelle usate sull'FX-720P, ma, grazie ad un secondo connettore, si può ampliare la memoria fino ad 8 Kbyte. Diversamente dall'FX-720P, la pressione dei tasti Shift e Function deve avvenire contemporaneamente ai tasti alfanumerici.
Dieci dei suoi tasti sono associati a importanti costanti fisiche: la velocità della luce, l'accelerazione gravitazionale della Terra, le costanti di Planck, Boltzmann e Avogadro, la carica elettronica, la massa atomica, la massa elettronica, la costante gravitazionale ed il volume molare. Queste costanti sono comunemente usate in campo chimico e fisico e la loro presenza in memoria evita di doverle immettere ogni volta che occorrono in un calcolo. Il computer possiede anche altre caratteristiche insolite per un normale BASIC: sei funzioni iperboliche ed una serie di funzioni statistiche, che comprendono la deviazione standard, la regressione lineare e la correlazione. Altre funzioni di questo BASIC sono state migliorate e alcune potrebbero dimostrarsi molto utili anche sugli home e personal computer.
Passiamo ora al terzo computer tascabile della Casio, il PB-700. L'apparecchio è largo 200 mm. e pesa 315 grammi. Il display contiene 4 righe di 20 caratteri e la memoria di 4 Kbyte può essere espansa fino a 16 Kbyte (innestando un'unità di memoria sul retro della macchina). Come gli altri computer tascabili, il PB-700 è corredato da un astuccio morbido di protezione.
La tastiera somiglia a quella dell'FX-750P ma, al posto del tasto Function, prevede un tasto Caps che, premuto contemporaneamente ai tasti alfabetici, consente di usare le lettere minuscole. Rispetto all'FX-750P, il PB-700 possiede alcuni comandi BASIC in più per visualizzare grafici sul display LCD, la cui risoluzione è di 32 x 160 punti. Dispone inoltre di comandi per tracciare grafici, che controllano un'unità d'interfaccia, chiamata FA-10, per le cassette e la stampante/plotter a colori. Quest'ultima ha una testina dotata di quattro penne a sfera colorate, in grado di scrivere testi oppure tracciare linee, cerchi o assi cartesiani per i grafici, su fogli di carta larghi 115 mm.
L'interfaccia FA-10 contiene anche una porta per registratore a cassette, che consente di salvare programmi e dati tramite un normale registratore. Tuttavia, sulla stampante/plotter può essere innestato un piccolo registratore a microcassette che, assieme alle altre unità, trasforma il PB-700 in un sistema di elaborazione portatile completo. Esiste anche un'interfaccia Centronics, venduta a parte, che permette di collegare l'apparecchio a comuni stampanti per computer.
Per l'FX-720P e l'FX-750P vengono prodotte diverse stampanti e interfacce per cassette. L'FX-720P usa la stampante FP-12S e un'interfaccia separata, detta FA-3, per il registratore a cassette. L'FX-750 adotta invece l'unità FA-20, una combinazione composta da una piccola stampante e da un'interfaccia per cassette. L'FA-20 possiede un connettore per schede di memoria aggiuntiva ed incorpora un set di batterie ricaricabili, un trasformatore per la corrente di rete ed una custodia protettiva.
I manuali di corredo ai tre computer sono esaurienti, ma in alcuni punti appaiono poco chiari per le difficoltà dovute alla traduzione dal Giapponese. Un altro grosso problema è la scarsa disponibilità di software: chi possiede questi computer tascabili deve quindi svilupparlo in proprio o limitarsi a copiare i semplici programmi contenuti nei manuali.

PRODOTTO	MISURE	PESO	MEMORIA	DISPLAY	CARATTERISTICHE
FX-720P	165 x 85 x 15 mm.	210 g.	2K	1 x 12 car.	Database interno. Tastierina numerica
FX-750P	185 x 85 x 15 mm.	250 g.	2K	1 x 24 car.	Slot per seconda cartuccia. Costanti fisiche preprogrammate. BASIC esteso
PB-700	200 x 85 x 15 mm.	315 g.	4K	4 x 20 car.	Capacità grafiche. Stampante/plotter disponibile



Divertirsi con la tartaruga

Il funzionamento in modo diretto e le procedure nel linguaggio LOGO

Molti programmatori di micro hanno scarsa dimestichezza con le procedure, tuttavia il principio di queste "sequenze di istruzioni" è veramente molto semplice. Proviamo a scrivere una procedura LOGO, assegnandole il nome QUADRATO. (Si rammenti che i comandi LOGO devono essere sempre battuti in caratteri maiuscoli, mentre i nomi di procedura possono essere in maiuscolo o in minuscolo). Inseriamo adesso il comando:

```
EDIT QUADRATO
```

Questa istruzione, dopo aver ripulito lo schermo, visualizza la scritta TO QUADRATO sulla prima riga (per ricordarci il nome della procedura). Sull'ultima, riservata invece ai messaggi di assistenza, compare EDIT CTRL-C TO DEFINE CTRL-G TO ABORT, che rimane finché ci troviamo nel modo di "edit". EDIT significa che il LOGO ha lasciato il modo "immediato" ed attende la creazione di una nuova procedura, che viene memorizzata (ossia incorporata nel LOGO) battendo Control-C, mentre Control-G serve per cancellarla, ossia rimuoverla dal LOGO. Nel modo di edit, il sistema non riconosce né esegue i normali comandi, ma si limita semplicemente ad annotarli nel proprio "dizionario", alla voce corrispondente al nome assegnato alla procedura. La versione finale della procedura QUADRATO è la seguente:

```
TO QUADRATO
   REPEAT 4 [FD 50 RT 90]
END
```

Battendo Control-C ordiniamo al LOGO di incorporare la nuova procedura. Eseguita questa operazione, il sistema visualizza il messaggio QUADRATO DEFINED. Se la procedura contiene un errore, possiamo battere Control-G per cancellarla e poi riscriverla, oppure tornare nel modo di edit ed apportare le correzioni necessarie. Nel secondo caso, l'inserimento e la correzione dei caratteri sono facilitati dall'uso dei tasti del cursore, che consentono di muoversi rapidamente sullo schermo. Nella "linea di edit" (in modo immediato), è consentito correggere soltanto l'ultima linea redatta. Più avanti verranno esaminati i comandi LOGO che facilitano la redazione e la correzione delle procedure di maggiori dimensioni. Vediamo adesso come funziona la procedura appena creata. Digitiamo il comando DRAW, per abilitare lo schermo grafico, e battiamo il nome QUADRATO: le istruzioni inserite nella procedura vengono ora interpretate come veri e propri comandi e la tartaruga traccia un quadrato sullo schermo. Si noti che l'uso delle procedure è identico a quello dei comandi base o "primitivi" del LOGO: infatti, battendo il nome di una procedura, vengono eseguite le istruzioni contenute al suo interno. Ogni nuova procedura, quindi, può essere considerata come un nuovo comando primitivo. In altre parole, il LOGO è un linguaggio "espandibile" e tale flessibilità permette di adattarlo alle esigenze più diverse.

La procedura QUADRATO è stata programmata per tracciare un quadrato con lati lunghi 50 unità: proviamo adesso a modificarne la misura per ottenere un quadrato con lati di 30 unità. Torniamo al modo di edit battendo il comando:

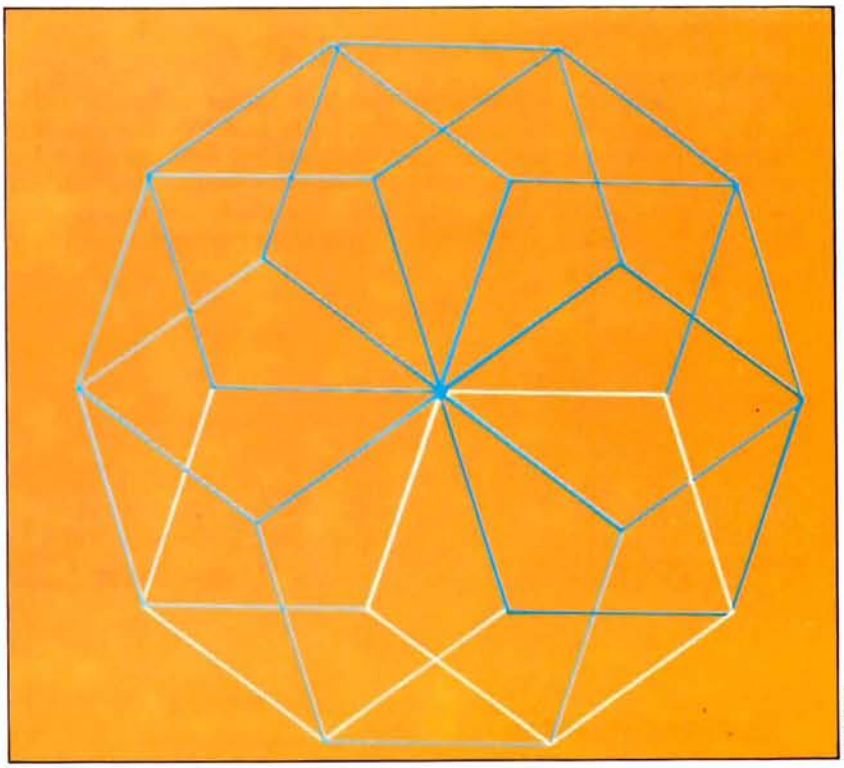
```
EDIT QUADRATO
```

Sullo schermo compare il listato della procedura, che modificheremo sostituendo 50 con 30. Fatto ciò, concludiamo nuovamente questa fase con Control-C e battiamo QUADRATO per controllare che il nuovo quadrato sia quello voluto. Come esercizio, si provino a scrivere le procedure per le figure illustrate a pag. 1013: triangoli, rettangoli, pentagoni, stelle, e via dicendo, magari ricorrendo anche al comando REPEAT.

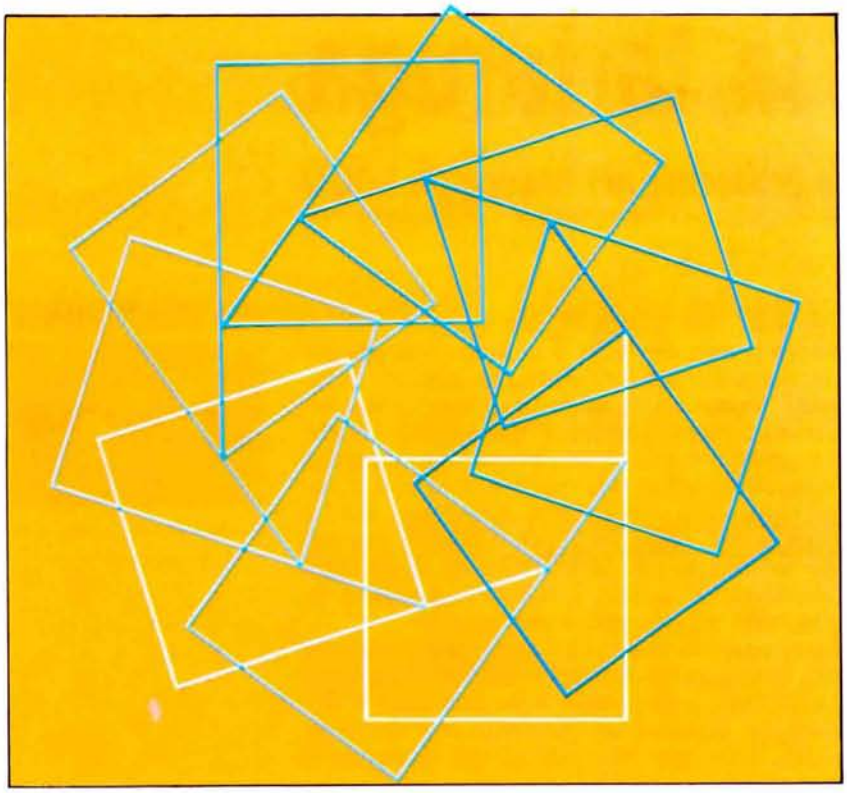
ABBREVIAZIONI

EDIT	ED
HIDETURTLE	HT
SHOWTURTLE	ST

PENTAGONO



LIZ DIXON



QUADRATO

Ad esempio, la figura PENT, definita con la sequenza:

```
REPEAT 5 [FD 50 RT 72]
```

può essere inserita in:

```
REPEAT 10 [PENT RT 36]
```

Il comando REPEAT può essere usato con qualsiasi figura e consente di creare facilmente e rapidamente immagini complesse.
Proviamo ora l'istruzione:

```
REPEAT 10 [QUADRATO RT 36 FD 25]
```

L'immagine tracciata sullo schermo dalla tartaruga è riprodotta nell'illustrazione. Nel tentativo di programmare le figure riportate a pag. 1013, si possono ottenere risultati del tutto inattesi. Se il primo tentativo per disegnare un triangolo è questo:

```
REPEAT 3 [FD 50 RT 60]
```

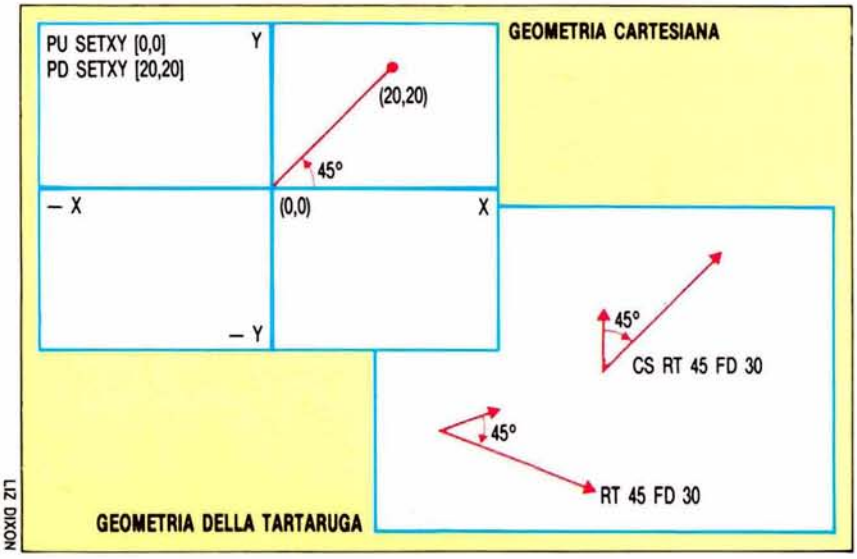
la tartaruga non traccia un triangolo, bensì un mezzo esagono! La creazione di una stella può dimostrarsi più difficile di quanto si possa pensare.Si provi allora un diverso approccio al problema: immaginiamo di trovarci al posto della tartaruga che si muove sullo schermo (corre voce che sia possibile distinguere i programmatori BASIC da quelli LOGO, osservando i movimenti delle loro spalle quando programmano!); dovendo tracciare una figura chiusa, la tartaruga percorrerà tutto il perimetro della figura, per poi tornare alla posizione di partenza, con la parte anteriore rivolta verso la direzione originale. Vale a dire che essa deve ruotare di una serie di angoli, la cui somma è un multiplo di 360 gradi. Se la figura è "convessa" (cioè nessuno dei suoi angoli interni supera i 180 gradi), la rotazione sarà esattamente di 360 gradi. Nel caso di un triangolo equilatero, la tartaruga deve ruotare tre volte e la somma dei tre angoli è data da 360/3=120 gradi. Il comando per disegnare un triangolo di questo tipo è:

```
REPEAT 3 [FD 50 RT 120]
```

Questo principio è valido anche per i poligoni non convessi, come la stella: in tal caso, però, la somma degli angoli tracciati sarà un numero intero multiplo di 360 gradi (perché le figure disegnate sono più di una). Questi sono concetti di carattere generale (non valgono solo per i poligoni) e vengono comunemente indicati con l'espressione "teorema del percorso della tartaruga". Se ci immaginiamo di essere la tartaruga, ci accorgiamo di non poter tracciare una figura quale una stella a sei punte muovendoci solo in avanti e ruotando sempre dello stesso angolo. È quindi necessaria una procedura più complessa.

"Cogito ergo ... LOGO"

Per guidare la tartaruga possiamo usare tanto la geometria cartesiana (assoluta) quanto quella della "Tartaruga" (relativa). Nel primo caso, le coordinate vengono misurate in base ad un sistema di assi immaginari X e Y, con origine [0,0] in posizione CS. Nel secondo, i comandi fanno riferimento alla posizione e alla direzione corrente della tartaruga



LIZ DIXON

Coordinate geometriche

I movimenti della tartaruga dipendono dalle proprietà delle figure disegnate e non dai rapporti che queste hanno con un punto di riferimento esterno, quale l'origine di un sistema di coordinate geometriche. Quella della tartaruga è quindi una geometria "relativa": i movimenti vengono misurati in base ad un parametro numerico detto "unità" ed hanno come punto di riferimento la corrente posizione della tartaruga sullo schermo. Al contrario, le coordinate geometriche usano valori "assoluti": lo schermo viene idealmente suddiviso in una griglia e le coordinate orizzontali e verticali vengono misurate a partire dall'origine specificata. Ciascun punto della griglia viene così definito mediante una coppia di valori fissi (i valori delle coordinate) ed i movimenti si basano su questi valori. Esiste una tecnica che permette di programmare i movimenti della tartaruga ricorrendo alle coordinate geometriche.

Consideriamo il comando SETXY 20 30, che sposta la tartaruga dalla sua posizione corrente al punto (20,30). Avendo specificato l'attributo PENDOWN, la tartaruga traccia una linea durante il movimento. Al contrario, usando PENUP, la tartaruga si sposta senza disegnare alcuna linea. L'origine (0,0) si trova nel centro dello schermo. Le seguenti procedure portano al medesimo risultato, ma chiariscono le differenze tra le coordinate della tartaruga e quelle geometriche:

```
TO QUADRATO1
   REPEAT 4 [FD 50 RT 90]
END

TO QUADRATO2
   SETXY 0 50
   SETXY 50 50
   SETXY 50 0
   SETXY 0 0
END
```

Impartito il comando DRAW, che abilita lo schermo grafico, battiamo i nomi di procedura QUADRATO1 e QUADRATO2: la tartaruga traccia figure identiche. Proviamo adesso a ruotare entrambi i quadrati di 30 gradi. Nel primo caso l'istruzione RT 30 QUADRATO1 lavora in modo corretto, mentre nel secondo occorre riscrivere l'intera procedura (in quanto usa coordinate di schermo "assolute"). Ciò non toglie che, in alcuni casi, sia più conveniente usare le coordinate assolute: SETXY traccia le linee molto più velocemente di FORWARD.
Un'altra particolarità della geometria della tartaruga è la ... "miopia". La tartaruga si muove compiendo un solo "passo" alla volta e traccia le figure con una serie di spostamenti successivi. Rimettiamoci nei panni della tartaruga ed immaginiamo di dover tracciare un cerchio: ci muoveremo in avanti per un breve tratto, poi ruoteremo leggermente, quindi faremo un altro "passo" e così via fino a chiudere la figura. La procedura LOGO per questi movimenti è la seguente:

```
TO CERCHIO
REPEAT 360 [FD 1 RT 1]
END
```

La sua esecuzione è molto lenta: per renderla più rapida, si può evitare la visualizzazione della tartaruga ad ogni passo: SHOWTURTLE abilita la visione della tartaruga, HIDETURTLE la rende "invisibile". La procedura vista, in realtà, traccia un poligono i cui 360 lati sono così corti da produrre l'effetto di una curva. Un poligono con 36 lati è già sufficiente per ottenere questo effetto ottico e la sua procedura è molto più rapida della precedente:

```
TO CERCHIO
   REPEAT 36 [FD 10 RT 10]
END
```

Proviamo adesso a scrivere le procedure necessarie per disegnare un semi-cerchio, un quarto di cerchio ed un 1/6 di cerchio oppure a combinarne due assieme, per ottenere l'immagine stilizzata di un fiore.



ALAN COODE PER CONCESSIONE DELLA VALIANT DESIGN LTD.

Come esercizio finale, si progetti una procedura che, anziché disegnare un cerchio, tracci una spirale attorno ad un punto centrale. La procedura qui presentata è piuttosto tortuosa, poiché vengono usati soltanto i comandi fin qui esaminati, tuttavia è sufficiente per chiarire il concetto:

```
TO SPIRALE
   FD 10 RT 10 FD 10 RT 20 FD 10 RT 30
   FD 10 RT 40 FD 10 RT 50 FD 10 RT 60
   FD 10 RT 70 FD 10 RT 80 FD 10 RT 90
END
```

Logo & Logo

I programmi "editor" del LOGO sono tutti più o meno simili: alcune piccole differenze sono dovute alle tastiere adottate nei vari apparecchi: si consiglia di consultare il manuale per il LOGO disponibile per il proprio apparecchio.
Per tutte le versioni LOGO della LCSI, il comando per editare la procedura QUADRATO è EDIT "QUADRATO (le virgolette devono precedere, ma non seguire il nome della procedura). Inoltre, l'istruzione per posizionare la tartaruga sul punto con coordinate (20 30) è SETPOS [2 30].

Soluzione agli esercizi

Triangolo
```
REPEAT 3 [FD 50 RT 120]
```
Pentagono
```
REPEAT 5 [FD 50 RT 72]
```
Esagono
```
REPEAT 6 [FD 60 RT 60]
```
Rettangolo
```
REPEAT 2 [FD 25 RT 90 FD 50 RT 90]
```
Parallelogramma
```
REPEAT 2 [FD 25 RT 70 FD 50 RT 110]
```
Rombo
```
REPEAT 2 [FD 50 RT 70 FD 50 RT 110]
```
Stelle:

1.
```
REPEAT 5 [FD 50 RT 144]
```
2.
```
REPEAT 8 [FD 50 RT 135]
```
3.
```
RT 30 REPEAT 3 [FD 50 RT 120]
PU LT 30 FD 29 RT 90 PD
REPEAT 3 [FD 50 RT 120]
```
4.
```
REPEAT 10 [FD 50 RT 108]
```

Impariamo col LOGO

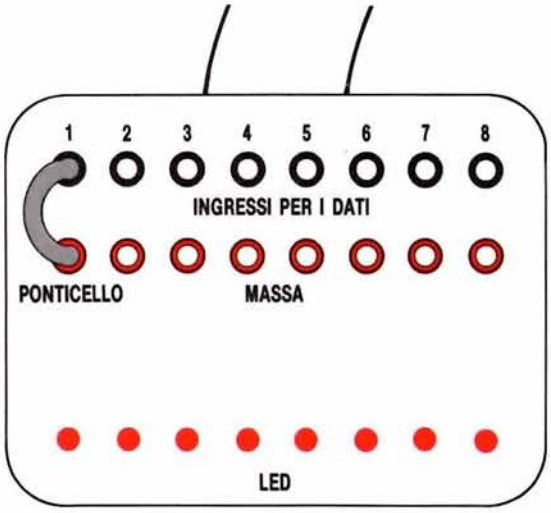
Il gioco Demolition Turtle, creato da Antony Ginn, si serve di una tartaruga da pavimento per sviluppare nei bambini la percezione dello spazio. Un giocatore pone sul pavimento una torre-giocattolo e stabilisce la posizione di partenza e la direzione della tartaruga. Il secondo giocatore deve cercare di colpire la torre con la tartaruga, programmandola col minor numero possibile di istruzioni



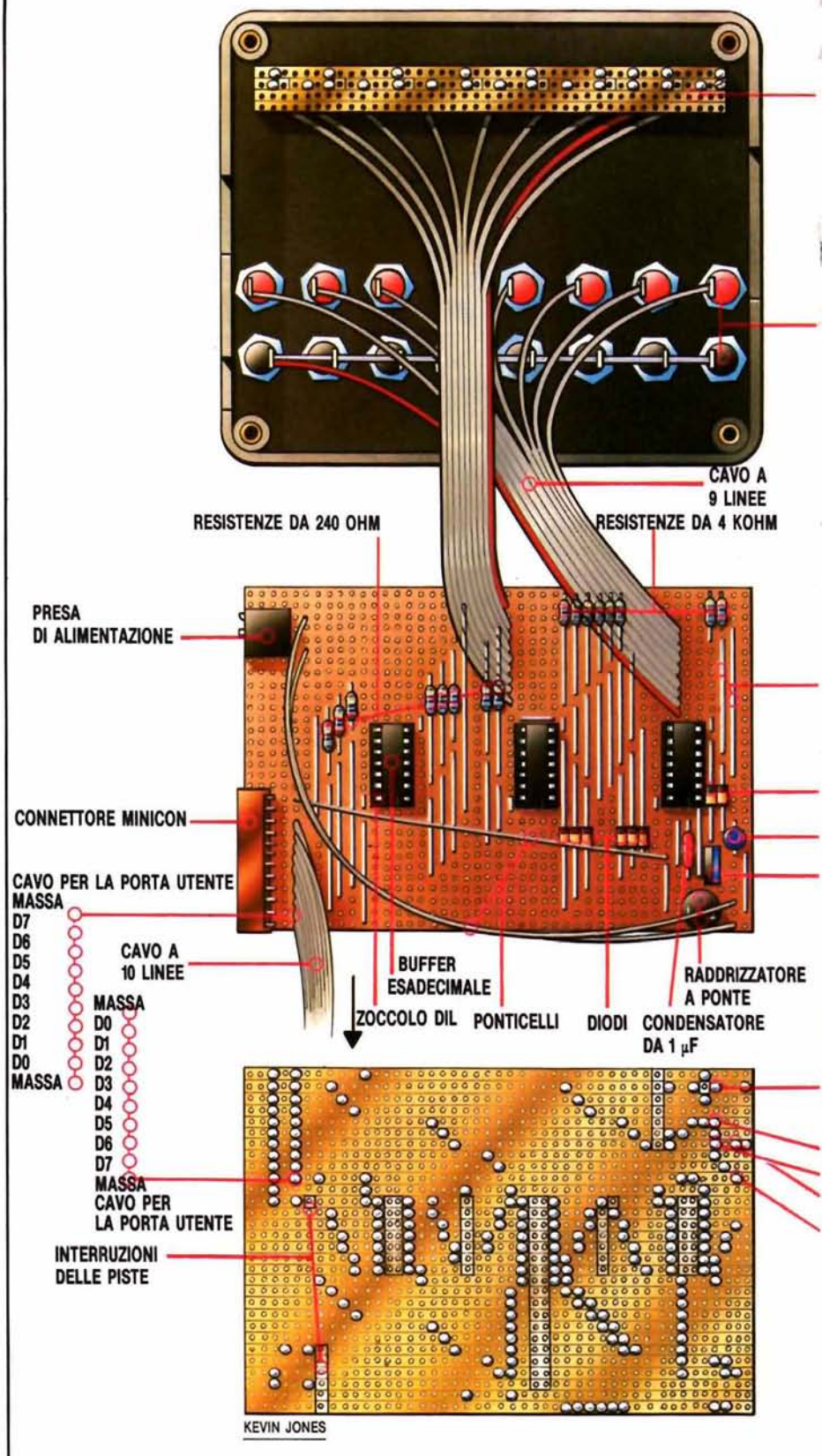
Collaudiamo la "scatola nera"

È giunto il momento di verificare il funzionamento del dispositivo costruito

Ultimata la costruzione del "buffer box", possiamo scrivere un semplice programma per usare l'apparecchio come strumento "provariflessi". Qui utilizziamo il bit 7 del registro per i dati dell'interruttore d'ingresso ed i bit da 0 a 6 per i LED. Il nostro scopo è sincronizzare tra loro l'accensione dei LED da parte del computer e l'interruttore di commutazione. Innanzi tutto dobbiamo collegare con un ponticello la presa d'ingresso 7 a quella di "massa" corrispondente. Tale connessione può essere di qualsiasi tipo: anche un ponticello a due spinotti, poiché viene impiegato esclusivamente per aprire e chiudere il circuito tra il bit 7 del registro dei dati e la massa. Effettuato il collegamento, dobbiamo assicurarci che il ponticello sia in grado di interrompere il circuito tra il bit 7 e la terra (il valore del bit 7 dovrebbe essere 1). La figura seguente mostra i collegamenti necessari:
Il primo compito del nostro programma di verifica è quello di organizzare il flusso dei dati, in modo da selezionare il bit 7 per l'input e tutti gli altri bit per l'output, azzerando poi



il registro dei dati. Dopo un intervallo di tempo casuale, vengono accesi 7 LED e viene inizializzato l'orologio interno del computer. La temporizzazione ha termine quando il bit 7 viene collegato a massa. Ricordiamo che i LED sono collegati in modo da accendersi quando i bit del registro sono posti a zero e spegnersi quando sono posti ad uno.



KEVIN JONES

CIRCUITERIA DEL BUFFER

Poggiare il filo di stagno su ciascun contatto delle prese e saldare, verificando poi il collegamento con un "tester". Di un cavo a nastro lungo 20 cm. si eliminino tre cavetti (ma non quello colorato) ottenendo così un nastro a nove linee. Togliere il rivestimento e stagnare le estremità dei cavetti. Saldare il cavetto colorato all'ultima presa a sinistra, quindi saldare gli altri 8 cavetti in successione ai contatti delle prese. Controllare col tester le saldature tra i cavetti ed i contatti delle prese

LED

PRESE

PONTICELLI

DIODI

CONDENSATORE ELETTROLITICO DA 1 µF

STABILIZZATORE DI TENSIONE

RADDRIZZATORE A PONTE

\+

REGOLATORE DI TENSIONE

–

\+

CONDENSATORE ELETTROLITICO DA 1 µF

–

COSTRUZIONE DELLA PIASTRA

La piastra qui mostrata possiede 30 piste con 45 fori ciascuna le cui misure si adattano perfettamente alle dimensioni della scatola del buffer. Per non commettere errori in fase di costruzione, si segua con attenzione la disposizione delle saldature, visibile sul retro della piastra. Per evitare sbavature di stagno, si adoperi un saldatore di piccole dimensioni. Prima di inserire un nuovo componente, si controlli più volte la sua esatta collocazione sulla piastra. I diodi, il condensatore elettrolitico, il raddrizzatore a ponte e lo stabilizzatore di tensione hanno un orientamento ben preciso, mostrato in figura (basarsi sui loro contrassegni più/meno): qualsiasi altro potrebbe danneggiarli. Tutti i componenti sono sensibili al calore: le saldature devono essere fatte rapidamente. Si presti attenzione nell'inserire i piedini del connettore minicon e di quello d'alimentazione nei giusti fori. Se necessario, piegare i piedini, ma con delicatezza. Per realizzare i ponticelli, si possono adoperare i cavetti asportati dal cavo a nastro.

Quando tutti i componenti del circuito sono al loro posto, si può passare al taglio delle piste di rame. Questa operazione può essere effettuata con un apposito strumento, oppure introducendo la punta di un trapano in un foro e facendola girare fino ad ottenere il taglio netto e senza sbavature della pista. Saldare adesso i cavi a nastro sulla scheda. Per la disposizione dei collegamenti per le prese rosse e nere e per i LED ci si riferisca alla posizione del cavetto colorato. Per i collegamenti della porta utente è necessaria la massima attenzione: le due linee di massa vanno collegate ai fori 1 e 10 (contando a partire dal bordo della scheda), mentre le altre linee di collegamento devono essere disposte in successione sui fori compresi tra 2 e 9. La linea 1 deve quindi trovarsi vicino al bordo della scheda.

Infine, usando la scheda come modello, si incidano due tacche sul fianco della scatola del buffer (corrispondenti ai connettori) ed una fessura per il passaggio del cavo a nastro per la porta utente.

Il programma impiega la variabile riservata TIME, che fornisce il valore del tempo intercorso (in centesimi di secondo) dal suo ultimo azzeramento. L'impiego di una AND logica, alla linea 130, isola il bit 7 e consente di esaminarlo indipendentemente dagli altri bit del registro per i dati (vedere a pag. 546).

```
 10 REM** PROGRAMMA PROVA-RIFLESSI PER IL BBC**
 15 :
 20 DDR=&FE62: REGDAT=&FE60
 30 ?DDR=127: REM LINEE DA 0 A 6 IN USCITA
 40 ?REGDAT=127: REM LED SPENTI
 50 :
 60 CLS: PRINT "PRONTO..."
 70 DELAY=3000+RND(9000)
 80 FOR I=1 TO DELAY: NEXT: REM CICLO DI RITARDO
 85 FOR D=1 TO 200: NEXT D
 90 :
 97 REPEAT UNTIL ?REGDAT AND 128=1
100 ?FE60=0: REM ACCENDE I LED
110 TIME=0: INIZIALIZZA L'OROLOGIO INTERNO
120 REPEAT
130 UNTIL ?REGDAT AND 128=0: REM CONTATTO INSERITO
140 :
150 PRINT "TEMPO IMPIEGATO="TIME/100"SECONDI"
160 END
```

Inserendo il ponticello, il valore del bit 7 cambia da 1 a 0.

Possiamo scrivere un programma simile per il Commodore 64, usando la variabile TI abbinata al suo orologio interno. TI funziona in modo diverso da TIME, poiché fornisce l'intervallo di tempo trascorso dal momento dell'accensione dell'apparecchio, misurandolo in sessantesimi di secondo. Per calcolare il tempo trascorso, occorre quindi sottrarre dal valore finale di TI quello letto all'inizio dell'intervallo di tempo da misurare.

In seguito verrà descritta la realizzazione di un dispositivo per l'invio di correnti di maggior intensità a motori elettrici, e lo sviluppo

```
 10 REM C8M 64 ** MISURA-RIFLESSI **
 20 :
 30 DDR=56579: REGDAT=56577
 40 POKE DDR,127: REM LINEE DA 0 A 6 IN USCITA
 50 POKE REGDAT, 127: REM LED SPENTI
 60 :
 70 PRINT CHR$(147): REM RIPULISCE LO SCHERMO
 80 PRINT "PRONTO..."
 90 DE=3000+INT(9000*RND(1))
100 FOR N=1 TO DE: NEXT: REM CICLO DI RITARDO
110 :
120 POKE REGDAT,0: REM ACCENDE I LED
130 T=TI: REM REGISTRA IL TEMPO INIZIALE
140 IF PEEK (REGDAT) AND 128<>0 THEN 140
150 :
160 TM=(TI-T)/60: REM CALCOLA INTERVALLO
170 PRINT "TEMPO IMPIEGATO=";TM;"SECONDI"
180 END
```

del software di controllo per motori bidirezionali e a velocità variabile.

ESPERIMENTI DA PROVARE

1) Scrivere un programmma che accenda i LED, uno alla volta, iniziando da quelli più a sinistra.
2) Scrivere un programma che accenda i LED in sequenza sulle linee da 1 a 6 (come il precedente), ma preveda un interruttore sulla linea 7, per invertire la direzione della sequenza. È possibile modificare il programma in modo da accendere in sequenza un "treno" di tre LED consecutivi ?
3) Scrivere un programma che simuli il lancio di un dado usando 6 LED ed un interruttore.
4) Scrivere un programma che, usando tre LED, simuli il funzionamento di un semaforo.
5) Scrivere un programma che conti le "automobili" (rappresentate dalle battute di un interruttore) in arrivo al semaforo rosso, facendo scattare il verde quando il numero delle automobili in attesa è maggiore di 10 oppure dopo che è trascorso un minuto dall'ultimo cambiamento del semaforo.

RISPOSTE AGLI ESERCIZI

Ecco alcune possibili soluzioni agli esercizi di pag. 1027:

SEQUENZA DI LED (I)

```
10 REM VERSIONE 2.1 - CBM 64
20 DDR=56579 : REGDAT=56577
30 POKE DDR,255 : REM TUTTE LINEE IN USCITA
40 POKE REGDAT, 255: REM TUTTI I LED SPENTI
50 GET A$
60 FOR N=1 TO 7
70 POKE REGDAT, 255-(2^N)
80 NEXT N
90 IF A$="" THEN 50
100 END
```

```
10 REM VERSIONE 2.1 - MICRO BBC
20 DDR=&FE62:REGDAT=&FE60
30 ?DDR=255: REM TUTTE LINEE IN USCITA
40 ?REGDAT=255: REM TUTTI I LED SPENTI
50 REPEAT
60 A$=INKEY$(1)
70 FOR N=0 TO 7
80 ?REGDAT=255-(2^N)
85 FOR D=1 TO 200: NEXT D
90 NEXT N
100 UNTIL A$<>""
110 END
```

SEQUENZA DI LED (II)

```
10 REM VERSIONE 2.2 - CBM 64
20 DDR=56579 : REGDAT=56577 : S=1
30 POKE DDR, 127: REM LINEA 7 INGRESSO
40 POKE REGDAT,255: REM TUTTI I LED SPENTI
50 GET A$
60 FOR N=0 TO 7 STEP S
70 POKE REGDAT,255-(2^N)
80 NEXT N
90 IF PEEK (REGDAT) AND 128=0 THEN S=-1
100 IF PEEK (REGDAT) AND 128=1 THEN S=1
110 IF A$="" THEN 50
120 END
```

```
10 REM VERSIONE 2.2 - MICRO BBC
20 DDR=&FE62:REGDAT=&FE60:S=1
30 ?DDR=127: REM LINEA 7 IN INGRESSO
40 ?REGDAT=255: REM TUTTI I LED SPENTI
50 REPEAT
60 A$=INKEY$(1)
70 FOR N=0 TO 7 STEP S
80 ?REGDAT=255-(2^N)
85 FOR D=1 TO 200: NEXT D
90 IF REGDAT AND 128=0 THEN S=-1 ELSE S=1
100 NEXT N
110 UNTIL A$<>""
120 END
```

SEQUENZA DI LED (III)

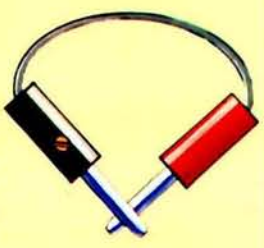
Per il "treno" di LED, apportare le seguenti modifiche alla risposta (II):

```
VERSIONE CBM
60 FOR N=2 TO 7 STEP S
70 POKE REGDAT,255-((2^N)+2^(N-1)+2^(N-2))
```

```
VERSIONE BBC
70 FOR N=2 TO 7 STEP S
80 ?REGDAT=255-((2^N)+2^(N-1)+2^(N-2))
```

PONTICELLO

Per collegare una presa nera ad una rossa si usi un filo elettrico isolato, lungo circa 8 cm. (vanno bene anche gli spezzoni tolti dal cavo a nastro). Occorrono otto ponticelli di questo tipo, ai quali fissare gli spinotti colorati



KEVIN JONES

LANCIO DI UN DADO

```
10 REM VERSIONE 2.3 - CBM 64
20 DDR=56579: REGDAT=56577
30 POKE DDR,127: REM INPUT SU LINEA 7
40 POKE REGDAT,255: REM TUTTI I LED SPENTI
50 GET A$
60 IF PEEK(REGDAT) AND 128<>0 THEN 60
70 NR=INT(RND(1)*6)+1: REM SELEZ. NUMERO
80 POKE REGDAT,255-((2^NR)-1)
90 IF PEEK(REGDAT) AND 128<>1 THEN 90
100 IF A$="" THEN 50
110 END
```

```
10 REM VERSIONE 2.3 - MICRO BBC
20 DDR=&FE62: REGDAT=&FE60
30 ?DDR=127: REM INPUT SU LINEA 7
40 ?REGDAT=255: REM TUTTI I LED SPENTI
50 REPEAT
60 A$=INKEY$(1)
70 REPEAT UNTIL (?REGDAT AND 128)=0
80 NUMERO=RND(6): REM SELEZ. NUMERO
85 FOR D=1 TO 200: NEXT D
90 ?REGDAT=255-((2^NUMERO)-1)
100 REPEAT UNTIL ?REGDAT AND 128=1
110 UNTIL A$<>""
120 END
```

SEMAFORO (I)

```
10 REM VERSIONE 2.4 - CBM 64
20 DDR=56579: REGDAT=56577
30 POKE DDR,255: REM TUTTE LINEE IN USCITA
40 POKE REGDAT,255: REM TUTTI LED SPENTI
45 REM BIT2=ROSSO,BIT1=GIALLO,BIT0=VERDE
50 RS=255-4: GL=255-2: VD=255-1
60 GET A$
70 POKE REGDAT,RS
80 FOR N=1 TO 200: NEXT: REM CICLO RITARDO
90 POKE REGDAT,RS+GL
100 FOR N=1 TO 40: NEXT: REM CICLO RITARDO
110 POKE REGDAT,VD
120 FOR N=1 TO 200: NEXT: REM CICLO RITARDO
130 IF A$="" THEN 60
140 END
```

```
10 REM VERSIONE 2.4 - MICRO BBC
20 DDR=&FE62: REGDAT=&FE60
30 ?DDR=255: REM TUTTE LINEE IN USCITA
40 ?REGDAT=255: REM TUTTI LED SPENTI
50 REM BIT2=ROSSO,BIT1=GIALLO,BIT0=VERDE
60 ROSSO=255-4: GIALLO=255-2: VERDE=255-1
70 REPEAT
80 A$=INKEY$(100)
85 FOR D=1 TO 200: NEXT D
90 ?REGDAT=ROSSO
100 FOR N=1 TO 200: NEXT: REM CICLO RITARDO
110 ?REGDAT=ROSSO+GIALLO
120 FOR N=1 TO 40: NEXT: REM CICLO RITARDO
130 ?REGDAT=VERDE
140 FOR N=1 TO 200: NEXT: REM CICLO RITARDO
150 UNTIL A$<>""
160 END
```

SEMAFORO (II)

Apportare le seguenti modifiche al precedente programma:

```
VERSIONE CBM 64
30 POKE DDR,127: REM INPUT SU LINEA 7
65 T=TI: REM LETTURA TEMPO
75 IF PEEK(REGDAT) AND 128=0 THEN C=C+1
76 IF PEEK(REGDAT) AND 128<>1 THEN 76
77 IF C<=10 AND T-TI<3600 THEN 75
```

```
VERSIONE BBC
30 ?DDR=127: REM INPUT SU LINEA 7
75 T=0: C=0
95 REPEAT
96 IF ?REGDAT AND 128=0 THEN C=C+1
97 REPEAT UNTIL ?REGDAT AND 128=1
98 UNTIL T>6000 OR C>10
```

DATABASE

La prima di due tabelle, dedicate al codice macchina del chip 6809

Istruzione	Formato	Modi d'indirizzamento: Immediato Op	Immediato ~	Immediato #	Diretto OP	Diretto ~	Diretto #	Indicizzato(1) OP	Indicizzato(1) ~	Indicizzato(1) #	Esteso OP	Esteso ~	Esteso #	Inerente OP	Inerente ~	Inerente #	Descrizione	5 H	3 N	2 Z	1 V	0 C
ABX														3A	3	1	B + X→X (Senza segno)	•	•	•	•	•
ADC	ADCA	89	2	2	99	4	2	A9	4+	2+	B9	5	3				A + M + C→A	↕	↕	↕	↕	↕
	ADCB	C9	2	2	D9	4	2	E9	4+	2+	F9	5	3				B + M + C→B	↕	↕	↕	↕	↕
ADD	ADDA	8B	2	2	9B	4	2	AB	4+	2+	BB	5	3				A + M→A	↕	↕	↕	↕	↕
	ADDB	CB	2	2	DB	4	2	EB	4+	2+	FB	5	3				B + M→B	↕	↕	↕	↕	↕
	ADDD	C3	4	3	D3	6	2	E3	6+	2+	F3	7	3				D + M:M + 1→D	•	↕	↕	↕	↕
AND	ANDA	84	2	2	94	4	2	A4	4+	2+	B4	5	3				A ∧ M→A	•	↕	↕	0	•
	ANDB	C4	2	2	D4	4	2	E4	4+	2+	F4	5	3				B ∧ M→B	•	↕	↕	0	•
	ANDCC	1C	3	2													CC ∧ IMM→CC					(3)
ASL	ASLA													48	2	1	A	(4)	↕	↕	↕	↕
	ASLB													58	2	1	B	(4)	↕	↕	↕	↕
	ASL				08	6	2	68	6+	2+	78	7	3				M	(4)	↕	↕	↕	↕
ASR	ASRB													47	2	1	A	(4)	↕	↕	•	↕
	ASR													57	2	1	B	(4)	↕	↕	•	↕
	ASR				07	6	2	67	6+	2+	77	7	3				M	(4)	↕	↕	•	↕
BIT	BITA	85	2	2	95	4	2	A5	4+	2+	B5	5	3				Verifica del Bit A (M ∧ A)	•	↕	↕	0	•
	BITB	C5	2	2	D5	4	2	E5	4+	2+	F5	5	3				Verifica del Bit B (M ∧ B)	•	↕	↕	0	•
CLR	CLRA													4F	2	1	0→A	•	0	1	0	0
	CLRB													5F	2	1	0→B	•	0	1	0	0
	CLR				0F	6	2	6F	6+	2+	7F	7	3				0→M	•	0	1	0	0
CMP	CMPA	81	2	2	91	4	2	A1	4+	2+	B1	5	3				Confronto tra M e A	(4)	↕	↕	↕	↕
	CMPB	C1	2	2	D1	4	2	E1	4+	2+	F1	5	3				Confronto tra M e B	(4)	↕	↕	↕	↕
	CMPD	10	5	4	10	7	3	10	7+	3+	10	8	4				Confronto tra M:M + 1 e D	•	↕	↕	↕	↕
		83			93			A3			B3											
	CMPS	11	5	4	11	7	3	11	7+	3+	11	8	4				Confronto tra M:M + 1 e S	•	↕	↕	↕	↕
		8C			9C			AC			BC											
	CMPU	11	5	4	11	7	3	11	7+	3+	11	8	4				Confronto tra M:M + 1 e U	•	↕	↕	↕	↕
		83			93			A3			B3											
	CMPX	8C	4	3	9C	6	2	AC	6+	2+	BC	7	3				Confronto tra M:M + 1 e X	•	↕	↕	↕	↕
	CMPY	10	5	4	10	7	3	10	7+	3+	10	8	4				Confronto tra M:M + 1 e Y	•	↕	↕	↕	↕
		8C			9C			AC			BC											
COM	COMA													43	2	1	$\overline{A}$→A	•	↕	↕	0	1
	COMB													53	2	1	$\overline{B}$→B	•	↕	↕	0	1
	COM				03	6	2	63	6+	2+	73	7	3				$\overline{M}$→M	•	↕	↕	0	1
CWAI		3C	≥20	2													CC ∧ IMM→CC In attesa di un interrupt					(3)
DAA														19	2	1	Aggiusta A in decimale	•	↕	↕	0	↕
DEC	DECA													4A	2	1	A−1→A	•	↕	↕	↕	•
	DECB													5A	2	1	B−1→B	•	↕	↕	↕	•
	DEC				0A	6	2	6A	6+	2+	7A	7	3				M−1→M	•	↕	↕	↕	•
EOR	EORA	88	2	2	98	4	2	A8	4+	2+	B8	5	3				A ⊻ M→A	•	↕	↕	0	•
	EORB	C8	2	2	D8	4	2	E8	4+	2+	F8	5	3				B ⊻ M→B	•	↕	↕	0	•
EXG	R1, R2	1E	8	2													R1↔R2(2)	•	•	•	•	•
INC	INCA													4C	2	1	A + 1→A	•	↕	↕	↕	•
	INCB													5C	2	1	B + 1→B	•	↕	↕	↕	•
	INC				0C	6	2	6C	6+	2+	7C	7	3				M + 1→M	•	↕	↕	↕	•
JMP					0E	3	2	6E	3+	2+	7E	4	3				IE(5)→PC	•	•	•	•	•
JSR					9D	7	2	AD	7+	2+	BD	8	3				Salto a subroutine	•	•	•	•	•
LD	LDA	86	2	2	96	4	2	A6	4+	2+	B6	5	3				M→A	•	↕	↕	0	•
	LDB	C6	2	2	D6	4	2	E6	4+	2+	F6	5	3				M→B	•	↕	↕	0	•
	LDD	CC	3	3	DC	5	2	EC	5+	2+	FC	6	3				M:M + 1→D	•	↕	↕	0	•
	LDS	10	4	4	10	6	3	10	6+	3+	10	7	4				M:M + 1→S	•	↕	↕	0	•
		CE			DE			EE			FE											
	LDU	CE	3	3	DE	5	2	EE	5+	2+	FE	6	3				M:M + 1→U	•	↕	↕	0	•
	LDX	8E	3	3	9E	5	2	AE	5+	2+	BE	6	3				M:M + 1→X	•	↕	↕	0	•
	LDY	10	4	4	10	6	3	10	6+	3+	10	7	4				M:M + 1→Y	•	↕	↕	0	•
		8E			9E			AE			BE											
LEA	LEAS							32	4+	2+							IE(5)→S	•	•	•	•	•
	LEAU							33	4+	2+							IE(5)→U	•	•	•	•	•
	LEAX							30	4+	2+							IE(5)→X	•	•	↕	•	•
	LEAY							31	4+	2+							IE(5)→Y	•	•	↕	•	•

Legenda:

- OP Codice Operativo (in esadecimale)
- ~ Numero di cicli macchina
- # Numero di byte di programma
- + Segno aritmetico di somma (numero positivo)
- — Segno aritmetico di sottrazione (numero negativo)
- • Segno aritmetico di moltiplicazione
- $\overline{M}$ Complemento di M
- → Trasferimento in
- H Half-Carry (semi riporto), dal bit 3
- N Negativo (bit del segno)
- Z Zero (corrispondente a stato "reset")
- V Overflow, in complemento a 2
- C Carry (bit del riporto)
- ↕ Esaminato e posto ad 1 se la condizione è vera, altrimenti posto a 0
- • Inalterato dall'operazione
- CC Registro di Condizione del Codice
- : Concatenazione
- ∨ Operatore OR logico
- ∧ Operatore AND logico
- ⊻ Operatore OR Esclusivo

Note:

(1) Questa colonna indica il numero di cicli ed il numero di byte dell'istruzione di base: per ottenere il conteggio totale, occorre sommare i valori che si ottengono dalla sezione MODO D'INDIRIZZAMENTO INDICIZZATO

(2) R1 ed R2 possono essere una qualsiasi coppia di registri a 8 bit, oppure a 16 bit.
I registri a 8 bit sono: A, B, CC e DP
I registri a 16 bit sono: X, Y, U, S, D e PC

(3) I codici di condizione sono azzerati o posti a 1, a seconda del risultato dell'operazione

(4) Il valore del flag "half-carry" (semi-riporto) è indefinito

(5) IE sta per "indirizzo effettivo"

per concessione della Motorola Inc.

Da un'idea al successo

Diventare ricchi sfondati inventando un programma è il sogno di tutti!

IAN McKINNELL



Vendita al dettaglio

I punti di vendita sono forse l'anello più importante di tutta la catena commerciale del software: è qui che si vedono i frutti di una buona campagna pubblicitaria o l'effettiva qualità di un programma. Inoltre, sul successo di un programma, influisce moltissimo la disponibilità: riuscire in una distribuzione capillare dei prodotti è l'ambizione di tutte le software house

Ecco due esempi di come un gioco possa fruttare una fortuna. Nell'estate del 1984, la Gremlin Graphics ha lanciato un nuovo gioco chiamato *Wanted: Monty Mole*. Se il programma diventa un best-seller, il guadagno per il suo ideatore (il diciannovenne Peter Harrap) dovrebbe aggirarsi su una decina di milioni. *Wanted: Monty Mole* è la sua prima esperienza commerciale.

Peter Harrap è un esponente di quella schiera di "ragazzi prodigio", capaci di farsi una fortuna con qualche gioco ben azzeccato. Alcuni di questi ragazzi hanno appena quattordici anni.

Come molti programmatori di giochi, Harrap è un autodidatta. Il suo primo computer è stato uno ZX81 della Sinclair e, vista la iniziale scarsità di giochi in commercio, egli decise di scriverne alcuni per conto proprio. Per far ciò, gli è stato necessario imparare il codice macchina del microprocessore Z80, poiché il BASIC è troppo lento per i giochi in stile "arcade". Ben presto Harrap è passato allo Spectrum, concentrandosi su alcune migliorie da apportare al noto *Ant Attack* della Quicksilva (vedere a pag. 486). Harrap riuscì a disassemblare il programma (nonostante le protezioni) e a modificare parte del paesaggio visualizzato durante il gioco. La versione risultante venne rifiutata dalla Quicksilva, ma una piccola software house, la Gremlin Graphics, in quel momento alla ricerca di nuovi "talenti", rimase colpita da questa nuova versione. Tuttavia, poiché il programma infrangeva le leggi sui diritti d'autore, la software house rimandò il gioco al mittente, ma assieme ad un'offerta di lavoro!

Tutte le software house sono alla costante ricerca di programmi di gioco insoliti ed originali, possibili candidati per i primi posti delle classifiche. Fu la Gremlin Graphics a concepire le idee basilari di *Wanted: Monty Mole*, confidando che l'implicito richiamo all'attualità (lo sciopero dei minatori inglesi) avrebbe fatto presa sul pubblico. È stato Harrap a tradurre l'idea in pratica, scrivendo un programma per lo Spectrum.

Lo sviluppo ha richiesto quattro mesi di lavoro, durante i quali Harrap ha scritto una routine "generatrice di sprite" (per creare e visualizzare rapidamente le immagini desiderate), ha "costruito" la miniera e, infine, ha impostato le regole del gioco in base alle quali si muovono i vari personaggi.

Il lancio pubblicitario è avvenuto, anche per televisione, su tutto il territorio britannico ed il successo non si è fatto attendere a lungo. Inoltre la stampa, anche quella non specializzata, ha messo in evidenza che lo stesso padre di Harrap era uno dei minatori aderenti al colossale sciopero e che la Gremlin Graphics avrebbe devoluto una piccola percentuale dei guadagni ad un fondo di carità per le famiglie dei minatori.

La Gremlin Graphics corrisponde ai propri programmatori, Harrap incluso, una "royalty" per ogni cassetta venduta. La percentuale della royalty varia da una software house all'altra e, in genere, viene fissata in base a "scaglioni" di vendita: il cinque per cento per le prime 3.000 copie vendute, poi il sette per cento per le successive 10.000. Tuttavia, tale percentuale viene talvolta calcolata sul prezzo netto (quello corrisposto dal distributore alla software house) oppure sull'effettivo prezzo al pubblico.

Spesso, le ditte sono anche disposte a versare un anticipo, che viene in seguito detratto dall'ammontare della royalty. L'accordo può

FAR COLPO

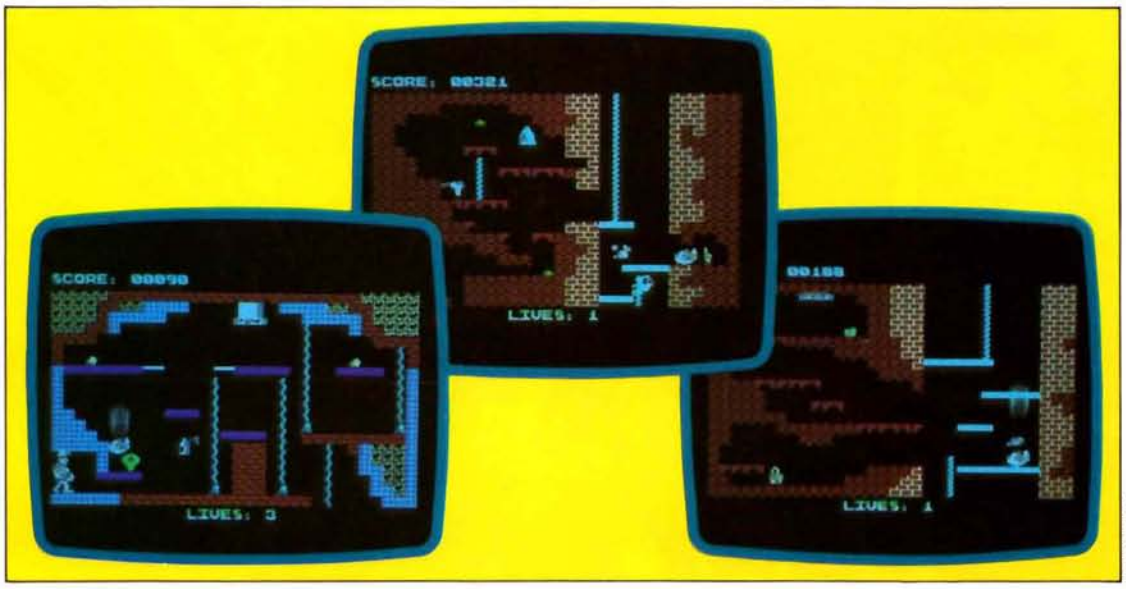
Migliaia di programmatori "casalinghi" si dedicano alla stesura di giochi più o meno insoliti, ma solo pochi arrivano a qualche successo di rilievo. L'originalità è, infatti, una delle qualità più importanti e nel gioco immaginario qui a fianco, questa caratteristica è stata volutamente esagerata...

PRANZO
5 ROYALTIES %
ACCONTO 2.000.000 £
QDOS
10 PUNTI
VALE 100.000
50
50
50

LOOT
Il gioco dei giochi
• Ogni giocatore è invitato a barare e defraudare gli altri, infrangendo tutte le possibili regole.
• Lo scopo è quello di far uscire gli altri dal gioco, accumulando tutte le carte QDOS e tutto il denaro. Ciò si ottiene consegnando le proprie IDEE ad un EDITORE, concludendo scanzionsi affari durante i PRANZI di lavoro. Il primo contratto prevede soltanto una royalty, ma nei successivi si possono avere anche contrattazioni ed immediati acconti.
• Una volta concluso un accordo, la cassetta del proprio programma viene venduta dal BUG MARKET a tutti i giocatori che vi si soffermano.
• Ad ogni passaggio da RETURN, viene conseguita una nuova IDEA, assieme alle royalty guadagnate (numero di QDOS per la percentuale della royalty).
• Quando si è costretti ad andare alla TASTIERA (niente RA, non si può passare da RETURN (niente guadagni!). Per lasciare la TASTIERA occorre fare un sei, oppure sottrarre un' IDEA ad un altro giocatore.
• Se si capita su BERGSTROM, l'ultimo gioco viene redatto nella versione per un nuovo microcro, raddoppiando così le entrate!
RECENSIONI
"Fondo PERDI
"Magnifico programma. Commenta la stampa
VINCI 45 QDOS
IMPREVISTI
READ ERROR formattare il disco 5 QDOS
BAD DATA Copie pirata del tuo programma invadono il mercato PERDI 25 QDOS
USCITA RIVISTA
PRENDI 1 RECENSIONI
EDITORE HYSTICK
GOTO PRANZO
BUG
SALTA 1 GIRO
BUG MARKET
PROGETTO DEL PACKAGE
PRENDI Q DOS SALTA 1 GIRO
EDITORE AR-K-DO
DISTRIBUZIONE
PRENDI 1 Q DOS
PRANZO GRATIS
BYTE A VOLONTÀ
PUBBLICITÀ
EDITORE PCK-OLO
VIA VENETO
PRENDI 1 Q DOS
BUG MARKET
EDITORE BASS-OTT
BUON SUCCESSO
TIRA ANCORA
SENZA PASSARE DA RETURN
GOTO TASTIERA
BERGSTROM
RIVISTA COMP65
PRENDI RECENSIONE
EDITORE PERDITEMPO
GOTO PRANZO
BUG MARKET
BUG
SALTA 1 GIRO
EDITORE PIGLIA&C.
GOTO PRANZO
DISTRIBUZIONE
PRENDI 1 Q DOS
TASTIERA
PRENDI LE ROYALTY PIÙ UN'IDEA
VIA MAZZINI
PRENDI 1 IDEA GRATIS
EDITORE STAMPAPICCOLA
GOTO PRANZO
GRAN FIASCO!
SALTA 1 GIRO
BUG MARKET
?
EDITORE PRENDITUTTO
GOTO PRANZO
CASSETTA "FUSA"

STEVE CROSS

anche prevedere l'acquisto di tutti i diritti, compensato da una cifra globale, senza alcuna royalty. Agli aspiranti programmatori conviene meditare a fondo su tutte queste forme di pagamento, per non trovarsi a sottoscrivere contratti "capestro".
Inoltre, c'è da considerare l'aspetto del copyright: quando l'accordo prevede il pagamento di royalty, il copyright rimane al programmatore. Ma, in tal caso, eventuali dispute giudiziarie lo coinvolgono direttamente. Quando invece l'acquisto del programma avviene con un pagamento "una tantum", il programmatore cede il copyright al compratore, che da tale momento diviene il rappresentante legale in ogni eventuale disputa.
Nel caso di Harrap, egli percepisce anche una sorta di "premio" di produzione, ma alla condizione che non presti la propria opera ad altre software house per almeno un anno. Certi contratti prevedono anche una clausola di "prelazione" (diritto di essere consultati per ogni nuova "idea", prima che essa venga offerta ad altri).
Un programma è considerato "di successo" se ne vengono vendute almeno 15.000 copie durante la sua "vita" (un periodo che in genere va dai quattro ai cinque mesi). Nel caso di *Wanted: Monty Mole* la Gremlin Graphics ha prodotto 10.000 copie per ogni versione (Spectrum e Commodore 64), con l'opzione di successive duplicazioni. Le previsioni sono per circa 20.000 vendite per ciascuna versione.
Veniamo al secondo caso: Christopher Kerry, un ex-studente diciassettenne che lavora come libero professionista per la House of Thor. Kerry ha scritto il gioco *Jack and the Beanstalk* a casa propria, su di uno Spectrum. In soli due mesi e senza alcuna forma di pubblicità, pare che ne siano state vendute ben diverse decine di migliaia. La House of Thor, molto gelosa di Kerry, temendo che possa venirgli "soffiato" da qualche concorrente, impiega altri quindici programmatori, tutti giovanissimi.
Come tutte le software house, anche la Thor è ben lieta di ricevere programmi "di prova" da parte di semplici appassionati: se il prodotto interessa, viene immediatamente preso contatto col programmatore (la rapidità conta moltissimo: guai a lasciarsi sfuggire un potenziale successo!).
La House of Thor è unicamente interessata a giochi del genere "arcade", quindi scritti in codice macchina e, inoltre, scarta tutte le proposte di programmi troppo simili ad altri già in commercio; non basta, dunque, modificare il paesaggio e qualche nome dei personaggi (invece di vendite si potrebbero ottenere soltanto beghe legali).
Un'altra software house sulla cresta dell'onda, la Salamander, impiega oltre 40 programmatori che, su suggerimento di Chris Holland direttore della Salamander, scrivono software per gli apparecchi nuovi, non appena essi sono introdotti sul mercato: a scrivere per lo Spectrum, ormai, sono già in tanti.
Alcuni programmatori, anziché affidarsi a terzi, preferiscono formare una ditta in proprio. La Llamasoft e la Bug-Byte sono un esempio di tale scelta. Sfortunatamente però, occorre un capitale notevole per questo tipo d'avventura. I quattro soci della Gremlin Graphics, quando questa lanciò il suo primo gioco *Potty Pigeon*, dovettero contribuire personalmente con quote piuttosto consistenti. Le spese maggiori inizialmente, sono quelle per la duplicazione, per la confezione e per la pubblicità; inoltre tutto ciò non garantisce il successo: la nota software house Imagine è fallita a metà del 1984, pur avendo prodotto innumerevoli giochi letteralmente andati "a ruba".
Creare una propria software house è un poco il sogno di tutti i programmatori, ma i rischi non sono pochi: occorre un buon consulente commerciale ed una discreta disponibilità di fondi. Tutto sommato, può esser molto più conveniente inviare qualche "saggio" delle proprie abilità a ditte già ben piazzate e con una solida rete distributiva.



LIZ HEANEY

Un gioco d'oro!
Il programmatore che ha scritto "Wanted: Monty Mole" è un tipico ragazzo prodigio: giovanissimo, autodidatta, brillante e... ben pagato! La trama del gioco, di per sé, non è poi molto originale, anche se contiene qualche esplicito riferimento ad una vertenza sindacale molto attuale in Gran Bretagna: lo sciopero dei minatori

Le interfacce musicali

Oltre alla parte hardware, occorrono adeguati programmi per far funzionare le interfacce

Il Micon (sigla derivata da MIDI controller) è prodotto dalla XRI Systems e rappresenta un package abbastanza tipico nel settore. Progettato per funzionare con lo Spectrum della Sinclair, consiste in un'interfaccia MIDI con annesso software su cassetta. Le funzioni di sequenziatore provvedono fino ad otto diversi canali, ciascuno con la capacità di memorizzare fino a 2.951 diversi "eventi" (note, pause, ecc.). Il brano musicale viene immesso tramite la tastiera del sintetizzatore collegato, mentre il tasto SPACE dello Spectrum serve per definire il numero di eventi da suonare. Durante l'immissione, la musica viene riprodotta sullo schermo del computer, in una versione approssimativa della consueta notazione su pentagramma. Comunque, sono previste tutte le notazioni (incluse la durata, gli accidenti, lo "staccato", ecc.) per la chiave di violino e quella di basso. C'è una certa carenza nelle indicazioni usate per le pause (le soste tra l'emissione di una nota e l'altra), che costituiscono un elemento fondamentale nella riproduzione musicale. Inoltre, le gambette delle note sono sempre rivolte verso l'alto (nell'uso normale, la direzione dipende dalla posizione sul rigo musicale). Il risultato, dunque, non è estremamente preciso, ma serve benissimo come guida approssimativa e se ne può ottenere una copia su carta, tramite la stampante ZX.

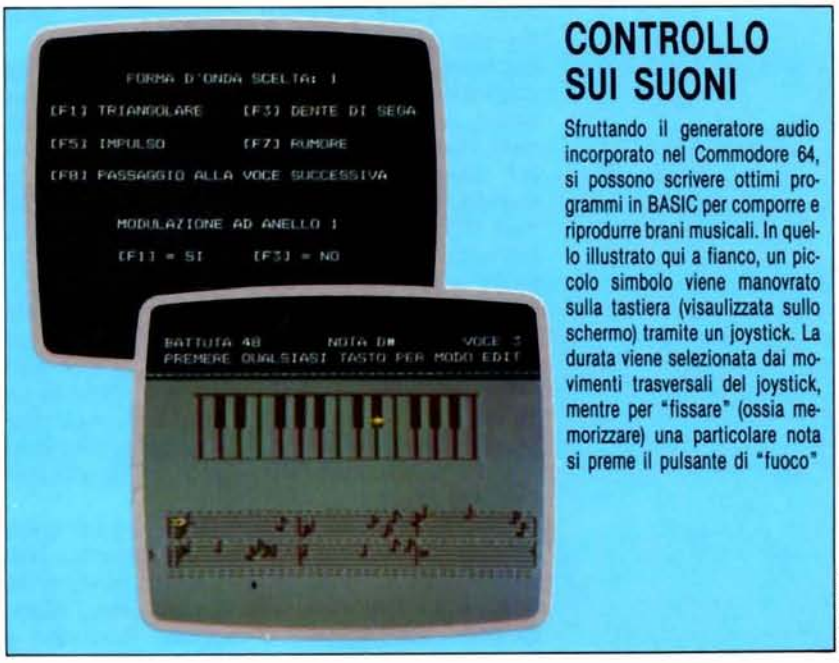
Quanto sia preziosa questa visualizzazione, anche se imprecisa, lo si apprezza quando si tratta di apportare modifiche al brano musicale: lo spartito è suddiviso in battute, separate da barre verticali. Richiamando una battuta, si possono modificare a piacere le note in essa contenute o, addirittura, eliminare l'intera battuta o anche riprodurla più volte in modo automatico. È possibile memorizzare su nastro fino a 10 sequenze (circa 24.000 note) assieme alle caratteristiche sonore ad esse assegnate. Il ritmo della riproduzione può esser definito con una precisione di quattro millisecondi, oppure controllato, tramite l'ingresso SYNC IN dell'interfaccia, da una batteria elettronica.

Per di più il sequenziatore del Micon può agire in "tempo reale", depositando tutte le informazioni nella memoria dello Spectrum durante l'esecuzione sul sintetizzatore collegato tramite MIDI. In tal caso sullo schermo non viene riprodotta la notazione musicale, ma nonostante ciò il vantaggio è duplice: in primo luogo l'esecutore può "registrare" un brano senza doversi preoccupare di livelli, risonanze o altri controlli; secondariamente, la registrazione di un particolare passaggio viene fatta senza dover percorrere un tedioso procedimento "passo per passo", a velocità ridotta.

Ciò non esenta dall'uso di un metronomo, altrimenti il ritmo di successive riproduzioni risente di tutti gli errori immessi durante la registrazione. Come accade per qualsiasi sistema MIDI, la qualità del risultato finale ottenuto mediante il Micon dipende da quanto è sofisticato il sintetizzatore collegato a questa interfaccia, il cui costo è veramente ragionevole.

Il package offerto dalla Jellinghaus Music Systems è leggermente più economico, tuttavia prevede il collegamento con più tipi di computer: oltre allo Spectrum, anche l'Apple II e il Commodore 64. Il suo funzionamento è sostanzialmente identico a quello del Micon, ma l'immissione delle sequenze avviene soltanto in "tempo reale" e la visualizzazione su schermo non segue la notazione classica.

L'organizzazione della musica in battute avviene specificando dei segni "temporali" (ossia il numero di crome o semiminime per gruppo). Sono previsti quattro segni "temporali": con tre, quattro o cinque semiminime per battuta (tempo in 3/4, 4/4 o 5/4) oppure sette crome (tempo 7/8). Come nel Micon, il ritmo dell'esecuzione deve essere coordinato ricorrendo ad un metronomo o ad una batteria elettronica collegata all'interfaccia: se



CONTROLLO SUI SUONI

Sfruttando il generatore audio incorporato nel Commodore 64, si possono scrivere ottimi programmi in BASIC per comporre e riprodurre brani musicali. In quello illustrato qui a fianco, un piccolo simbolo viene manovrato sulla tastiera (visaulizzata sullo schermo) tramite un joystick. La durata viene selezionata dai movimenti trasversali del joystick, mentre per "fissare" (ossia memorizzare) una particolare nota si preme il pulsante di "fuoco"

LIZ HEANEY



1. Controllo col computer
Sul Commodore 64 è possibile, adoperando software adeguato, raggiungere un notevole livello di controllo. Grazie all'unità a nastri o dischetti si possono memorizzare i brani musicali composti

l'immissione viene fatta sbagliando ritmo, il risultato sarà una sequenza di note "fuori battuta". Tutto procede abbastanza bene, finché i raggruppamenti di note escono dagli schemi consentiti (3/4, 4/4, 5/4 o 7/8). Non appena si verifica un tale evento, il sequenziatore va fuori passo. Tale limitazione può essere anche piuttosto grave: non è possibile impostare un tempo di 12/8, ad esempio (anche se la ripartizione è formalmente corretta), mentre questo è un tempo comunissimo nei pezzi di jazz, rock, funk e reggae.
Supponendo che una certa sequenza venga immessa correttamente, i dati vengono mostrati, incolonnati, sullo schermo. Sono indicati: l'ottava cui si riferisce la nota (o l'evento), l'altezza della nota in tale ottava, la durata in termini musicali (crome o semiminime), il "tempo di porta" (un parametro che influisce sull'effetto di una frase musicale) e i valori abbinati alla tastiera (nell'arco 0-9).
Questo genere di visualizzazione, in effetti, fornisce molte più informazioni di quello più tradizionale, offerto dal Micon, in cui non esiste alcun esplicito riferimento numerico né ai tempi di porta, né ai livelli di velocità. Questi due parametri musicali sono nati solo recentemente, con l'avvento delle nuove tecniche elettroniche.
Tuttavia, anche musicisti privi di un tirocinio formale nella notazione classica riescono ben presto a trarre le necessarie informazioni da entrambi i metodi di visualizzazione: l'altezza delle note oppure i rapporti armonici e melodici tra esse, gli intervalli, e via dicendo. Ovviamente, l'impostazione dei dati su colonne è ottima per il controllo dei parametri, ma è ben difficile canticchiare un brano in base a simili informazioni! L'ideale sarebbe un sistema che preveda entrambe le forme di visualizzazione.

Sintetizzatori concettuali

Il problema principale, per un possessore di home computer dotato di una certa inclinazione alla musica, non è quello di trovare un'interfaccia MIDI adatta al proprio computer, semmai è quello di acquistare un sintetizzatore sufficientemente sofisticato, il cui costo è però quasi sempre inaccessibile ad un semplice appassionato. Volendo costruire appena il 50% degli effetti contenuti in un qualsiasi brano di musica moderna, occorrono apparecchi dal costo superiore a quello di un personal computer. Al di sotto di tale livello, la maggior parte dei sintetizzatori ha notevoli limiti.
Una possibilità da non sottovalutare è l'impiego di un "sintetizzatore concettuale". Vengono così chiamati alcuni speciali programmi per computer abbinati ad apposite tastiere, che sfruttano le capacità musicali del generatore audio incorporato nei computer. Questi dispositivi si possono considerare i progenitori delle attuali interfacce MIDI.
Un esempio di tali apparecchi è il PDSG

2. Il potente Six-Trak
In questo sofisticato sintetizzatore è incorporato anche un registratore multitraccia. Il nome Six-Trak deriva proprio dalla possibilità di registrare su ben sei piste magnetiche. Diversamente da molti altri sintetizzatori, questo consente di usare una voce diversa per ogni traccia registrata, permettendo così la creazione di sonorità anche molto ricche e complesse

3. Il Modello 64 della Sequential Circuits
Innestando questa unità nel connettore per espansioni del Commodore 64 è possibile combinare un sintetizzatore interfacciato con un MIDI e la memoria del computer, quella offerta dai nastri o dai dischetti. Il Modello 64 memorizza il ritmo, l'altezza e la modulazione delle note, fino ad un massimo di 4.000. Nella riproduzione, l'interfaccia può limitarsi ad inoltrare i suoni così come li riceve dalla tastiera, oppure modificarne alcune caratteristiche



IAN McKINNELL

(Programmable Digital Sound Generator) prodotto dalla Clef Products. Questo apparecchio, collegabile al Micro BBC Modello B, include una tastiera musicale con 61 note, tasti "sensibili" alla velocità dell'esecuzione e due pedali. Per sensibilità alla velocità s'intende la capacità di modificare il suono emesso, in base alla pressione esercitata su ciascun tasto. L'unità permette di registrare e riprodurre una notevole varietà di suoni: ciascuna delle 32 unità per la generazione di suoni (non semplici oscillatori!) consente di specificare ben 11 attributi diversi. All'occorrenza, tutte le 32 unità possono essere impiegate per produrre una sola nota. Alcuni dei suoni ottenibili reggono il confronto anche con quelli prodotti da sintetizzatori professionali. Abbinando ciascuna unità ad una nota, si possono ottenere interessanti sequenze programmate e le combinazioni sono limitate soltanto dalla fantasia del suonatore. Sullo schermo vengono riportate le caratteristiche relative alle forme d'onda, fatto particolarmente utile anche nel campo didattico. Il solo software del PDSG, però, costa circa il doppio del Micon ed anche la tastiera si aggira su cifre cospicue.

Il maggior inconveniente del sistema PDSG è dovuto alla conversione digitale-analogica dei suoni. L'orecchio umano è in grado di percepire suoni nella gamma che va dai 20 Hz a circa 20 KHz. I suoni, in natura, superano questa larghezza di banda. Il PDSG, tuttavia, ha un limite massimo di 12 KHz. Ciò può andar bene per applicazioni amatoriali, ma non soddisfa certo i professionisti. La Clef Products ha annunciato versioni del PDSG adatte anche ad altri computer.

L'interfaccia MIDI rappresenta una grande innovazione nel campo musicale professionale, ma per chi abbia intenzione di soddisfare esigenze meno ambiziose, il PDSG è un'alternativa da prendere in considerazione.

ATTRAZIONE MUSICALE

Lo sviluppo dell'interfaccia MIDI apre ai possessori di home e personal computer orizzonti illimitati nella creatività musicale. Al tempo stesso, però, si corre il rischio di acquistare una serie di apparecchi relativamente costosi, per perdersi poi nella complessità del sistema.

Un'alternativa accettabile può essere quella di cominciare con uno strumento economico ed elementare, purché dotato di sufficienti caratteristiche da renderne stimolante l'impiego. Un economico sistema di questo genere può essere il package Ultisynth 64, prodotto dalla Quicksilva per il Commodore 64.

Con tale programma, a ciascun tasto è abbinata una precisa funzione modificatrice del suono prodotto. Sono disponibili le quattro forme d'onda fondamentali (sinusoidale, triangolare, quadra e a dente di sega), assieme a controlli per definire l'attacco, la durata ed il decadimento dei singoli suoni (inviluppo). Inoltre le varie frequenze possono essere filtrate per caratterizzare maggiormente le note emesse. Anche la modulazione "ad anello" (la simultanea somma e sottrazione tra due frequenze date) è un arricchimento molto interessante, col quale si possono ottenere suoni molto simili a quelli di campane. È possibile costruire sequenze composte da un massimo di 2.048 note ed è disponibile anche una base ritmica prefissata.

Gli effetti ottenibili con l'Ultisynth sono paragonabili a quelli prodotti dal VCS 3, un sintetizzatore molto diffuso negli anni '60 e '70, oggi quasi un pezzo da museo.

Un altro sistema, anch'esso basato su software fornito su cassetta, è il Multisound della Romik. Il suo funzionamento e le sue caratteristiche sono paragonabili a quanto ottenibile con l'Ultisynth, ma il controllo avviene tramite la tastiera del computer, anziché con un joystick.

La scelta... di una scelta

Spesso nei programmi esistono più funzioni: ci sono alternative al solito menu?

In un programma, un menu può essere una semplice lista di elementi numerati oppure un'elaborata serie di icone, ma il principio rimane identico in entrambi i casi: quando un procedimento giunge ad una diramazione multipla, l'operatore deve in qualche modo poter indicare quale debba essere il passo successivo. I programmi pilotati da menu tendono ad avere una struttura ramificata o "ad albero": partendo da un menu generale, si segue un percorso fino alla "foglia" desiderata.

Il vantaggio di tale metodo consiste nel fatto che l'operatore può ignorare del tutto la struttura del programma, poiché gli stessi menu servono da buoni segnalatori per il percorso da seguire. Tuttavia, gli utenti più esperti trovano noioso dover percorrere per intero tutto un percorso composto da più menu senza poter accedere immediatamente alla funzione voluta.

Anche i meno esperti si trovano talvolta in difficoltà, specialmente se, dopo aver selezionato un'opzione sbagliata, tentano di tornare indietro nel percorso.

Un esempio di struttura profondamente ramificata è data dai sistemi tipo Prestel. Comunque, anche se la struttura ramificata è la più usata, non è l'unica: si possono anche collegare i vari menu in una forma reticolare, ma il loro impiego diventa più complicato per i meno esperti.

Il progetto di una struttura pilotata da menu è molto complesso, anche se la sua conversione in programma eseguibile è, al contrario, relativamente facile. Il principale problema è che tutte le parti del programma applicativo devono essere ben specificate, prima che si possa passare efficacemente a tale progetto. Infatti, anche l'aggiunta di una sola nuova opzione, ad uno stadio già avanzato, può implicare tutta una serie di modifiche ai menu previsti in origine. Nel redigere un programma complesso, composto da più sottoprogrammi, conviene riunire tutta la parte relativa ai menu in un'unica routine, che richiami direttamente i vari moduli del programma.

Ciò serve a tenere sotto controllo, in modo ordinato, tutto lo svolgimento del programma e, inoltre, a permettere lo sviluppo indipendente di ciascuna delle sezioni.

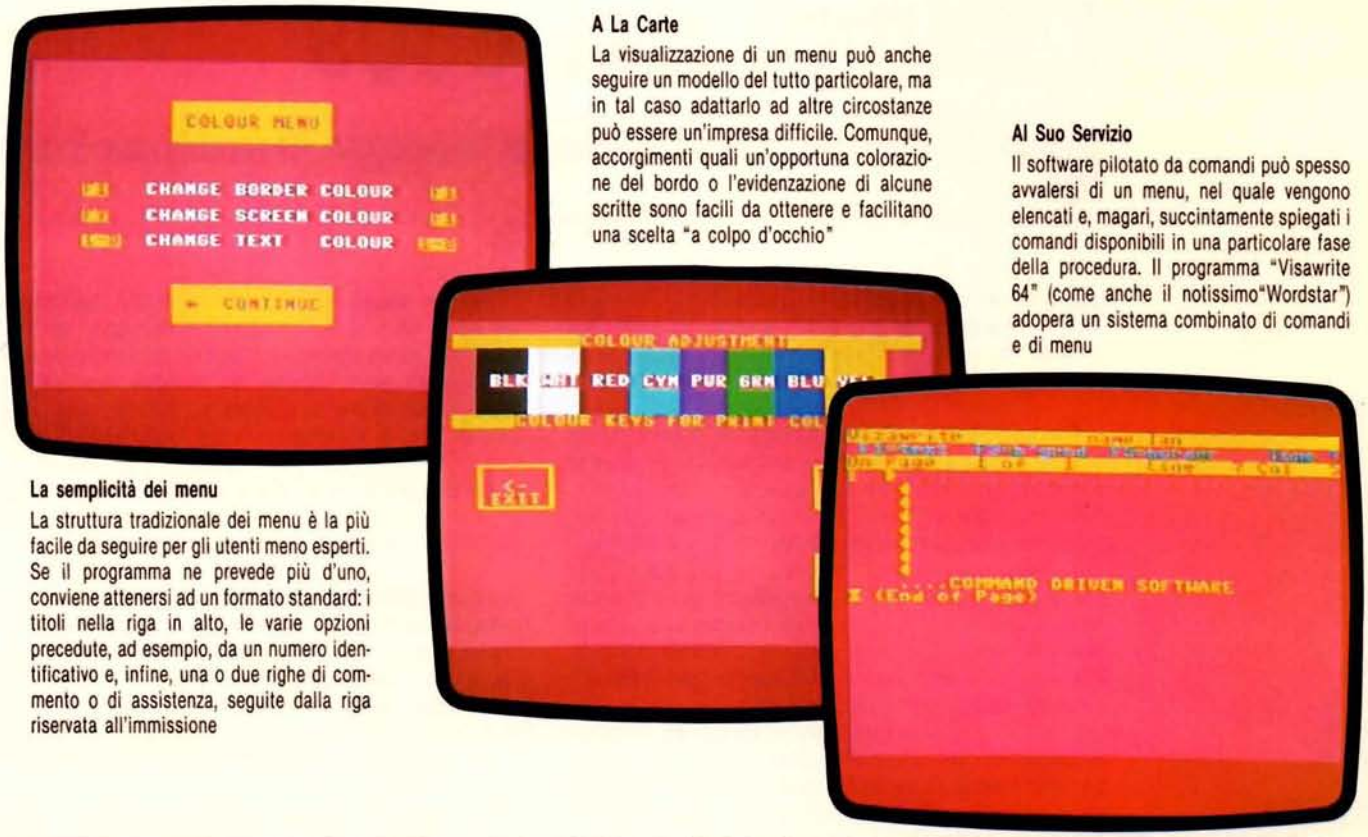
In un sistema così concepito, il flusso logico segue un percorso ben preciso: per ogni menu incontrato, la routine di controllo fornisce le scritte necessarie ad un'apposita routine, che le visualizza in forma ordinata sullo schermo, sfruttando sempre uno stesso modulo di base. Quest'ultimo deve contenere una o due righe (generalmente in alto) riservate al titolo del menu e una o due righe in basso per eventuali spiegazioni o chiarimenti, con spazio sufficiente per l'immissione da parte dell'utente, oltre ovviamente alle opzioni stesse. Queste vengono di solito disposte una sotto l'altra, precedute da un numero o una lettera che le identifica e ne permette la selezione.

La routine del menu ne richiama poi una di immissione, magari passando ad essa alcuni parametri (ad esempio l'arco di valori ammesso per le risposte). In base al valore immesso (tipicamente un singolo carattere) la routine del menu provvede a richiamare un ulteriore menu oppure direttamente la sezione di programma interessata. Al termine di questa, il controllo può essere restituito al menu di provenienza, oppure a quello generale di partenza.

I menu utilizzano una notevole quantità di scritte, alcune delle quali sono ricorrenti (ad esempio le informazioni su come selezionare le opzioni, su come tornare al menu principale, ecc.). Per risparmiare memoria ed evitare ripetizioni, conviene assegnare le scritte ricorrenti a delle variabili stringa o addirittura depositarle in piccoli file ad accesso casuale su disco. Per usarle, basterà utilizzare un numero come indice.

Un'alternativa ai menu è la struttura pilotata tramite comandi: l'utente dispone di un vocabolario più o meno vasto di speciali parole o comandi, che vanno digitati per esteso. Ciascuno di questi comandi richiama direttamente la sezione di programma interessata. In tal caso, un'apposita routine di immissione si occupa di distinguere, fra i dati digitati, quali siano i comandi. In alternativa, può essere previsto l'uso di particolari codici (spesso si tratta del tasto Control).

In strutture di questo genere, per interpretare i comandi occorre, una routine, che deve svolgere quattro compiti ben precisi. Il primo è di attendere semplicemente un'immissione, il secondo riguarda la separazione di quanto immesso in parti elementari o singole parole, il terzo concerne l'individuazione e la preparazione di tutti i parametri occorrenti alla routine da eseguire, il quarto compito, infine, è quello di eseguire la chiamata alla routine; una volta terminata questa, il controllo torna al primo compito: aspettare!



A La Carte

La visualizzazione di un menu può anche seguire un modello del tutto particolare, ma in tal caso adattarlo ad altre circostanze può essere un'impresa difficile. Comunque, accorgimenti quali un'opportuna colorazione del bordo o l'evidenzazione di alcune scritte sono facili da ottenere e facilitano una scelta "a colpo d'occhio"

Al Suo Servizio

Il software pilotato da comandi può spesso avvalersi di un menu, nel quale vengono elencati e, magari, succintamente spiegati i comandi disponibili in una particolare fase della procedura. Il programma "Visawrite 64" (come anche il notissimo"Wordstar") adopera un sistema combinato di comandi e di menu

La semplicità dei menu

La struttura tradizionale dei menu è la più facile da seguire per gli utenti meno esperti. Se il programma ne prevede più d'uno, conviene attenersi ad un formato standard: i titoli nella riga in alto, le varie opzioni precedute, ad esempio, da un numero identificativo e, infine, una o due righe di commento o di assistenza, seguite dalla riga riservata all'immissione

IAN McKINNELL

Il formato dei comandi può raggiungere livelli di complessità notevoli e, addirittura, in certi casi si tratta di un vero e proprio linguaggio, anche se semplificato. Il nucleo del sistema operativo Unix, ad esempio, prevede comandi secondo il formato «comando + eventuale lista di parametri». Nell'Unix, per ottenere un elenco dei file, si può usare il comando L, ma usando il formato L-1 (dove -1 è un parametro) viene visualizzato un elenco con molte più indicazioni.

In questo caso, la routine che separa i vari elementi di un comando (chiamata in gergo "parser", ossia "analizzatore grammaticale") è abbastanza semplice: la prima parola viene sempre considerata un comando ed i parametri sono individuati da un prefisso (il segno meno).

I parametri specificati non vengono usati dal parser, ma in genere sono passati alla routine interessata. Nel progettare una struttura pilotata da comandi, conviene sempre fissare un formato standard ed attenervisi in ogni circostanza: ciò consente di semplificare il passaggio dei parametri, magari sotto forma di un'unica stringa.

Non si può affermare in assoluto se sia più facile il progetto di un sistema a menu o quello di una struttura pilotata da comandi. Si può dire, però, che i programmatori esperti preferiscono il secondo tipo, mentre gli utenti meno pratici preferiscono i menu. La maggior parte dei sistemi operativi funziona tramite comandi (suscitando spesso forti perplessità nell'utente, specialmente se non esiste una routine di "assistenza").

UNA SCELTA GUIDATA

Come visualizzare sullo schermo i percorsi consentiti all'interno di una procedura dipende molto dai limiti concessi dal computer, ma anche dalla fantasia e dalla bravura del programmatore. Le scritte o lo sfondo possono essere evidenziati con l'impiego moderato di colori. Ecco alcuni esempi, ottenuti su di un Commodore 64

Ciò significa che anche gli operatori che hanno una maggiore familiarità col sistema sono frequentemente costretti a consultare i manuali.

Comunque, esistono applicazioni di gran successo in cui vengono applicati sistemi "ibridi": il programma per l'elaborazione di testi Wordstar, ad esempio, è uno di questi. Pur essendo essenzialmente un programma pilotato da comandi, l'utente ha l'impressione di lavorare con un sistema di menu. I comandi sono quasi sempre semplici codici di controllo, talvolta seguiti da parametri. Sui menu i sintetici comandi vengono abbinati ad una scritta, cosicché l'utente ne impara gradualmente il significato, almeno di quelli più ricorrenti.

Inoltre si può scegliere il livello di assistenza, ossia la quantità di informazioni riportate sui menu: ciò permette all'operatore esperto di utilizzare lo schermo per intero. Il Wordstar adopera un sistema a soli due livelli di "profondità": realizzare una struttura con un numero maggiore di livelli è un compito molto difficile.

I campi del 6809

Esaminiamo alcune caratteristiche del linguaggio Assembly per il chip 6809

Ogni linea del linguaggio Assembly consiste di tre elementi o "campi", anche se talvolta alcuni di essi non vengono adoperati. Essi sono:

• **Campo delle etichette,** che si trova nella colonna più a sinistra del listato. Le etichette sono semplicemente dei nomi scelti dal programmatore ai quali l'Assembler associa un indirizzo di memoria. Un'etichetta è composta da una parola, con un massimo di sei caratteri, il primo dei quali deve essere alfabetico. Non si possono però adoperare i nomi dei registri, né i nomi delle istruzioni Assembly e neppure nomi di etichette già definite altrove nel programma. Queste sono le normali convenzioni impiegate da quasi tutti gli Assembler, ma possono esserci eccezioni. Se in una particolare linea non occorre indicare un'etichetta, allora è necessario digitare almeno uno spazio, indicando così all'Assembler che il campo delle etichette è vuoto. Analogamente, tale campo è delimitato da uno spazio: LABLDA, ad esempio, è un'etichetta consentita, ma LAB LDA viene interpretato come etichetta LAB, seguita dall'istruzione Assembly LDA.
Il programma Assembler adopera un "contatore di locazione", equivalente al contatore di programma del microprocessore, dove viene conservato l'indirizzo di memoria in cui va depositato il successivo codice macchina (istruzione o dato). Quando l'Assembler incontra un'etichetta ne verifica la presenza in quella che viene comunemente chiamata "symbol table" (tabella dei simboli). Se non trova l'etichetta, allora il nome viene aggiunto nella tabella. Se la posizione dell'etichetta nel listato è nel campo delle etichette, assieme al nome viene anche conservato l'indirizzo che in quel momento si trova nel contatore di locazione (l'etichetta risulta "definita"). Da questo momento in poi, ogni volta che incontra l'etichetta, l'Assembler può estrarre l'indirizzo per depositarlo nel codice macchina.

• **Campo delle Istruzioni o degli Op-code.** Questo è situato all'immediata destra del campo delle etichette. In questa colonna vengono scritte le istruzioni Assembly, generalmente formate da un codice mnemonico di tre lettere seguito, se necessario, dal nome di un registro. ADDA , per esempio, è composto dal codice mnemonico ADD, seguito dal nome del registro A. L'op-code rappresenta l'istruzione da far eseguire al processore. Come il campo delle etichette, anche questo è delimitato da uno spazio.

• **Campo degli Operandi o degli Indirizzi,** che contiene le informazioni relative ai dati sui quali deve operare l'istruzione specificata nel campo precedente. In genere, queste informazioni sono indirizzi o, più spesso, etichette che si riferiscono ad indirizzi.

Si consideri la seguente linea di Assembly:

```
LABEL1      ADDA      NUM1
```

il cui significato è «somma il dato contenuto nella locazione rappresentata dall'etichetta NUM1 a quello contenuto nell'accumulatore A». L'indirizzo dell'istruzione viene abbinato all'etichetta LABEL1, cosicché, volendo eseguire un salto o una diramazione a questo indirizzo, possiamo tranquillamente adoperare l'etichetta LABEL1.
Naturalmente, altrove nel programma, è definita l'etichetta NUM1. Adesso consideriamo un altro esempio:

```
            CLRA
```

che significa «azzera l'accumulatore A». Questa istruzione non richiede alcun operando e inoltre, si noti che non è stata indicata alcuna etichetta per l'indirizzo di tale istruzione. L'uso delle etichette è opzionale e, in realtà, serve soltanto se occorre far riferimento ad una particolare locazione del programma oppure per evidenziarne una (come se si trattasse di una REM). Questo secondo uso, tuttavia, non è molto pratico e può confondere le idee. Per inserire commenti conviene affiancarli sulla colonna più a destra del listato: la maggior parte degli Assembler considera commento tutto ciò che segue un carattere di punto e virgola oppure un asterisco.
Nel campo degli operandi si possono usare valori numerici costanti o stringhe. Generalmente, i numeri non espressi in forma decimale vanno preceduti da un simbolo che ne stabilisce la base: il prefisso $ o il suffisso H indicano un numero esadecimale (come in $AF08 oppure AF08H) il prefisso @ o il suffisso Q indicano un numero ottale (base otto, come in @6712 oppure 6712Q). Per i numeri binari si usa il prefisso % oppure il suffisso B (%11010011 oppure 11010011B). I codici ASCII dei caratteri si specificano racchiudendo la lettera cui si riferiscono tra apici o virgolette: 'A' corrisponde al codice 65 (ossia $41).
Un valore particolarmente utile è quello del contenuto del contatore di locazione, che può essere indicato con un asterisco e, spesso,

addirittura incluso in una semplice espressione aritmetica. Per esempio:

```
LDA     *+5
```

significa "carica nell'accumulatore il contenuto della locazione il cui indirizzo è quello corrente più 5 byte".

Direttive Assembly

L'Assembler accetta un certo numero di direttive (o "pseudo-op") che vengono adoperate come comuni istruzioni. Ne abbiamo già impiegate due (vedere a pag. 978):

```
CR      FCB     13
```

La direttiva FCB riserva un byte alla locazione corrente del programma, depositandovi il valore specificato nel campo degli operandi. Qui nella locazione indicata con CR viene collocato il valore 13.

```
MEMTOP  FDB     $7FFF
```

La direttiva FDB funziona come la precedente, ma riserva due locazioni (16 bit).
Si possono riservare intere zone di memoria, ma senza assegnare loro un valore specifico, come nel seguente caso:

```
TABEL1  RMB     7
TABEL2  FCB     $F6
```

Qui vengono riservati 7 byte, con inizio a TABEL1. In pratica, ciò significa che se TABEL1 corrisponde all'indirizzo $C104, allora TABEL2 rappresenta quello che è sette byte più in alto, ossia $C10B.
Esiste anche una direttiva per depositare i caratteri di una stringa in locazioni consecutive:

```
MESSG   FCC     'ERRORE'
```

A partire dall'indirizzo contraddistinto da MESSG, i byte contengono i codici dei caratteri E,R,R,O,R,E. Ciò facilita l'inclusione di scritte nei programmi in codice macchina.
Una direttiva molto importante è la ORG (da ORiGine), con la quale si può modificare l'indirizzo contenuto nel contatore di locazione dell'Assembler. In pratica, la ORG viene sempre usata all'inizio di un listato Assembly, per stabilire da quale indirizzo debba iniziare il codice macchina finale. In un programma è consentito usare anche più di una ORG, se necessario.
Un'altra utile direttiva, generalmente usata in principio ad un listato, è la EQU, la cui funzione è paragonabile al segno aritmetico "uguale". Ad esempio:

```
RESET   EQU     $F100
```

significa che al nome RESET viene assegnato il valore $F100. In questo caso RESET non è un'etichetta riferita ad un indirizzo, ma semplicemente il nome assegnato alla costante $F100.
Non a caso, l'ultima direttiva esaminata è la END, che va sempre inserita al termine del listato Assembly.
Come la ORG, anche la END non richiede alcun operando.

Indirizzamento sul 6809

Una misura della qualità di un microprocessore è data dal numero di modi d'indirizzamento ammessi.
Il 6809 ne possiede molti. Tutti gli esempi di indirizzamento forniti fin qui sono del tipo "diretto" o del tipo "esteso": ciò significa che il valore specificato nel campo degli operandi è l'indirizzo della locazione in cui si trova un dato.
Nell'indirizzamento "diretto" è sufficiente indicare un solo byte (la parte "bassa") dell'indirizzo, poiché la metà più significativa viene estratta da un apposito registro del processore, chiamato "registro di pagina diretta". L'indirizzamento diretto offre indiscutibili vantaggi: flessibilità e genericità. Ad esempio, una subruotine, scritta usando tale metodo, può essere collocata ovunque in memoria: basterà caricare nel registro di pagina diretta il valore appropriato, prima di richiamarla.
Nell'indirizzamento esteso, viceversa, l'operando di un'istruzione è un indirizzo specificato per intero (entrambi i byte), quindi tale istruzione è sempre riferita ad una ben precisa zona di memoria.
L'Assembler riconosce l'indirizzamento diretto da quello esteso esaminando la natura dell'operando.
Un altro modo d'indirizzamento molto usato è quello "immediato", in cui l'operando è il dato stesso sul quale operare. In tal caso, l'operando è preceduto dal simbolo #. Per esempio:

ORG $1000	il codice macchina inizia a $1000
NUM1 FDB $FFFF	inizializza la locazione NUM1 col valore $FFFF
LDD NUM1	carica il contenuto di NUM1 (ossia il valore $FFFF) nel registro D
LDD #NUM1	carica il valore NUM1 ($1000 in questo caso) nel registro D

Si noti che l'etichetta NUM1 corrisponde all'indirizzo $1000: la direttiva ORG, infatti, non genera alcun codice macchina, ma si limita a comunicare l'indirizzo di partenza all'Assembler.
Si noti che LDD NUM1 carica in D il contenuto della locazione NUM1, mentre LDD #NUM1 carica in D il valore assegnato a NUM1 (ossia l'indirizzo $1000).

La Casio

Questa ditta giapponese è nota soprattutto per gli orologi e per le calcolatrici tascabili

La Casio, pur essendo una compagnia di medie dimensioni (i suoi dipendenti sono circa 3.000), controlla circa il 50% del mercato mondiale delle calcolatrici.
Nel 1983, nonostante gli alti standard di produttività, il suo fatturato è stato di appena 29 milioni di dollari. Questi modesti risultati sono dovuti a precise scelte imprenditoriali: la Casio segue infatti una politica di mercato piuttosto conservatrice e rifugge dalle più moderne tecniche di lancio pubblicitario.
La ditta, fondata dopo la Seconda Guerra Mondiale dai cinque fratelli Kashio col nome di Kashio Seisakujo, si dedicò inizialmente alla produzione di apparecchiature per ufficio.
Agli inizi degli anni '50 lanciò il 14-A Relay Calculator, una delle prime calcolatrici elettriche del mondo: una struttura ingombrante, che pesava circa 130 chili.
Negli anni seguenti la ditta produsse altre calcolatrici e nel 1957 cambiò la propria ragione sociale in Casio Computer Company Limited. Nel 1982 realizzò il suo primo computer tascabile, il Casio FX-702P. Questo apparecchio era particolarmente indicato per applicazioni scientifiche, ma i suoi tasti, anziché seguire il comune formato QWERTY, erano curiosamente disposti in ordine alfabetico. L'FX-702P ebbe scarso successo e fu ben presto rimpiazzato dall'FX-700P, dotato di normale tastiera QWERTY.
Ad esso seguì il computer tascabile PB-100, rivolto soprattutto ad utenti impegnati in attività commerciali. Il suo progetto richiamava quello dell'FX-702P, ma l'apparecchio era dotato di un display a cristalli liquidi e di una tastierina numerica. Per queste macchine la Casio produsse, oltre a pacchetti RAM, registratori a cassetta e stampanti/plotter "su misura".
L'FX-750P, che ha recentemente sostituito il PB-100, prevede l'uso di due "RAMcard", ossia sottilissime schede che possono contenere fino a 4 Kbyte. Ogni scheda è dotata di una batteria da tre volt ed è in grado di conservare le informazioni ed i programmi anche quando viene rimossa dall'apparecchio. Le batterie durano circa un anno: prima della sostituzione, i programmi (o i dati) vengono trasferiti su nastro e successivamente ricaricati sulla scheda.
La Casio produce anche l'FP-200, un computer tascabile con 8 Kbyte di RAM. Il suo display a cristalli liquidi possiede 8 righe di 20 caratteri ed offre una risoluzione grafica di 160x64 pixel.
L'ultima creazione della Casio è l'LS-800, una calcolatrice di basso costo dalle dimensioni di una carta di credito. Per fabbricare un apparecchio a sezione così ridotta, viene impiegata una tecnica detta "filming" o "a impressione", mediante la quale i componenti del circuito, invece di essere saldati su una scheda convenzionale, vengono letteralmente stampati su una pellicola metallica.
La Casio esporta in Europa computer gestionali ed home computer che adottano gli standard MSX. Tuttavia, in certe nazioni dove la concorrenza nel settore dei micro è molto forte, punta soprattutto sulle calcolatrici e sui computer tascabili.
Per il futuro la Casio, oltre a cercare di ampliare il proprio mercato, intende sfruttare la propria esperienza nel campo delle tastiere elettroniche per produrre una serie di apparecchi basati sull'interfaccia digitale MIDI (vedere a pag. 1014).



Il Centro di Ricerca e Progettazione della Casio, a Tokyo



I magnifici cinque
Tadao Kashio, presidente della Casio, è uno dei cinque fratelli Kashio, fondatori di questa compagnia giapponese

Musica ... sintetica

Le innumerevoli applicazioni dell'interfaccia MIDI nel settore amatoriale e didattico



MARCUS WILSON-SMITH

Imparare a leggere la musica non è facile, anche per chi, magari, suona già uno strumento. Infatti, talvolta non si riesce ad abbinare la notazione musicale, composta da strani simboli e segnature, con l'effettiva sequenza di suoni e pause del brano in questione.

Il problema principale è causato proprio dalla natura stessa della musica: per aver senso, essa deve rappresentare una serie di eventi sonori che avvengono in un ben preciso intervallo di tempo ed è proprio questo che scoraggia i principianti. Infatti, dovendo continuamente interrompere l'esecuzione per leggere e interpretare i vari simboli, l'effetto finale risulta talmente frammentato nel tempo da non assomigliare più al brano originale.

L'interfaccia MIDI può rimuovere tale ostacolo, offrendo un valido strumento, dunque, anche in campo didattico. Un pezzo musicale precedentemente memorizzato (anche a piccoli passi) tramite un sintetizzatore, può essere riprodotto al ritmo corretto.

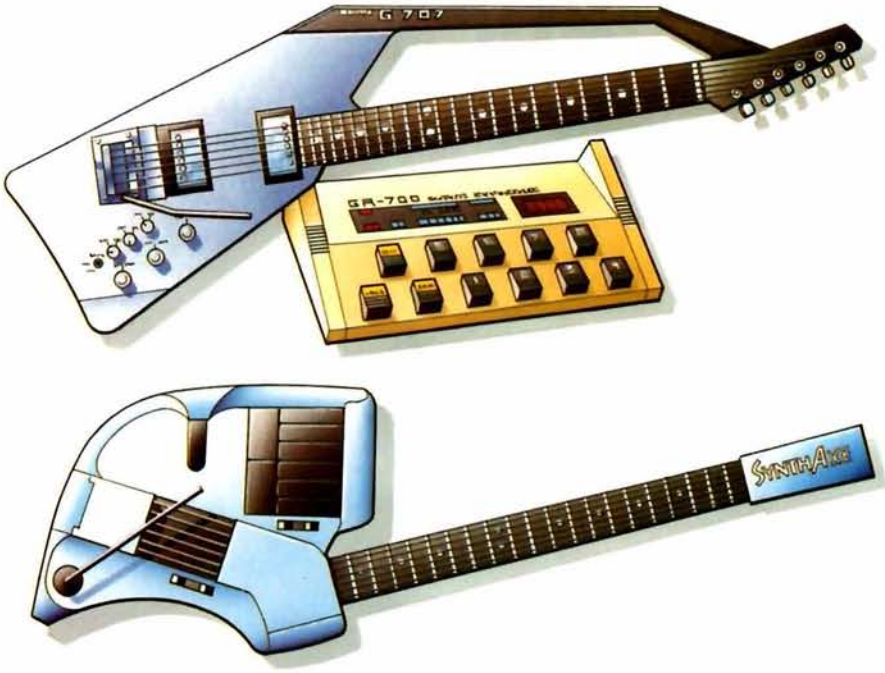
Nel frattempo, col software adatto, su di uno schermo può comparire una visualizzazione grafica della notazione musicale relativa al brano suonato. Volendo, la rappresentazione su pentagramma può avvenire simultaneamente alla pressione dei tasti e l'allievo può facilmente abbinare i tasti alle note riprodotte sullo spartito. Supponiamo che venga suonato, per un certo tempo, un accordo di Si minore: sullo schermo compaiono le note che lo compongono (Si, Re e Fa diesis), assieme alla durata espressa in termini musicali.

Si può andar oltre e far sì che sullo schermo appaia, nota per nota ed al ritmo voluto, la notazione relativa ad un brano precedentemente memorizzato, durante la stessa esecuzione. Se si incontra un passaggio difficile basta fermare l'esecuzione, ritornare al punto giusto e risentirlo (e rivederlo) eventualmente rallentandone il ritmo. Un'altra possibilità è quella di poter variare i timbri ed i parametri del sintetizzatore, introducendo così l'allievo ad un altro aspetto della musica: quello dell'arrangiamento musicale.

Inoltre, poiché la musica obbedisce a precisi rapporti matematici, si può addirittura arrivare a comporre un pezzo in termini aritmetici, per poi apprezzarne la validità sullo strumento. Una volta presa sufficiente pratica con lo strumento, si può passare ad uno dei linguaggi specifici, quale l'MCL (Music Composition Language), lasciando da parte la notazione musicale classica. Trattandosi di apparecchi elettronici e dell'immissione di dati numerici in una memoria, l'MCL è senz'altro un sistema più adatto allo scopo, poiché prevede anche l'inclusione di parametri elettronici, non previsti dalla notazione tradizionale. Tuttavia, non esiste un linguaggio MCL standard: ogni sistema adotta il

Un nuovo stile, un nuovo "sound"

Nel DX7, uno tra i molti nuovi sintetizzatori della gamma DX prodotta dalla Yamaha, la generazione dei suoni avviene in base ad un metodo precedentemente adottato soltanto in apparecchi costosissimi: la sintesi a modulazione di frequenza (FM). Anziché partire da un suono fondamentale preesistente e modificarlo tramite l'impiego di speciali filtri, il DX7 crea suoni complessi combinando in svariati modi sei forme d'onda fondamentali. Ciò consente di riprodurre i suoni tipici di molti strumenti tradizionali con una fedeltà enormemente superiore a quella offerta dai sintetizzatori della precedente generazione. Nel DX7 è incorporato anche un controllo "a fiato", per arricchire i suoni emessi con variazioni altrimenti impossibili da ottenere. Il DX7 adopera cartucce di ROM contenenti suoni predefiniti, ma è possibile anche memorizzare quelli creati dal musicista nella RAM del sistema. Il costo del DX7 è paragonabile a quello di un personal computer



KEVIN JONES

SUONO SINTETIZZATO

Il controllo di molti sintetizzatori più recenti non avviene, come in passato, tramite una tastiera simile a quella di un pianoforte, ma mediante forme alternative, come ad esempio i segnali emessi dalle corde di una chitarra elettrica. Il Roland GR700 è uno di questi apparecchi.
I complessi segnali prodotti dalla vibrazione delle corde vengono captati da appositi rivelatori a sei canali ed inviati al sintetizzatore, dove vengono combinati con altri parametri.
Un sistema ancor più originale è quello offerto dal SynthAxe, un sintetizzatore controllato da un chip 6809. Le informazioni elettriche vengono rilevate, tramite speciali sensori situati tra le corde e le sbarrette dei tasti, sul manico della chitarra. Altri sensori, collocati in cima al manico, captano altre particolarità, quali lo sforzo sostenuto dalle corde. Tramite una tastierina ed un secondo gruppo di corde si possono ulteriormente arricchire i suoni

proprio dialetto, un po' come è successo per il BASIC sui computer.
Imparare il linguaggio MCL significa poter sfruttare appieno le possibilità offerte dal sistema MIDI. Sfortunatamente, l'insegnamento della musica nelle scuole e nei conservatori è tuttora impostato esclusivamente sullo studio della musica classica: molti insegnanti ritengono che la musica elettronica sia il frutto di una "corrente" troppo giovane per essere inclusa nei programmi di studio, se non addirittura qualcosa di completamente estraneo ad essi!
Di conseguenza, è ragionevole supporre che per molto tempo ancora l'enorme potenzialità del sistema MIDI venga esclusa dai normali corsi d'insegnamento. Per di più, il fatto che occorrano alcune nozioni basilari sulla programmazione dei computer aggrava la situazione. Né si può sperare molto di più da eventuali corsi scolastici sull'informatica, in cui la musica elettronica viene considerata un aspetto secondario, come l'acquisto delle relative apparecchiature (sintetizzatore, interfaccia MIDI ecc.).
Nel ramo professionale, ovviamente, le cose sono ben diverse: qualsiasi novità viene immediatamente recepita ed utilizzata. Il sistema MIDI è già impiegato efficacemente da molti musicisti, anche nei concerti dal vivo, per coordinare tutta una serie di strumenti elettronici.
Infatti, la maggior preoccupazione di un musicista, nell'adoperare sequenziatori, sintetizzatori e batterie elettroniche è proprio quella di far sì che essi suonino in perfetto sincronismo.
Pare che alcuni musicisti di successo, quali i Thompson Twins e gli Howard Jones, abbiano più volte preferito esibirsi dal vivo affidandosi a pezzi registrati in studio, piuttosto che rischiare qualche difficile passaggio su strumenti così nuovi.
Agli inizi degli anni '70, molti complessi musicali adottarono i primi sintetizzatori, composti da una o più tastiere e da un'infinità di manopole e cavetti di collegamento. Talvolta, se in un concerto occorreva più di un "synth", la bravura del musicista non era tanto nel suonare un particolare brano, quanto nel saper coordinare i vari suoni emessi, raggiungendo l'effetto desiderato. Oggi basta una sola tastiera; a sincronizzare tutta la miriade di nuove "scatole nere" pensa il sistema MIDI.
In effetti, le innovazioni introdotte dai sistemi digitali possono sconvolgere totalmente il ruolo tradizionale dei musicisti: nulla vieta, ad esempio, di abbandonare del tutto l'uso di tastiere. Esistono già in commercio speciali apparecchi, che convertono i segnali provenienti da qualsiasi strumento (anche la voce umana) in codici di controllo digitali, da usare su un sistema computerizzato. Ciò significa che può essere un suonatore di sassofono a controllare tutto il settore ritmico ed armonico di sottofondo, creato dal sintetizzatore, soffiando nel proprio strumento. La Yamaha, ad esempio, ha previsto un controllo "a fiato" sul proprio sintetizzatore DX7, mentre la SynthAxe (una rivoluzionaria chitarra digitale) è stata progettata per usare il sistema MIDI nel controllo di un sintetizzatore Fairlight.
Non c'è più da stupirsi, quindi, se il tipico suono degli strumenti a corda viene ottenuto soffiando in un clarino, né se il suono di un trombone viene provocato dal pizzicare una corda di chitarra. Si può facilmente immaginare quale sarà la tendenza musicale nei prossimi anni, a seguito di tali cambiamenti radicali.

Per la fine degli anni '80, probabilmente, i più conformisti proveranno soddisfazione nel veder usare gli strumenti "di una volta": una chitarra da jazz, un sassofono, un contrabbasso ed una batteria. Tuttavia, il loro stupore sarà grande nel constatare che il chitarrista, in effetti, suona un vibrafono e che, magari, il batterista produce i suoni di un violino!

Alla prima esperienza in uno studio di registrazione contemporaneo, molti complessi abituati ad esibirsi "dal vivo" si trovano intimiditi dalla quantità e dalla qualità delle apparecchiature. Vengono messi di fronte a strumenti musicali ed a sistemi operativi mai incontrati prima. A peggiorare le cose, i tecnici dello studio possono non conoscere lo stile o il "sound" del complesso, quindi il raggiungimento di particolari effetti può richiedere molto più impegno e fatica del necessario. Poiché i tempi devono essere necessariamente brevi, talvolta i risultati non sono tra i più esaltanti, nonostante la buona preparazione dei musicisti e l'indiscussa potenza delle apparecchiature usate. Si tratta, comunque, di un periodo di transizione ("doloroso", per alcuni), che verrà presto superato dalla sempre più crescente diffusione di apparecchi pilotati tramite l'interfaccia MIDI o sistemi similari.

Naturalmente, questa vera e propria irruzione dell'elettronica e dell'informatica nel campo musicale ha sconcertato e sconcerta tuttora i musicisti, che si trovano a dover imparare come funzioni il sistema operativo di un computer, l'unità a dischetti e via dicendo. I risultati non possono essere così immediati quanto forse possono desiderare le case discografiche, sempre alla ricerca di novità per far colpo sul pubblico. In certi casi alcuni complessi musicali hanno superato la transizione senza traumi, ma non è sempre così.

Può succedere, ad esempio, che un complesso si trovi costretto ad abbandonare gli strumenti ai quali è abituato (e che caratterizzano un particolare "sound") per avventurarsi in qualcosa di freddamente anonimo sui nuovi strumenti elettronici messi a disposizione dagli studi di registrazione.

Ovviamente, si troveranno avvantaggiati coloro che hanno già una certa dimestichezza con il sistema MIDI: per creare e memorizzare una ben precisa sequenza di suoni può bastare il semplice uso di un sintetizzatore di un tipo al posto di un altro.

Una certa pratica del sistema MIDI torna utile anche in tutte le attività accessorie della produzione musicale, non direttamente connesse alla generazione di suoni. Ad esempio, le più moderne consolle da mixaggio contengono l'ormai onnipresente computer, che consente di memorizzare, riprodurre e modificare a piacimento tutte le sequenze di comandi relative alla fusione dei suoni prodotti da ciascun musicista.

Infatti, poiché la registrazione di un brano viene solitamente fatta su nastri con 24 piste (una o più per ciascun strumento o voce), le possibilità di variare l'effetto finale (il "sound") dosando accuratamente il volume o particolari effetti su ciascuna pista, sono infinite. Tanto per citare un caso, si pensi che talvolta, in un preciso momento, occorre aggiungere "reverbero" ad una sola nota prodotta da una tromba: far ciò, mentre sulle altre 23 piste sono presenti altri segnali, è un compito veramente arduo. Usare un computer in queste operazioni è come impiegare un sistema MIDI per coordinare più strumenti dal vivo.

Un'altra tecnica, originariamente realizzata per la sincronizzazione nei filmati, sta prendendo piede anche in campo musicale: si tratta della sincronizzazione tramite un codice numerico di tempo. Questo codice è paragonabile agli impulsi di un orologio digitale, ma viene registrato simultaneamente alla musica. Nelle fasi di doppiaggio, viene adoperato il codice di tempo che è composto da "parole" di ben 80 bit: tale frammentazione consente di misurare con estrema precisione il momento in cui intervenire con una modifica o un'aggiunta sulla registrazione originale.

Da quanto detto fin qui, si può comprendere quanto l'informatica stia influenzando anche un settore altamente crativo quale quello musicale. Oltre alle doti artistiche, i musicisti contemporanei devono possedere anche una profonda conoscenza di sistemi computerizzati quali il MIDI. Inoltre, si creano nuove opportunità anche per coloro che, pur non essendo particolarmente preparati nella teoria musicale, si sono formati una pratica sui sistemi computerizzati.

Nella parte che sarà prossimamente dedicata a questo argomento, verranno esaminati alcuni dei più sofisticati apparecchi attualmente in commercio.



Una vera scienza

Laurie Anderson, un'artista e poetessa di New York, crea effetti sonori interessantissimi, combinando suoni provenienti da ogni sorta di apparecchiature. In brani quali "O Superman" e "Mr. Heartbreak" sono impiegate, ad esempio, dalle campane africane alle più raffinate sfumature di suono ottenibili con strumenti elettronici quali il Vocoder e il Synclavier

Giochi d'incastro

Una procedura LOGO può contenerne altre definite precedentemente

Per meglio chiarire come si possa definire una procedura in funzione di un'altra definita in precedenza, consideriamo il gioco d'incastro *Tangram*. Il gioco è composto da un quadrato (generalmente in legno) suddiviso in 7 sagome geometriche, che vengono combinate per creare nuove figure. Nel nostro esempio useremo i sette pezzi per creare l'immagine di un cane. Il lavoro inizia con la programmazione delle procedure per la creazione dei vari pezzi del gioco, successivamente inserite in una sola procedura, qui chiamata CANE. Le procedure occorrenti per il corretto posizionamento della tartaruga, durante il disegno di ciascun pezzo, sono chiamate MOVE1, MOVE2 ecc., fino a MOVE7.

Per creare l'immagine del cane, è sufficiente disporre i vari comandi nell'opportuna sequenza all'interno di una procedura più lunga. La tecnica di progettazione adottata è quella "top-down", descritta in dettaglio a pagina 956. In pratica, si tratta di suddividere il problema in un gruppo di operazioni elementari, che vengono poi risolte l'una dopo l'altra. Per capire come funzioni questo procedimento, analizziamo la struttura del programma che traccia la figura del cane.

Per prima è stata redatta la procedura CANE, anche se le procedure che la compongono non erano ancora state sviluppate; poi, separatamente, le procedure che tracciano i vari pezzi e, successivamente, quelle che li posizionano sullo schermo. Alla fine è stata eseguita la procedura CANE, per verificare l'esatta collocazione dei vari pezzi. Ovviamente, quando il LOGO giunge ad una procedura

IL "TANGRAM"

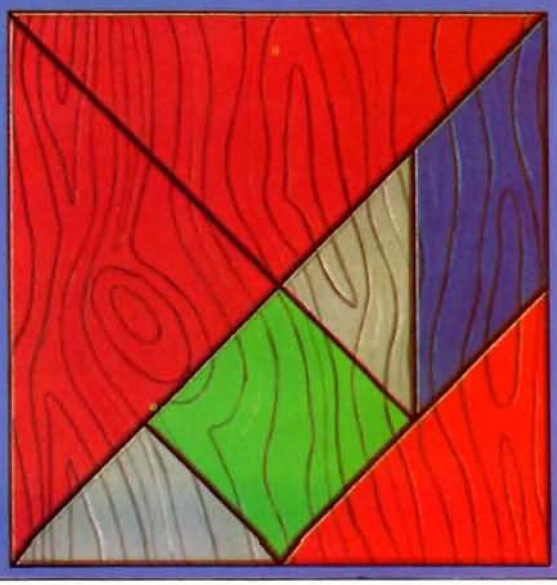
Il "Tangram" è un antico gioco ad incastro, composto da 7 pezzi che vengono combinati in vario modo per creare figure stilizzate. Il primo dei due listati qui presentati traccia i 7 pezzi di base, mentre il secondo disegna l'immagine di un cane. La procedura CANE inizia con il triangolo che rappresenta la zampa posteriore del cane

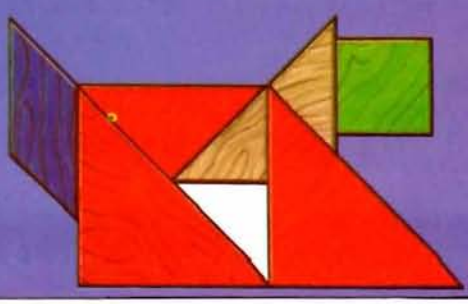
Procedure Tangram

```
TO QUADRATO
   REPEAT 4 [FD 25 RT 90]
END
TO PAR
   REPEAT 2 [FD 25 RT 45 FD 35 RT 135]
END
TO TRI1
   FD 25 RT 135 FD 35 RT 135 FD 25 RT 90
END
(esistono due triangoli di queste dimensioni)
TO TRI2
   FD 35 RT 135 FD 50 RT 135 FD 35 RT 90
END
TO TRI3
   FD 50 RT 135 FD 71 RT 135 FD 50 RT 90
END
(ci sono due triangoli di queste dimensioni)
```

Programma CANE

```
TO CANE
   TRI3 MOVE1 PAR MOVE2 TRI2 MOVE3
   TRI1 MOVE4 TRI3 MOVE5 TRI1 MOVE6
   QUADRATO MOVE7
END
TO MOVE1
   PU FD 15 LT 45 PD
END
TO MOVE2
   PU RT 45 FD 35 LT 45 BK 35 PD
END
TO MOVE3
   PU LT 45 BK 25 PD
END
TO MOVE4
   PU RT 90 BK 25 PD
END
TO MOVE5
   PU FD 50 RT 45 PD
END
TO MOVE6
   PU FD 25 RT 135 FD 5 LT 90 PD
END
TO MOVE7
   PU LT 90 FD 5 RT 45 BK 25 RT 45
   BK 50 LT 90 BK 50 PD
END
```





KEVIN JONES

MOVE non ancora creata, l'esecuzione si ferma e sullo schermo compare un messaggio d'errore. Tuttavia, questa tecnica permette di controllare per gradi se l'immagine via via disegnata è corretta o se si è commesso un errore nell'ultima procedura MOVE aggiunta.
Questo modo mette in luce un altro importante principio: tutte le procedure usate per disegnare la figura, compresa la procedura CANE, lasciano inalterato lo stato (posizione e direzione del movimento) della tartaruga: ciò significa che dopo ogni esecuzione, essa si trova nella stessa posizione ed è rivolta nella stessa direzione che aveva all'inizio. Simili procedure vengono dette "trasparenti" e questa caratteristica facilita notevolmente lo sviluppo di immagini anche molto complesse. Consideriamo, ad esempio, la procedura CANE: la tartaruga, dopo essere stata posizionata ed aver disegnato un pezzo, si trova nella stessa posizione che aveva prima di cominciare; ne consegue che, per disporre sullo schermo i vari pezzi, non è affatto necessario conoscere il funzionamento interno di ciascuna procedura. Se rendessimo "trasparente" anche l'intera procedura CANE, potremmo usarla più agevolmente, come parte di un'altra procedura, per visualizzare sullo schermo una serie di cani.

Area di lavoro LOGO

A questo punto abbiamo in memoria numerose procedure: è quindi utile esaminare più dettagliatamente come il LOGO organizzi la memoria del computer. Quando si carica il LOGO, la sua memoria di lavoro presenta una serie di "nodi" (ciascuno composto da 5 byte) il cui numero varia da 1.000 a 3.000, a seconda dell'apparecchio usato. Alcuni nodi vengono "occupati" dalle procedure create dal programmatore mentre altri vengono usati durante la loro esecuzione oppure dalle variabili in esse contenute (più avanti nel corso vedremo come ciò avvenga).
Le procedure sviluppate vanno a comporre la nostra area di lavoro. Inserendo il comando POTS (contrazione di PRINTOUT TITLES) si ottiene un listato delle procedure presenti, mentre per visualizzarne una in particolare occorre digitare PO (ovvero PRINTOUT), ad esempio: PO QUADRATO. Se una procedura non serve più, l'istruzione ERASE la cancella. Ad esempio ERASE QUADRATO libera i nodi da essa impiegati. Il LOGO marca questi nodi, ma prima di aggiungerli alla tabella dei nodi liberi, deve esaurire quelli di quest'ultima. A questo punto, il LOGO esamina tutta la memoria, recupera i nodi liberi e crea una nuova tabella di nodi. Questa operazione, detta in gergo *garbage collection* (raccolta di rifiuti), spiega perché, periodicamente, il LOGO sembra fermarsi per un secondo o due.

Salvataggio delle procedure

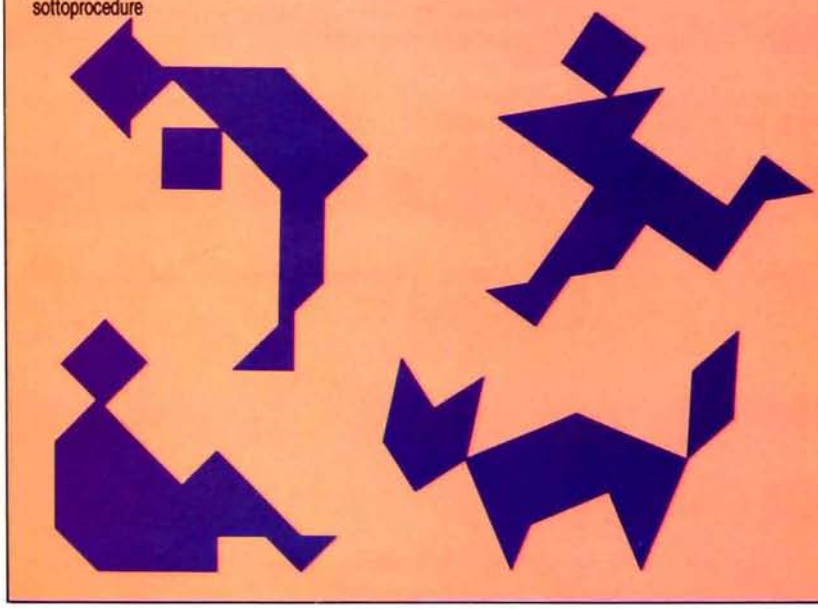
Per archiviare le procedure in modo permanente su dischetto, è necessario salvare l'area di lavoro LOGO come se si trattasse di un file. Il comando di salvataggio è SAVE "MYPROCS, dove MYPROCS è un nome di file scelto dal programmatore (si noti che le virgolette precedono, ma non seguono il nome). Questa istruzione non modifica in alcun modo il contenuto dell'area di lavoro ma ne esegue semplicemente una copia su disco. Il file può essere ricaricato in memoria tramite il comando READ "MYPROCS, che definisce le procedure e le aggiunge a quelle presenti nell'area di lavoro. Se una delle procedure che vengono caricate ha lo stesso nome di un'altra che già si trova in memoria, la nuova procedura sostituisce quella preesistente.
Altri utili comandi di gestione del disco sono CATALOG ed ERASEFILE: il primo produce una lista di tutti i file contenuti nel disco, mentre ERASEFILE "MYPROCS cancella il file "MYPROCS. Le diverse versioni LOGO, specialmente quelle su cassetta, prevedono comandi diversi; quindi è bene consultare il relativo manuale.

ABBREVIAZIONI

ERASE	ER
PRINTOUT	PO
PRINTOUT TITLES	POTS

ESERCIZI SULLE PROCEDURE

1) Scrivere le procedure per tracciare le figure qui illustrate. (ovviamente, prima occorre capire da quali elementi esse sono composte!)
2) Scrivere una procedura che disegni una casa (composta da un triangolo equilatero sovrapposto ad un quadrato)
3) Scrivere una procedura che tracci una griglia di 5 x 5 quadrati
4) Riscrivere la precedente procedura, in modo da disegnare una stella a sei punte, servendosi di più sottoprocedure



LIZ DIXON

Logo & Logo

In tutte le versioni LCSI, i nomi di procedura usati con le istruzioni PO ed ERASE devono essere preceduti da virgolette: ad esempio, PO "QUADRATO ed ERASE "QUADRATO.

Usare il comando LOAD "MYPROCS per leggere un file su disco.

Le versioni LOGO su cassetta o microdrive possiedono comandi leggermente diversi da quelle su dischetto. Occorre quindi consultare i rispettivi manuali



Le "librerie" di routine

Si può evitare di riscrivere più volte identiche sezioni di programma

Le tecniche di progettazione strutturata descritte in precedenza possono sembrare lunghe e complicate ma, in realtà, permettono di risparmiare tempo sia nella fase di codifica sia durante la messa a punto o debugging di un programma. Infatti, i programmi creati "di getto" alla tastiera spesso presentano strutture e algoritmi non lineari, richiedono più tempo per la loro trascrizione e sono più frequentemente soggetti ad errori. Ciò avviene perché tale modo di programmare non consente di seguire la logica del programma in modo adeguato e richiede un maggior impegno nelle operazioni di verifica e debugging. Al contrario, un'analisi preventiva semplifica la struttura e gli algoritmi di un programma, riduce gli errori di codifica e facilita le operazioni di verifica e di messa a punto.

La progettazione preventiva evita al programmatore di scrivere strutture o procedure di controllo che potrebbero dimostrarsi inadeguate (ad esempio campi troppo piccoli all'interno di un file). L'insorgere di simili problemi rischia di compromettere il funzionamento di un programmma e spesso costringe il programmatore a riscrivere parti del codice, ampliandole più del necessario.

Per quanto riguarda la battitura del programma sulla tastiera, si può fare ben poco per ridurne il tempo necessario. Il processo di codifica, invece, può essere velocizzato in molti modi. Il più semplice è ideare ed adottare una serie di "convenzioni" di codifica. Ad esempio, usare per le variabili locali particolari tipi di nomi che permettano di distinguerle da quelle del programma principale; iniziare tutte le subroutine con numeri di linea che terminano con 000; chiudere tutte le subroutine con una linea che contenga la sola istruzione RETURN; assegnare a ciascun tipo di subroutine uno specifico "blocco" di linee (ad esempio, inserire le routine di gestione dei file tra le linee 9000 e 9999 e quelle di utilità a partire dalla linea 50000, e così via).

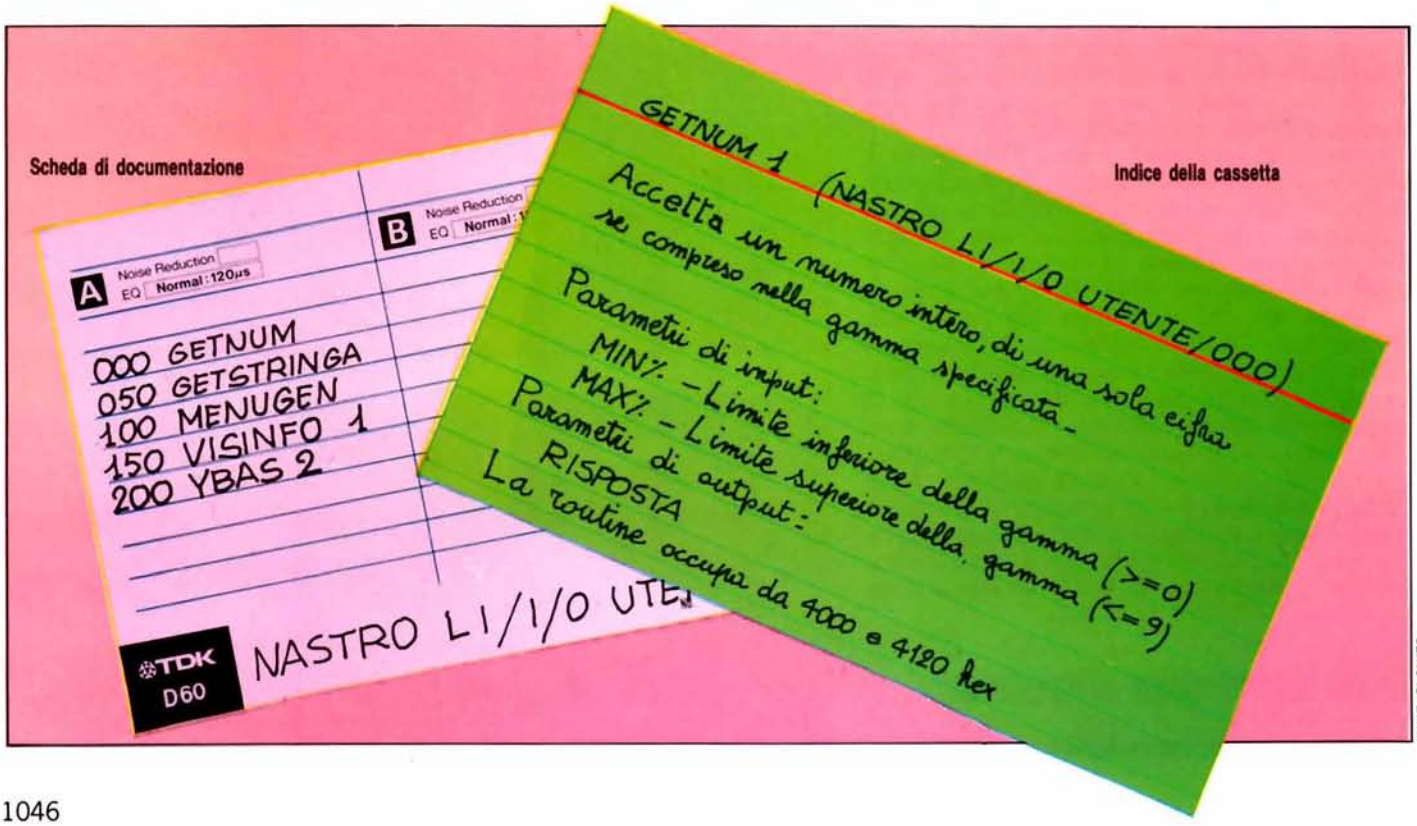
L'impiego di queste convenzioni comporta numerosi vantaggi: non è necessario ispezionare il menu per trovare l'indirizzo delle routine, poiché queste occupano sempre la stessa posizione all'interno del programma e, inoltre, non si corre il rischio di usare nomi identici per le variabili del programma principale e per quelle di una subroutine.

Sistema omogeneo

Le librerie di subroutine sono strumenti inutili, se non vengono corredate da una documentazione esauriente ed omogenea. Questo è vero soprattutto per le versioni su cassetta: in questi casi, l'utente che desideri esaminare il contenuto di una cassetta priva di documentazione, è costretto a caricare e listare in successione tutti i programmi in essa esistenti

Librerie di programmi

Queste tecniche di programmazione si rivelano altrettanto utili per la creazione delle librerie di programmi. Una libreria di subroutine ben organizzata consente infatti di dimezzare



il tempo di codifica. Il miglior sistema per creare una libreria di questo tipo consiste nell'esaminare i programmi che già si possiedono ed estrarre da essi le subroutine migliori e quelle più "generiche" (routine di I/O, immissione di date, di conversione da maiuscolo a minuscolo ecc.). Ciascuna routine viene poi salvata in un file indipendente oppure raggruppata con altre in base alla sua funzione (nel caso di archiviazione su nastro, conviene memorizzare ciascun gruppo su una cassetta separata). I nomi dei file dovrebbero far riferimento allo scopo delle routine. Infine, è opportuno mantenere un indice o un database, nel quale abbinare al nome del file anche una succinta descrizione del suo funzionamento.
Ovviamente, tutte le routine della libreria devono essere totalmente esenti da errori. Inoltre, siccome non sono state appositamente progettate per i programmi nei quali verranno impiegate, occorre inserire opportune verifiche sulla validità dei valori passati dal programma alla routine e viceversa. La documentazione interna alla routine (i commenti nelle linee REM, per intendersi) dovrebbe essere sintetica, ma esauriente. Si ricordi che una "libreria" così concepita deve essere polivalente, ma non sovrabbondante: deve contenere soltanto le routine necessarie ed utili per una buona programmazione. Numerare le linee delle varie routine secondo le convenzioni adottate è molto importante: ciò evita di doverne modificare la numerazione ogni volta che occorre fonderle (MERGE) assieme ad un nuovo programma. Routine o programmi particolarmente utili possono essere anche copiati da pubblicazioni specializzate ed inseriti nella propria libreria.
Naturalmente, è necessario un sistema che permetta di estrarre ed assemblare le varie routine in un programma completo. È il "link-loader" (uno speciale programma di conversione del codice intermedio in codice eseguibile) che si occupa di estrarre le subroutine specificate dalla libreria, quando si adoperano compilatori di programmi, anziché linguaggi "interpreti". La maggior parte degli home computer, tuttavia, non prevede l'impiego di tali compilatori, ma si può ottenere un risultato pressoché identico usando i comandi RENUMBER e MERGE. Per inserire una routine in un programma, prima di tutto si carica in memoria il programma, poi si decide dove inserire la routine, assicurandosi che siano presenti un numero di linee sufficienti ad accoglierla (all'occorrenza, si può modificare la numerazione della routine stessa). Infine, dopo aver fuso assieme i due programmi con il comando MERGE, è opportuno controllare che tutto funzioni a dovere. A questo punto si può salvare il nuovo programma, che contiene la routine di libreria.

Lavoro di gruppo

Spesso gli utenti di home computer lavorano in equipe alla stesura dei programmi: in questi casi la progettazione preventiva del programma e l'abilità del singolo programmatore assumono un'importanza ancora maggiore. In effetti, molte delle idee e dei concetti su cui poggia la programmazione strutturata sono nati dalla necessità di spartire tra i membri di uno "staff" il lavoro di programmazione relativo al software commerciale.
Un gruppo di programmatori BASIC che intenda lavorare in questo modo deve ovviamente adottare le stesse convenzioni di codifica e ciascun membro del gruppo deve avere ben chiari i seguenti punti:

1) Quali file vanno sviluppati e qual è la loro organizzazione interna.
2) Quali convenzioni vanno seguite per i nomi delle variabili (ad esempio, per le variabili locali); i nomi delle variabili più importanti (quelle usate in tutto il programma principale) devono essere stabiliti in precedenza. I nomi delle variabili interessate allo scambio di valori tra i vari moduli devono essere definiti in precedenza e uniformati, ad esempio facendo seguire al nome il numero del modulo di provenienza.
3) Quali routine esistano già nella "libreria", il loro formato, i nomi delle variabili usate, la loro funzione ed il criterio delle eventuali verifiche di validità incorporate.
4) Come sono organizzate le routine per la gestione degli errori (per esempio se le routine segnalano direttamente gli errori o se invece assegnano un valore ad un "flag", in seguito esaminato da una routine di controllo).
5) L'esatta gamma e tipologia dei dati accettati in ingresso o trasmessi in uscita da ciascun modulo.

Tutto questo implica una lunga fase di pianificazione e di organizzazione del lavoro seguita da quella, spesso più breve, di effettiva programmazione. Le operazioni di verifica, compreso l'esame dei programmi prodotti da "equipe", verranno approfondite più avanti, dopo aver esaminato la possibilità di velocizzare e snellire i propri programmi, riducendo l'occupazione di memoria.

MERGE

● Lo Spectrum è dotato di un utile comando MERGE, che "fonde" il file specificato con il programma in memoria. Se due o più numeri di linea coincidono, la linea del file sostituisce quella del programma ricevente

● Il Micro BBC possiede il comando *SPOOL, col quale si può creare un file contenente il programma in formato ASCII. A questo punto basta un qualsiasi programma per l'elaborazione di testi in modo da apportare modifiche rapidamente. Il comando MERGE fonde due file in formato ASCII in un terzo file anch'esso ASCII. Per trasformarlo in un programma eseguibile, si usa il comando *EXEC

● Sul Commodore la sequenza di istruzioni: OPEN1,1: CMD1: LIST: PRINT#1: CLOSE1 crea su nastro un file (senza attribuirgli un nome) contenente il programma, ma in formato ASCII. Si deve quindi caricare il secondo programma al quale va aggiunta una piccola ruotine che legga e listi su schermo il file ASCII precedentemente memorizzato. Eseguita e conclusa questa piccola routine, i due programmi sono riuniti in uno solo

L'Apple IIc

Anatomia di un computer a diffusione europea: tecnologie a confronto

Il grande successo ottenuto dal Macintosh, la crescente concorrenza dei prodotti Commodore nel settore degli home computer e degli elaboratori IBM e ACT per il mercato degli apparecchi gestionali, hanno momentaneamente offuscato il futuro della serie Apple II. Molti rivenditori e analisti di mercato ritengono che questa gamma di apparecchi sia destinata ad una rapida scomparsa, nonostante la Apple sembri ancora puntare su computer basati sul microprocessore 6502. Recentemente è stato lanciato l'Apple IIc assieme ad una serie di novità hardware e software che dovrebbero prolungare di almeno tre anni la permanenza sul mercato della gamma Apple II.

I diversi modelli di questa serie (II, II+, IIe) si sono infatti creati una propria area commerciale, sono stati per alcuni anni in testa alle vendite in più paesi del mondo ed hanno contribuito in modo significativo al record totale di vendite della nota ditta americana. Il numero degli apparecchi Apple venduti in tutto il mondo supera i 2 milioni di unità, anche se il successo riscosso dall'Apple II in Europa non è paragonabile a quello ottenuto negli Stati Uniti. Il prezzo europeo dell'apparecchio, comprensivo di monitor e di unità a dischetti, si è dimostrato troppo elevato per l'utente medio di home computer e le interferenze della casa madre californiana sulla politica di commercializzazione adottata in Europa hanno in parte frenato il successo dell'Apple II come apparecchio gestionale e didattico.

L'ultima versione dell'Apple II è il IIc (dove *c* sta per "compact"). Questo apparecchio è più piccolo dei suoi predecessori (circa la metà) incorpora un'unità a dischetti da 5 pollici e 1/4, pesa 3,4 Kg e può quindi essere considerato un "portatile" utilizzabile tanto in ufficio quanto a casa propria. Il trasporto è reso più agevole da una piccola maniglia che viene estratta dal dorso dell'unità centrale. Il computer, inoltre, è dotato di connettori per collegarlo ad un monitor a video composito (o RGB) oppure ad un comune apparecchio televisivo. Infine, inclinando la maniglia verso il basso, si può far assumere alla macchina una confortevole angolazione di lavoro che, contemporaneamente, favorisce la circolazione dell'aria al suo interno.

Diversamente dalle altre versioni Apple II, il IIc è un sistema chiuso, privo cioè di connettori d'espansione, poiché all'interno dell'apparecchio sono già presenti le principali periferiche. Queste comprendono due porte d'interfaccia per monitor e per televisore, una porta per joystick (alla quale può essere anche collegato un "mouse" opzionale), una porta per modem ed una per stampante. Esistono inoltre, un'uscita audio ed una porta d'interfaccia per una seconda unità a dischetti. Le diverse porte sono contrassegnate da "icone", piccole figure stilizzate che ne indicano la funzione. Il IIc visualizza inoltre uno schermo ad 80 colonne e possiede una RAM di 128 Kbyte. Molte di queste caratteristiche erano opzionali sul IIe ed implicavano l'acquisto di almeno tre schede d'espansione.

L'Apple IIc possiede una tastiera di tipo QWERTY con 63 tasti, símile a quella del IIe. Il tasto di Reset è stato però trasferito sopra l'angolo sinistro della tastiera ed al suo fianco si trovano due piccoli interruttori: quello di sinistra seleziona gli schermi a 40 e 80 colonne. Il manuale d'uso consiglia di impiegare il video a 40 colonne quando si lavora con un apparecchio televisivo e quello a 80 colonne con un monitor (parte del software Apple visualizza soltanto su 40 colonne, indipendentemente dalla posizione dell'interruttore). Con l'interruttore di destra si seleziona-

Il portatile IIc

Il IIc, l'ultimo apparecchio creato dalla Apple, è una versione portatile dell'Apple II. Ha 128 Kbyte di RAM, un formato video di 80 colonne ed incorpora numerose interfacce oltre ad un'unità a dischetti.
Nell'illustrazione è collegato al monitor opzionale a fosfori verdi e al mouse



CHRIS STEVENS

no il set di caratteri Europeo o quello Americano (riprodotto sulla tastiera). Questa possibilità di scelta permette di utilizzare con facilità i caratteri presenti in una sola delle due versioni (ad esempio, il carattere # che nella versione europea è sostituito dal simbolo della Sterlina). Sopra l'angolo destro della tastiera ci sono due spie luminose: una indica che la macchina è accesa, l'altra che l'unità a dischetti è in funzione.
Quando si accende l'apparecchio, l'unità a dischetti entra automaticamente in funzione e interrompe la sua rotazione solo quando viene inserito un dischetto o quando vengono battuti contemporaneamente i tasti Reset e Control. Se non vengono introdotti dischetti, il IIc carica dalla ROM il BASIC Applesoft che è praticamente identico a quello dell'Apple II+ e presenta solo piccole variazioni rispetto alle prime versioni del BASIC Microsoft standard. L'impiego dell'Applesoft consente al IIc di utilizzare i programmi scritti per le precedenti versioni dell'apparecchio. Sfortunatamente. però, questo BASIC è privo di molti comandi presenti invece nei dialetti più avanzati (quale quello disponibile sul micro BBC). Per esempio, manca l'istruzione RANDOMIZE, l'opzione AUTO per la numerazione automatica delle linee, la struttura IF...THEN...ELSE ed il comando WHILE. Per quanto riguarda la grafica, sono assenti i comandi, ormai divenuti fondamentali, CIRCLE e PAINT.
Quando nel drive viene inserito un dischetto, il controllo del IIc passa al DOS 3.3 (dischetti Apple II+ e IIe) oppure al PRODOS, il nuovo sistema operativo della Apple. Quest'ultimo deriva dal sistema operativo appositamente progettato dalla Apple per il suo primo apparecchio gestionale, l'Apple III. Il PRODOS è caratterizzato da un rigido sistema "ad albero" nella creazione e nella gestione dei file. I file vengono archiviati seguendo la stessa tecnica usata per immagazzinare pratiche in uno schedario tradizionale. Pertanto, tutti i file che si riferiscono ad uno stesso progetto identificato da una qualsiasi sigla, per esempio ZED, possono essere immagazzinati su disco sotto l'unica intestazione ZED. I file di contabilità relativi al progetto ZED (Uscite, Vendite, Entrate ecc.) possono essere raccolti in un unico gruppo, magari chiamato "Conti". In un comune sistema di gestione "manuale", quando si vuol lavorare sulla pratica "Vendite", si deve innanzi tutto aprire la pratica principale ZED, poi aprire la pratica "Conti" ed, infine, tirar fuori la pratica "Vendite". Con il sistema operativo PRODOS, tutte queste operazioni vengono sintetizzate in un comando "di percorso" (pathname), che contiene, nel corretto ordine, l'elenco dei nomi di file che devono essere aperti in successione. I nomi dei file devono essere separati nel modo seguente:

/ZED/CONTI/VENDITE/

Un nome di percorso può contenere fino a 64 caratteri. Questa tecnica può inizialmente creare qualche difficoltà, dovuta soprattutto all'inesperienza ma, con l'andar del tempo, facilita l'organizzazione e la gestione dei file su disco. Un sistema simile, con una gestione dei file "ad albero", è impiegato nell'MS/DOS, il sistema operativo per il PC della IBM. Anche il sistema di visualizzazione del IIc è un aggiornamento delle versioni precedenti. Oltre alle due opzioni per i testi (24 linee x 40 o 80 colonne), il IIc possiede tre modi grafici: 40 x 40 (Bassa Risoluzione), 280 x 192 (Alta Risoluzione) e 560 x 192 (detto Alta Risoluzione Doppia), tutti a 16 colori.

PARLANDO DELLA APPLE ...



CHRIS STEVENS

L'Apple III

Il Lisa

Rendendosi conto dell'importanza del settore dei computer gestionali, la Apple ha concepito l'Apple III, un brillante computer da ufficio, dotato di due chip 6502, necessari per la gestione di 512 Kbyte di memoria. Il sistema operativo adottato dall'Apple III è il SOS (Sistema Operativo Sofisticato) che permette alla macchina di comunicare con unità a dischi rigidi e di organizzare i file con una struttura gerarchica. La stessa struttura "ad albero" è stata utilizzata anche per il sistema operativo del Lisa ed alcune caratteristiche sono comuni a quelle del sistema MS/DOS, usato sul PC della IBM. Sfortunatamente, la popolarità dell'Apple III ha avuto una partenza molto difficile, a causa di alcuni inconvenienti operativi. La Apple ha ritirato e sostituito tutti gli apparecchi difettosi, ma ciò non è valso ad eliminare la cattiva reputazione venutasi a creare. Come risultato, la produzione dell'Apple III è cessata.
Il Lisa, lanciato nel 1983, è stato il primo personal computer a grande diffusione ad usare software integrato con uso di "finestre", immagini (icone) al posto dei consueti comandi e ad operare tramite il mouse, un dispositivo insolito. Il lancio del Lisa provocò un tale entusiasmo, che le azioni della Apple salirono vertiginosamente in pochi mesi.
Nonostante ciò, il Lisa è stato quasi un secondo fallimento. Il prezzo troppo alto e, forse, l'aver orientato il prodotto verso una fascia di mercato non ideale, ha ridimensionato le aspettative iniziali (assieme alle quotazioni in borsa). Il Lisa è stato sostituito da una versione più economica: il Lisa II.
L'apparecchio che invece ha riscosso vasti consensi è stato il Macintosh, che forse ha permesso alla Apple di recuperare in parte le perdite subite con l'Apple III ed il Lisa. Anche il più recente IIc, con la sua immagine di "trasportabile" sembra destinato ad un successo piuttosto ampio

Inizialmente, è previsto soltanto un monitor a fosfori verdi, ma in seguito dovrebbe essere disponibile uno schermo piatto a cristalli liquidi (LCD), fabbricato dalla Sharp, con un display di 24 linee x 80 colonne. Un'altra novità dovrebbe essere il sistema di alimentazione a pile, che trasforma il IIc in un vero e proprio computer portatile.

Uno dei punti di forza del IIc è la sua immensa libreria software. Infatti, pare che per la gamma Apple II siano stati scritti oltre 17.000 programmi, alcuni dei quali altamente specialistici. Purtroppo, molti di questi programmi sono disponibili solo sul mercato americano. Il software di questa vasta gamma per il IIc comprende alcuni dei migliori programmi di gioco esistenti al mondo, tra i quali: il *Chess 7.0*, lo *Zork*, il *Simulatore di volo* della Microsoft ed il *Pinball Construction Set* (un programma per la costruzione sullo schermo di un "flipper"). Inoltre: una vasta gamma di word processor, database e lavagne elettroniche, programmi di contabilità, programmi grafici, di controllo per laboratori scientifici e anche didattici (da semplici esercizi di lettura per bambini a programmi di calcolo avanzato).

Oltre al software esistente per i sistemi II e IIe, l'Apple ha sviluppato un programma chiamato *Appleworks* che integra l'uso di "finestre" con le funzioni di word processor, database e lavagna elettronica. Il pacchetto di dischi che corredano il IIc comprende un dischetto di introduzione all'*Appleworks* (che non è però il vero dischetto di lavoro, che viene venduto a parte), un'introduzione simile alla precedente, ma per il LOGO della Apple, l'*Apple Presents Apple* (un'introduzione interattiva al sistema di base), un dischetto di introduzione alla programmazione BASIC ed un altro contenente i programmi di utilità del sistema operativo PRODOS. Infine, la Apple offre come pacchetto opzionale il MousePaint, un sistema basato sul MacPaint e composto da un mouse e da un programma che ne prevede l'impiego grafico. La documentazione di corredo al IIc comprende un breve opuscolo per l'installazione del sistema, un manuale a colori di oltre 100 pagine, che descrive in modo chiaro e sintetico le funzioni di sistema e l'uso dei 5 dischetti che arrivano assieme all'apparecchio.

La Apple ha prestato particolare attenzione all'estetica della macchina: è infatti passata dal colore beige, adottato per il II, il Lisa ed il Macintosh, al bianco brillante. Alcune fessure (visibili sulla parte superiore della custodia) permettono una miglior circolazione dell'aria all'interno dell'apparecchio.

L'Apple IIc, come i suoi predecessori, si dimostra un ottimo apparecchio da ufficio e, grazie al nuovo schermo piatto a LCD e all'alimentazione a batteria di prossima introduzione, può oggi essere considerato un ottimo compagno di lavoro portatile.



Il Disk Pack del IIc

Il IIc è venduto con 5 dischetti di corredo. Quattro sono di introduzione all'uso dell'apparecchio, alla programmazione BASIC e ai programmi applicativi (acquistabili a parte). Il quinto contiene il sistema operativo PRODOS, il nuovo sistema per la gestione di file, sviluppato dalla Apple per la gamma di apparecchi Apple II



Uscita per video composito

Porta RS232 per stampante

Alimentazione

Il IIc prevede un sistema di alimentazione interna a 12 volt: ha quindi bisogno di un trasformatore per la corrente di rete a 220 volt

Uscita video RGB

Innestando in questa porta un piccolo adattatore PAL, è possibile collegare il IIc ad un comune televisore

Chip di controllo di I/O

Questi chip controllano la tastiera e le porte di input e di output

Presa per l'audio

Permette di collegare il IIc ad un amplificatore HI-FI esterno



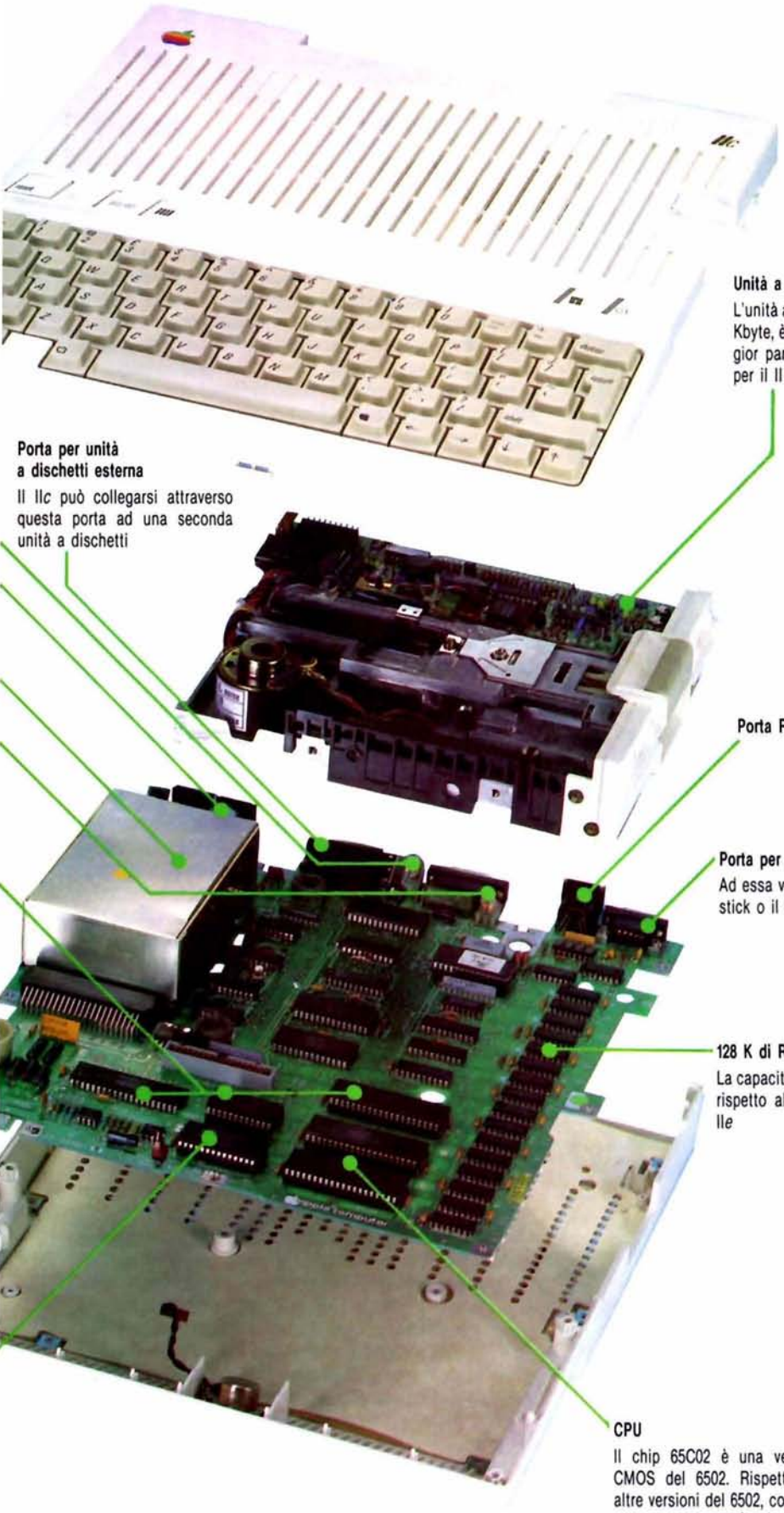
Il "MousePaint"

Il "MousePaint" è un pacchetto, composto da un mouse e da un software grafico, disponibile per i modelli II+, IIe e IIc. Il programma, basato sul "MacPaint", è una versione ridotta del software per il Macintosh. Muovendo il mouse su di un piano d'appoggio, si possono tracciare accurati disegni sullo schermo ad alta risoluzione del IIc. Per poter usare il mouse sui computer della serie Apple II, occorre acquistare un'interfaccia aggiuntiva ed inserirla in una delle due porte d'espansione



ROM

Contiene il Basic Applesoft ed i sottoprogrammi di servizio (ossia le routine di sistema)



Unità a dischetti
L'unità a dischetti interna, da 143 Kbyte, è compatibile con la maggior parte dei dischetti prodotti per il II+ ed il IIe

Porta per unità a dischetti esterna
Il IIc può collegarsi attraverso questa porta ad una seconda unità a dischetti

Porta RS232 per modem

Porta per il controllo manuale
Ad essa vengono collegati i joystick o il mouse

128 K di RAM utente
La capacità di memoria è doppia rispetto alla RAM standard del IIe

CPU
Il chip 65C02 è una versione CMOS del 6502. Rispetto alle altre versioni del 6502, consuma meno corrente e può quindi essere alimentato anche a pile

APPLE IIc

DIMENSIONI

mm. 305 x 292 x 64 (tastiera)

CPU

6502, funzionante a 1 MHz

MEMORIA

128 Kbyte di RAM, 16 Kbyte di ROM

VIDEO

24 righe di 40 oppure 80 caratteri. Tre modi grafici con massima risoluzione di 560 x 192 pixel, in 16 colori

INTERFACCE

Porta joystick a 9 piedini (utilizzabile anche per l'apposito mouse), interfaccia RS232 per modem, uscita RGB, uscita video composito (PAL per televisorre tramite adattatore a parte), porta per unità a dischetti esterna, porta RS232 per stampante

LINGUAGGI FORNITI

BASIC Applesoft (residente in ROM), LOGO, PASCAL, FORTRAN

TASTIERA

Tipo QWERTY, con 63 tasti di cui 4 per il cursore. Set di caratteri internazionale

DOCUMENTAZIONE

Il manuale d'accompagnamento, vivacemente colorato, è di facile comprensione: è molto adatto anche ai meno esperti. Vengono forniti anche due manuali (uno per l'installazione dell'apparecchio ed uno per spiegare l'uso dei programmi di sistema)

PRO

Il maggior punto di forza dell'Apple IIc consiste nella compatibilità software con i modelli della gamma Apple II (oltre 17.000 programmi disponibili)

CONTRO

Il BASIC della Applesoft, migliorato ben poco in più di sei anni di vita, comincia a dare segni d'invecchiamento: manca di flessibilità. Il prezzo, relativamente alto, può essere un fattore deterrente

Un generatore di caratteri

Un programma per il Commodore 64, per definire un 'set' di caratteri

Le caratteristiche grafiche e audio del Commodore 64 sono di ottimo livello, ma il BASIC è privo di istruzioni specifiche per sfruttarle appieno. Il BASIC dello Spectrum, ad esempio, possiede comandi quali BEEP, DRAW, INK e PAPER, mentre sul Commodore 64 mancano totalmente istruzioni equivalenti a queste. Ecco perché molti programmi BASIC sviluppati per il Commodore 64 presentano grafica e suoni piuttosto primitivi e, anche quelli di miglior qualità, contengono numerose istruzioni DATA ed innumerevoli POKE, come nel listato alla pagina seguente. Il programma generatore di caratteri qui descritto facilita il processo di creazione di nuovi caratteri e permette di progettarli direttamente sullo schermo (elimina quindi la procedura d'inserimento in RAM tramite istruzioni POKE). I nuovi caratteri vengono quindi depositati automaticamente in memoria.

Le procedure usate per creare caratteri definiti dall'utente (UDG) sul Commodore 64 sono già state esaminate in dettaglio (vedere a pag. 712). Le principali operazioni preliminari vengono svolte dalla subroutine alla linea 61000. L'indirizzo iniziale della memoria utente viene "abbassato", spostandolo dalla locazione 40959 alla 14335. La serie completa delle maiuscole residenti nella ROM del Commodore 64 (2 Kbyte a partire dall'indirizzo 53248) viene trasferita in RAM, a partire dall'indirizzo 14336. Qui i caratteri possono essere raggiunti e modificati usando le solite PEEK e POKE. Infine, la subroutine abilita il VIC (chip d'interfaccia per il video) affinché utilizzi il set di caratteri appena ridefinito.



Caratteri ... alla finestra

La finestra di Edit (sulla sinistra) mostra la nuova versione del carattere corrispondente al codice video 55, ossia il numero 7. Dal carattere standard è stata eliminata la "terminazione" ed è stato aggiunto un trattino trasversale. Nella finestra testuale (sulla destra) sono visibili 4 copie a dimensione normale del nuovo 7. I cambiamenti apportati sono presenti anche in tutti gli altri 7 presenti sullo schermo, con esclusione di quelli visualizzati nel riquadro bianco, in cui si trova il carattere originale, prima dell'intervento



Chele di granchio

Gli sprite sono paragonabili ad un'insieme di caratteri multipli definiti dall'utente. Nella finestra di Edit della figura a fianco sono stati ridefiniti nove caratteri, poi trasferiti nella finestra testuale. La sezione "in lavorazione" corrisponde alla parte destra del granchio

Eseguita questa operazione, la routine d'inizializzazione apre due "finestre" sullo schermo e trasferisce il controllo alla routine di immissione, che inizia alla linea 2500. Questa "esplora" la tastiera ed accende un cursore lampeggiante nell'angolo superiore sinistro dello schermo, ossia all'interno della finestra di Edit. In essa viene visualizzato (ingrandito e posto all'interno di una griglia) il carattere da modificare, assieme ad un'indicazione numerica dei byte che lo compongono.

I tasti di funzione f1, f3, f5 ed f7 controllano i movimenti del cursore all'interno della finestra. La cella che si trova sotto il cursore (corrispondente ad un bit di uno degli otto byte che definiscono il carattere) può essere accesa o spenta (attivata o inattivata) usando il tasto di funzione f2. Quando ciò accade, tutti gli otto byte vengono aggiornati, e tutti i caratteri presenti sul video, con codice identico a quello del carattere "in lavorazione", vengono immediatamente modificati.

Il tasto di funzione f4 permette di inserire un nuovo carattere nella finestra di Edit. I caratteri sono definiti dai valori usati per le POKE: una tabella dei codici video è riportata in appendice al "Manuale per l'Utente" del Commodore 64. Tali valori (diversi dai codici CHR$, pur avendo certe corrispondenze) sono qui più convenienti, poiché i valori che definiscono i caratteri vengono organizzati in memoria seguendo l'ordine dei codici video.

Il tasto di funzione f6 consente di riprodurre, all'interno della finestra che si trova sulla destra dello schermo (detta "finestra testuale") una versione a dimensioni normali del carattere modificato. Il carattere compare nella posizione indicata dal cursore della finestra di Edit. Se, ad esempio, il cursore si trova nell'angolo in alto a sinistra ed il carattere è una "A", tale carattere, di dimensioni standard, compare nell'angolo in alto a sinistra della finestra testuale.

Infine, il tasto del punto esclamativo interrompe il programma, mentre CONT lo fa ripartire. Una volta usciti dal programma, tramite le opzioni Exit o Quit, si può tranquillamente impartire un NEW, seguito da una LOAD, per caricare in memoria un altro programma. Ciò non danneggia i caratteri appena modificati; tuttavia, questa operazione presenta alcuni problemi. Innanzi tutto l'indirizzo iniziale della memoria utente è stato abbassato, quindi il nuovo programma dispone solo di 12 Kbyte. Inoltre, spegnendo l'apparecchio, si perdono i nuovi caratteri. Entrambi i problemi verranno affrontati in seguito (vedere a pag. 1068). Limitiamoci a far esercizio con questo programma ed impieghiamolo come indicato dal programma di pagina 713.

IAN McKINNELL

DEFINIZIONE DI CARATTERI

```
  19 REM***********C64**************
  20 REM* GENERATORE DI CARATTERI *
  21 REM***********C64**************
  50 POKE 52,56:POKE 56,56:CLR
  60 PRINT CHR$(147)"ATTENDERE CIRCA 22
     SECONDI"
  70 GOSUB 61000: REM COPIA SET CARATT.
 100 GOSUB 1000: REM ROUTINE INIZZALIZZ.
 120 FOR CT=0 TO 1 STEP 0
 140 GOSUB 2500: REM IMMISSIONE
 160 GOSUB 3000: REM CONVALIDA IMMISSIONE
 180 ON PT GOSUB 3500,4000,4500,7000
 200 NEXT CT
 900 END
 999 REM*******************************
1000 REM* ROUTINE INIZIALIZZAZIONE *
1001 REM*******************************
1020 DIM BD(8,8),C$(2,2),OF(2,7),CX(4)
1040 SH$=CHR$(19):SC$=CHR$(147):R$=
     CHR$(18):N$=CHR$(146):CD$=CHR$(17)
1060 P$=CD$+CD$+CD$:P$=P$+P$+P$+P$:
     P$=P$+P$:P$=SH$+P$
1080 C1$=CHR$(144):C2$=CHR$(5)
1200 REM---INIZIALIZZA LO SCHERMO---
1210 S0=1024:PRINT SC$C1$
1220 R0=4:C0=3:RL=8:CL=8:OF=16
1230 Z$="CARATTERI DEFINITI DA UTENTE"
1240 C=4:GOSUB2100:R=24:GOSUB2100
1250 L$=" 76543210 ":S$="          "
1260 Z$=L$:R=R0:C=C0:GOSUB 2100
1270 C=C0-1:FOR R=R0+1 TO R0+8
1280 Z$=STR$(R-R0-1):Z$=Z$+S$+Z$
1290 GOSUB 2100:C=C+OF:GOSUB2100
1300 C=C-OF:NEXT R
1310 C=C0:Z$=L$:GOSUB2100
1320 L$=" 01234567 "
1330 Z$=L$:R=R0:C=C0+OF:GOSUB 2100
1350 C=C0+OF:R=R0+CL+1 :GOSUB2100
1370 PRINT C2$
1400 C$(1,1)=R$+" "+N$:C$(1,2)=" "
1410 C$(2,1)=R$+"*"+N$:C$(2,2)="*"
1420 REM---SPOSTAMENTI DEL CURSORE---
1440 DATA 0,-1,+1,0,-1,0,0,+1
1460 FOR K=1 TO 2:FOR L=1 TO 4:
1480 READ OF(K,L): NEXT L,K
1500 REM---CONVERSIONE DEL CARATTERE---
1520 DATA 64,0,32,64
1540 FOR K=0 TO 3:READ CX(K):NEXT K
1620 RP=1:CP=1:CN=1:GOSUB 6000
1990 RETURN
1999 REM*************************************
2000 REM* INSERISCE CURSORE IN POS. R,C *
2001 REM*************************************
2050 PRINT LEFT$ (P$,R+1)TAB(C);
2070 RETURN
2099 REM*******************************
2100 REM* VISUALIZZA Z$ IN POS. R,C *
2101 REM*******************************
2150 PRINT LEFT$(P$,R+1)TAB(C)Z$;
2170 RETURN
2499 REM*************************************
2500 REM* LAMPEGGIO CURSORE IN POS. RP,CP *
2501 REM*************************************
2520 CF=1+BD(RP,CP):R=RP+R0:C=CP+C0
2540 FOR LP=0 TO 1 STEP 0
2560 FOR CS=1 TO 2:DE=10:GOSUB 2800
2580 GET GT$
2600 IF GT$<>"" THEN LP=1:CS=2
2620 Z$=C$(CF,CS):GOSUB 2100
2640 DE=10:GOSUB 2800
2660 NEXT CS,LP:RETURN
2799 REM*************************************
2800 REM* CICLO RITARDO (VALORE IN DE) *
2801 REM*************************************
2820 FOR NN=1 TO DE:NEXT:RETURN
2999 REM*******************************
3000 REM* CONVALIDA DELL'IMMISSIONE *
3001 REM*******************************
3020 IF GT$="!" THEN R=18:C=0:
     GOSUB2000:STOP
3040 GT=ASC(GT$)-132:F=2*INT(GT/2)
3060 IF (GT<1)OR (GT>8)  THEN PT=0:
     RETURN
3080 IF GT<5 THEN PT=1:RETURN
3100 PT=GT-3
3490 RETURN
3499 REM************************
3500 REM* SPOSTAMENTO CURSORE *
3501 REM************************
3520 NY=RP+OF(2,GT):NX=CP+OF(1,GT)
3540 IF  (NY<1)OR(NY>RL)  THEN RETURN
3560 IF  (NX<1)OR(NX>CL)  THEN RETURN
3580 RP=NY:CP=NX
3620 RETURN
3999 REM*****************************
4000 REM* ATTIVAZ.UNA CELLA VIDEO *
4001 REM*****************************
4020 TG=1-BD(RP,CP):Z$=C$(1+TG,2)
4040 R=R0+RP:C=C0+CP:GOSUB2100
4060 BD(RP,CP)=TG:MP=NCGEN+CN*8-1
4120 PE=PEEK(MP+RP)
4140 PE=PE+(TG*2-1)*(2^(8-CP))
4160 POKE (MP+RP),PE:GOSUB 6500:
     RETURN
4499 REM*******************************
4500 REM* DEFINISCE NUOVO CARATTERE *
4501 REM*******************************
4520 FOR K=1 TO 1
4540 Z$=R$+" CAMBIO CARATTERE "
4550 R=14:C=7:GOSUB2100
4560 Z$=" NUOVO NUMERO "
4570 R=15:GOSUB2100
4580 C=19:GOSUB2000:INPUT A$
4600 CN=VAL(A$):PRINT C2$
4620 IF (CN<0)OR(CN>127) THEN K=0
4640 NEXT K:GOSUB 6000
4660 Z$="                ":C=7
4670 R=14:GOSUB2100:R=15:GOSUB2100
4680 RETURN
5999 REM*************************
6000 REM* VISUALIZZA CARATTERE *
6001 REM*************************
6020 MP=NCGEN+CN*8-1
6040 FOR RP=1 TO 8:PE=PEEK(MP+RP):
     Z$=""
6060 FOR CP=8 TO 1 STEP-1:
     N=INT(PE/2)
6080 Q=PE-2*N:BD(RP,CP)=Q:PE=N
6100 Z$=C$(Q+1,2)+Z$:NEXT CP
6120 R=R0+RP: C=C0+1:GOSUB2100
6130 NEXT RP
6140 X$=CHR$(CN+CX(INT(CN/32)))
6150 Z$=R$+"CARATTERE":R=1:C=11
6160 GOSUB2100:R=R+2:GOSUB2100
6170 Z$=" "+X$+" =        "
6180 R=R-1:GOSUB2100
6190 C=C+4:Z$=STR$(CN)+N$:GOSUB2100
6220 RP=1:CP=1
6490 GOSUB6500:RETURN
6499 REM*************************
6500 REM* VISUALIZZA BYTE *
6501 REM*************************
6540 C=C0+CL+2:FOR R=R0+1 TO R0+8
6560 Z$=STR$(PEEK(MP+R-R0))+"  "
6580 GOSUB2100:NEXT:RETURN
6999 REM*****************************
7000 REM* POSIZIONA UN CARATTERE *
7001 REM*****************************
7020 Z$=X$:C=C+OF:GOSUB2100:RETURN
60999 REM*******************************
61000 REM* TRASFERISCE SET CARATTERI *
61001 REM*******************************
61100 CGEN=53248:NCGEN=14336
61120 ITRPT=56334:IOPT=1:PO=53272
61125 RETURN
61150 POKE IT,PEEK(IT)AND254
61200 POKE IO,PEEK(IO)AND251
61250 FOR J=0 TO 2047
61300 POKE (NCGEN+J),PEEK(CGEN+J)
61350 NEXT J
61400 POKEIO,PEEK(IO)OR4
61450 POKE IT,PEEK(IT)OR1
61500 POKE PO,(PEEK(PO)AND240)OR14
61990 RETURN
```

Dispositivi di controllo

La realizzazione di un'interfaccia per il controllo di piccoli motori e lampadine



IAN McKINNELL

L'interfaccia

Sulla scatola dell'interfaccia sono visibili la presa minicon ed i cavetti di ingresso/uscita. Le misure della piastra devono corrispondere esattamente alla dimensione della scatola e la presa minicon deve inserirsi saldamente nella fessura laterale, cosicché quando viene inserito il connettore, la piastra non si muoverà nella scatola

Questa interfaccia permette al nostro computer di controllare periferiche a basso voltaggio e a basso assorbimento. L'output proveniente da ciascuno dei quattro bit della porta utente del computer viene temporaneamente immagazzinato nello stesso chip usato per la costruzione del buffer box (vedere a pag. 1003). Il segnale proveniente da questo chip viene usato per commutare un transistor di potenza, in grado di controllare correnti di maggior ampiezza di quelle applicabili al buffer. La particolare disposizione dei transistor (in elettronica chiamata "complementare") consente sia di alimentare un carico sia di assorbire le correnti di drenaggio del carico applicato. I due diodi in uscita proteggono i transistor da eventuali correnti di ritorno, spesso generate da carichi induttivi, come nel caso di relais o di motorini elettrici. L'interfaccia può essere usata per il controllo bidirezionale di un motore, i cui terminali vanno collegati al connettore di uscita ed a quello di terra. Quando il computer trasmette un segnale di livello logico 1, il motore si mette in moto. Quando, al contrario, il segnale di output è zero, il motore si ferma. Collegando il motore alle due uscite dell'interfaccia, possiamo controllare la direzione della rotazione. Se il computer invia segnali di output uguali (entrambi zero o entrambi uno) sulle uscite dell'interfaccia compare lo stesso voltaggio ed il motore non riceve corrente. Segnali di output diversi tra loro (zero e uno) producono invece una differenza di voltaggio ed il motore gira in una direzione; invertendo i segnali di output (uno e zero) cambia la direzione del flusso della corrente e, di conseguenza, quella della rotazione del motore.

COSTRUZIONE DELL'INTERFACCIA

Innanzi tutto fabbrichiamo la scatola che contiene l'interfaccia. La scheda dei circuiti deve avere le stesse dimensioni della scatola usata nel progetto del buffer box. La scatola va forata per accogliere le prese rosse e nere e quella del bus (e, se necessario, anche la presa minicon).

Le prese nere (massa) vengono collegate tra loro con un ponticello ben stagnato. Quindi si prenda un cavo piatto a 9 linee e si saldi uno dei nove cavetti al ponticello che collega le prese di massa. Gli altri otto capi, uniti a coppie, vanno collegati alle quattro prese di uscita (rosse).

Prendiamo adesso una piastrina di "veroboard" di dimensioni adeguate (45 fori x 16 piste) e rimuoviamo metà riga di fori ad una delle estremità (si conservi la parte rimossa, che verrà in seguito usata per la costruzione di un'altra interfaccia). Si interrompano le piste ramate come mostrato nell'illustrazione A.

Prima vanno saldati i componenti inerti: lo zoccolo del chip, la presa per il bus ed i ponticelli di filo stagnato. Volendo applicare anche la presa del bus d'espansione, la si può fissare adesso, ma si deve attendere la fine del lavoro per innestare il cavo che la collega al connettore del bus.

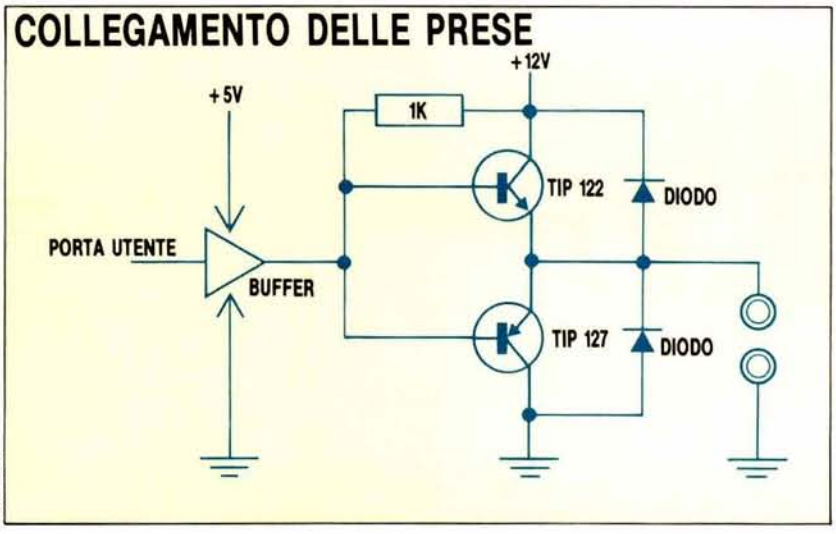
Si saldino adesso le resistenze, i diodi e gli 8 transistor, facendo attenzione a inserire questi componenti sulla pista giusta. Si saldino poi i fili di collegamento per le prese come mostrato nell'illustrazione B si inserisca il chip sullo zoccolo. Infine, si inserisca la scheda nella scatola: l'interfaccia è pronta per l'uso

LISTA DEI COMPONENTI

Quantità	Articolo
1	Veroboard a 50 fori x 16 piste
4	Resistenze da 1 Kohm
8	Diodi 1N4001
4	Transistor TIP 122
4	Transistor TIP 127
1	Circuito integrato 7407
1	Presa DIL a 14 piedini
4	Prese rosse da 4 mm.
4	Prese nere da 4 mm.
4	Spinotti rossi da 4 mm.
4	Spinotti neri da 4 mm.
1	Connettore minicon a 12 canali
1	Scatola, dimensioni 120 x 65 x 40 mm.
1	Presa minicon a 12 canali*

* L'ultimo articolo è opzionale: eventualmente serve per collegare contemporaneamente più di un'interfaccia. Dopo la realizzazione della precedente interfaccia, dovrebbero essere avanzati alcuni fili stagnati e qualche spezzone di cavo a nastro che possono tornare utili (mai gettare niente in elettronica: le minuterie fanno sempre comodo!)

COLLEGAMENTO DELLE PRESE

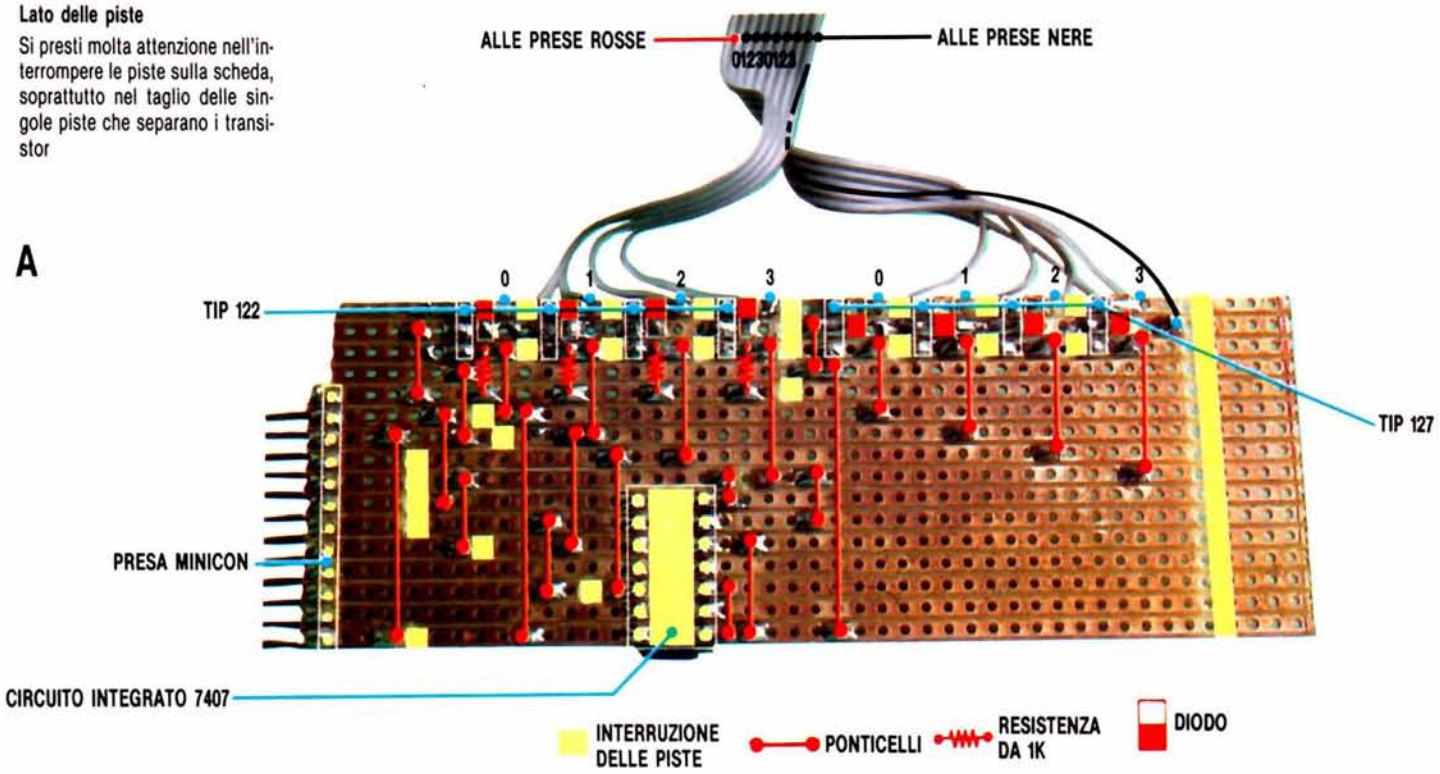


LIZ DIXON

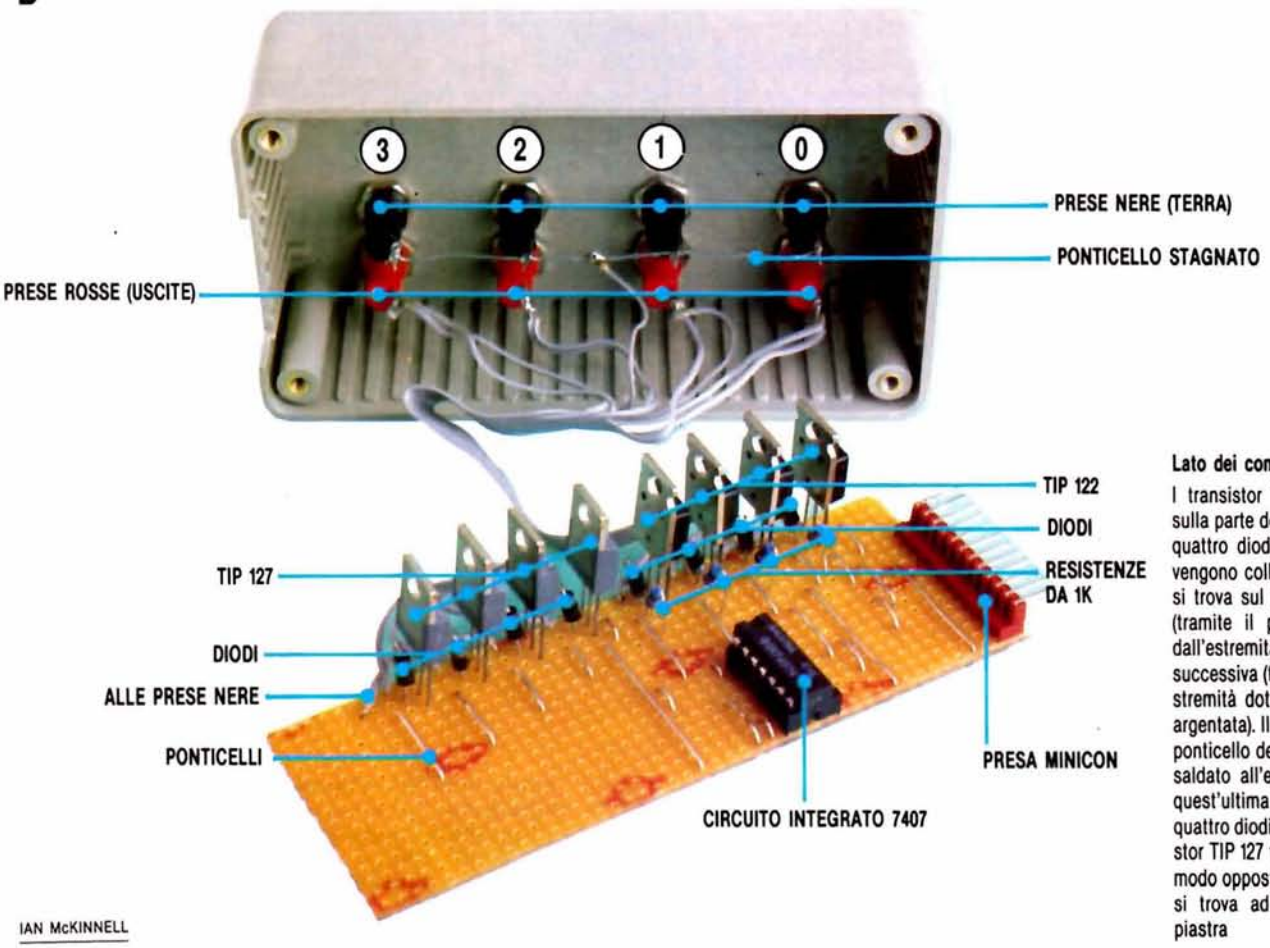
Lato delle piste

Si presti molta attenzione nell'interrompere le piste sulla scheda, soprattutto nel taglio delle singole piste che separano i transistor

A



B



Lato dei componenti

I transistor TIP 122 si trovano sulla parte destra della scheda. I quattro diodi, posti tra di essi, vengono collegati alla pista che si trova sul bordo della scheda (tramite il piedino proveniente dall'estremità nera) ed alla pista successiva (tramite quello dell'estremità dotata di una fascetta argentata). Il cavetto collegato al ponticello delle prese nere viene saldato all'estremità sinistra di quest'ultima traccia. Gli altri quattro diodi disposti tra i transistor TIP 127 vengono collegati in modo opposto. La presa minicon si trova ad un'estremità della piastra

IAN McKINNELL

DATABASE

La seconda di due tabelle, dedicate al codice macchina del chip 6809

Legenda:

- OP Codice Operativo (in esadecimale)
- ~ Numero di cicli macchina
- # Numero di byte di programma
- + Segno aritmetico di somma (numero positivo)
- – Segno aritmetico di sottrazione (numero negativo)
- • Segno aritmetico di moltiplicazione
- $\overline{M}$ Complemento di M
- → Trasferimento in
- H Half-Carry (semi riporto), dal bit 3
- N Negativo (bit del segno)
- Z Zero (corrispondente a stato "reset")
- V Overflow, in complemento a 2
- C Carry (bit del riporto)
- ↕ Esaminato e posto ad 1 se la condizione è vera, altrimenti posto a 0
- • Inalterato dall'operazione
- CC Registro di Condizione del Codice
- : Concatenazione
- v Operatore OR logico
- ∧ Operatore AND logico
- ⊻ Operatore OR Esclusivo

		Modi d'indirizzamento																5	3	2	1	0
		Immediato			Diretto			Indicizzato(1)			Esteso			Inerente				H	N	Z	V	C
Istruzione	Formato	Op	~	#	OP	~	#	OP	~	#	OP	~	#	OP	~	#	Descrizione					
LSL	LSLA													48	2	1	A	•	↕	↕	↕	↕
	LSLB													58	2	1	B	•	↕	↕	↕	↕
	LSL				08	6	2	68	6+	2+	78	7	3				M	•	↕	↕	↕	↕
LSR	LSRA													44	2	1	A	•	0	↕	•	↕
	LSRB													54	2	1	B	•	0	↕	•	↕
	LSR				04	6	2	64	6+	2+	74		3				M	•	0	↕	•	↕
MUL														3D	11	1	A × B → D (senza segno)	•	•	↕	•	(5)
NEG	NEGA													40	2	1	$\overline{A}+1 \rightarrow A$	(4)	↕	↕	↕	↕
	NEGB													50	2	1	$\overline{B}+1 \rightarrow B$	(4)	↕	↕	↕	↕
	NEG				00	6	2	60	6+	2+	70	7	3				$\overline{M}+1 \rightarrow M$	(4)	↕	↕	↕	↕
NOP														12	2	1	nessuna operazione	•	•	•	•	•
OR	ORA	8A	2	2	9A	4	2	AA	4+	2+	BA	5	3				A v M → A	•	↕	↕	0	•
	ORB	CA	2	2	DA	4	2	EA	+	2+	FA	5	3				B v M → B	•	↕	↕	0	•
	ORCC	1A	3	2													CC v IMM → CC				(3)	
PSH	PSHS	34	5+(6)	2													Deposita i registri sullo stack S	•	•	•	•	•
	PSHU	36	5+(6)	2													Deposita i registri sullo stack U	•	•	•	•	•
PUL	PULS	35	5+(6)	2													Recupera i registri dallo stack S	•	•	•	•	•
	PULU	37	5+(6)	2													Recupera i registri dallo stack U	•	•	•	•	•
ROL	ROLA													49	2	1	A	•	↕	↕	↕	↕
	ROLB													59	2	1	B	•	↕	↕	↕	↕
	ROL				09	6	2	69	6+	2+	79	7	3				M	•	↕	↕	↕	↕
ROR	RORA													46	2	1	A	•	↕	↕	•	↕
	RORB													56	2	1	B	•	↕	↕	•	↕
	ROR				06	6	2	66	6+	2+	76	7	3				M	•	↕	↕	•	↕
RTI														3B	6/15	1	Ritorno da una interrupt					(3)
RTS														39	5	1	Ritorno da una subroutine	•	•	•	•	•
SBC	SBCA	82	2	2	92	4	2	A2	4+	2+	B2	5	3				A – M – C → A	(4)	↕	↕	↕	↕
	SBCB	C2	2	2	D2	4	2	E2	4+	2+	F2	5	3				B – M – C → B	(4)	↕	↕	↕	↕
SEX														1D	2	1	Estensione del segno da B in A	•	↕	↕	0	•
ST	STA				97	4	2	A7	4+	2+	B7	5	3				A → M	•	↕	↕	0	•
	STB				D7	4	2	E7	4+	2+	F7	5	3				B → M	•	↕	↕	0	•
	STD				DD	5	2	ED	5+	2+	FD	6	3				D → M:M+1	•	↕	↕	0	•
	STS				10	6	3	10	6+	3+	10	7	4				S → M:M+1	•	↕	↕	0	•
					DF			EF			FF											
	STU				DF	5	2	EF	5+	2+	FF	6	3				U → M:M+1	•	↕	↕	0	•
	STX				9F	5	2	AF	5+	2+	BF	6	3				X → M:M+1	•	↕	↕	0	•
	STY				10	6	3	10			10	7	4				Y → M:M+1	•	↕	↕	0	•
					9F			AF	6+	3+	BF											
SUB	SUBA	80	2	2	90	4	2	A0	4+	2+	B0	5	3				A – M → A	(4)	↕	↕	↕	↕
	SUBB	C0	2	2	D0	4	2	F0	4+	2+	E0	5	3				B – M → B	(4)	↕	↕	↕	↕
	SUBD	83	4	3	93	6	2	A3	6+	2+	B3	7	3				D – M:M+1 → D	•	↕	↕	↕	↕
SWI	SWI(7)													3F	19	1	Interrupt di software 1	•	•	•	•	•
	SWI2(7)													10 3F	20	2	Interrupt di software 2	•	•	•	•	•
	SWI3(7)													11 3F	20	1	Interrupt di software 3	•	•	•	•	•
SYNC														13	≥4	1	Sincronizzazione con l'interrupt	•	•	•	•	•
TFR	R1, R2	1F	6	2													R1 → R2(2)	•	•	•	•	•
TST	TSTA													4D	2	1	Verifica su A	•	↕	↕	0	•
	TSTB													5D	2	1	Verifica su B	•	↕	↕	0	•
	TST				0D	6	2	6D	6+	2+	7D	7	3				Verifica su M	•	↕	↕	0	•

Note:

(1) Questa colonna indica il numero di cicli ed il numero di byte dell'istruzione di base: per ottenere il conteggio totale, occorre sommare i valori che si ottengono dalla sezione MODO D'INDIRIZZAMENTO INDICIZZATO

(2) R1 ed R2 possono essere una qualsiasi coppia di registri a 8 bit, oppure a 16 bit.
I registri a 8 bit sono: A, B, CC e DP
I registri a 16 bit sono: X, Y, U, S, D e PC

(3) I codici di condizione sono azzerati o posti a 1, a seconda del risultato dell'operazione

(4) Il valore del flag "half-carry" (semi-riporto) è indefinito

(5) Caso Speciale – il flag di Carry è posto a 1 se il bit 7 (b7) vale 1

(6) Le istruzioni PSH e PUL richiedono 5 cicli, più un ciclo per ogni byte appoggiato o prelevato dallo stack

(7) SWI pone ad 1 i bit I e F. SWI2 e SWI3 non hanno alcun effetto su I ed F

per concessione della Motorola Inc.

L'Aritmetica del 6809

Calcoli in codice macchina e concetti relativi ai numeri binari negativi

Esaminiamo adesso una serie di nuove istruzioni in codice macchina ed alcune tecniche per rappresentare i dati che permettono di scrivere un programma capace di trasformare un "numero decimale in codice binario (BCD)" nella sua rappresentazione binaria.
La numerazione decimale in codice binario BCD è di grande utilità quando si lavora con processori a 8 bit. Con questo sistema, ciascuna cifra del numero decimale viene "tradotta" nel suo equivalente binario. Per esempio, al numero 69 corrisponde la rappresentazione BCD %01101001: i primi quattro bit (0110) sono l'equivalente binario del decimale 6, i secondi quattro (1001) del decimale 9. Usando la numerazione BCD, otteniamo quindi un equivalente decimale completamente diverso da quello che otterremmo convertendo il numero "binario" %01101001 (che, preso nella sua forma integrale, equivale al decimale 105). Vediamo adesso come funzionano le nuove istruzioni in codice macchina, impiegate nel programma di conversione:

• LSR (Logical Shift Right): fa scorrere di una posizione a destra ciascun bit dell'operando. Il bit che si trova all'estremità destra viene trasferito nel bit di riporto (Carry) del registro di condizione, mentre nel primo bit a sinistra dell'operando viene inserito uno zero.

• AND: Pone ciascun bit del registro in AND logico rispetto ai corrispondenti bit dell'operando e copia il risultato nel registro. Questa istruzione viene spesso usata per "mascherare" specifici bit: se un bit del registro contiene il valore uno, ponendolo in AND con un bit dell'operando si ottiene il trasferimento di quest'ultimo bit nel registro. Se invece il bit del registro contiene uno zero, il risultato ottenuto con l'operazione AND è sempre zero. Ad esempio, ponendo in AND il valore %00001111 contenuto nel registro con una particolare locazione di memoria, si ottiene il trasferimento nel registro dei quattro bit a destra (1111) della locazione. Quindi:

%00001111 Valore del registro
%10110110 Valore della locazione di memoria posta in AND
%00000110 Risultato finale nel registro.

• MUL: moltiplica i contenuti dei registri A e B e copia il risultato nel registro D (il registro a 16 bit composto dai due registri A e B usati assieme). Pochi altri processori ad 8 bit possiedono un op-code per la moltiplicazione.

• SWI (Software Interrupt): questa istruzione, che esamineremo in modo più dettagliato quando ci occuperemo del sistema d'interruzione, viene usata per porre fine ad un programma in codice macchina e trasferire il controllo al sistema operativo.

Descriviamo adesso il programma di conversione da BCD in Binario:

• Inizializzazione del valore contenuto nel contatore di locazione:

```
ORG $10
```

• Immagazzina il valore 58 (BCD) in BCDNUM, riserva un byte alla locazione BINNUM:

```
BCDNUM FCB %01011000
BINNUM RMB 1
```

• Carica il valore 58 (BCD) nell'accumulatore A, maschera la cifra contenuta nel semibyte basso (gli ultimi 4 bit) e la deposita in BINNUM:

```
START LDA BCDNUM
      ANDA #%00001111
      STA BINNUM
```

• Carica 58 (BCD) nell'accumulatore A e fa scorrere verso destra di una posizione la cifra contenuta nel semibyte alto (i primi 4 bit):

```
SHIFT LSRA
      LSRA
      LSRA
      LSRA
```

• Carica 10 (decimale) nel registro B e lo moltiplica per il contenuto del registro A:

```
MULT LDB  #10
     MUL
```

• Il risultato viene copiato nel registro D a 16 bit di cui viene impiegato solo il byte più basso, in quanto il risultato non può essere maggiore di 90 (10x9=90). Dato che il byte più basso del registro D corrisponde al registro B, sommiamo il contenuto di questo registro a BINNUM e depositiamo in BINNUM stesso il risultato:

```
ADDIT ADDB BINNUM
      STB BINNUM
```

• Come risultato si ottiene che il numero BCD è depositato in BCDNUM ed il suo equivalente binario in BINNUM. Trasferiamo infine il controllo al sistema operativo e terminiamo (END) il codice sorgente:

```
RETURN SWI
END
```

Gli esempi di programmazione sin qui descritti hanno coinvolto soltanto operazioni aritmetiche semplici. Affrontiamo adesso i problemi legati al "segno" di un numero, cioè i numeri binari positivi e negativi. Il metodo più comune per rappresentare i numeri negativi, in una locazione di memoria del computer o in un registro, è quello noto come "complemento a due". Il complemento a due di un numero binario si ottiene invertendo tutte le cifre che lo compongono (cambiando cioè tutti gli zero in uno e viceversa) e

Decimale	Binario
−7	1001
−6	1010
−5	1011
−4	1100
−3	1101
−2	1110
−1	1111
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111

sommando uno al numero. Quindi, il complemento a due di 0101 è 1011.
Vediamo adesso come usare il complemento a due per eseguire operazioni matematiche con numeri positivi e negativi. Innanzi tutto si ricordi che un registro ad 8 bit può contenere fino a 256 diverse configurazioni di bit, che vengono usate per rappresentare numeri positivi compresi tra 0 e 255 oppure numeri negativi e positivi compresi tra −128 e 127 (mentre un registro a 16 bit può contenere numeri compresi tra 0 e 65535 oppure tra −32768 e 32767). La tabella a sinistra mostra la rappresentazione binaria a 4 bit dei valori decimali compresi tra −7 e 7.
Osservando la tabella si può notare che in tutti i numeri negativi in essa riportati, il bit più significativo (quello più a sinistra) vale sempre 1, contrariamente a quanto accade per i numeri positivi. Da questa osservazione si possono dedurre le seguenti proprietà o regole fondamentali per la matematica *segnata*, basata sul complemento a due:

- Il complemento a due di un numero negativo dà come risultato l'equivalente positivo di quel numero e viceversa.
- Il bit più a sinistra è sempre zero nei numeri positivi e uno nei numeri negativi. È quindi assai facile distinguere i numeri positivi dai negativi.
- Il complemento a due di zero è zero (1111 più 1 = 0000).
- L'addizione e la sottrazione vengono eseguite nel modo abituale ed i risultati che si ottengono hanno il segno giusto.

Mentre è possibile verificare quest'ultima proprietà con semplici esempi, le cose si complicano quando si vogliono moltiplicare tra loro numeri preceduti dal segno. L'istruzione MUL, usata nel programma di conversione da BCD a Binario prima descritto, tratta tutti i valori contenuti nei registri A e B come numeri privi di segno, ossia positivi. Quindi la moltiplicazione tra due numeri preceduti dal segno va programmata in altro modo. Chi ha una certa esperienza di programmazione sa che la semplice programmazione "lineare", fin qui adottata per i nostri esempi, presenta notevoli limitazioni, mentre l'impiego di una o di alcune delle seguenti strutture di controllo consente di realizzare programmi molto più efficienti:

- La "Selezione" permette di scegliere tra due differenti linee di esecuzione del programma (come l'istruzione IF in BASIC).
- La "Ripetizione" consente di ripetere una serie di operazioni:

1) mentre una data condizione resta vera (la struttura WHILE...WEND);
2) finché non si realizza una data condizione (REPEAT...UNTIL);
3) un numero di volte definito (FOR...NEXT).

Tutte queste strutture dipendono dalla capacità di verificare se si è realizzata o meno una data condizione (di solito, il raggiungimento di un particolare valore da parte di una variabile), vale a dire: se la condizione è vera o falsa. Queste strutture vengono impiegate anche nel linguaggio Assembly ed il programma deve quindi essere capace di controllare i valori contenuti nei registri. Di solito, la verifica prende in esame solo due possibilità: se il valore è zero o no e se è positivo o negativo. Con una serie di istruzioni supplementari è tuttavia possibile eseguire anche controlli d'altro genere.

Il registro di condizione

Per adoperare queste strutture condizionali si sfrutta il registro di condizione del processore (CC), sommariamente descritto a pag. 1017. A differenza degli altri registri del chip 6809, il valore globale di questo registro a 8 bit non riveste alcun interesse per il programmatore: conta esclusivamente lo stato individuale (1 o 0) di ciascuno degli 8 bit. Tre di questi riguardano la gestione degli interrupt (esaminata in un secondo momento) mentre gli altri cinque sono dedicati a condizioni del tipo appena descritto. Uno di questi, H (il flag di carry "mediano") è quasi esclusivamente dedicato all'aritmetica BCD ed il suo funzionamento viene descritto più avanti. Gli altri quattro, che riguardano più direttamente l'aritmetica dei numeri positivi e negativi, sono:

- C: il flag di Carry, che riceve la cifra di riporto (o il resto nel caso di una sottrazione) proveniente dal bit più significativo dopo un'operazione aritmetica. Questo flag viene utilizzato anche quando si vuol far scorrere di un bit verso destra il contenuto di un accumulatore. Questo flag può essere usato, ad esempio, per controllare se un numero è dispari. Il flag C è il bit 0 (il bit meno significativo) del registro CC.
- V: il flag di Overflow, che viene posto ad uno quando il risultato di un'operazione aritmetica è troppo grande per il registro che lo deve contenere. Il flag V è il bit 1 del CC.
- Z: il flag di Zero, che viene posto ad uno quando il valore contenuto in un registro è zero. Z è il bit due del registro CC.
- N: il flag del Segno, nel quale viene copiato il bit più significativo (cioè il bit del segno) del numero contenuto in un registro. In altre parole, il flag N contiene 1 quando il numero è negativo e 0 quando questo è positivo. Il flag N è il bit 3 del registro CC.

Non tutte le istruzioni influenzano lo stato dei flag e, inoltre, alcuni vengono posti ad uno, a seconda del contenuto dell'accumulatore, mentre altri possono dipendere dai valori di altri registri. La procedura più sicura è quella di eseguire verifiche soltanto sui valori contenuti nell'accumulatore ed eseguirle nel preciso momento in cui nell'accumulatore compare il valore da controllare: rimandando la verifica, è difficile esser certi che il flag non sia stato nel frattempo modificato da altre istruzioni.

I flag vengono controllati per mezzo delle istruzioni di "salto", che si possono paragonare al comando BASIC GOTO. Il 6809 adopera quasi esclusivamente salti "relativi", anziché quelli "assoluti": i primi trasferiscono il controllo in avanti o indietro di alcuni byte, mentre i secondi passano il controllo all'indirizzo specificato nella stessa istruzione. Il risultato, comunque, è identico poiché tale distinzione, in realtà, significa soltanto che i salti possono essere "corti" se l'ampiezza del salto è espressa da un singolo byte (da −128 a 127) e "lunghi" se il controllo viene trasferito ad un qualsiasi indirizzo della memoria.
Il 6809 possiede una discreta gamma di istruzioni di salto, la cui scelta dipende dalle necessità del programma. I seguenti esempi descrivono le istruzioni utilizzate per verificare e confrontare i valori contenuti negli accumulatori e spiegano come impiegare le istruzioni di salto per selezionare e per ripetere parti di un programma.

• ANDCC: dato che non è possibile caricare i valori direttamente nel registro di stato del processore, è buona abitudine azzerare i flag di cui si ha bisogno prima di usarli. Questa operazione viene eseguita con l'istruzione ANDCC, il cui funzionamento è analogo a quello della AND: gli zero sono usati come "maschere" per azzerare i flag interessati.

• SUB: sottrae l'operando dall'accumulatore. Il risultato dell'operazione influenza i flag C, V, Z ed N (se la sottrazione è a 8 bit, è interessato anche il flag H).

• CMP: confronto tra due registri. Funziona esattamente come SUB, ma lascia inalterato il contenuto del registro. Come in SUB, vengono modificati i flag C, V, Z, N ed eventualmente H.

• BRA (Salto assoluto): è identico al comando BASIC GOTO.

• BGT (Salta se maggiore di zero): esegue una verifica sui numeri (positivi e negativi). Il salto avviene se il flag Z vale zero (ossia se il numero esaminato è diverso da zero). Per non trascurare l'eventualità che il bit del segno venga interpretato in modo errato, a seguito di un "overflow", N e V devono essere entrambi a zero (l'esecuzione procede perché il numero non è negativo) oppure entrambi ad uno (segno negativo errato a causa dell'overflow). BGE, BLT e BLE sono altre istruzioni di verifica per i numeri preceduti dal segno.

• BLO (Salta se minore di zero). Questa è un'istruzione di verifica su numeri non "segnati". Infatti, è inutile controllare il flag N in caso di numeri privi di segno. Il salto si verifica se il flag C viene posto ad uno, il che avviene in presenza di un resto dopo una sottrazione. Altre istruzioni di verifica senza segno sono BLS, BHI e BHS.

Ecco un programma per trovare il maggiore tra due numeri ad 8 bit, dotati di segno, immagazzinati in $3000 e $3001. Il numero più grande tra i due viene depositato in $3002. Innanzi tutto va assegnata un'etichetta ai numeri:

```
NUM1    EQU     $3000
NUM2    EQU     $3001
RISP    EQU     $3002
        ORG     $1000
```

• Inizia l'esecuzione del programma: i flag di stato vengono posti a zero. Il primo numero viene caricato e confrontato con il secondo:

```
        ANDCC   #%11110000
        LDA     NUM1
        CMPA    NUM2
```

• Se NUM1 è più grande di NUM2, il programma salta a FINE, altrimenti carica il secondo numero nel registro A. Quando si raggiunge l'etichetta FINE, il numero caricato nel registro viene immagazzinato in RISP. L'istruzione END restituisce il controllo al sistema operativo:

```
        BGT     FINE
        LDA     NUM2
FINE    STA     RISP
        SWI
        END
```

Le direttive e l'Assembler

Il seguente esempio mostra i diversi effetti che le direttive (chiamate anche pseudo-op) dell'assembler e le istruzioni del linguaggio Assembly hanno sul contatore di locazione e sui valori contenuti in memoria

DIRETTIVE...ORIGINALI

```
CAMPO     CAMPO   CAMPO          CONTATORE   CONTENUTO
ETICHETTE OP-CODE OPERANDI       LOCAZIONE   MEMORIA
*----------DIMOSTRAZIONE-------------
RESET     EQU     $F100          $0400       ???
INDEX     EQU     16             $0400       ???
MASK1     EQU     %01101010      $0400       ???

          ORG     $1000          $1000       ???

CR        FCB     16             $1000       $10

MEMTOP    FDB     $7FFF          $1001       $7F
                                 $1002       $FF

TABLE1    RMB     7              $1003       ???
                                 $1004       ???
                                 $1005       ???
                                 $1006       ???
                                 $1007       ???
                                 $1008       ???
                                 $1009       ???

ERRMSG    FCC     /ERROR/        $100A       $45
                                 $100B       $52
                                 $100C       $52
                                 $100D       $4F
                                 $100E       $52

          CLRA                   $100F       $4F

          END                    $100F       ???
```

Poiché non è presente l'istruzione ORG, l'indirizzo di locazione viene automaticamente definito dall'Assembler. Si noti che EQU non incide sull'indirizzo di locazione e che il contenuto della memoria non è ancora definito

Questa direttiva fissa l'indirizzo indicato, ma il contenuto della memoria rimane imprecisato

FCB memorizza l'operando nel byte attualmente indirizzato dal contatore di locazione

FDB provoca l'inizializzazione di due byte

RMB riserva 7 byte (di contenuto non definito) aggiungendo il valore 7 al contatore di locazione

La direttiva FCC deposita in memoria i codici ASCII della stringa specificata

Finalmente un'operazione in linguaggio Assembly! Non c'è alcun operando, quindi l'op-code occupa un solo byte

Un'altra direttiva dell'assemblatore che non modifica il contatore di locazione

```
*--------TAVOLA DEI SIMBOLI---------
RESET   F100   INDEX   0010   MASK1   006A
CR      1000   MEMTOP  1001   TABLE1  1003
ERRMSG  100A
```

Ecco come vengono memorizzati i simboli nell'apposita area di lavoro dell'Assembler che consulta continuamente la tabella durante la conversione in codice macchina

La Memotech

Una ditta britannica inizialmente specializzata in accessori per lo ZX81 della Sinclair



La "linea" ZX81

La Memotech ha iniziato la propria attività producendo accessori per lo ZX81 della Sinclair. Il Memopak, mostrato in figura, è appunto un'espansione RAM da 32 Kbyte per lo ZX81

La Memotech venne fondata sull'onda dell'immenso interesse suscitato dai primi computer fabbricati dalla Sinclair. Lo ZX80 e lo ZX81 ottennero un grande successo, ma possedevano una memoria piuttosto limitata: questo fatto portò alla nascita di un fiorente mercato, incentrato sulla produzione di schede di memoria aggiuntive.

I due fondatori della Memotech erano entrambi assistenti presso l'Università di Oxford: Geoff Boyd teneva corsi in metallurgia, mentre Robert Branton insegnava matematica. I due si conobbero ad una mostra di computer nel 1981 e, l'anno successivo, decisero di lavorare assieme alla creazione di una serie di accessori per lo ZX81. Il loro primo prodotto fu una scheda d'espansione da 16 Kbyte. In seguito fabbricarono una completa gamma di schede di memoria (Memopaks) che comprendeva: due unità RAM da 32 e da 64 Kbyte, una scheda grafica ad alta risoluzione, un programma di lavagna elettronica (Memocalc), un word processor (Memotext), le interfacce Centronics ed RS232 e, infine, una tastiera.

Quando, nel 1982, la Sinclair lanciò lo Spectrum, la Memotech decise di non produrre accessori per questo apparecchio, bensì di impiegare l'esperienza accumulata per progettare e fabbricare computer propri. Questa decisione si basava sulla valutazione, in seguito dimostratasi errata, che il successo commerciale dello ZX81 non sarebbe durato a lungo e che era economicamente più conveniente dedicarsi ad un'attività produttiva propria.

La Memotech valuta che le vendite dei propri Memopak abbiano superato le 250.000 unità. Questi "pacchetti" vengono fabbricati, assieme agli apparecchi MTX, nella sede principale della ditta, che si trova a Witney, nello Oxfordshire.

La gamma MTX è ufficialmente apparsa sul mercato nel febbraio dell'84 e la compagnia dichiara di aver finora venduto circa 25.000 apparecchi. Come il Micro BBC, l'MTX viene prodotto in due versioni: l'MTX500 con 32 Kbyte e l'MTX512 con 64 Kbyte. Il computer impiega un microprocessore Z80, dispone di 16 colori nel modo di alta risoluzione (a 256 x 192 pixel) ed il BASIC è molto simile a quello del Micro BBC. L'apparecchio possiede inoltre un assemblatore/disassemblatore incorporato.

Il computer, una volta espanso, può usare il pacchetto grafico HRX, prodotto dalla stessa Memotech. Al modello MTX500 non espanso l'utente può aggiungere le unità a dischetti e tre schede di controllo per la grafica: una scheda principale di controllo del processore (a 96 bit!), un "Frame Grabber" o fissatore d'immagine ed un convertitore A/D a tre canali. Il sistema risultante è in grado di produrre animazioni, immagini e grafica di grande effetto compositivo. Queste caratteristiche fanno intuire che la gamma degli apparecchi MTX è stata progettata soprattutto per usi commerciali e per il mercato dei computer gestionali. Tuttavia, su queste macchine si possono far girare con soddisfacenti risultati anche programmi di gioco.



Computer "vincenti"

Dopo il successo ottenuto con le unità di espansione per lo ZX81, la Memotech ha lanciato una serie di periferiche per tale computer, tra le quali la DMX 80, una stampante ad aghi a 80 colonne. L'ultimo apparecchio fabbricato dalla compagnia è l'MTX 512, un piccolo microcomputer gestionale con 64 Kbyte di memoria, al quale possono essere collegate sia unità a nastri che a dischetti

L'MTX è in grado di funzionare col sistema operativo CP/M e può quindi far affidamento su una discreta quantità di software, anche se la scarsità di software su cassetta può suscitare qualche perplessità, soprattutto tra gli utenti meno interessati alle caratteristiche gestionali. Oggi l'MTX dispone di solo 40 programmi su cassette, ma la Memotech si è fortemente impegnata per colmare questa lacuna ed ha già preso contatti con la Continental Software e la PSS, due software house da sempre impegnate nello sviluppo di nuovi programmi, anche a carattere didattico.

Gran finale

Una panoramica sui più recenti apparecchi elettronici per il settore musicale

Il più significativo sviluppo, nella musica elettronica, è rappresentato dai nuovi sistemi digitali di registrazione. Non solo è migliorata in modo superlativo la qualità di riproduzione, ma lo stesso termine "registrazione" ha assunto significati completamente nuovi riferiti a tutta una serie di nuove tecniche. La sua definizione suona all'incirca così: «codifica digitale dei suoni e della loro organizzazione sotto forma di musica».
Solo afferrando il senso di ciò si ha un'idea precisa di cosa stia accadendo oggi in campo musicale.
Dopo la Seconda Guerra Mondiale, le registrazioni musicali sono state fatte su nastro magnetico (dalle minuscole "microcassette" alle gigantesche bobine per nastri a 24 piste, usate negli studi professionali): i suoni, convertiti in segnali elettromagnetici, modificano la struttura delle particelle di ossido metallico, presenti sulla superficie del nastro, la cui disposizione corrisponde al complesso di suoni originali.
Durante l'ascolto, queste configurazioni magnetiche vengono rilevate da un'apposita testina di lettura, per essere riconvertite in segnali elettrici e poi in suoni.
È abbastanza facile individuare con sufficiente esattezza dove si trovi la configurazione corrispondente ad una particolare sequenza o combinazione di suoni su un nastro. Pertanto, una delle attività dei tecnici del suono impegnati a costruire effetti speciali è stata, fino a poco tempo fa, proprio quella di sezionare spezzoni di nastro mediante affilatissime lamette smagnetizzate.
Nella registrazione digitale i suoni sono memorizzati su nastro sotto forma di codici numerici e la testina di lettura è corredata da un convertitore D/A (Digitale/Analogico). Supponendo che vi siano sufficienti informazioni per la conversione, la qualità di questo tipo di registrazione è nettamente superiore a quella tradizionale. Inoltre, purché le informazioni digitali rimangano intatte, si possono eseguire infinite copie, senza mai diminuirne la qualità, mentre nelle normali registrazioni magnetiche, ad ogni "passaggio", aumentano inevitabilmente sia i fruscii che le distorsioni.
Questa progressiva degradazione è stata, per anni, il maggior grattacapo dei tecnici del suono.
Oggi questo problema non esiste più e molti



Il Synclavier
Prodotto dalla New England Digital, è considerato l'apparecchio più sofisticato attualmente disponibile. Oltre alle normali funzioni di sintetizzatore, ha la facoltà di memorizzare i dati relativi ai suoni su dischi rigidi da 10 Megabyte



Il CMI della Fairlight
Questo è stato uno dei primi sistemi musicali computerizzati. Il suo sistema operativo è pilotato da menu e consente un'ampia facoltà d'interventi sulla configurazione delle forme d'onda generate. È anche prevista la possibilità di riportare su carta, tramite stampante, i dati caratteristici di un particolare suono



Il CX5 della Yamaha
Dotato di un'interfaccia MIDI, questo apparecchio può essere collegato ad altri sintetizzatori, oltre che all'home computer prodotto dalla stessa Yamaha secondo gli standard MSX



L'MSQ-700 della Roland
Questo sequenziatore ha la fama di essere stato il primo compatibile col sistema MIDI. L'MSQ-700 prevede una vasta gamma di facilitazioni per i collegamenti via MIDI e permette di memorizzare fino a 6.500 note





Il Drumulator
Questa batteria elettronica. prodotta dalla Emu Systems, ha una capacità di 10.088 note, all'interno di 64 brani. Con l'aggiunta di moduli ROM, si può ampliare la gamma di effetti, con strumenti a percussione di tipo latino o africano

per concessione DE "SYCO SYSTEM"

IAN McKINNELL

studi di registrazione sono dotati di sofisticati registratori digitali a 24 piste. La loro qualità è talmente perfetta, che spesso anche gli stessi tecnici non sanno distinguere se un suono prodotto da un altoparlante proviene da un nastro o dall'effettiva esecuzione di uno dei musicisti in sala.
Nonostante ciò sono sorti nuovi problemi: non è più possibile individuare con esattezza la posizione di un particolare suono o sequenza sul nastro, quindi gli interventi di tipo "taglia e cuci" non sono più attuabili. Inoltre, la sensibilità degli apparecchi è talmente alta, che molti rumori di sottofondo (di livello troppo basso per esser rilevati dalle apparecchiature tradizionali) vengono captati e riprodotti nella registrazione.
Mentre la registrazione digitale a 24 piste è una prerogativa esclusiva degli studi professionali, quella a traccia singola è molto più accessibile; viene impiegata, ad esempio, anche dai videoregistratori che adottano il sistema Betamax. Il nastro per video è un supporto magnetico digitale, sul quale è possibile codificare ogni sorta di dati. IL PCM (Pulse Code Modulator, ossia modulatore di impulsi codificati) della Sony è un dispositivo che trasforma un videoregistratore Betamax in un sofisticato registratore digitale per audio. La qualità di questa unità, dal costo relativamente basso, è incredibilmente superiore a quella di qualsiasi registratore magnetico tradizionale di pari dimensioni.
La codifica digitale dei suoni detta anche "sampling" o campionatura, è alla base del CMI (Computer Musical Instrument) della Fairlight, uno dei sistemi più avanzati. Il Fairlight è in grado di eseguire una campionatura di qualsiasi suono (fino a una durata massima di due secondi), per poi riprodurlo in qualsiasi tonalità sull'arco di sei ottave.
Questa tecnica rappresenta una vera rivoluzione in campo musicale. Per anni i tecnici del suono hanno tentato un'infinità di strade per riprodurre fedelmente, tramite sintetizzatori, i suoni degli strumenti tradizionali. In certi casi i risultati sono stati apprezzabili, ma niente in confronto a quanto si può ottenere col sampling: non solo si può ricreare fedelmente il suono di un qualsiasi violino, ma addirittura quello di un violino particolare, o di un altro strumento particolare, modificato dalle caratteristiche acustiche di un ben preciso ambiente!
I faticosi e lunghi tentativi di "musique concrète" degli anni '50 si possono ottenere nel giro di qualche minuto con i sofisticati apparecchi di oggi.

Echi dal passato

Questo brano è tratto da "Nuova Atlantide", una visione utopistica descritta da Francesco Bacone (1561-1626). Le descrizioni fatte da questo scrittore sembrano adattarsi perfettamente all'odierna situazione della musica elettronica



MARY EVANS PICTURE LIBRARY

Abbiamo anche le Case dei Suoni, dove imparare e studiare tutti i suoni e la loro origine.
Armonie che voi non avete, quarti di suono e suoni ancora piu' sottili. Strumenti diversi a voi altrettanto ignoti, alcuni molto piu' soavi dei vostri, assieme a campane e anelli dai suoni preziosi e dolci.
Generiamo suoni lievi ed anche gravi e profondi, e ancora grandi, indeboliti e netti.
Sappiamo tremolare e vibrare i suoni in piu' modi, che se pur originali, son sempre integri. Noi imitiamo e riproduciamo ogni suono articolato e lettere e voce e nota di bestie e uccelli.
Possediamo certi ausili che all'orecchio fan crescere l'udito e poi anche diversi echi strani e artificiali, che ricalcan piu' volte la voce, da sembrare che la ruzzolino. Altri ancora che tornan la voce piu' forte, o piu' squillante o piu' profonda, di quanto parlata sia.. Ed altri poi che rendono la voce che vi entro', mutata nelle lettere o nei suoni articolati che la compongono.
E ancora, abbiamo mezzi per convogliare li suoni in botti e tube, in strani condotti e a distanza.

(da Nuova Atlantide, Francesco Bacone, pubblicata nel 1624)

Strumenti di campionatura

Con un apparecchio quale il Fairlight, basato sul sampling, si possono rimuovere i limiti fisici degli strumenti musicali tradizionali. È facile, per un bravo suonatore di flauto, ottenere un timbro caldo e vibrante sulle note basse: ma, per quanto bravo egli possa essere, non riesce ad ottenere un simile effetto sulle note più acute dello strumento, proprio a causa delle caratteristiche fisiche di quest'ultimo. Un operatore di Fairlight, al contrario, può eseguire una campionatura del suono prodotto dal flauto sulle note basse, per poi riprodurlo due ottave sopra: il risultato è ancora un suono di flauto, ma tale da non poter essere ottenuto con nessun altro strumento reale.
Tra le varie opzioni offerte dal Fairlight, esiste quella di visualizzare sullo schermo le caratteristiche di un suono e quella di conservarle su dischetto magnetico ad 8 pollici. Le singole caratteristiche relative ad un particolare suono possono essere esaminate e modificate: talvolta è più facile trovare qualche incongruenza in un suono vedendone la forma che non ascoltandolo. Se occorre un suono di durata superiore al limite massimo di campionatura (2 secondi), se ne può esaminare la struttura e fissare un punto preciso dal quale iniziare un ciclo di ripetizione (loop) la cui durata è fissata a piacere. Se il punto è stato scelto bene, il suono avrà una tale continuità da non potersi più distinguere l'originale dalla parte aggiunta. Tuttavia i suoni naturali non si comportano in modo così matematicamente ripetitivo, quindi gli effetti possono risultare anche del tutto nuovi.
L'operatore di Fairlight dispone di due nuovi

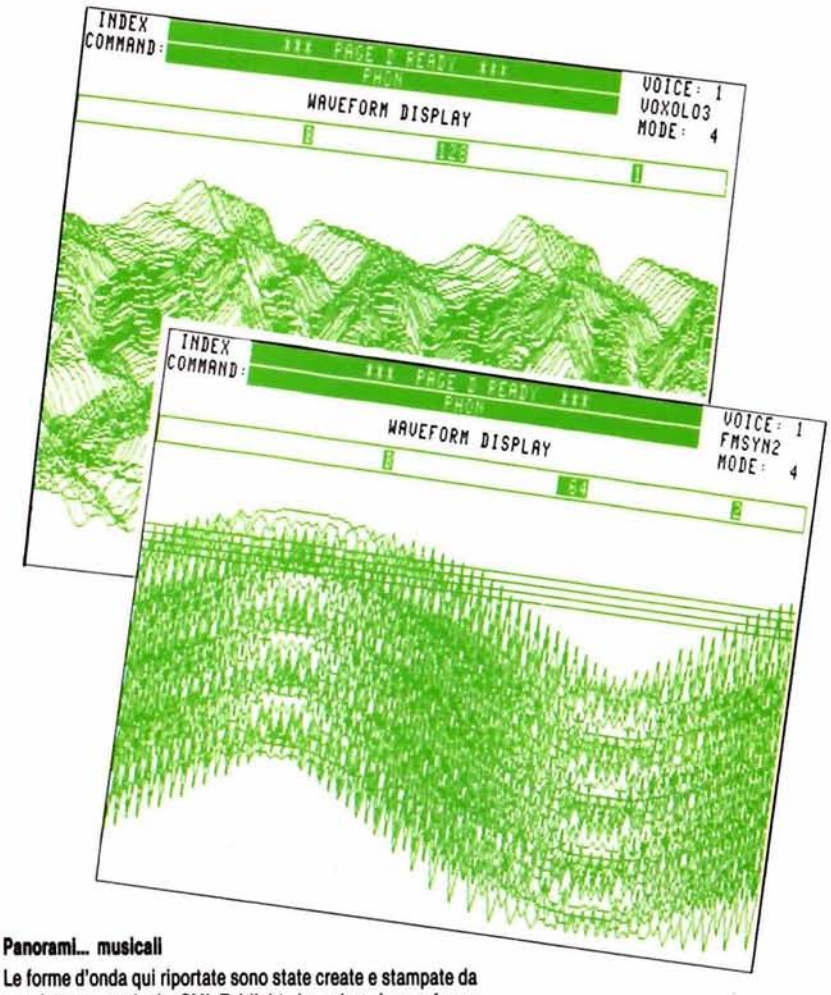
metodi per l'immissione dei brani musicali, oltre quello di suonare in "tempo reale" sulla tastiera. Il primo (noto come "Pagina R") prevede l'impiego dello schermo, sul quale viene riprodotto un pentagramma e il musicista vi inserisce i simboli delle note tramite la tastiera.
Eventuali errori nei valori musicali delle note vengono automaticamente corretti dal computer, in base al metro, o misura, musicale precedentemente fissato.
Il secondo metodo consiste nell'uso di un linguaggio MCL. L'MCL del Fairlight prevede che ogni nota (o "evento") venga immessa tramite tastiera numerica, ma consente di mutare il ritmo e l'accento su ogni singola nota.
Il Fairlight possiede otto voci, quindi l'operatore può immettere otto diverse sequenze di note, suonate da altrettanti diversi strumenti o voci. Ogni nota può anche essere leggermente sfalsata rispetto alle altre (nell'ordine dei millisecondi) poiché è l'orologio interno del Fairlight a coordinare il tutto.
Il motivo per cui è previsto questo genere di "inaccuratezza" (proprio da un computer!) è semplice: nel suonare uno strumento reale le note non vengono mai emesse in perfetto unisono; inoltre i musicisti, specialmente quelli di jazz, tendono a "stirare" di proposito le note.
Il sistema operativo del Fairlight, dunque, permette di simulare anche certi tipi di stili musicali. Tali simulazioni vengono spesso usate a scopo sperimentale, così come avviene con alcuni programmi per computer (simulazioni di volo oppure simulazioni di "gallerie del vento", ecc.).
Molti musicisti temono che apparecchi così sofisticati possano un giorno sostituire i musicisti "umani" e, in effetti, molti complessi dei giorni nostri (quali i Wang Chung, i Duran Duran e i Culture Club) usano regolarmente il Fairlight nelle proprie composizioni rendendo talvolta impossibile distinguere quanto il brano sia suonato realmente e quanto sia invece prodotto dal Fairlight. Quando si arriva a conoscere a fondo il funzionamento di questo strumento, si comprende che non si tratta di un semplice giocattolo creato per "scimmiottare" altri strumenti.
Il Fairlight non è l'unico prodotto del genere: il Synclavier, che costa più o meno il doppio, ha funzioni simili, ma possiede una scala cromatica più ampia. I dati vengono memorizzati su dischi rigidi, con una capacità massima di 40 Megabyte. Sul Synclavier sarebbe possibile memorizzare un intero "album", rendendo superfluo l'impiego di un registratore a 24 piste.
Nonostante ciò, anche il Synclavier ha dei limiti, uno dei quali è comune a tutti gli apparecchi basati sul sampling: il suono (nella sua versione originale o modificata) è riprodotto "troppo" fedelmente. Un trombettista, durante un concerto, ha la facoltà di introdurre piccoli effetti, decisi lì per lì, magari anche inconsciamente: basta variare la potenza del fiato o la posizione delle labbra. Sui sintetizzatori tale facoltà di intervento immediato è ancora un obiettivo da raggiungere.
Il Kurzweil, un apparecchio ancora allo stadio di prototipo, è dotato di un programma per il riconoscimento di "modelli" sonori: quando viene premuto uno dei tasti, viene eseguita una scansione su diversi campioni e le caratteristiche di ciascun campione vengono combinate assieme per produrre il suono. Un tale sistema consente di modellare l'esecuzione di un brano e dovrebbe riuscire a simulare con più successo il comportamento acustico degli strumenti tradizionali. Rimane un solo inconveniente: nessun pianoforte, ad esempio, è esattamente identico ad un altro, mentre il suono riprodotto da tutti i Kurzweil è quello di un solo pianoforte (a meno che l'operatore non scriva del software proprio per modificarne le caratteristiche).
Un sistema ancor più sofisticato dei tre citati messi assieme, dovrebbe essere quello in via di sviluppo presso la Lucasfilm (la compagnia cinematografica che ha prodotto *Guerre Stellari*). L'apparecchio è stato chiamato ASP (Audio Signal Processor) e si dice che nel suo



L'MCL del Fairlight

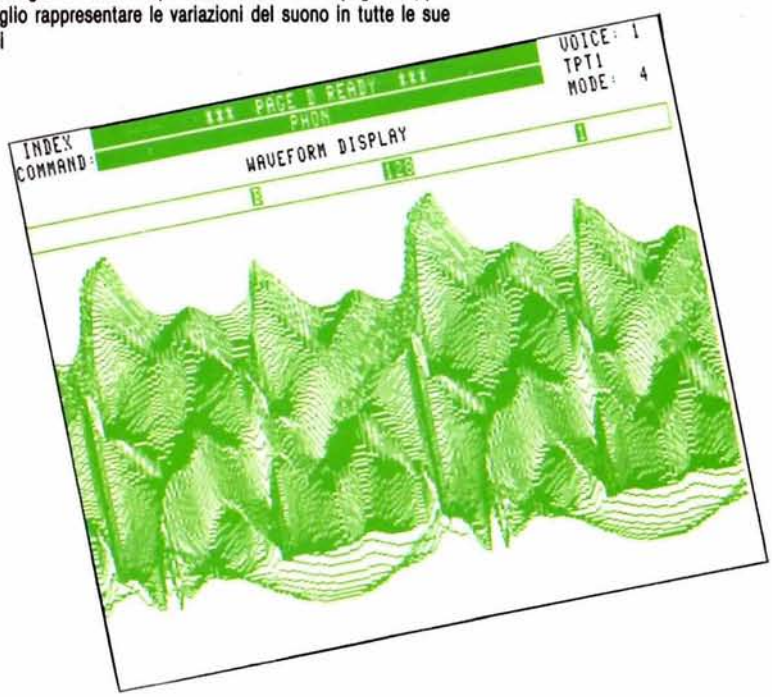
Il linguaggio per la composizione musicale (MCL) adottato sul Fairlight è pilotato da menu, facilitando la scelta tra le varie opzioni disponibili. Le forme d'onda illustrate a pag. 1064 sono state generate mediante un elenco di parametri, simile ad una lista di frasi DATA in un programma BASIC. L'elenco è stato formulato, visualizzato e quindi stampato usando le scelte indicate dal menu

```
0001 !B=24:G=1/24:V=15
0002 R,36:<(Bf4:D4),5:(Bf4:D4),7
0003 R,5:(Bf4:D4)V10,7/3:(Bf4:D4)V11,7/3:(F4:C4)V13,7/3
0004 (Bf4:D4),5:(Bf4:D4),7
0005 R,5:(Bf4:D4)V10,7/3:(Bf4:D4)V11,7/3:(F4:C4:Ef4)V13,7/3
0006 (D4:F5),5:(D4:F5),7
0007 R,5:(D4:F5)V10,7/3:(D4:F5)V11,7/3:(G4:B4:Ef4)V13,7/3
0008 (Bf4:D4),5:(Bf4:D4),7
0009 R,5:(Bf4:D4)V10,7/3:(Bf4:D4)V11,7/3:(F4:C4)V13,7/3>2
0010 <(Bf4:D4),7/3:(Bf4:D4),7/3:(F4:C4),7/3:T=+3>3:T=0
0011 <(Bf4:D4:A5),5:(C4:Ef4),7
0012 (Bf4:D4:A5),5:(C4:Ef4),7/2:(D4:Bf5),7/2>2
0013 (C4:Ef4:Bf5),5:(Bf4:D4),7
0014 (C4:Ef4:Bf5),5:(Bf4:D4),7/3:(Ef4:C4),7/3:(A5:C5),7/3
0015 (C4:Ef4:Bf5),5:(Bf4:D4),7
0016 (C4:Ef4:Bf5),5:(Bf4:D4),7/4:R,5/6
0017 (Bf4:D4),7/8:R,5/12
0018 (G4:B4:Ef4),75/24:V=15
0019 <(Bf4:D4),5:(Bf4:D4),7
0020 R,5:(Bf4:D4)V10,7/3:(Bf4:D4)V11,7/3:(F4:C4)V13,7/3
0021 (Bf4:D4),5:(Bf4:D4),7
0022 R,5:(Bf4:D4)V10,7/3:(Bf4:D4)V11,7/3:(F4:C4:Ef4)V13,7/3
0023 (D4:F5),5:(D4:F5),7
0024 R,5:(D4:F5)V10,7/3:(D4:F5)V11,7/3:(G4:B4,Ef4)V13,7/3
0025 (Bf4:D4),5:(Bf4:D4),7
0026 R,5:(Bf4:D4)V10,7/3:(Bf4:D4)V11,7/3:(F4:C4)V13,7/3>2
```



Panorami... musicali

Le forme d'onda qui riportate sono state create e stampate da un sistema musicale CMI Fairlight. La prima è una forma d'onda sinusoidale generata usando la sintesi a modulazione di frequenza (FM) del Fairlight e combinando assieme, elettronicamente, più forme d'onda. Gli altri diagrammi mostrano esempi di "campionatura" di suoni: il secondo si riferisce ad un campione di voce umana, il terzo alla voce di una tromba. Tutti i grafici sono di tipo tridimensionale o topografico, per meglio rappresentare le variazioni del suono in tutte le sue fasi



sistema operativo siano previste tutte le più disparate facilitazioni attualmente offerte dagli studi di registrazione più sofisticati. C'è dunque da aspettarsi, in un futuro molto prossimo, uno sviluppo simile a quello avvenuto per i computer: non più enormi apparecchi intrasportabili, ma un vero e proprio studio di registrazione contenuto in una normalissima ventiquattrore.

Le applicazioni della tecnologia digitale non sono confinate alla sola generazione e riproduzione dei suoni; alcuni dei più aggiornati studi di registrazione sono dotati di tutta una serie di apparecchi dedicati all'elaborazione dei suoni, come le unità di riverbero, attraverso le quali la musica viene dirottata per aggiungervi echi o riverberi, indispensabili in certi generi musicali.

Il Quantec è un'unità digitale che non si limita ad aggiungere un riverbero "generico": produce infatti quello di particolari ambienti o sale di diverse dimensioni. Il più piccolo "ambiente acustico" disponibile è quello corrispondente ad una scatola con un volume di un metro cubo e la gamma di simulazioni predefinite include cattedrali, hangar, ambienti domestici e sale da concerto. Uno degli aspetti più interessanti del Quantec è la possibilità di prolungare il tempo di riverbero ben oltre la durata naturale. Ciò significa che, se il suono è riprodotto nell'ambiente "cattedrale" e la durata è portata al massimo, si ottiene un effetto che perdura svariati minuti, paragonabile all'eco ottenibile con 10 Gran Canyon consecutivi!

Il Quantec viene comunemente usato nel doppiaggio di film e telefilm: la voce degli attori, registrata nell'ambiente afono di uno studio, viene passata a questo apparecchio, che aggiunge l'effetto di riverbero tipico dell'ambiente in cui si svolge l'azione del filmato.

Valori "umani"

Molte persone, musicisti e non, ritengono che la tecnologia digitale, applicata alla musica, abbia un effetto deleterio. Viene fatto notare che la musica è creata ed eseguita in maniera talmente artificiale, che i veri valori "umani" di spontaneità ed espressività vengono sacrificati sull'altare del maggior controllo tecnico. Questo è un ragionamento abbastanza persuasivo, ma si consideri ciò che è avvenuto in un altro settore artistico: quello del cinema. Si può forse pretendere che un buon film venga girato tutto d'un fiato, magari improvvisando, come accade in un brano musicale? Eppure, nonostante gli innumerevoli e risaputi effetti speciali e gli interventi di montaggio (per non parlare di scene ripetute più volte per raggiungere la perfezione) alcuni film sono considerati opere d'arte. In linea di principio, si può dire che anche la musica stia attraversando un periodo di metamorfosi, simile a quanto già avvenuto per teatro e cinema; ma non per questo, la gente che apprezza un buon film ha smesso di andare a teatro!

Controllo di un motore

Un semplice sistema di "feedback" per controllare i movimenti di un motorino

Il controllo di dispositivi esterni tramite microprocessore è oggi un fatto comune nell'industria: dal semplice convogliare bottiglie su di un nastro trasportatore al complesso montaggio dello sportello di un'auto. I principi alla base di qualsiasi sistema di controllo sono: l'immissione di appropriati dati in un formato accettato dal microprocessore, l'analisi di tali dati e, infine, l'emissione di segnali di controllo tramite i quali ottenere una determinata sequenza di azioni sui dispositivi esterni. In un sistema "a controllo di feedback", queste tre operazioni sono ripetute in un ciclo continuo.

Per chiarire i concetti di un simile sistema di controllo, consideriamo un esempio molto pratico. Supponiamo di voler cuocere un minestrone su di un fornello. Il calore deve essere tale da far bollire il contenuto della pentola, ma senza provocarne la fuoriuscita. Un cuoco, probabilmente, si comporterebbe così: regolerebbe il calore al massimo per portare il minestrone all'ebollizione, poi abbasserebbe la fiamma ad un livello sufficiente per non provocare fuoriuscite. Se l'ebollizione dovesse aumentare occorrerebbe abbassare leggermente la fiamma. Questo metodo è basato su di un continuo controllo "visivo": si analizza la situazione e si agisce di conseguenza, ripetendo il procedimento fino a cottura completata. Un microcomputer può seguire un sistema simile, ma non identico. La principale differenza consiste nel metodo adottato per rilevare lo stato del minestrone: ad un cuoco basta un'occhiata per agire in base alla propria esperienza, mentre un computer deve affidarsi a misurazioni di proprietà fisiche più precise, quali ad esempio la temperatura. Prima di poter "computerizzare" la cottura di un minestrone, andrebbero fatti numerosi esperimenti, per determinare con esattezza la temperatura occorrente per una buona cottura e quella che provocherebbe una fuoriuscita del minestrone. Dopo tutto questo però, avendo dotato il computer di adeguate unità di lettura, la macchina potrebbe per sempre sostituire il cuoco in questa operazione.

Usando il "buffer box" (vedere a pag. 1026) e il "box di uscita" (vedere a pag. 1054) da noi realizzati, un motore elettrico può esser controllato in più modi. L'ideale, per mettere alla prova il nostro software di controllo, è il motore di un trenino elettrico o uno della Lego. L'unico accorgimento da prendere è vedere se il motore da collegare al nostro box di uscita funziona a corrente continua e ad un voltaggio pari o leggermente superiore a quello fornito dal trasformatore. Collegando i due terminali del motore alla linea 0 del box, possiamo azionarlo o arrestarlo tramite i tasti Z e X della tastiera sul computer.

MICRO BBC

```
10 REM CONTROLLO SEMPLICE — BBC
20 DDR=&FE62:REGDAT=&FE60
30 ?DDR=255:REM TUTTE LINEE IN USCITA
40 ?REGDAT=0:REM MOTORE SPENTO
50 REPEAT
60 A$=INKEY$(1):REM TASTO PREMUTO?
70 IF A$="Z" THEN?REGDAT=1:REM ACCENDE
80 UNTIL A$="X"
90 ?REGDAT=0:REM SPEGNE IL MOTORE
```

COMMODORE 64

```
10 REM CONTROLLO SEMPLICE — CBM64
20 DDR=56579:REGDAT=56577
30 POKEDDR,255:REM TUTTE LINEE IN USCITA
40 POKEREGDAT,0:REM MOTORE SPENTO
50 GETA$:IFA$<>"Z"ANDA$<>"X"THEN50:REM
   ATTESA PER UN CARATTERE VALIDO
60 IFA$="Z"THEN POKEREGDAT,1:GOTO50
70 IFA$="X"THEN POKEREGDAT,0:END
```

Possiamo controllare la direzione della rota-



IAN McKINNELL

Un progetto in espansione

Il buffer box, realizzato per primo, provvede a proteggere i circuiti del computer da correnti troppo alte sia in ingresso che in uscita. Il circuito di alimentazione autonomo fornisce la corrente necessaria anche per il secondo modulo realizzato e consente al software di commutare tensioni a 12 volt

ESERCIZI

Possiamo adesso avventurarci in una serie di interessanti esperimenti di controllo, adoperando i due moduli realizzati in "Laboratorio". I sensori citati negli esercizi possono essere unità fotorivelatrici o contatti magnetici (reed), azionati da una piccola calamita. Tali dispositivi sono reperibili, a prezzi modestissimi, presso qualsiasi fornitore di materiali elettronici. Negli interruttori termici il contatto tra i due terminali avviene ad una temperatura prefissata. Anche questi sono facilmente reperibili.

1) Scrivere un programma che faccia muovere un veicolo avanti e indietro lungo un cammino delimitato da due sensori.

2) Scrivere un programma che faccia fermare il veicolo immediatamente sopra uno dei due sensori (facendogli invertire la marcia se necessario).

3) Scrivere un programma per far sì che la temperatura dell'acqua in un bricco sia compresa tra 70° e 100°C, usando un riscaldatore elettrico a basso voltaggio e due interruttori termici.

4) Scrivere un programma per calcolare la velocità (in metri al secondo) di un veicolo che viaggia tra due precise posizioni (occorre conoscere la distanza tra i punti e registrare il tempo necessario per coprire tale distanza)

zione, collegando i terminali del motore alle uscite adiacenti a quelle usate, come illustrato nel diagramma. Per selezionare la direzione basta collegare un semplice pulsante alla linea 7 del buffer box.
Nel seguente programma, depositando un 1 nel registro dati si provoca un flusso di corrente in una direzione, mentre depositando un 2 il flusso cambia direzione. Il programma controlla continuamente la linea 7 e deposita il valore 2 soltanto se la linea si trova a livello basso (il pulsante "chiude" il contatto). In questo modo, agendo sul pulsante si inverte la direzione della rotazione: un esempio molto elementare di controllo basato sul feedback.

MICRO BBC

```
10 REM CONTROLLO DIREZIONE — BBC
20 DDR=&FE62:REGDAT=&FE60
30 ?DDR=127:REM LINEA 7 IN INGRESSO
40 ?REGDAT=0:REM MOTORE SPENTO
50 A$=GET$:REM ATTESA DI UN TASTO
60 REPEAT
70 A$=INKEY$(1)
80 IF (?REGDAT AND 128)=0 THEN DIR=2 ELSE DIR=1
90 ?REGDAT=DIR
100 UNTIL A$="X":REM TASTO X PER FINIRE
110 ?REGDAT=0:REM SPEGNE IL MOTORE
```

COMMODORE 64

```
10 REM CONTROLLO DIREZIONE — CBM64
20 DDR=56579:REGDAT=56577
30 POKEDDR,127:REM LINEA 7 IN INGRESSO
40 POKEREGDAT,0:REM MOTORE SPENTO
50 GETA$:IFA$=""THEN50:REM ATTENDE UN CARATTERE
60 GETA$
70 IF (PEEK(REGDAT)AND128)=0THENPOKEREGDAT,2:
   GOTO90
80 POKEREGDAT,1
90 IFA$<>"X"THEN60
100 POKEREGDAT,0:REM SPEGNE IL MOTORE
```

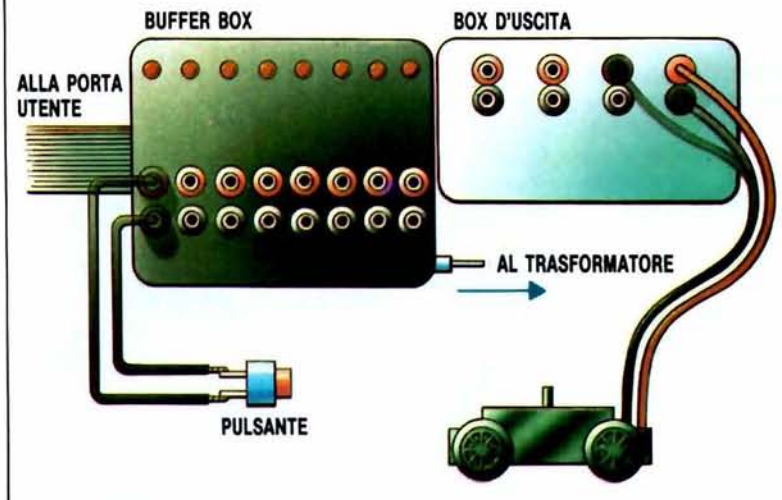
Oltre a poter controllare la direzione della rotazione, possiamo anche variarne la velocità tramite il box di uscita. Il metodo non ricorre a dispositivi aggiuntivi (quali un convertitore digitale/analogico) ma è basato sull'invio di brevi impulsi che in rapida successione avviano o arrestano il motore. Se la frequenza degli impulsi è sufficientemente alta, il motore sembrerà ruotare in modo continuo. È l'intervallo tra gli impulsi che regola la velocità del motore. Per ottenere tutto ciò da un programma, basta disporre di due cicli di ritardo, la cui durata sia facilmente variabile, all'interno di una struttura ripetitiva più ampia.

MICRO BBC

```
10 REM CONTROLLO VELOCITÀ — BBC
20 DDR=&FE62:REGDAT=&FE60:VELOC=30
30 ?DDR=255:REM TUTTE LINEE IN USCITA
40 ?REGDAT=0:REM MOTORE SPENTO
50 A$=GET$:REM ATTESA DI UN TASTO
60 REPEAT
70 A$=INKEY$(1)
80 ?REGDAT=0:REM SPEGNE IL MOTORE
90 FORI=1TO(100—VELOC):NEXT:REM RITARDO1
100 ?REGDAT=1:REM ACCENDE IL MOTORE
110 FORI=1TOVELOC:NEXT:REM RITARDO2
120 IF A$="D"THEN VELOC=VELOC—5
130 IF A$="Z"THEN VELOC=VELOC+5
140 UNTIL A$="X"
150 ?REGDAT=0:REM SPEGNE IL MOTORE
```

COMMODORE 64

```
10 REM CONTROLLO VELOCITÀ — CBM64
20 DDR=56579:REGDAT=56577:VELOC=30
30 POKEDDR,255:REM TUTTE LINEE IN USCITA
40 POKEREGDAT,0:REM MOTORE SPENTO
50 GETA$:IFA$=""THEN50:REM ATTESA TASTO
60 GETA$
70 POKEREGDAT=0:REM SPEGNE IL MOTORE
80 FORI=1TO(100—VELOC):NEXT:REM RITARDO1
90 POKEREGDAT=1:REM AVVIA IL MOTORE
100 FORI=1TOVELOC:NEXT:REM RITARDO2
110 IFA$="D"THENVELOC=VELOC-5
```



Una periferica su quattro ruote
I due box realizzati in "Laboratorio" hanno in comune il bus dei dati e l'alimentazione. Se le prese minicon sono state fissate adeguatamente, le due scatole possono essere saldamente connesse una accanto all'altra.
Il piccolo veicolo è dotato di un motorino a 12 volt, pilotato dallo stesso box di uscita e dalla linea 7 del buffer box. La polarità dell'uscita può esser invertita tramite software, facendo muovere l'auto in avanti o indietro

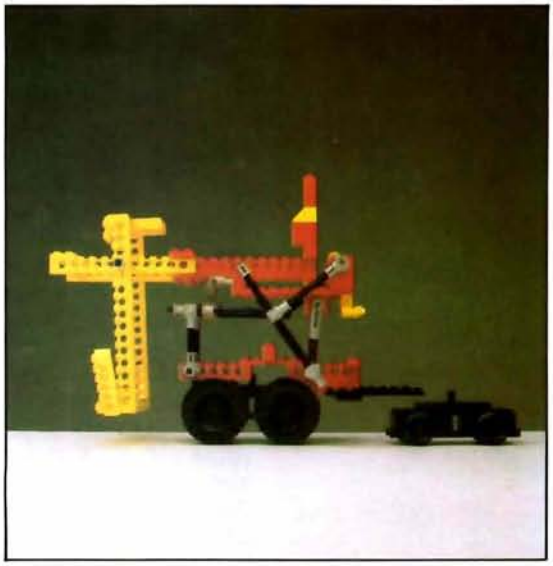
KEVIN JONES

```
120 IFA$="Z"THENVELOC=VELOC+5
130 IFA$<>"X"THEN60
140 POKEREGDAT,0:REM SPEGNE IL MOTORE
```

In questo programma, la variabile VELOC è usata per fissare la durata di ciascun ciclo di ritardo. I cicli sono formulati in modo che all'aumentare del ritardo prodotto da uno corrisponda una diminuzione nell'altro e viceversa. RITARDO1 si riferisce al periodo di accensione, mentre RITARDO2 al periodo di sospensione nell'erogazione di corrente. Per valori alti di VELOC, il primo ciclo dura poco e il secondo dura molto: il motore gira velocemente. Diminuendo VELOC, si aumenta la durata dei periodi durante i quali il motore è spento, quindi la velocità diminuisce. Durante l'esperimento, il LED sulla linea 1 lampeggia, evidenziando come il segnale sia formato da impulsi.

Imparare divertendosi

Ad alcuni può sembrare che il fatto di poter controllare gli spostamenti di un piccolo veicolo giocattolo sia un magro risultato, considerato l'investimento necessario per l'acquisto di un microcomputer. Tuttavia, anche questi semplici esperimenti possono far comprendere i principi che sono alla base dei sistemi industriali, dove il software controlla procedimenti ben più complessi



IAN McKINNELL

Costruzioni LEGO di DAVE WHELAN

RISPOSTE AGLI ESERCIZI

1) Supponendo che i sensori siano collegati alle linee 6 e 7 e che i terminali del motore siano connessi alle uscite positive 0 e 1:

```
10 REM VERSIONE 3.1 - BBC
20 DDR=&FE62:REGDAT=&FE60
30 ?DDR=63:REM LINEE 6 & 7 IN INGRESSO
40 avanti=1:indietro=2
50 ?REGDAT=avanti
60 FORI=1TO2000:NEXT:REM RITARDO
70 REPEAT UNTIL(?REGDAT AND 192)<>192
80 ?REGDAT=indietro
90 FORI=1TO2000:NEXT:REM RITARDO
100 IF (?REGDAT AND 192)<>192 THEN 50
    ELSE GOTO 100
```

```
10 REM VERSIONE 3.1 - CBM64
20 DDR=56579:REGDAT=56577
30 POKEDDR,63:REM LINEE 6 & 7 IN INGRESSO
40 AV=1:RI=2
50 POKE REGDAT,AV
60 FORI=1TO1000:NEXT:REM RITARDO
70 IF (PEEK(REGDAT)AND192)=192THEN70
80 POKE REGDAT,RI
90 FORI=1TO1000:NEXT:REM RITARDO
100 IF (PEEK(REGDAT)AND192)<>192THEN50
110 GOTO100
```

3) Supponendo che i sensori termici a 40 e 70 gradi siano collegati rispettivamente alle linee 6 e 7 e il riscaldatore alla linea 0:

```
10 REM VERSIONE 3.3 - BBC
20 DDR=&FE62:REGDAT=&FE60
30 ?DDR=63:REM LINEE 6 & 7 IN INGRESSO
40 REPEAT
50 A$=INKEY$(1)
60 ?REGDAT=1:REM ACCENDE RISCALDATORE
70 REPEAT
80 UNTIL(?REGDAT AND 192)=0:REM 70 GRADI
90 ?REGDAT=0:REM SPEGNE RISCALDATORE
100 REPEAT UNTIL(?REGDAT AND192)=192
110 UNTIL A$<>"":REM PRESSIONE TASTO
```

```
10 REM VERSIONE 3.3 - CBM64
20 DDR=56579:REGDAT=56577
30 POKEDDR,63:REM LINEE 6 & 7 IN INGRESSO
40 GET A$
50 POKE REGDAT=1:REM ACCENDE RISCALDATORE
60 IF (PEEK(REGDAT)AND192)<>0THEN60
70 POKE REGDAT,0:REM SPEGNE RISCALDATORE
80 IF (PEEK(REGDAT)AND192)<>192THEN80
90 IF A$="" THEN40
```

2) Supponendo che i sensori siano collegati alla linea 7 ed il motore alle linee 0 e 1:

```
10 REM VERSIONE 3.2 - BBC
20 DDR=&FE62:REGDAT=&FE60
30 ?DDR=127:REM LINEA 7 IN INGRESSO
40 veloc=30:avanti=1:indietro=2
50 direz=avanti
60 REPEAT
70 ?REGDAT=0:MOTORE SPENTO
80 FOR I = 1 TO (100-veloc):NEXT
90 ?REGDAT=direz
100 FOR I = 1 TO veloc:NEXT
110 UNTIL(?REGDAT AND 128)=0:REM INTERRUTT.
120 FORI=1TO1000:NEXT:REM RITARDO
130 REM VERIFICA SE VEICOLO SOPRA SENSORE
140 IF(?REGDAT AND 128)=0 THEN ?REGDAT=0:END
150 REM RETROMARCIA LENTA
160 veloc=2:direz=indietro:GOTO 60
```

```
10 REM VERSIONE 3.2 - CBM64
20 DDR=56579:REGDAT=56577
30 POKEDDR,127:REM LINEA 7 IN INGRESSO
40 VE=30:AV=1:RI=2
50 DI=AV
60 POKE REGDAT,0:REM MOTORE SPENTO
70 FORI=1TO(100-VE):NEXT
80 POKE REGDAT,DI
90 FOR I=1 TO VE:NEXT
100 IF (PEEK(REGDAT)AND128)<>0THEN60
110 FORI=1TO1000:NEXT:REM RITARDO
120 IF (PEEK(REGDAT)AND128)=0THENPOKE
    REGDAT,0:END
130 REM RETROMARCIA LENTA
140 VE=2:DI=RI:GOTO60
```

4) Supponendo che il primo interruttore sia collegato alla linea 6 ed il secondo alla linea 7:

```
10 REM VERSIONE 3.4 - BBC
20 DDR=&FE62:REGDAT=&FE60:DIST=1
30 ?DDR=63:REM LINEE 6 & 7 IN INGRESSO
40 REPEAT UNTIL (?REGDAT AND 64)=0
50 ?REGDAT=1
60 TIME=0:REM AZZERA CONTATEMPO
70 REPEAT UNTIL(?REGDAT AND 128)=0
80 PRINT "VELOCITA'="DIST/(TIME/100)
90 ?REGDAT=0
```

```
10 REM VERSIONE 3.4 - CBM64
20 DDR=56579:REGDAT=56577:DS=1
30 POKEDDR,63:REM LINEE 6 & 7 IN INGRESSO
40 IF (PEEK(REGDAT)AND64)<>0THEN40
50 POKE REGDAT,1
60 T=TI:REM LEGGE CONTATEMPO
70 IF (PEEK(REGDAT)AND128)<>0THEN60
80 PRINT "VELOCITA'="DS/((TI-T)/60)
90 POKE REGDAT,0
```

La creazione di caratteri

È la volta di un generatore di caratteri per lo Spectrum ed il BBC

Per la definizione di nuovi caratteri, sullo Spectrum, esiste un'apposita area di memoria. Infatti, il sistema operativo riserva a tale scopo 168 byte (lo spazio per 21 definizioni) nella parte alta della RAM (questa piccola zona può essere altrimenti adoperata per depositarvi piccoli programmi in codice macchina o dati). A ciascuna definizione di carattere, il sistema operativo abbina un codice CHR$, da 144 a 164, corrispondenti ai caratteri da "a" ad "u" del modo grafico (G). La variabile di sistema per i caratteri definiti da utente (più brevemente UDG) è situata nella locazione 32600 ed il suo indirizzo punta all'inizio della zona riservata. Tuttavia, non occorrono complicati calcoli per ricavare l'indirizzo iniziale per ciascun carattere, poiché il comando LET indir=USR "A" assegna alla variabile indir l'indirizzo della definizione relativa al carattere tra virgolette.
Paragonando il listato per il Commodore con quello per lo Spectrum, si può notare quanto il secondo sia relativamente meno complesso. In effetti, la versione per lo Spectrum è una trasposizione di quella per il Commodore (vedere a pag. 1053). I normali tasti per il cursore dello Spectrum (Shift 5, Shift 6, ecc.) controllano il cursore di Edit, mentre i tasti 6, 7 e 8 (senza Shift) corrispondono rispettivamente a: accensione/spegnimento di una cella, selezione del carattere da modificare, collocazione di un nuovo carattere nella griglia; il punto esclamativo serve per uscire dal programma.
Terminata la creazione di uno o più caratteri, si può memorizzare l'intera area UDG su nastro tramite il comando:

```
SAVE "nomeudg" CODE (USR"A"),168
```

Per rileggere l'area in memoria:

```
LOAD "nomeudg" CODE
```

Il procedimento sul BBC è simile a quello per lo Spectrum, almeno a prima vista. I 256 byte situati tra &0C00 e &0CFF sono riservati per le definizioni di UDG, abbinati ai codici ASCII da 224 a 255. Qualora il set di caratteri sia "imploso" (come descritto nel Manuale per l'utente esperto del BBC) allora tali definizioni sono anche associate ai codici ASCII da 128 a 159, da 160 a 191 e da 192 a 233. Se, invece, il set di caratteri è "esploso" l'intero gruppo di caratteri visualizzabili, da CHR$(32) a CHR$(255), può esser ridefinito, ma sacrificando una buona dose di RAM. Comunque, in genere 32 UDG sono sufficienti per la maggior parte delle applicazioni.
Il programma funziona in modo analogo alle versioni per Commodore e Spectrum: i tasti con le freccette controllano i movimenti del cursore, mentre i tasti di funzione da f1 a f3 sono dedicati alle operazioni di accensione/spegnimento delle celle e di selezione ed estrazione dei caratteri; il tasto del punto esclamativo serve per uscire dal programma. Si notino le chiamate OS al sistema operativo, nelle linee 70 e 180: queste abilitano e disabilitano la funzione COPY, in modo da permettere al programma di adoperare i tasti con le freccette. Se capitasse di uscire dal programma senza ricorrere al tasto "!", conviene ripristinare le condizioni normali impartendo subito il comando *FX4,0.
Il Micro BBC consente la definizione di caratteri mediante l'istruzione VDU23, ma nel caso in esame è più semplice calcolare gli indirizzi occorrenti ed eseguire singole PEEK e POKE.
Il set di caratteri UDG appena creato si salva su nastro (o disco) con:

```
*SAVE "nomefile" 0C00 0CFF
```

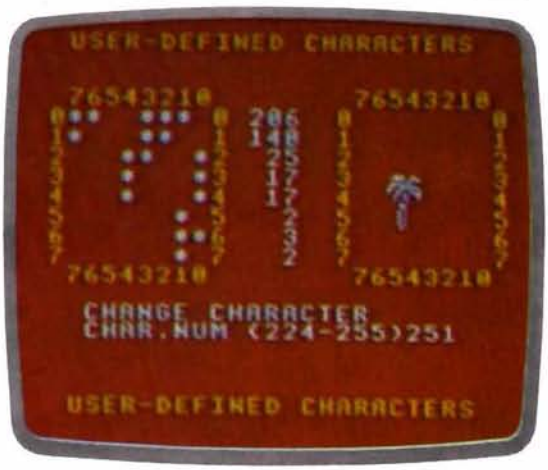
e lo si rilegge con

```
*LOAD ""
```

Spectrum



BBC Micro



IAN McKINNELL

SPECTRUM

```
 19 REM ********************************
 20 REM *          Spectrum            *
 21 REM *  generenatore caratteri UDG  *
 22 REM ********************************
100 GO SUB 1000: REM inizializza
110 FOR j=0 TO 1 STEP 0
120 GO SUB 2500: REM immissione
130 GO SUB 3000: REM convalida
140 GO SUB puntat: REM esegue
150 NEXT j
200 STOP
999 REM ********************************
1000 REM *          inizializza         *
1001 REM ********************************
1020 DIM b(8,8): DIM c$(2,2): DIM o(2,7):
     DIM r$ (8): DIM d$(1): DIM t(8,8)
1100 LET errato=2000: LET movicur=3500:
     LET comando=3000: LET aggior=6500
1200 REM **** inizializza schermo ****
1220 LET r0=4: LETc0=3: LET ri=8:
     LET c1=8: LET of=16
1230 LET nero=0: LET blu=1: LETazzu=5:
     LET bianco=7
1240 PAPER azzu: INK nero
1250 LET z$="Caratteri UDG": PRINT AT
     1,4; z$: PRINT AT 20,4;z$
1260 LET z$=" 76543210": PRINT AT
     r0,c0;z$;: PRINT TAB c0+of;z$
1270 PRINT AT r0+c1+1,c0;z$;:PRINT
     TAB c0+of;z$
1280 FOR r=r0+1 TO r0+c1
1290 LET z$=STR$ (r-r0-1): LET z$=
     z$+"  "+z$
1300 PRINT AT r,c0;z$;AT r,c0+of;z$
1310 NEXT r
1420 REM ****     offset cursore     ****
1440 DATA -1,+1,0,0
1445 DATA 0,0,+1,-1
1450 DATA 0,0,+1,-1
1460 FOR k=1 TO 2: FOR l=1 TO 4
1470 READ o(k,l): NEXT l: NEXT k
1600 REM ****  inizio finestra testo ****
1620 FOR r=1 TO r1
1640 FOR c=1 TO c1
1660 LET t(r,c)=32
1680 NEXT c: NEXT r
1800 LET rpos=1: LET cpos=1: LET cn=144:
     LETe$="A"
1820 GO SUB 6000
1990 RETURN
1999 REM ********************************
2000 REM *        tasto non valido      *
2001 REM ********************************
2020 BEEP .2,-3: RETURN
2120 RETURN
2499 REM ********************************
2500 REM *    cursore in  rpos,cpos     *
2501 REM ********************************
2520 LET rp=r0+rpos: LET cp=c0+cpos: LET
     cf=i+b(rpos,cpos): LET d$=" "(cf)
2540 PRINT AT rp,cp; flash 1;d$
2600 REM LET t$=INKEY$: IF t$<>"" THEN
     GO TO 2600
2620 LET t$=INKEY$: IF T$="" THEN GO
     TO 2620
2640 PRINT AT rp,cp; FLASH 0;d$
2700 RETURN
2999 REM ********************************
3000 REM *     convalida immissione     *
3001 REM ********************************
3020 IF t$="!" THEN LET puntat=errato:
     LET j=2: RETURN
3040 LET tasto=CODE (t$)-7:
     LET k=tasto-45: LET puntat=errato
3060 IF (tasto>=1 AND tasto<=4) THEN LET
     puntat=movicur: RETURN
3080 IF (T$>="&" AND t$<="8") THEN LET
     tasto=VAL (t$)-4 : LET puntat=comando
     +tasto*500: RETURN
3100 RETURN
3499 REM ********************************
3500 REM *       muove il cursore       *
3501 REM ********************************
3520 LET ny=rpos+o(2,tasto): LET
     nx=cpos+o(1,tasto)
3540 IF (ny<1 OR ny>r1) THEN RETURN
3560 IF (nx<1 OR nx>r1) THEN RETURN
3580 LET rpos=ny: LET cpos=nx
3600 RETURN
3999 REM ********************************
4000 REM *       alterna una cella      *
4001 REM ********************************
4020 LET tg=1-b(rpos,cpos): LET
     d$=" "(1+tg)
4040 PRINT AT  rpos+r0,cpos+c0;d$
4060 LET b(rpos,cpos)=tg
4120 LET pe=PEEK (mpos+rpos)
4140 LET pe=pe+(tg*2-1)*(2^(8-cpos))
4160 POKE (mpos+rpos),pe
4200 PRINT AT r0+rpos,c0+c1+3;PEEK
     (mpos+rpos);" "
4220 NEXT r
4300 RETURN
4499 REM ********************************
4500 REM * definizione nuovo carattere  *
4501 REM ********************************
4520 PRINT AT r0+r1+3,c0+2; FLASH 1;
     "Cambio Carattere"
4530 GO SUB aggior
4540 PRINT AT r0+r1+4,c0+2;"Premi un
     tasto (a -u)"
4550 FOR z=1 TO 1
4560 LET e$=INKEY$: IF e$<>"" THEN GO TO
     4560
4570 LET e$=INKEY$: IF e$="" THEN GO TO
     4570
4580 IF (e$<"a" OR e$>"u") THEN GO SUB
     errato: LET z=0
4590 NEXT z
4600 LET cn=CODE (e$)+79
4610 IF cn>164 THEN LET cn=cn-32
4620 PRINT AT r0+r1+3,c0+2;
     FLASH 0;"                "
4630 PRINT AT r0+r1+4,c0+
     2;"                "
4640 GO SUB 6000
4990 RETURN
4999 REM ********************************
5000 REM *       colloca carattere      *
5001 REM ********************************
5020 PRINT AT rpos+r0,cpos+c0+of;CHR$(cn)
5040 LET t(rpos,cpos)=cn
5490 RETURN
5999 REM ********************************
6000 REM *     visualizza carattere     *
6001 REM ********************************
6020 LET apos=USR (e$)-1: INK blu
6040 FOR r=1 TO c1: LET pe=PEEK(mpos+r):
     LET p=pe: LET z$=""
6060 FOR c=c1 TO 1 STEP -1: LET n=INT
     (pe/2): LET q=pe-2*n
6080 LET b(r,c)=q: LET pe=n
6100 LET r$(c)=" *"(q+1): NEXT c
6120 PRINT AT r0+r,c0+1;r$;AT r0+r,c0+c1+
     3;p;" "
6130 NEXT r
6140 RETURN
6499 REM ********************************
6500 REM *    aggiorna finestra testo   *
6501 REM ********************************
6520 FOR r= TO 8: LET y=r0+r
6540 FOR c=1 TO 8: LET x=c0+c+of
6560 PRINT AT y,x;CHR$ (t(r,c))
6580 NEXT c: NEXT r
6600 RETURN
```

BBC MICRO

```
  19 REM ********************************
  20 REM *        GENERATORE UDG        *
  21 REM *        PER  MICRO BBC        *
  22 REM ********************************
  50 MODE 1
  60 @%=3
  70 *FX4,1
 100 PROCinizializza
 110 REPEAT
 120 PROCaccetta_comando
 130 PROCconvalida_comando
 140 PROCesegue(COMANDO)
 160 UNTIL COMANDO=FINITO
 180 *FX4,0
 200 END
 999 REM ********************************
1000 DEFPROCinizializza
1001 REM ********************************
1040 DIM BD(8,8),C$(2,2)
1045 DIMOFST(2,7),D$(1),TX(8,8)
1060 D$(0)=" ":D$(1)="*"
1080 CGEN=&0C00-1
1100 FINITO=-1:ERRATO=0:MOVCUR=1:
1105 ALTER=2:DFIN=3:DEPOS=4
1200 REM*** INIZ. SCHERMO ****
1220 RO=4:CO=3:RL=8:CL=8
1225 C9=CL+3:OF=16
1230 NERO=0:ROSSO=1:GIALLO=2
1235 BIANCO=3:SFONDO=128
1240 COLOUR SFONDO+ROSSO
1245 COLOUR GIALLO:CLS
1250 Z$="CARATTERI UDG"
1255 PRINTTAB(4,1)Z$;TAB(4,20)Z$
1260 Z$=" 76543210"
1264 PRINTTAB(CO,RO)Z$
1268 PRINTTAB(CO+OF,RO)Z$
1270 Y=RO+RL+1:X=CO
1274 PRINTTAB(X,Y)Z$
1278 PRINTTAB(X+OF,Y)Z$
1280 FOR R=RO+1 TO RO+RL
1290 Z$=STR$(R-RO-1)
1295 Z$=Z$+"     "+Z$
1300 PRINTTAB(CO,R)Z$
1304 PRINTTAB(CO+OF,R)Z$
1310 NEXT R
1320 COLOUR BIANCO
1420 REM*********OFFSET  CURSORE*********
1440 DATA -1,+1,0,0,
1450 DATA 0,0,+1,-1
1460 FOR K=1 TO 2:FOR L=1 TO 4
1470 READ OFST(K,L) :NEXT L,K
1500 REM********DEFINIZIONE TASTI********
1520 *KEY1"T"
1540 *KEY2"D"
1560 *KEY3"P"
1600 REM***INIZIALIZZA FINESTRA TESTO***
1620 FOR R=1 TO RL
1640 FO C=1 TO RL
1660 TX(R,C)=32
1680 NEXT C,R
1800 RPOS=1:CPOS=1:CN=224
1805 PROCvisual_carattere
1990 ENDPROC
1999 REM********************************
2000 DEFPROCerrato
2001 REM********************************
2020 SOUND 1,-15,48,10
2120 ENDPROC
2499 REM********************************
2500 DEFPROCaccetta_comando
2501 REM********************************
2520 RP=RO+RPOS:CP=CO+CPOS
2540 PRINTTAB(CP,RP);
2620 T$=GET$
2700 ENDPROC
2999 REM********************************
3000 DEFPROCconvalida_comando
3001 REM********************************
3020 COMANDO=MOVCUR
3040 TASTO=ASC(T$)-135
3060 IF TASTO<1 THEN COMANDO=ERRATO
3065 IF TASTO>4 THEN COMANDO=ERRATO
3080 IF T$="T"THEN COMANDO=ALTER
3100 IF T$="D"THEN COMANDO=DFIN
3120 IF T$="P"THEN COMANDO=DEPOS
3150 IF T$="!"THEN COMANDO=FINITO
3160 ENDPROC
3299 REM********************************
3300 DEFPROCesegue(comando)
3301 REM********************************
3320 IF comando=FINITO THEN PROCerrato
3340 IF comando=MOVCUR THEN
     PROCmovicurs(TASTO)
3360 IF comando=ALTER THEN
     PROCalterna_cella
3380 IF comando=DFIN THEN
     PROCcrea_carattere
3400 IF comando=DEPOS THEN
     PROCmetti_carattere
3420 IF comando=ERRATO THEN PROCerrato
3450 ENDPROC
3499 REM********************************
3500 DEFPROCmovicurs(tasto)
3501 REM********************************
3520 NY=RPOS+OFST(2,Key)
3525 NX=CPOS+OFST(1,Key)
3540 IF NY>=1 AND NY<=RL THEN RPOS=NY
     ELSE PROCerrato
3560 IF NX>=1 AND NX<=CL THEN CPOS=NX
     ELSE PROCerrato
3600 ENDPROC
3999 REM********************************
4000 DEFPROCalterna_cella
4001 REM********************************
4020 TG=1-BD(RPOS,CPOS)
4025 Y=RPOS+RO:X=CPOS+CO
4040 PRINTTAB(X,Y)D$(TG)
4060 BD(RPOS,CPOS)=TG
4120 PE=?(MPOS+RPOS)
4140 PE=PE+(TG*2-1)*(2^(8-CPOS))
4160 ?(MPOS+RPOS)=PE
4200 PRINTTAB(CO+C9,RO+RPOS)PE
4220 PROCaggior_testo
4300 ENDPROC
4499 REM********************************
4500 DEFPROCcrea_carattere
4501 REM********************************
4510 REPEAT
4520 PRINTTAB(CO+2,RO+RL+3);
4525 PRINT"                "
4540 PRINTTAB(CO+2,RO+RL+4);
4545 PRINT"                "
4550 PRINTTAB(CO+2,RO+RL+4);
4560 INPUT"NUMERO (224-255)"CHNUM
4580 UNTIL CHNUM>223 AND CHNUM<256
4600 CN=CHNUM
4620 PROCvisual_carattere
4900 ENDPROC
4999 REM********************************
5000 DEFPROCmetti_carattere
5001 REM********************************
5015 X=CPOS+CO+OF:Y=RPOS+RO
5020 PRINTTAB(X,Y);CHR$(CN)
5040 TX(CPOS,RPOS)=CN
5400 ENDPROC
5999 REM********************************
6000 DEFPROCvisual_carattere
6001 REM********************************
6020 MPOS=(CN-224)*8+CGEN
6040 FOR R=1 TO CL
6045 PE=?(MPOS+R):P=PE:Z$=""
6060 FOR C=CL TO 1 STEP -1
6065 N=INT(PE/2):Q=PE-2*N
6080 BD(R,C)=Q:PE=N
6100 Z$=D$(Q)+Z$:NEXT C
6120 PRINTTA(CO+1,RO+R)Z$
6125 PRINTTAB(CO+C9,RO+R)P
6130 NEXT R
6140 ENDPROC
6499 REM********************************
6500 DEFPROCaggior_testo
6501 REM********************************
6520 FOR R=1 TO RL:Y=RO+R
6540 FOR C=1 TO RL:X=CO+C+OF
6560 PRINTTAB(X,Y)CHR$(TX(C,R))
6580 NEXT C,R
6900 ENDPROC
```

Joypad senza fili

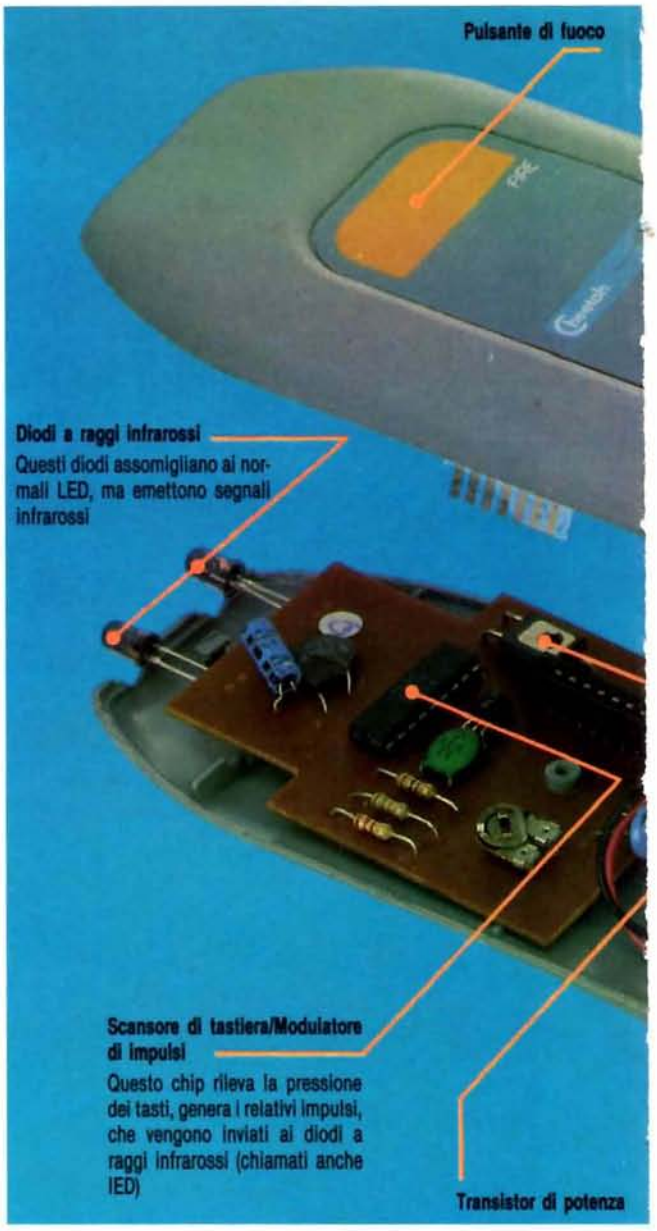
Il RAT è uno dei più sofisticati accessori per lo Spectrum

Gli appassionati di giochi "arcade" sono sempre alla ricerca di qualsiasi novità che possa migliorare le proprie prestazioni di gioco ed accrescere il punteggio finale. Proprio per questo il metodo adottato per il controllo del gioco è della massima importanza. Se viene usata la tastiera, occorre scegliere i tasti di controllo tra quelli più comodi ed accessibili e questo compito ricade sui programmatori del gioco. Se vengono usati apparecchi di controllo esterni, quali i joystick, l'elemento cruciale è soprattutto il progetto dell'hardware.

La qualità di un joystick è determinata da fattori quali la libertà di movimento, la velocità di risposta e la compatibilità col software. Questi, a loro volta, sono condizionati da alcuni limiti fisici del progetto: la lunghezza dei cavi di collegamento, le dimensioni, la forma e la posizione dei controlli e del tasto di "fuoco". Quest'ultimo è spesso collocato in modo svantaggioso per i mancini. Benché molte ditte abbiano tentato di rimuovere alcuni degli inconvenienti descritti, nessuna è forse arrivata a realizzare un accessorio di comando come il RAT, prodotto dalla Cheetah Marketing per lo Spectrum: un joypad senza fili, funzionante a raggi infrarossi.

Il nome RAT deriva dalle parole inglesi "Remote Action Transmitter" (trasmettitore di azioni a distanza) ma certamente c'è di mezzo un gioco di parole tra "rat" (ratto) e "mouse" (topo), il termine usato per descrivere un altro comando a distanza (come quello adottato, ad esempio, dal Macintosh della Apple). Il RAT ha una forma oblunga, che assomiglia alle armi laser usate nella serie di fantascienza *Star Trek*. L'involucro è colorato in grigio, mentre il disco di controllo è blu ed il pulsante di fuoco è arancione. Dalla parte frontale del RAT sporgono due diodi, che servono per trasmettere i raggi infrarossi. Le prime volte che si impugna il RAT, ci si aspetta quasi che da esso si sprigionino lunghi raggi colorati!

Il sistema include un'apposita interfaccia da innestare direttamente sul connettore posteriore dello Spectrum; essa è dotata di una porta d'espansione che consente di adoperare anche altri accessori. Sulla parte frontale dell'interfaccia è situato un unico diodo di ricezione per i raggi infrarossi, che permette di comunicare col RAT.





CHRIS STEVENS

L'INTERFACCIA RICEVENTE

Il trasmettitore del RAT, tramite un apposito circuito e speciali diodi fotoemittenti, produce un raggio di luce infrarossa, simile alla luce normale, ma la cui frequenza è molto più lunga. Nel ricevitore avviene un processo di tipo opposto: il segnale infrarosso, captato da uno speciale diodo, viene decodificato e convertito in un segnale elettrico e inviato allo Spectrum tramite l'apposita interfaccia. Questa si innesta direttamente sul connettore posteriore del computer



CHRIS STEVENS

Le istruzioni allegate spiegano come collegare ed usare il joypad, oltre ad elencare i giochi coi quali funziona il RAT (in pratica tutti i programmi che prevedono joystick compatibili con i Kempston). Molto saggiamente, la Cheetah Marketing fornisce anche le routine in BASIC, da aggiungere ai propri programmi, necessarie per il funzionamento del RAT.

La ditta produttrice afferma che il RAT può essere comodamente usato fino a più di 4 metri di distanza dall'interfaccia (purché sia, ovviamente, puntato verso di essa). I movimenti si ottengono premendo il bordo del disco blu: otto sensori, disposti lungo il perimetro, forniscono altrettanti segnali che vengono interpretati dal computer come indicatori di direzione e otto segnetti, sul disco, indicano la posizione dei sensori. Mentre una mano regge il RAT e controlla il comando di movimento, l'altra può essere impiegata per il pulsante di fuoco. Il RAT è progettato in modo da non comportare problemi per i giocatori mancini. L'alimentazione del RAT è garantita da una pila da 9 volt, alloggiata dentro lo stesso involucro, proprio sotto il disco di comando.

L'occorrente per il funzionamento del RAT è ben poco: basta collegare l'interfaccia allo Spectrum, controllare che nel RAT ci sia una pila carica e che il software preveda l'uso di joystick adatti. Dato che non viene emesso alcun segnale per assicurarsi che il RAT stia funzionando, inizialmente si è indotti ad usarlo stando letteralmente incollati al computer (molto più vicini che con qualsiasi altro joystick). Tuttavia, quando ci si rende conto che la portata del trasmettitore è di oltre quattro metri, si torna comodamente seduti sulla poltrona preferita.

In realtà, la distanza massima è stata garantita dal produttore con molta prudenza: alcune prove col RAT hanno dimostrato che funziona egregiamente anche a distanze superiori ai quattro metri e nemmeno puntando il trasmettitore in direzione dell'interfaccia (il RAT ha funzionato perfino manovrandolo da dietro la schiena, sebbene sia difficile giocare in simile posizione!).

Il RAT della Cheetah consente, dunque, la più ampia libertà di movimento al giocatore. Semmai, possono sembrare limitate le sole otto direzioni consentite dal disco di controllo, anche se sono più che sufficienti per la maggior parte dei giochi.

Il fatto che nel RAT non esistano componenti mobili, è un grosso vantaggio in quanto non è soggetto ad usura quanto lo può essere un joystick meccanico. Inoltre, il costo è molto ragionevole, considerato che lo strumento possiede un'interfaccia propria.



Ricezione mista

La tastiera del sospirato PC Junior della IBM è stata apprezzata per l'elegante e pratico design, ma soprattutto, perché ha rappresentato il primo caso di una tastiera collegata al computer tramite raggi infrarossi, anziché con il tradizionale cavetto

IL JOYPAD "REMOTE ACTION TRANSMITTER" (RAT) DELLA CHEETAH MARKETING

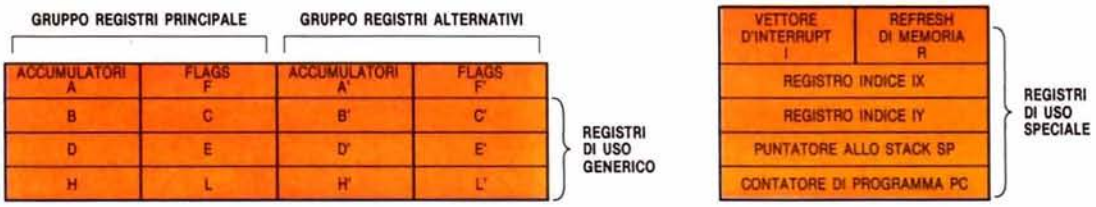
PER: Specrum ZX della Sinclair

COMPATIBILITÀ: Il RAT funziona con tutti i programmi che prevedono l'uso di joystick compatibili con i Kempston

NOTE: Funzionamento senza fili (a raggi infrarossi). La massima distanza operativa supera i quattro metri

DATABASE

Prima scheda delle istruzioni per lo Z80 della Zilog



Istruzioni	D_7 S	Z	H	P/V	N	D_0 C	Commenti
ADD A, s, ADC A, s	↕	↕	X	↕	X	V	0	↕	Somma a 8 bit o somma con riporto (Carry)
SUB s, SBC A, s, CP s, NEG	↕	↕	X	↕	X	V	1	↕	Sottrazione a 8 bit, sottr. con riporto, confronto e negazione dell'accumulatore
AND s	↕	↕	X	1	X	P	0	0	Operazioni logiche
OR s, XOR s	↕	↕	X	0	X	P	0	0	
INC s	↕	↕	X	↕	X	V	0	•	Incremento a 8 bit
DEC s	↕	↕	X	↕	X	V	1	•	Decremento a 8 bit
ADD DD, ss	•	•	X	X	X	•	0	↕	Somma a 16 bit
ADC HL, ss	↕	↕	X	X	X	V	0	↕	Somma con riporto a 16 bit
SBC HL, ss	↕	↕	X	X	X	V	1	↕	Sottrazione con riporto a 16 bit
RLA, RLCA, RRA, RRCA	•	•	X	0	X	•	0	↕	Rotazione dell'accumulatore
RL m, RLC m, RR m, RRC m, SLA m, SRA m, SRL m	↕	↕	X	0	X	P	0	↕	Rotazione e scorrimento di locazioni
RLD, RRD	↕	↕	X	0	X	P	0	•	Rotazione di un bit a sinistra e a destra
DAA	↕	↕	X	↕	X	P	•	↕	Aggiusta il risultato nell'accumulatore in forma decimale
CPL	•	•	X	1	X	•	1	•	Complemento dell'accumulatore
SCF	•	•	X	0	X	•	0	1	Pone a 1 il flag di Riporto (Carry)
CCF	•	•	X	X	X	•	0	↕	Complementa il flag di riporto (Carry)
IN r (C)	↕	↕	X	0	X	P	0	•	Legge un byte da una porta (indirizzamento indiretto)
INI, IND, OUTI, OUTD	X	↕	X	X	X	X	1	•	Input e output di blocchi tramite porta Z = 0 se B 0, altrimenti Z = 0
INIR, INDR, OTIR, OTDR	X	1	X	X	X	X	1	•	
LDI, LDD	X	X	X	0	X	↕	0	•	Trasferimento di blocchi P/V = 1 se BC 0, altrimenti P/V = 0
LDIR, LDDR	X	X	X	0	X	0	0	•	
CPI, CPIR, CPD, CPDR	X	↕	X	X	X	↕	1	•	Ricerca di blocchi Z = 1 se A = (HL), altrimenti Z = 0 P/V = 1 se BC 0, altrimenti P/V = 0
LD A, I, LD A, R	↕	↕	X	0	X	IFF	0	•	Carica nell'accumulatore il contenuto del vettore d'interrupt o di refresh
BIT b, s	X	↕	X	1	X	X	0	•	Lo stato del bit b viene copiato nel flag Z

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 r, r'	Reg.
LD r, r'	r←r'	•	•	X	•	X	•	•	•	01 r r'		1	1	4	000	B
LD r, n	r←n	•	•	X	•	X	•	•	•	00 r 110 ←n→		2	2	7	001	C
LD r, (HL)	r←(HL)	•	•	X	•	X	•	•	•	01 r 110		1	2	7	010	D
LD r, (IX + d)	r←(IX + d)	•	•	X	•	X	•	•	•	11 011 101 01 r 101 ←d→	DD	3	5	19	011	E
LD r, (IY + d)	r←(IY + d)	•	•	X	•	X	•	•	•	11 111 101 01 r 110 ←d→	FD	3	5	19	100	H
LD (HL), r	(HL)←r	•	•	X	•	X	•	•	•	01 110 r		1	2	7	101	L
LD (IX + d), r	(IX + d)←r	•	•	X	•	X	•	•	•	11 011 101 01 110 r ←d→	DD	3	5	19	111	A
LD (IY + d), r	(IY + d)←r	•	•	X	•	X	•	•	•	11 111 101 01 110 r ←d→	FD	3	5	19		
LD (HL), n	(HL)←n	•	•	X	•	X	•	•	•	00 110 110 ←n→	36	2	3	10		
LD (IX + d), n	(IX + d)←n	•	•	X	•	X	•	•	•	11 011 101 00 110 110 ←d→ ←n→	DD 36	4	5	19		
LD (IY + d), n	(IY + d)←n	•	•	X	•	X	•	•	•	11 111 101 00 110 110 ←d→ ←n→	FD 36	4	5	19		
LD A, (BC)	A←(BC)	•	•	X	•	X	•	•	•	00 001 010	0A	1	2	7		
LD A, (DE)	A←(DE)	•	•	X	•	X	•	•	•	00 011 010	1A	1	2	7		
LD A, (nn)	A←(nn)	•	•	X	•	X	•	•	•	00 111 010 ←n→ ←n→	3A	3	4	13		
LD (BC), A	(BC)←A	•	•	X	•	X	•	•	•	00 000 010	02	1	2	7		
LD (DE), A	(DE)←A	•	•	X	•	X	•	•	•	00 010 010	12	1	2	7		
LD (nn), A	(nn)←A	•	•	X	•	X	•	•	•	00 110 010 ←n→ ←n→	32	3	4	13		
LD A, I	A←I	↕	↕	X	0	X	IFF	0	•	11 101 101 01 010 111	ED 57	2	2	9		
LD A, R	A←R	↕	↕	X	0	X	IFF	0	•	11 101 101 01 011 111	ED 5F	2	2	9		
LD I, A	I←A	•	•	X	•	X	•	•	•	11 101 101 01 000 111	ED 47	2	2	9		
LD R, A	R←A	•	•	X	•	X	•	•	•	11 101 101 01 001 111	ED 4F	2	2	9		

Simboli	Operazioni
S	Flag del segno: S = 1 se il bit più significativo del risultato è 1
Z	Flag di Zero: Z = 1 se il risultato di un'operazione è 0
P/V	Flag di Parità/Overflow. I due flag condividono lo stesso bit. Le operazioni logiche influiscono sul flag di Parità, mentre quelle aritmetiche sull'Overflow (sconfinamento). Quando usato come flag di Parità, se P/V = 1 il risultato dell'operazione è pari, altrimenti dispari. Quando usato come flag di Overflow, P/V = 1 indica la condizione di Overflow.
H	Flag di Half-Carry (riporto intermedio). H = 1 se viene generato un riporto dal bit 4 dell'accumulatore dopo una somma o una sottrazione
N	Flag di Sommma/Sottrazione. N = 1 se l'ultima operazione è una sottrazione
H&N	I due flag H e N vengono usati unitamente all'istruzione DAA per portare il risultato nell'accumulatore in forma decimale (BCD) dopo un'operazione su numeri BCD
C	Flag di Carry (Riporto). C = 1 se l'ultima operazione comporta un riporto
↕	Il flag è modificato in base al risultato dell'operazione
•	Il flag non è influenzato dal risultato dell'operazione
0	Il flag viene azzerato dall'operazione
1	Il flag viene posto ad uno dall'operazione
X	Lo stato del flag è indefinito
V	Il flag P/V rispecchia l'eventuale condizione di Overflow, a seguito di un'operazione aritmetica
P	Il flag P/V rispecchia l'eventuale condizione di Parità, a seguito di un'operazione logica
r	Un qualsiasi registro a 8 bit della CPU: A, B, C, D, E, H, L
s	Una qualsiasi locazione a 8 bit
ss	Una qualsiasi locazione a 16 bit
*	Uno qualsiasi dei due registri indice IX o IY
R	Il registro di refresh della memoria
n	Un qualsiasi valore a 8 bit (compreso tra 0 e 255)
nn	Un qualsiasi valore a 16 bit (compreso tra 0 e 65535)

Note

rr'	significa uno qualsiasi dei registri A, B, C, D, E, H, L
IFF	il contenuto del flip-flop per l'abilitazione dell'interrupt è copiato nel flag P/V

Notazione per i flag

•	nessun effetto
1	flag posto a 1
0	flag azzerato
X	effetto non definito
↕	Il flag rispecchia il risultato dell'operazione

per concessione della Zilog Inc.

Variazioni su... una tartaruga

Le procedure LOGO offrono una maggiore elasticità d'uso con l'impiego di parametri

In LOGO, una parola può essere usata in tre modi diversi e, per distinguere fra i tre casi, viene usata una notazione particolare. Ad esempio, se la parola è "misura", si ha: MISURA oppure :MISURA (pronunciato "punti misura") o ancora "MISURA (pronunciato "virgolette misura"). Come già esaminato, se il LOGO incontra la parola MISURA, senza punteggiatura, la interpreta come il nome di una procedura, passando ad eseguire i comandi contenuti nella sua definizione. Nel caso di :MISURA, il LOGO prende in considerazione, invece, il valore contenuto nella variabile MISURA. Infine, la notazione "MISURA viene interpretata come un riferimento al nome stesso della procedura e della variabile, senza tener conto dei comandi o dei valori ad essa associati. Si noti l'uso insolito delle virgolette: esse precedono, ma non seguono la parola in questione: il nome è seguito da uno spazio.

Il LOGO del MIT non è completamente uniforme nell'uso di tale notazione. Ad esempio, dopo i comandi EDIT, ERASE e PO il nome della procedura non va preceduto da virgolette. Formalmente si tratta di un errore, poiché nel comando EDIT QUADRATO, ad esempio, ci si riferisce al nome della procedura e non si tratta di una chiamata alla procedura, pertanto il nome andrebbe preceduto da virgolette. Il LOGO della LCSI si dimostra più coerente, e richiede l'uso di virgolette in comandi di questo tipo.

Per usare una qualsiasi procedura LOGO da noi definita, basta digitarne il nome, così come accade per altri comandi "intrinseci", quali DRAW e HIDETURTLE. Tuttavia, in alcuni comandi occorre indicare anche certe informazioni: il comando FORWARD (avanti) non significa nulla, se non è seguito da alcuni valori, che indichino di quanto spostarsi in avanti. Inserendo il nome di una variabile all'interno di una procedura, possiamo richiamare quest'ultima specificando un valore da assegnare a tale variabile. Ciò consente di ottenere risultati diversi da una singola procedura.

Consideriamo la procedura per disegnare un quadrato, definita in precedenza:

```
TO QUADRATO
   REPEAT 4 [FD 50 RT 90]
END
```

Così com'è, la procedura traccia un quadrato, i cui lati sono lunghi 50 unità. Sarebbe molto più utile se la procedura fosse in grado di tracciare quadrati di qualsiasi dimensione. Per far ciò dovremmo specificare, di volta in volta, la misura desiderata. La modifica è presto fatta: basta sostituire il nome di una variabile, "LATO nell'esempio, alla costante 50 ed aggiungere :LATO accanto al nome della procedura. Ecco il risultato:

```
TO QUADRATO :LATO
   REPEAT 4 [FD :LATO RT 90]
END
```

Adesso la procedura va richiamata indicando un preciso valore per :LATO. Si provi QUADRATO 40, QUADRATO 10 e via dicendo, per verificare la variazione nelle dimensioni del quadrato.

Esaminiamo cosa accade quando si digita QUADRATO 30. Il LOGO, per prima cosa, cerca la definizione di QUADRATO e, dalla riga del titolo, capisce che è richiesta l'indicazione di un valore da associare a "LATO. Il valore da noi immesso (in questo caso 30) è assegnato alla variabile "LATO, dopo di che vengono regolarmente eseguiti i comandi che formano la procedura. Per meglio visualizzare il concetto, si pensi alla variabile "LATO come ad una scatola, nella quale viene depositato un valore. Quando il LOGO incontra il comando FORWARD :LATO, va a cercare nella scatola chiamata "LATO il valore da usare per il comando FORWARD. La "scatola" "LATO è usata soltanto nella procedura cui essa si riferisce: se un'altra procedura usa anch'essa "LATO come nome di variabile, viene adoperata una "scatola" diversa. In altre parole, "LATO è una variabile "locale".

Il passaggio di parametri può avvenire anche nelle sottoprocedure. La procedura CASA, definita qui sotto, è una delle tante possibili soluzioni al problema posto nella lezione precedente. Tuttavia, in questo caso il disegno può avere dimensioni fissate a piacere:

```
TO CASA :GRANDEZZA
   QUADRATO :GRANDEZZA
   FD :GRANDEZZA RT 30
   TRI :GRANDEZZA
   LT 30 BK :GRANDEZZA
END

TO QUADRATO :LATO
   REPEAT 4 [FD :LATO RT 90]
END

TO TRI :DIME
   REPEAT 3 [FD :DIME RT 120]
END
```

Sono stati usati tre diversi nomi di variabili: "GRANDEZZA, "LATO e "DIME. Si potrebbe usare un solo nome per tutte, poiché si tratta di variabili locali, ma ciò provocherebbe un'inutile confusione.

Per vedere queste procedure all'opera, proviamo a digitare CASA 30. Il LOGO legge il comando ed assegna il valore 30 a "GRANDEZZA, nella procedura CASA. È come se adesso la pri-



ma linea di questa fosse QUADRATO 30, quindi, nell'esecuzione di QUADRATO, alla variabile "LATO è assegnato il valore 30. Vale a dire che FD :LATO equivale a FD 30. Un procedimento analogo accade per la procedura TRI.
Adesso proviamo ad adattare le procedure per disegnare un reticolo di 5 x 5 caselle, in modo da poter fissare di volta in volta, tramite il metodo fin qui descritto, le dimensioni di ciascuna casella.
Ecco una procedura per disegnare poligoni, in cui il numero di lati costituisce l'input:

```
TO POLIGONO :LATI
   REPEAT :LATI [FD 50 RT 360 /:LATI]
END
```

Usando il comando POLIGONO 3 si ottiene un triangolo, con POLIGONO 4 un quadrato e così via. Tuttavia, tutti i poligoni tracciati da questa procedura hanno lati che misurano 50 unità. Una procedura di impiego più generico richiede quindi due parametri: uno per il numero di lati e l'altro per la lunghezza di questi. Bastano poche modifiche alla procedura POLIGONO:

```
TO POLIGONO :LATI :MISURA
   REPEAT :LATI [FD :MISURA RT 360/:LATI]
END
```

Adesso, POLIGONO 5 30 disegna un pentagono, i cui lati misurano 30 unità. Può esser utile modificare la procedura per il reticolo, in modo da fissare a piacimento il numero di caselle e la loro dimensione.

Variabili globali

Fin qui abbiamo esaminato soltanto le variabili locali, ossia riferite ad una singola procedura; ne esiste però un secondo tipo, le variabili "globali", accessibili a tutte le procedure. Queste variabili sono utili per passare informazioni da una procedura all'altra, ma vanno usate con cautela, poiché complicano la ricerca di eventuali errori nel programma.
Per assegnare valori alle variabili globali, esiste il comando MAKE. Pertanto, MAKE "LATO 3 assegna 3 al valore della variabile "LATO. Inoltre, con MAKE "LATO :LATO + 1, si incrementa di uno il valore di "LATO. Il significato di questa seconda notazione si può così riassumere: viene cercato il valore abbinato alla variabile "LATO, ad esso viene sommato 1 ed il risultato viene nuovamente posto in "LATO. In entrambi i casi, il comando MAKE richiede due operandi: il nome della variabile ed il valore da assegnare ad essa.
Adesso esaminiamo alcune procedure, per il disegno di spirali, che sintetizzano tutte le nozioni di programmazione esposte in questa parte del corso sul LOGO. La procedura principale è EQSPI. Essa prevede l'immissione di tre parametri: la lunghezza iniziale della linea da tracciare, l'ampiezza dell'angolo usato ad ogni cambiamento di direzione, ed il fattore di scala, da moltiplicare per la lunghezza iniziale, per ottenere l'andamento a spirale voluto. L'effetto globale può essere variato enormemente, specificando gruppi di parametri diversi. Ecco alcuni gruppi da usare per prova (ma se ne esperimentino altri ancora): 70 283 0.95, 70 143 0.95 e 20 243 1.05.
NOWRAP è un comando mai incontrato finora: esso impedisce che la tartaruga "giri" attorno allo schermo. Vale a dire che, invece di "uscire", ad esempio, dal margine destro dello schermo per "rientrare" da quello sinistro, la tartaruga si ferma e viene segnalato un errore del tipo "out of bounds" (superati i margini). In certi casi il "giro" crea effetti interessanti, ma in questa particolare procedura rovinerebbe il disegno.
La procedura principale EQSPI traccia una serie di segmenti (la lunghezza dei quali è determinata dal fattore di scala). Al termine di ciascun segmento viene eseguita una lieve rotazione e calcolato il nuovo fattore di scala. La lunghezza dei segmenti aumenta (o diminuisce) via via, a seconda che il valore del fattore scala sia maggiore o minore di 1. L'alto valore che segue il comando REPEAT serve per garantire un numero sufficiente di segmenti. Per fermare il programma, si prema Control-G (o Break).
In genere le variabili sono locali, con l'eccezione di "SCALA, che è globale. Ciò è necessario perché il valore di "AUMEN cresce di volta in volta e tale valore occorre nella procedura S.AVANTI. La variabile "SCALA serve dunque da intermediaria per le due procedure.

PROCEDURA "SPIRALE"

```
TO EQSPI :DIMEN :ANGOLO :FATT
   PREPARA
   REPEAT 1000 [S.AVANTI :DIMEN RIGHT :ANGOLO AU-
   MEN :FATT]
END
TO PREPARA
   DRAW NOWRAP MAKE "SCALA 1
END
TO AUMEN :NUMERO
   MAKE "SCALA :SCALA 1 :NUMERO
END
TO S.AVANTI :DIST
   FORWARD :SCALA 1 :DIST
END
```

Logo & Logo

La versione LOGO della LCSI prevede l'uso di FENCE (barriera) al posto di NOWRAP per evitare fuoriuscite dai margini. La versione per Atari usa il comando WINDOW (finestra) simile a FENCE e NOWRAP, salvo il fatto che il programma non si interrompe con la segnalazione di un errore quando la tartaruga tenta di uscire dai margini dello schermo.
Nella procedura per la spirale, sullo Spectrum, sostituire a DRAW (nella sottoprocedura PREPARA) il comando CS

ESERCIZI CON LE PROCEDURE 2

1) Si scriva un programma per tracciare un cerchio di raggio 50. Poi si modifichi la procedura, affinché tracci cerchi di qualsiasi raggio specificato
2) Si scriva una procedura che disegni un bersaglio, composto da cinque cerchi concentrici

Risposte agli esercizi

1) Il Tangram

Le sagome dell'uomo seduto, dell'uomo inchinato e del gatto usano tutte l'altro lato del pezzo a parallelogramma, impiegato per la sagoma del cane. Per ribaltare una figura, in LOGO, basta cambiare tutti i comandi RIGHT, usati nella sua definizione, in LEFT (o viceversa). Così:

```
TO PAR
   REPEAT 2 [FD 25 RT 45 FD 35 RT 135]
END
```

diventa:

```
TO PAR1
   REPEAT 2 [FD 25 LT 45 FD 35 LT 135]
END
```

Per i restanti pezzi le definizioni rimangono inalterate.

Uomo che corre

```
TO CORRIDORE
   MOVE1 TRI1 MOVE2 PAR MOVE3 TRI3 MOVE4
   TRI3 MOVE5 QUADRATO MOVE6 TRI1 MOVE7
   TRI2 MOVE8
END
TO MOVE1
   LT 45
END
TO MOVE2
   PU FD 25 RT 135 FD 17.5 LT 45 PD
END
TO MOVE3
   PU FD 75 RT 90 PD
END
TO MOVE4
   PU RT 90 FD 25 RT 90 PD
END
TO MOVE5
   PU FD 50 RT 135 FD 50 LT 135 PD
END
TO MOVE6
   PU RT 135 FD 21 RT 135 FD 25 LT 90 FD 50
   LT 90 FD 25 RT 90 PD
END
TO MOVE7
   PU FD 25 RT 135 FD 71 RT 45 BK 35 PD
END
TO MOVE8
   PU FD 35 LT 90 FD 25 RT 45 FD 17.5 LT 45
   FD 25 RT 135 PD
END
```

Uomo seduto

```
TO SEDUTO
   MOVE1 TRI1 MOVE2 TRI2 MOVE3 TRI3 MOVE4
   TRI1 PAR1 MOVE5 QUADRATO MOVE6 TRI3 MOVE7
END
TO MOVE1
   LT 45
END
TO MOVE2
   PU FD 25 LT 45 FD 17.5 RT 90 PD
END
TO MOVE3
   PU BK 15 LT 90 PD
END
TO MOVE4
   PU FD 50 RT 45 FD 25 RT 90 PD
END
TO MOVE5
   PU FD 25 LT 45 FD 35 LT 45 PD
END
TO MOVE6
   PU BK 50 LT 90 PD
END
TO MOVE7
   PU BK 21 RT 135 BK 50 RT 90 FD 35
   LT 90 PD
END
```

Uomo inchinato

```
TO INCHINO
   MOVE1 TRI1 MOVE2 PAR1 MOVE3 TRI3 MOVE4
   TRI3 MOVE5 TRI1 MOVE6 TRI2 MOVE7 QUADRATO
   MOVE8
END
TO MOVE1
   LT 90
END
TO MOVE2
   PU FD 25 RT 135 FD 30 PD
END
TO MOVE3
   PU LT 45 FD 35 LT 135 BK 50 PD
END
TO MOVE4
   PU RT 90 FD 50 LT 135 PD
END
TO MOVE5
   PU RT 90 FD 50 LT 135 FD 5 RT 90 PD
END
TO MOVE6
   PU RT 90 FD 25 RT 45 FD 7.5 RT 45 BK 35
   PD
END
TO MOVE7
   PU FD 35 RT 135 FD 7.5 LT 45 FD 55 RT 45
   PD
   END
TO MOVE8
   PU LT 45 FD 36 RT 45 FD 56 LT 135 FD 5
   LT 45 BK 25 PD
END
```

Gatto

```
TO GATTO
   MOVE1 TRI3 MOVE2 QUADRATO MOVE3 TRI1 MOVE4
   TRI1 MOVE5 TRI3 MOVE6 PAR1 MOVE7 TRI2
   MOVE8
END
TO MOVE1
   PU FD 50 RT 90 PD
END
TO MOVE2
   RT 170
END
TO MOVE3
   PU RT 90 FD 25 LT 90 PD
END
TO MOVE4
   RT 180
END
TO MOVE5
   PU RT 90 FD 25 LT 80 FD 50 RT 45 FD 50
   RT 90 PD
END
TO MOVE6
   LT 155
END
TO MOVE7
   PU LT 160 FD 35 PD
END
TO MOVE8
   PU FD 35 LT 45 FD 21 RT 135 PD
END
```



Ecco alcune delle possibili soluzioni agli esercizi sulle procedure, dati a pag. 1045:

2)

```
TO CASA
   QUADRATO FD 50 RT 30
   TRI LT 30 BK 50
END
TO QUADRATO
   REPEAT 4 [FD 50 RT 90]
END
TO TRI
   REPEAT 3 [FD 50 RT 120]
END
```

3)

```
TO TABELLA
   REPEAT 5 [LINEA FD 10] BK
   50
END
TO LINEA
   REPEAT 5 [QUADRATO RT
   90 FD 10 LT 90] LT 90 FD 50
   RT 90
END
TO QUADRATO
   REPEAT 4 [FD 10 RT 90]
END
```

4)

```
TO STELLA.SEI
   TRI MOVE TRI MOVE.IND
END
TO TRI
   RT 30 REPEAT 3
   [FD 50 RT 120]
END
TO MOVE
   PU FD 29 RT 60 PD
END
TO MOVE.IND
   PU LT 60 BK 29 PD
END
```

Software calibrato

Esaminiamo le tecniche per rendere più rapida l'esecuzione dei programmi

La programmazione strutturata e una buona e chiara stesura del programma facilitano la realizzazione di una procedura, ma non ne migliorano l'efficienza. Talvolta, per rendere più rapida l'esecuzione di un programma e tenere al minimo l'occupazione di memoria, occorre sacrificare una parte della chiarezza del progetto. Infatti, quando si esegue la "messa a punto" di una parte di programma, per renderlo più veloce o efficiente, si rischia sempre di comprometterne la leggibilità, la comprensione e la ricerca degli errori.

A causa dell'intrinseca lentezza di tutti i linguaggi "interpretati" (come il BASIC degli home computer) vi sono occasioni in cui l'esecuzione è talmente "pigra", che è indispensabile intervenire. La soluzione più semplice sarebbe quella di adoperare un programma compilatore. Tale opportunità non è sempre disponibile; infatti, pur esistendo diversi compilatori in commercio (sia su cassetta che su dischetto), nella maggioranza dei casi si tratta di BASIC che prevedono soltanto numeri interi e, comunque, può esser necessario rivedere completamente il formato del programma originale, prima di poterlo compilare. Lo stesso processo di compilazione richiede tempo, specialmente durante lo sviluppo di un programma e, ancor più, se il sistema è dotato soltanto di unità a cassette. Il programma compilatore occupa memoria e, tanto più esso è sofisticato, tanta più RAM consuma, portandone via al programma dell'utente. In linea di principio, la compilazione dei programmi sugli home computer dovrebbe avvenire soltanto su programmi già ben verificati e collaudati.

Ciò che rallenta maggiormente l'esecuzione di un programma è l'accesso ai file. In programmi che prevedono frequenti letture o scritture su file, i rallentamenti sono inevitabili. Un singolo accesso ad un record di un file "random", su dischetto, richiede mediamente circa un quarto di secondo. Se l'accesso è diretto ad un file sequenziale, specialmente su nastro, il tempo richiesto è notevolmente più lungo (e dipende anche dalla lunghezza dello stesso file). Un sistema per ridurre frequenti attese può esser quello di copiare un consistente blocco di dati nella RAM e rimandare la registrazione di tutte le modifiche o gli aggiornamenti a quando si esce dal programma. Molti programmi fatti passare per "interattivi", in realtà causano molte perplessità nell'operatore, poiché vi sono lunghi intervalli di tempo in cui il computer pare non far niente ed essersi bloccato. Uno stratagemma è quello di accedere ai file mentre l'operatore è intento a far qualcos'altro (a leggere una videata di istruzioni, per esempio).

Un'altra causa di lentezza è l'uso di un'aritmetica "reale" (in cui i numeri sono "reali", ossia prevedono la virgola ed alcune cifre dopo di essa). Proprio per la presenza della parte dopo la virgola, il tempo occorrente per estrarre, convertire ed operare su tali numeri è maggiore di quello occorrente per i numeri "interi" (senza virgola). Quindi conviene sempre adoperare variabili intere (ad esempio SOMMA% al posto di SOMMA) quando i valori in esse sono numeri interi. Con questo accorgimento si può arrivare ad un risparmio di tempo globale di circa il 20% nell'esecuzione del programma, e ricorrendo a sistemi di "compattazione" dei numeri si giunge addirittura al 50%.

La lentezza del programma può anche dipendere da quella degli algoritmi usati: produrne di più efficienti è questione di ingegno e creatività. I dialetti BASIC offrono una gamma di funzioni predefinite (quali INSTR, SGN, LOG e via dicendo) la cui velocità è abbastanza alta; ciò dipende dal fatto che gli algoritmi adoperati per esse sono tra i migliori e, naturalmente, scritti in codice macchina. Spesso vale la pena di consultare attentamente il manuale del proprio BASIC, per vedere quali funzioni sono già disponibili, prima di scervellarsi a reinventarne la propria versione. Se una particolare funzione non esiste, conviene definirla ricorrendo alla struttura DEF FN, disponibile in quasi tutti i BASIC: l'esecuzione è più veloce di quella per una subroutine. Le funzioni definite dall' utente, tramite DEF FN appunto, sono particolarmente utili quando in un programma ricorre spesso uno stesso calcolo o una stessa sequenza di operazioni su stringhe.

Le subroutine scritte in codice macchina garantiscono quasi sempre una maggior velocità. Infatti, come spiegato in precedenza, i linguaggi "interpreti" perdono tempo a decifrare i comandi in una linea BASIC, prima di poterla effettivamente eseguire. Una subroutine in codice macchina salta a piè pari la procedura di traduzione del BASIC. Disgraziatamente, scrivere programmi in codice macchina è più complicato che in BASIC e spesso il tempo e l'impegno necessari non ripagano quel poco che si può guadagnare in rapidità d'esecuzione. D'altro canto, in applicazioni quali i giochi impostati sull'animazione grafica, è impensabile non ricorrere a sezioni di programma in codice macchina.

Esistono altri trucchi per velocizzare i pro-

grammi. Ad esempio, conviene sempre usare una variabile al posto di un valore costante (ossia MAX invece di 363.5), specialmente nei valori usati all'interno di cicli iterativi. Nell'assegnare i nomi alle variabili, si faccia in modo che le loro lettere iniziali siano ben distribuite nell'arco dell'intero alfabeto. Si usino linee contenenti più di un comando (se il BASIC lo consente) e si lasci un adeguato intervallo (ad esempio di 10) nella numerazione delle linee. Sempre se la versione di BASIC lo permette, si tralasci il nome del contatore, nei cicli FOR...NEXT, dopo l'istruzione NEXT (vale a dire, si usi il semplice NEXT e non NEXT J o NEXT CONTO). All'interno dei cicli, infine, si eviti di includere un calcolo che non dipende dal ciclo stesso: conviene collocarlo esternamente al ciclo, ed assegnarne il risultato ad una variabile.

Economizzare spazio

L'aritmetica su numeri interi non solo riduce il tempo d'esecuzione, ma occupa anche meno memoria: laddove occorrono 5 o 6 byte per un numero reale, ne bastano due per un numero intero. La differenza è ancor più evidente quando si usano matrici di considerevoli dimensioni. Alcuni accorgimenti che velocizzano il programma servono anche ad economizzare memoria: le funzioni predefinite (o quelle definite da utente) occupano meno spazio, lo stesso accade con subroutine in codice macchina e con linee contenenti quanti più comandi BASIC possibile. La compilazione di un programma, in genere, tende ad allungare i programmi molto brevi, mentre il risparmio di spazio su quelli lunghi può essere notevole.
Anche l'eliminazione di tutte le REM e, ovviamente, l'uso di messaggi e scritte più brevi portano a risparmiare byte. Se non si possono abbreviare le scritte, va considerata la possibilità di raggrupparle tutte (o parte di esse) in un file, da leggere e visualizzare all'occorrenza. A questa categoria appartengono, ad esempio, tutte le scritte relative a istruzioni o a messaggi d'assistenza. Altri accorgimenti consistono nel rimuovere il maggior numero possibile di spaziature tra i comandi, numerare le linee con valori bassi ed impiegare nomi brevi per le variabili. Se una matrice va dimensionata, ma non si conosce con esattezza il numero di elementi, si eviti di adoperare un valore a caso, quasi certamente superiore all'effettiva necessità: si dimensioni la matrice quando l'informazione è nota o viene immessa, tramite una variabile. Come nel seguente esempio:

```
10 INPUT "Quanti elementi in questa categoria?";ELEM%
20 DIM MATRIC%(ELEM%)
```

Questo metodo è chiamato "dimensionamento dinamico". In genere in linguaggi diversi dal BASIC non è possibile applicarlo, quindi, visto che qui è possibile, conviene sfruttarlo!
Un'altra tecnica prevede un'intervento sulla quantità di RAM accessibile da parte del BASIC e ciò si ottiene usando comandi del tipo HIMEM. Di solito, questo genere di comandi varia il rapporto tra la quantità di memoria dedicata al programma e quella riservata alle variabili e viene usato per proteggere una subroutine in codice macchina da eventuali sovrascritture. Tuttavia, se non preoccupa "sporcare" momentaneamente lo schermo, si può tentare di estendere l'uso della RAM anche alla zona riservata al video: è sempre qualche "kappa" in più! Qualora non sia disponibile un comando HIMEM si può sempre ricorrere a delle POKE e PEEK, riferite anch'esse alla RAM del video.
Se ogni altro tentativo dovesse fallire, in alcuni BASIC esiste il comando CHAIN, col quale è possibile accodare automaticamente l'esecuzione di un programma a quella di un altro. Abbinato a CHAIN, in genere, c'è il comando COMMON (o qualche sistema equivalente) che permette al secondo programma di usare le variabili inizializzate da quello precedente, mettendole in "comune".
Se un programma è redatto in modo strutturato, l'esecuzione di ogni singola routine può esser sottoposta ad una verifica di durata. Ecco un semplice programma per effettuare tale verifica:

```
100 REM Questa prima sezione va usata per
105 REM inizializzare tutte le variabili
110 REM occorrenti alla subroutine da esaminare
115 REM (si rammenti di dimensionare le matrici
120 REM e di inizializzare anche queste con opportuni
125 REM valori). Qui viene usata la variabile TIME,
130 REM supponendo che in essa venga scandito il tempo
135 REM in centesimi di secondo (come sul Micro BBC)
200 INIZIO = TIME
210 GOSUB 2000: REM chiamata alla routine da verificare
220 FINE = TIME
230 PRINT "Tempo impiegato: ";(FINE-INIZIO)/100;" secondi."
240 END
```

Questo breve programma consente di verificare l'efficienza di un algoritmo o di confrontare l'effettivo risparmio di tempo introdotto da eventuali modifiche alla procedura.

PROGRAMMI... A TAVOLETTA!

- Si consideri attentamente se conviene rinunciare ad un corretto stile di programmazione, rendendo eventualmente il programma illeggibile, a favore di qualche scorciatoia
- Si compili il programma se si può, altrimenti si ricorra alla definizione di funzioni o procedure
- Si eviti, quando possibile, l'accesso ai file
- Si evitino numeri reali. Inizializzare ed usare variabili intere se possibile
- Si progettino con cura gli algoritmi, confrontandone la rapidità con altri
- Si considerino i pro e i contro della programmazione in codice macchina: l'esecuzione è più veloce, ma lo sviluppo di una routine può richiedere molto tempo
- Ultimato e collaudato il programma, si eliminino tutte le REM e le spaziature inutili

Cambio d'indirizzo

I segreti dell'indirizzamento indicizzato tramite i registri X e Y del 6809

Una delle ragioni per cui furono concepiti i programmi per computer deriva da un'idea molto elementare: se una serie di istruzioni risiede nello stesso luogo (e con lo stesso formato) dei dati sui quali operano tali istruzioni, il programma può automodificarsi durante la stessa esecuzione. Lo scopo principale di questo sistema non è tanto di modificare le stesse istruzioni, quanto di variare gli indirizzi ai quali accedere per estrarre e manipolare i dati. Si immagini che problema si dovrebbe affrontare se, dovendo accedere a una tabella contenente svariate migliaia di valori, fosse necessario scrivere un'istruzione di richiamo per ciascuno di questi valori!

Il concetto di "indirizzamento variabile" ha indubbiamente consentito di ridimensionare questo insormontabile problema. In base ad esso, una stessa istruzione può essere adoperata per accedere ai dati depositati in qualsiasi area di memoria, semplicemente variando l'indirizzo in un particolare registro del computer. Questo metodo è usato più spesso di quanto non si creda. Ad esempio:

```
FOR I = 1 TO N
PRINT ELEM(I)
NEXT I
```

In questo semplice ciclo, l'istruzione PRINT viene adoperata N volte, variando ad ogni passaggio il dato da visualizzare, ossia ELEM, tramite l'indice (I).

Il principio che è alla base dell'"indirizzamento indicizzato" è simile: il contenuto di un registro indice viene di volta in volta sommato ad un indirizzo di base, per ottenere l'effettivo indirizzo di una particolare locazione di memoria. Qualora l'operazione avvenga all'interno di un ciclo iterativo, il registro indice può essere incrementato (o decrementato) opportunamente, per consentire l'accesso ad un'intera tabella di valori.

Il microprocessore 6809 non solo possiede due appositi registri indice (X e Y), ma anche altri due registri (S e U) utilizzabili all'occorrenza. Addirittura, in certe occasioni, si può anche usare il contatore di programma. Ad ampliare le cose (o a complicarle, a seconda dei punti di vista) esiste tutta una gamma di variazioni dell'indirizzamento indicizzato, tali da soddisfare praticamente tutte le esigenze di programmazione. Da questo punto in poi, verranno esaminati ed impiegati quasi tutti questi metodi, e si avrà quindi la possibilità di apprezzarne i pregi e gli svantaggi.

L'indirizzamento indicizzato è caratterizzato dalla presenza del registro X (o Y) nel campo degli operandi. Il formato tipico delle istruzioni indicizzate è:

```
Op-code     Offset, registro indice
LDA         TABEL1,X
STA         TABEL2,Y
```

In certe occasioni, l'offset (o scarto) è nullo e viene quindi omesso, come in:

```
Op-code     , registro indice
LDA         ,X
STA         ,Y
```

Vediamo come tutto ciò funzioni nella pratica. Supponiamo di avere una tabella di 64 valori ad otto bit, a partire dalla locazione $3000, e di voler accedere a ciascun byte sequenzialmente. Possiamo definire l'indirizzo di base e riservare lo spazio per la tabella con:

```
            ORG     $3000
TABEL       RMB     64
```

Queste istruzioni assegnano l'indirizzo $3000 al contatore di locazione, e definiscono un'area di 64 byte, chiamata TABEL, a partire dalla locazione $3000. Le istruzioni che seguono servono per accedere alla tabella. Il nuovo valore attribuito alla ORG significa che il codice viene depositato in una zona diversa della memoria e questa è una precauzione da prendere nei primi tentativi di indirizzamento indicizzato, per evitare che il programma venga distrutto da accidentali sovrascritture.

```
            ORG     $1000
CONTO       FCB     0
            LDX     #0
CICLO       LDA     TABEL,X
```

Adesso modifichiamo il valore del registro X:

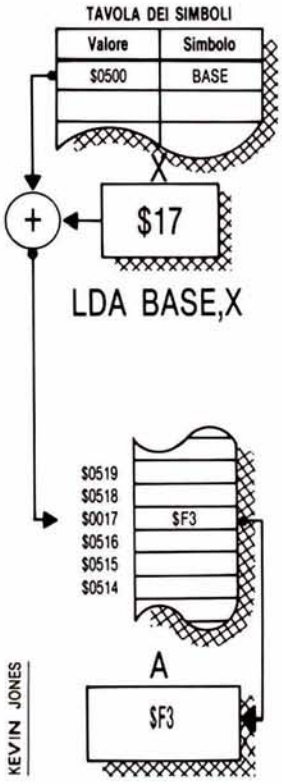
```
            TFR     X,D
            ADDD    #1
            TFR     D,X
```

Questo è un sistema contorto per incrementare X di una sola unità, ma torna utile per valori maggiori (più avanti nel corso vengono descritti altri metodi). L'ultima sezione di codice aggiorna un contatore e controlla che questo non sia arrivato a 64. In tal caso il programma termina, poiché non occorre ripetere il ciclo:

```
            INC     CONTO
            LDB     CONTO
            CMPB    #64
            BLT     CICLO
```

Collegamenti indicizzati

Il simbolo BASE (inizializzato col valore $0500) rappresenta l'indirizzo del primo byte di una tabella di valori. L'istruzione LDA BASE,X, appartenente al gruppo di quelle progettate per l'indirizzamento indicizzato, prende l'indirizzo BASE e vi somma il contenuto del registro X, per ottenere l'effettivo indirizzo del byte che viene poi caricato nell'accumulatore. Se tale istruzione si trova all'interno di un ciclo in cui X funge da contatore, allora si può accedere sequenzialmente all'intera tabella, un byte alla volta. Poiché X è un registro a 16 bit, il ciclo potrebbe spaziare sull'intero arco di memoria (da $0000 a $FFFF) previsto da un sistema ad 8 bit, quale il chip 6809



KEVIN JONES

Esistono diversi modi per migliorare l'efficienza di questo programma e uno tra i più utili è quello di usare il metodo dell'"autoincremento". L'istruzione:

```
LDA     TABEL,X+
```

provoca un incremento del registro X, subito dopo l'esecuzione dell'istruzione stessa. Se la tabella è composta da valori a 16 bit, si usa allora:

```
LDA     TABEL,X++
```

che provoca un duplice incremento di X. Il ciclo del nostro programma originale diventa così altamente lineare:

```
CICLO   LDA     TABEL,X+
        INC     CONTO
        LDB     CONTO
        CMPB    #64
        BLT     CICLO
```

Un'utile alternativa al metodo di accesso descritto consiste in una lettura in ordine inverso, magari adoperando il metodo dell'"autodecremento". Ciò offre il vantaggio di ottenere in X il valore 0, una volta esauriti tutti gli elementi della tabella. Inoltre, il decremento automatico del registro indice influenza lo stato dei flag del processore, cosicché l'istruzione di confronto, CMP, risulta superflua: si può direttamente verificare se X è giunto a zero. Un risultato identico, tuttavia, si ottiene inizializzando il registro X con un opportuno valore negativo: esso crescerà fino a raggiungere lo zero. Il fatto che l'incremento (o decremento) automatico modifichi lo stato dei flag consente di accorciare notevolmente i programmi. Nonostante ciò, si rammenti che in generale è buona norma adoperare l'accumulatore per confronti e verifiche, poiché tra l'istruzione di incremento/decremento e l'effettiva verifica dei flag possono essere interposte altre istruzioni che alterano lo stato dei flag. Scartando la possibilità di caricare il registro X con un valore negativo, è utile esaminare il caso in cui X progressivamente decresce fino a zero.

Un'importante distinzione da ricordare è che nell'autodecremento il registro indice è modificato *prima* di calcolare l'indirizzo, mentre nell'autoincremento l'indirizzo è incrementato *dopo* il calcolo. Pertanto, se X contiene 7 e TABEL inizia da $1000, allora l'istruzione LDA TABEL,X+ carica nell'accumulatore il contenuto di $1007 per poi incrementare X da 7 a 8. LDA TABEL,-X (si noti il segno meno prima di X), invece, decrementa X da 7 a 6 e poi carica nell'accumulatore il contenuto di $1006.

Muovendosi all'indietro e usando il registro B come contatore, il nostro ciclo diventa:

```
        LDX     #64
        LDB     #64
CICLO   LDA     TABEL,-X
        DECB
        BGE     CICLO
```

Il primo dei due programmi portati ad esempio (qui sotto) mostra l'uso di un ciclo per scorrere in avanti una tabella di valori ad 8 bit. Contemporaneamente, viene contato il numero di valori negativi.

Il contatore nel registro B, inoltre, viene usato come offset per il valore costante in X. Il secondo programma mostra il simultaneo impiego dei due registri indice, ma con un offset pari a zero, per copiare una sequenza di caratteri da una zona di memoria (ad esempio, il "buffer" della tastiera) ad un'altra. La lunghezza della stringa è ignota, ma inferiore a 255 byte, e termina con un codice di ritorno carrello (0D hex). Tale carattere viene eliminato nella nuova copia della stringa, ma nel primo byte di essa viene depositata la lunghezza della stessa stringa.

CONTATORE DI VALORI NEGATIVI

• Dalla locazione $3001 in poi è conservata una tabella di valori ad otto bit. Il numero dei valori, inferiore a 256, è situato nella locazione $3000. Questo programma conta quanti valori negativi ci sono nella tabella

```
TABEL   EQU     $3000     TABEL contiene il numero di elementi
        ORG     $1000
NEGS    FCB     0         NEGS è una locazione usata per tenere il conto dei valori negativi
        LDB     TABEL
        LDX     # TABEL   L'indirizzo della tabella va in X
CICLO   LDA     B,X       Estrae il primo valore dalla tabella
        BGE     FINEC     Salta a FINEC se il valore è positivo
        INC     NEGS      Altrimenti incrementa il conto dei valori negativi
FINEC   DECB              Decrementa il contatore del ciclo
        BGT     CICLO     Salta a CICLO se esistono altri elementi
        SWI               Altrimenti torna al sistema operativo
        END
```

COPIA DI UNA STRINGA DI CARATTERI

• Questo programma copia una sequenza di caratteri che inizia da $3000 in una zona di memoria prefissata, che inizia a $0010

```
CR      EQU     13        Codice ASCII per il ritorno carrello
STRING1 EQU     $3000     Indirizzo della stringa originale
STRING2 EQU     $0010     STRING2 è l'indirizzo al quale copiare la stringa
        ORG     $1000
        LDX     #STRING1  Indirizzo della stringa originale nel registro X (inizialmente $3000)
        LDY     STRING2   Indirizzo di destinazione (inizialmente $10,$11) nel registro Y
        TRF     Y,U       Trasferisce l'indirizzo del byte per la lunghezza in U
        LDA     #0        Deposita uno zero nella prima locazione della destinazione
        STA     ,Y+
CICLO   LDA     ,X+       Prende il successivo carattere della stringa originale
        CMPA    #0        È un Ritorno Carrello?
        BEQ     FINITO    Fine del programma se lo è
        STA     ,Y+       Altrimenti lo copia
        INC     ,U        Incrementa la lunghezza
        BRA     CICLO     Salta ad estrarre un nuovo carattere
FINITO  SWI               Torna al sistema operativo
        END
```

Classic racing

Una "corsa ippica" della Salamander Software per l'Oric-1

In Gran Bretagna, le corse di cavalli sono considerate lo "sport dei Re". Non c'è da meravigliarsi, dunque, se la ditta inglese Salamander Software ha prodotto un gioco su questo argomento. In effetti, il giocatore assume il ruolo di un allenatore, per tutta la durata di una "stagione". Il programma prevede da uno a sei giocatori, ma anche se il loro numero è inferiore al massimo, i fantini sono sempre sei. Un'intera stagione è composta da 16 manifestazioni, ognuna delle quali conta 6 gare, ma può esser scelta una stagione di minor durata. L'obiettivo del gioco (cioè guadagnare il più possibile) si può raggiungere in due modi: vincendo i premi abbinati ai tre cavalli meglio addestrati (dal giocatore) oppure semplicemente puntando sui risultati. Il giocatore è sempre abbinato ad un cavallo di sua scelta, ma nulla vieta di puntare su quello di un altro giocatore, se lo si ritiene più forte.

La scuderia del giocatore comprende 16 cavalli e, all'inizio della stagione, i loro meriti sono ignoti. Poiché all'inizio della stagione i premi sono bassi, questo è il momento di sperimentare i propri cavalli su terreni e distanze diverse. Qui si tratta unicamente di applicare un metodo empirico: la strategia va impostata sull'osservazione del comportamento di ciascun cavallo. Per far ciò occorre prender buona nota di tutta una serie di componenti: la distanza percorsa, il peso trasportato, le condizioni del terreno e, ovviamente, il risultato della corsa. Peccato che il programma non preveda la stampa di tali informazioni: la fatica sarebbe certo inferiore.

Una volta selezionati i sei cavalli da far correre nella prima gara, vengono forniti i dati sugli avversari: i nomi dei cavalli e il peso del fantino. A questo punto, il computer decide la proporzione da assegnare alle vincite abbinate a ciascun cavallo. All'inizio della stagione la scelta può apparire casuale, ma in seguito ci si accorge che dipende effettivamente dalle doti del cavallo. Scommettere è obbligatorio (le "poste" vanno da 5 a 500 sterline) e le vincite (ovviamente fittizie) possono essere considerevoli.

Nella strategia di gioco si può anche prevedere di far correre un cavallo in specialità per le quali non è adatto: si è sicuri di perdere, ma ciò fa salire il rapporto tra puntata e vincita abbinato al cavallo. Facendolo in seguito correre nella sua specialità le scommesse sono più redditizie. Non si cada nella tentazione di far correre sempre il miglior cavallo: come accade nella realtà, deve essergli concesso qualche periodo di riposo!

Dopo tutti questi preliminari, si passa alla gara vera e propria: le sequenze grafiche sono eccellenti e molto realistiche, tanto che sembra quasi di assistere ad una gara in un vero ippodromo (si sente perfino il rumore degli zoccoli!). I giocatori più appassionati non mancheranno di infervorarsi mentre seguono il proprio cavallo sulla pista.

Terminata la gara, vengono "pagate" le puntate e si passa alla gara successiva, fino a conclusione di una manifestazione. Ciascuna di queste prevede terreni e specialità diverse, offrendo così un'ampia varietà al gioco. L'eliminazione di un cavallo da una squadra è semplice: basta non farlo correre per le manifestazioni successive.

Verso la fine della stagione, vincere le gare diventa sempre più difficile; i giocatori infatti hanno perfettamente inquadrato le possibilità dei propri cavalli, quindi non rischiano di farli correre se hanno poche probabilità di vincere. Di pari passo, aumentano anche i premi corrisposti ai tre allenatori migliori: nel leggendario "Derby", corso durante l'ultima manifestazione, il premio raggiunge le 90.000 sterline!

Classic racing è uno dei migliori giochi mai concepiti per l'Oric e l'Atmos: la grafica è molto accattivante e la strategia da applicare è talmente varia, che il gioco può essere ripetuto innumerevoli volte, senza mai stancarsi. In una sola stagione, si possono vincere fino a 250.000 sterline: il problema rimane come ritirare la vincita!



Passaggi vincenti

In qualsiasi momento del gioco si può consultare il programma delle corse (figura in alto), in cui sono riportate le caratteristiche di ciascuna manifestazione. Selezionati i cavalli per una corsa, i giocatori hanno la possibilità di fare le proprie puntate (seconda figura dall'alto) e l'entità di queste va da 5 a 500 sterline. A questo punto, i cavalli vengono allineati alla partenza (terza figura) in attesa del "via". La corsa è completamente gestita dal computer: ciò accresce la tensione nei giocatori che non possono far altro che attendere ansiosamente il risultato della gara. L'ultima immagine mostra i cavalli che stanno rallentando, dopo aver superato il traguardo

Classic racing: per Oric-1 e Atmos con 48 Kbyte
Produttori: Salamander Software
Autore: Paul Neal
Joystick: non richiesti
Formato: cassetta

Paperi e droidi

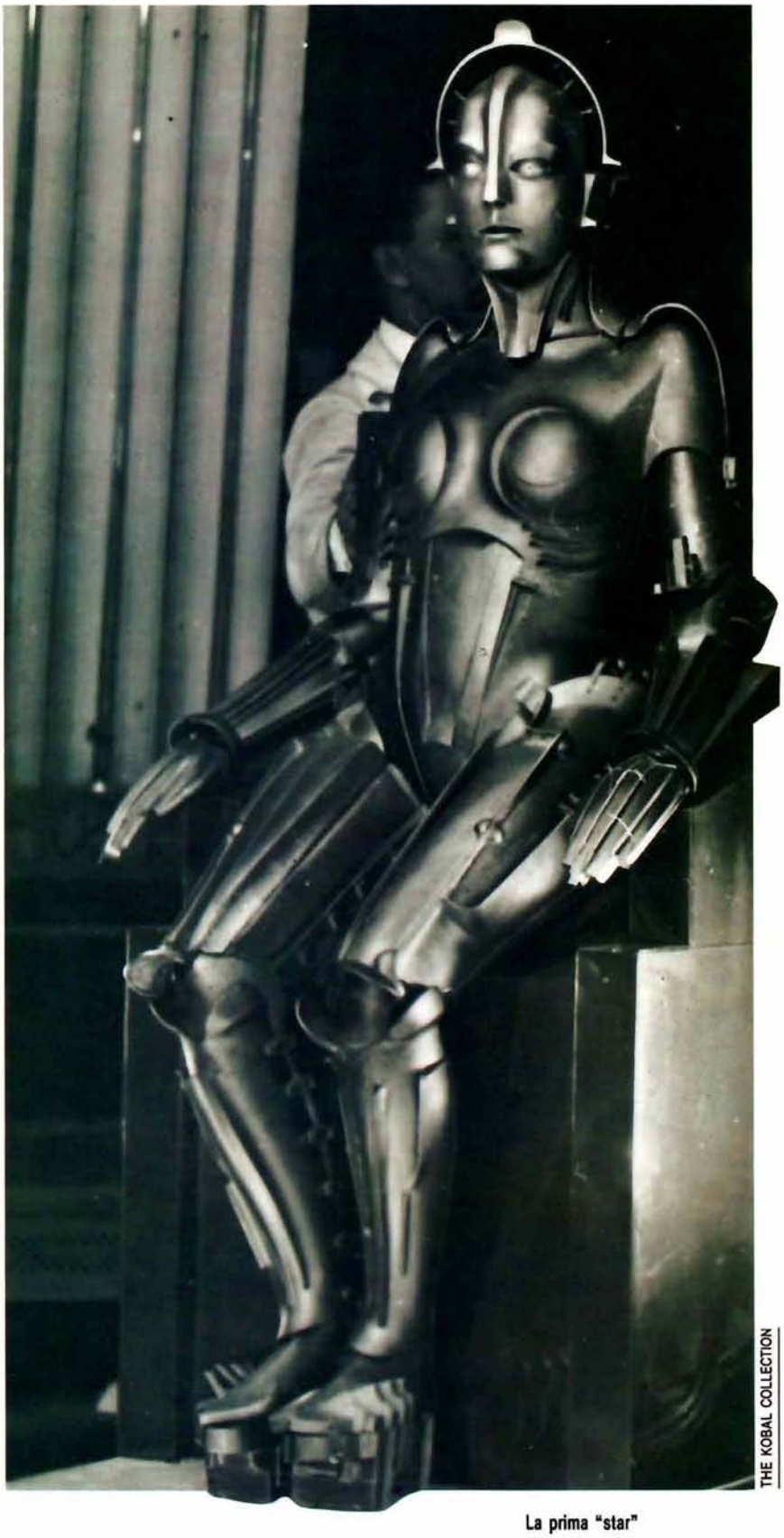
Dai primi animali meccanici del XVIII secolo ai sofisticati robot del futuro

Nel corso di centinaia di anni, l'uomo è stato in vario modo affascinato dall'idea di poter creare una copia meccanica di se stesso. Filosofi, ingegneri ed inventori hanno pensato e cercato di fabbricare macchine che imitassero fedelmente il comportamento degli uomini. Raramente gli odierni robot possiedono un aspetto umano e, in genere, vengono programmati per eseguire solo particolari tipi di lavoro; invece nei primi "uomini meccanici" la somiglianza con l'uomo era molto più accentuata.

Nonostante ciò, il prodotto di uno dei primi tentativi di cui si hanno notizie non aveva sembianze umane. Nel 1738, l'ingegnere francese Jacques de Vaucanson (1709-1782) presentò un papero meccanico all'Accademia Reale delle Scienze di Parigi. Il papero muoveva le ali, emetteva versi ed era capace di "mangiare". Verso la fine del XVIII Secolo, l'inventore svizzero Pierre Jaquet-Droz (1721-1790) fabbricò una serie di pupazzi meccanici capaci di muoversi: uno scriveva, un altro eseguiva disegni, un terzo suonava brani musicali su di un organo. Dalla fine del XIX secolo sono state create moltissime macchine di questo tipo, tutte basate su meccanismi a molla.

A cavallo tra i due secoli fecero la loro comparsa numerosi robot meccanici di grande realismo, alcuni dei quali non funzionavano a molla. Nel 1893, George Moore costruì un uomo meccanico che funzionava a vapore: il vapore veniva fatto uscire dalla bocca e dava l'impressione che il robot stesse fumando un sigaro.

Lo sviluppo di nuove tecnologie ha permesso, in seguito, la creazione di macchine più sofisticate: dal semplice uomo meccanico in grado di camminare, fabbricato con una scatola di Meccano, al famoso Elektro, costruito dalla compagnia americana Westinghouse. Elektro era un uomo meccanico alto 2 metri e 15 che poteva pronunciare 80 parole, contare, camminare, conversare, salutare e distingue-



THE KOBAL COLLECTION

La prima "star"
"Metropolis", realizzato da Fritz

mobile. In effetti, uomini meccanici di questo tipo non erano che macchine alquanto complicate: le loro azioni non richiedevano capacità decisionali ed erano totalmente privi di quella che oggi viene chiamata intelligenza artificiale.

Robot nella Fantascienza

Ad un certo punto, gli inventori e gli ingegneri abbandonarono il campo mentre la fantasia degli scrittori di fantascienza trovò nei robot un'inesauribile fonte d'ispirazione. In effetti, gran parte del loro fascino continua ad essere alimentato proprio dai romanzi di fantascienza. Nel 1923, lo scrittore cecoslovacco Karel Čapek (1889-1938) scrisse un racconto intitolato *R.U.R.* (il titolo era un'abbreviazione di Rossum's Universal Robots). La trama descriveva la creazione di un uomo meccanico talmente perfetto da poter imitare qualsiasi attività umana. Il racconto si concludeva con la presa di potere da parte dei robot, che si accorsero di non aver più alcun bisogno degli uomini. In Cecoslovacco la parola "robota" significa lavoratore ma, dopo l'uscita del romanzo, questo termine ha assunto un significato completamente diverso ed oggi viene universalmente usato per indicare qualsiasi uomo meccanico con capacità paragonabili a quelle umane.

Uno dei primi romanzi fantastici in cui si parla di creature costruite a somiglianza dell'uomo è *Frankenstein*, il famoso racconto scritto da Mary Shelley nel 1818. Il mostro creato dal dottor Victor Frankenstein non era un uomo meccanico, bensì un macabro essere composto da parti umane provenienti da più corpi. Al contrario, gli invasori marziani descritti nel romanzo *La Guerra dei Mondi* di H.G. Wells erano, almeno in parte, dei robot.

I romanzieri del XX secolo hanno descritto fin nei minimi particolari mondi fantastici popolati da robot. Il contributo più significativo è stato dato da Isaac Asimov, il famoso scrittore di fantascienza che già nel 1940 scriveva brevi storie sui robot e sui loro problemi di funzionamento. Il mondo dei robot immaginato da Asimov era così realistico che, nel suo *Manuale della Robotica*, formulò le "Tre Leggi della Robotica", ossia i principi che dovrebbero governare il comportamento di tutti i robot.

I robot hanno fatto la loro comparsa anche nel cinema ed in televisione. La nota serie televisiva *Dr Who* è popolata di Daleks e Cyberman e nel film *Guerre Stellari* i robot C3PO e R2D2 sono dei veri e propri protagonisti. Tuttavia, le reali applicazioni dei robot sono molto meno eccitanti: di grande attualità sono i robot industriali, massicciamente impiegati nelle catene di montaggio delle automobili.

Un' indagine di mercato afferma che, alla fine del 1985, i robot industriali installati in Giappone saranno ben 25.000, negli Stati Uniti 15.000, in Germania Ovest 8.000 e che il mercato europeo dei robot industriali è destinato a crescere in modo travolgente.

Queste apparecchiature che verniciano o assemblano in modo ripetitivo i pezzi di un'automobile hanno poco a che fare con i robot descritti dalla fantascienza. Molte sono le ipo-

Fantasia e realtà

I più noti robot televisivi sono probabilmente i Daleks della serie "Dr Who", veri e propri carrelli corazzati dalle sembianze vagamente umane che i loro creatori controllano dall'interno.

Robbie il Robot, protagonista del film "Il Pianeta Proibito", rappresenta invece il robot dolce e sensibile, una figura ricorrente in molti romanzi di fantascienza.

Topo, il robot domestico fabbricato per un certo periodo dalla Prism, è stato un tentativo più o meno serio di introdurre la robotica tra le mura domestiche



I Daleks





Robbie il Robot

tesi che vengono fatte sui futuri sviluppi della robotica ma è difficile, per il momento, dire se i robot del domani avranno un aspetto più umano di quelli attuali.
Questa breve panoramica sulla storia della robotica introduce un tema di vastissima risonanza che verrà in seguito maggiormente approfondito.

GLOSSARIO

Questo breve elenco riporta alcuni dei più comuni termini usati, soprattutto nella fantascienza, per indicare robot di vario tipo. Detto per inciso, molti dei termini riportati si riferiscono a macchine puramente immaginarie!

Androide: Un robot che ha caratteristiche e capacità vagamente umane.

Antropomorfico: Letteralmente "simile all'uomo". Un androide può essere definito come antropomorfico mentre molti robot presentano solo limitate rassomiglianze con l'uomo, come per esempio, i bracci meccanici vagamente simili alle braccia umane.

Automazione: Il controllo automatico di un processo produttivo.

Automa: Una macchina dotata di un meccanismo di funzionamento nascosto, che di solito esegue solo una serie di azioni programmate in precedenza (i primi uomini meccanici erano automi). Il termine assume un significato più tecnico nella definizione "Teoria degli automi", che indica un sistema analitico per lo studio e la descrizione di apparecchi di diverso tipo (robot, computer ecc.).

Cibernetica: La scienza che studia i sistemi di controllo e comunicazione. Sviluppata da Norbert Weiner nel 1947, la cibernetica viene applicata anche allo studio dei sistemi biologici.

Cybert: Un umanoide meccanico della letteratura fantascientifica.

Cybot: Un robot dotato di intelligenza artificiale (fantascienza).

Cyborg: Organismo cibernetico. Una via di mezzo tra l'uomo e la macchina.

Doppelgänger: Adoperato talvolta come sinonimo di "Clonazione", ossia la creazione di una copia esatta di un essere vivente.

Droide: Un robot che ubbidisce alle Tre Leggi della Robotica concepite da Asimov.

End effector: Terminologia corrente per indicare la "mano" di un robot industriale.

Omunculi: Piccoli uomini o manichini.

Manipolatore: Un altro termine per la "mano" di un robot industriale.

Metal-collar workers: Robot industriali. Letteralmente, "colletti metallici". Nei paesi anglosassoni gli impiegati vengono spesso chiamati "colletti bianchi" e gli operai "colletti blu". È quindi naturale che i robot industriali siano stati soprannominati "colletti metallici".

Robot: Una macchina in grado di svolgere alcune delle attività tipiche dell'uomo il cui aspetto non è necessariamente umano.

Robotica: La scienza che studia i robot.

LE "LEGGI DELLA ROBOTICA" DI ASIMOV

1. Un robot non deve recar danno ad un essere vivente o, attraverso una sua omissione d'azione, permettere che ciò accada.
2. Un robot deve ubbidire agli ordini impartiti da un essere umano, purché essi non contravvengano alla Prima Legge.
3. Un robot deve proteggere la propria esistenza purché, nel far ciò, non vengano infrante la Prima e la Seconda Legge.



Il "mondo" di Asimov

Con le "Tre Leggi della Robotica", Asimov ha codificato il comportamento dei robot che popolano i suoi romanzi. Nella realtà i robot non hanno ancora simili doti, né sono in grado di distinguere gli esseri umani dagli altri oggetti inanimati



Ford · La linea di assemblaggio della "Sierra"

Fiat · Assemblaggio degli assali di torsione



Assemblaggio robotizzato

I robot industriali vengono impiegati soprattutto nelle linee di produzione meccaniche. Infatti, la loro economicità li rende particolarmente adatti alle catene di montaggio, come già avviene alla Ford e alla Fiat. Di solito questi robot sono macchine ad alta specializzazione con uno o due bracci dotati di utensili di vario genere

COS'È UN ROBOT?

La definizione data nel glossario alla parola robot è volutamente generica e può essere applicata, per esempio, ai computer (che eseguono operazioni di calcolo tipiche del cervello umano).
Di solito però, un robot possiede alcune appariscenti caratteristiche pseudo-umane: può muoversi o addirittura camminare, possiede uno o due bracci simili alle braccia dell'uomo, può vedere ed udire. Infine, può mostrare un certo grado di "intelligenza".
L'aspetto esteriore e le capacità dei robot dipendono essenzialmente da due fattori: il loro compito e il modo in cui lo devono eseguire. Ad esempio, un robot industriale per saldature è una macchina "statica" non perché non la si possa far muovere, ma perché il suo compito è di mantenere una posizione fissa e saldare dei pezzi. Un robot domestico, invece, può essere capace di fare una tazza di tè, benché non sia in grado di salire le scale senza rovesciarla!
La parola robot viene oggi genericamente usata per tutte le macchine che imitano un'attività umana. I limiti dei robot dipendono dalle modalità di progettazione e costruzione, ma la gamma e la complessità delle loro attività cresce di giorno in giorno

La ricursione del LOGO

I risultati ottenibili con procedure che richiamino se stesse sono spettacolari



Uno dei primi programmi LOGO sviluppati era una procedura per il disegno di un quadrato. Il programma faceva avanzare la tartaruga per un certo tratto, quindi, dopo una rotazione a destra di 90 gradi, faceva ripetere queste due operazioni altre tre volte. La seguente procedura traccia lo stesso quadrato, ma con un altro metodo:

```
TO QUADRATO
   FD 50
   RT 90
   QUADRATO
END
```

La tartaruga continua a muoversi sul perimetro, finché non viene battuto BREAK o Control-G. La cosa più interessante da notare è che questa nuova procedura QUADRATO "richiama se stessa": in altre parole essa è "ricursiva". Quando viene eseguita, il LOGO chiama la definizione di QUADRATO e comincia ad eseguire le istruzioni. La tartaruga si sposta in avanti di 50 unità (FORWARD 50) e ruota a destra di 90 gradi (RIGHT 90). L'istruzione successiva è QUADRATO per cui il LOGO chiama nuovamente la definizione di QUADRATO e la esegue. Se il programma non viene interrotto, questo ciclo prosegue all'infinito.
È possibile usare chiamate ricursive anche in procedure che prevedono il passaggio di parametri:

```
TO POLIGONO :LATO :ANGOLO
   FD :LATO
   RT :ANGOLO
   POLIGONO :LATO :ANGOLO
END
```

Questa permette di tracciare tutti i poligoni già definiti (vedere a pag. 1025) ed altri ancora (assegnando all'angolo di rotazione, ad esempio, un valore di 89 gradi). Inoltre, nella chiamata ricursiva, è possibile modificare i valori dei parametri:

```
TO POLISPIR :LATO :ANGOLO
   FD :LATO
   RT :ANGOLO
   POLISPIR (:LATO + 5) :ANGOLO
END
```

L'unica differenza tra questa procedura e POLIGONO è l'aggiunta di 5 al valore del lato dopo ogni chiamata ricursiva. Quindi, iniziando con POLISPIR 10 90, la prima chiamata fa tracciare alla tartaruga una linea di lunghezza 10, la seconda di 15, la terza di 20 e così via. Come risultato si ha una spirale. Altri valori di partenza potrebbero essere: 10 95, 10 120, 10 117. Si può anche provare a modificarla, sostituendo all'addizione una sottrazione o una moltiplicazione.
Nella seguente procedura, invece, viene incrementato il valore dell'angolo di rotazione:

```
TO SPIRALE :LATO :ANGOLO :INC
   FD :LATO
   RT :ANGOLO
   SPIRALE :LATO (:ANGOLO + :INC) :INC
END
```

Anche in questo caso, si provi a usare diversi valori iniziali: 5 0 7, 10 40 30, 15 2 20 e via dicendo. Perché alcune figure risultano chiuse ed altre no? È possibile definire una regola generale? La semplice ripetizione di una parte del programma viene detta "iterazione". Per creare un'iterazione, il LOGO usa il comando REPEAT, mentre in altri linguaggi sono previste strutture del tipo FOR...NEXT, REPEAT...UNTIL e WHILE...WEND. Tuttavia, nel LOGO l'impiego della ricursione è molto più frequente di quello della iterazione. Se si è abituati ad usare altri linguaggi, si incontra una certa difficoltà ad abbandonare l'uso dell'iterazione: la grafica della tartaruga è quindi lo strumento ideale per abituarsi ad usare le chiamate ricursive.

L'istruzione primitiva "STOP"

Tutte le procedure ricursive viste finora si ripetono all'infinito: occorre quindi un metodo per fermarle al momento giusto. La procedura QUADRATO, ad esempio, deve essere interrotta dopo che la tartaruga ha tracciato un quadrato completo ed è tornata alla posizione iniziale 0. Per far ciò si deve introdurre nella procedura l'istruzione "primitiva" STOP:

```
TO QUADRATO
   FD :LATO
   RT 90
   IF POSIZIONE = 0 THEN STOP
   QUADRATO :LATO
END
```

Le nuove primitive sono STOP e IF. STOP interrompe l'esecuzione di una procedura e restituisce il controllo alla procedura di chiamata (assomiglia alla RETURN del BASIC). IF (se) rappresenta il modo usato dal LOGO per prendere una decisione. La IF è sempre seguita da una condizione, mentre THEN da un'operazione che viene eseguita se la condizione è vera.

Osserviamo adesso cosa accade quando viene usata la primitiva STOP in una versione modificata di POLISPIR:

```
TO POLISPIR :LUNGHEZZA
   IF :LUNGHEZZA > 15 THEN STOP
   FD :LUNGHEZZA
   RT 90
   POLISPIR (:LUNGHEZZA + 5)
END
```

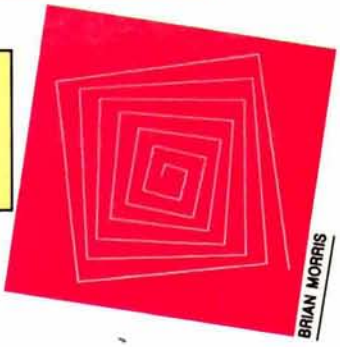
Impartendo il comando POLISPIR 10, si richiama la procedura POLISPIR e alla variabile locale LUNGHEZZA viene assegnato il valore 10. Poiché questo è minore di 15, il LOGO fa eseguire alla tartaruga il movimento FD 10 RT 90 e chiama nuovamente POLISPIR, che adesso ha un valore di 15. Dato che la variabile locale LUNGHEZZA è minore di 15, la tartaruga esegue un secondo movimento FD 15 RT 90, dopodiché il LOGO torna a chiamare POLISPIR. Stavolta, però, il valore della variabile LUNGHEZZA è 20, quindi l'esecuzione della procedura si arresta ed il controllo passa alla procedura precedente (POLISPIR 15). Questa ha raggiunto a sua volta la linea finale e trasferisce il controllo alla procedura chiamante. Ma anche questa è giunta al termine, quindi il programma si conclude.
È importante sottolineare che le chiamate ricursive sono possibili soltanto in presenza della procedura originale, pur lavorando in modo spesso completamente diverso. Terminato il loro compito, esse trasferiscono il controllo alla procedura che le ha chiamate. Per meglio chiarire questo procedimento di "ritorno", modifichiamo POLISPIR nel modo seguente:

```
TO POLISPIR :LUNGHEZZA
   IF :LUNGHEZZA > 15 THEN STOP
   POLISPIR (:LUNGHEZZA + 15)
   FD :LUNGHEZZA
   RT 90
END
```

Questa procedura traccia la spirale in senso inverso, cioè partendo dall'esterno e portandosi verso il centro (questo modo di procedere risulta più chiaro se nell'istruzione di condizione inseriamo un valore maggiore di 15: ad esempio, 50). Negli esempi precedenti, prima veniva tracciata una linea e poi il controllo veniva passato a un'altra procedura.
In questo caso, invece, il LOGO prima chiama tutte le procedure ricursive, quindi comincia a tracciare la figura. L'ultima procedura creata (che presenta il valore massimo nella variabile locale LUNGHEZZA) è la prima ad essere eseguita.
Si rammenti, infine, che la ricursione è una tecnica che consuma una discreta quantità di memoria. Le procedure nelle quali la chiamata ricursiva si trova nell'ultima linea, lavorano in modo più efficiente. Dato che questa tecnica non occupa spazio di memoria extra, non ci si deve preoccupare di quante volte viene ripetuta la chiamata e conviene quindi usarla il più spesso possibile.

ESERCIZIO 3

Scrivere una procedura ricursiva che tracci una serie di quadrati sovrapposti e , di volta in volta, dimezzi la lunghezza del lato

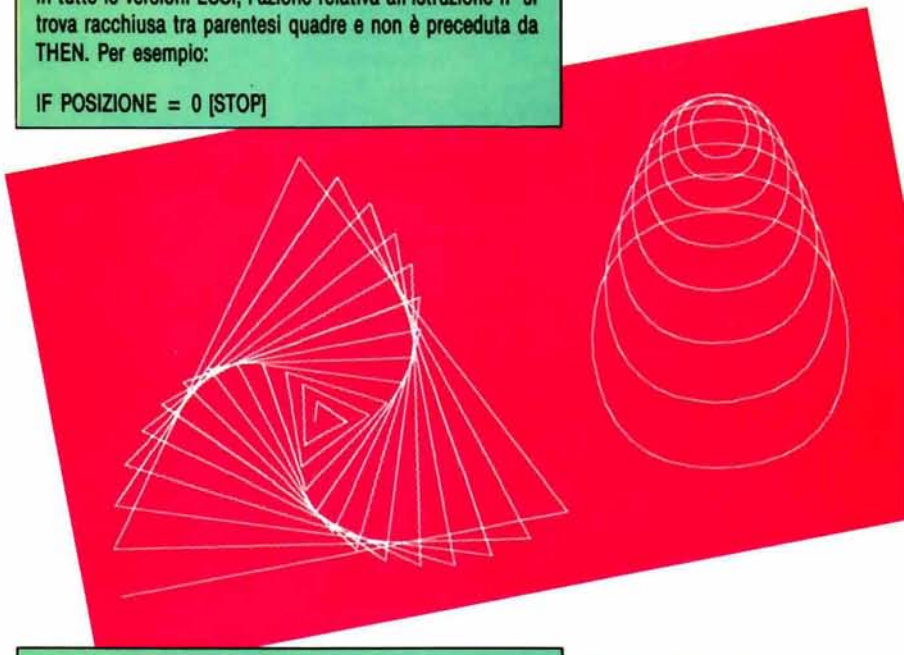




Logo & Logo

In tutte le versioni LCSI, l'azione relativa all'istruzione IF si trova racchiusa tra parentesi quadre e non è preceduta da THEN. Per esempio:

IF POSIZIONE = 0 [STOP]



SOLUZIONI AGLI ESERCIZI

La seguente procedura traccia un cerchio, il cui raggio viene passato come parametro. La posizione di partenza della tartaruga è un punto sul cerchio:

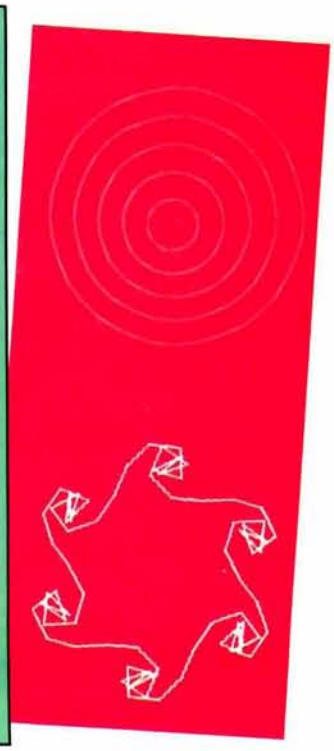
```
TO CERCHIO :RAGGIO
   REPEAT 36 [FD(2* :PI* :RAGGIO/36) RT 10]
END
MAKE "PI (3.14159)
```

Volendo modificare la procedura, in modo che il centro del cerchio corrisponda alla posizione di partenza della tartaruga, si ha:

```
TO C.CERCHIO :RAGGIO
   PU LT 90 FD :RAGGIO RT 90 PD
   CERCHIO :RAGGIO
   PU LT 90 BK :RAGGIO RT 90 PD
END
```

La procedura BERSAGLIO usa C.CERCHIO per tracciare 5 cerchi concentrici:

```
TO BERSAGLIO
   C.CERCHIO 10 C.CERCHIO 20 C.CERCHIO 30
   C.CERCHIO 40 C.CERCHIO 50
END
```



Il check-up di un programma

Alcuni aspetti della verifica e del collaudo finale del proprio software

Uno dei principali vantaggi nella programmazione con linguaggi "interpretati" (anziché "compilati") è che i programmi possono essere provati durante la loro stesura; il programmatore può impartire il comando RUN in qualsiasi momento ed osservare cosa accade. Sulla maggior parte dei computer è facile interrompere l'esecuzione di un programma, visualizzare i valori contenuti in variabili di particolare interesse e, dopo averli eventualmente modificati, riprendere l'esecuzione. Questo procedimento permette di individuare e correggere immediatamente la maggior parte degli errori, ma non può sostituire il "debugging" finale del programma.

Un importante collaudo è quello di accertarsi che il programma faccia esattamente ciò che deve fare. Vale a dire che deve produrre i giusti risultati per qualsiasi serie di dati "legali" in ingresso, mentre deve applicare appropriate procedure d'errore per quelli "illegali".

Un semplice metodo per controllare un programma potrebbe essere quello di introdurre un dato legale di prova ed accertarsi che il risultato ottenuto sia quello atteso. Purtroppo questa tecnica non è sempre applicabile: nella maggior parte dei casi il collaudo dovrebbe prevedere la verifica di tutti i possibili valori, legali ed illegali, anche nel caso di un semplice programma che, immessi due numeri, li sommi e stampi il risultato.

Un'altra tecnica prevede l'analisi di tutti i "percorsi logici" del diagramma di flusso del programma. Ogni blocco decisionale determina due percorsi alternativi ed ogni ciclo iterativo ne aggiunge degli altri. La figura 1 mostra un programma il cui ciclo principale presenta numerose istruzioni IF...THEN. Il ciclo contiene quattro percorsi e viene eseguito 10 volte; quindi il numero dei percorsi possibili durante l'esecuzione è 1.398.100: un numero enorme, sebbene il programma sia lungo solo una decina di linee. Di conseguenza anche questa tecnica va scartata.

Da quanto detto, si può dedurre che non esiste un metodo per verificare un programma alquanto complesso in modo assoluto ed in un tempo ragionevole. Perciò l'operazione di verifica segue il principio della cosiddetta "risposta decrescente"; la verifica va interrotta quando il costo dell'impegno richiesto supera quello provocato da eventuali (e non individuati) errori.

SENTIERI SELVAGGI

Anche un programma semplice come questo ciclo iterativo non può essere verificato in modo sicuro per il gran numero delle possibili combinazioni di valori in ingresso: il flusso di controllo può infatti seguire più di un milione di diversi percorsi!

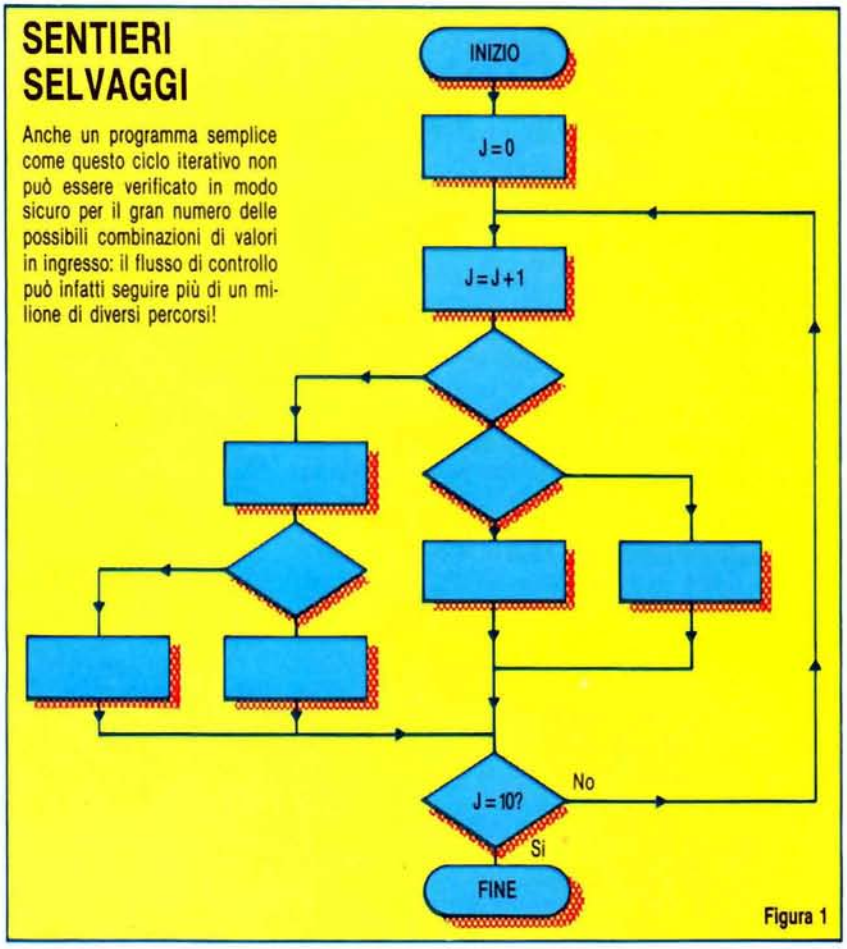


Figura 1

Nonostante tutto, si può formulare qualche teoria di verifica. Se il computer, ad esempio, elabora correttamente il dato di un particolare gruppo (ad esempio, un numero intero positivo), è logico supporre un analogo comportamento per tutti i dati dello stesso tipo. Su questo principio poggia la cosiddetta "verifica per classi equivalenti", che prevede l'impiego di una serie di dati campione, ciascuno rappresentativo di un gruppo i cui elementi dovrebbero comportarsi tutti allo stesso modo. Quindi, se una sezione del programma controlla che il valore immesso sia compreso tra 1 e 100, la procedura dovrà verificare se i dati in ingresso sono maggiori, minori oppure compresi tra 1 e 100 (valore < 1; valore > 100; 1 = < valore < 100).

L'esame di tutti i percorsi logici può essere semplificato richiamando separatamente ogni routine (teoricamente ogni routine dovrebbe possedere un solo ingresso) e prevedendo tutti i possibili risultati delle istruzioni decisionali in essa contenute. Nella figura 2 è mostrata una routine che gestisce i "premi" (bonus) in un gioco. Alla routine vengono passati i parametri BONUS, LIVELLO e CENTRI, in base ai quali essa calcola il nuovo valore per BONUS:

```
6030 IF LIVELLO>2 AND CENTRI=10 THEN BONUS=BONUS
     *LIVELLO
6040 IF LIVELLO=6 OR BONUS>2000 THEN BONUS=
     BONUS+100
```

Per prevedere l'uscita di entrambe le espres-

Tabella di controllo

Questa tabella riporta una serie di valori di controllo, calcolati a mano, per la routine descritta dai diagrammi di flusso alla pagina seguente:

DATI IN INGRESSO			USCITA
LIVELLO	CENTRI	BONUS	BONUS
6	10	200	1300
4	10	550	2300
7	10	550	3950
4	10	200	800
7	10	200	1400
1	20	2500	2600
1	20	550	550
6	5	200	300
6	50	200	300
4	5	2500	2600
7	50	2500	2600
4	50	550	550
7	5	550	550

sioni decisionali, si devono quindi considerare i valori in ingresso che possono dare come risultato un "vero" o un "falso" e si deve valutare l'effetto "combinato" delle due variabili legate da un operatore logico (AND e OR). Vediamo cosa accade se verifichiamo il primo blocco di decisione con i valori 1 e 4 per la variabile LIVELLO e 10, 5 e 20 per la variabile CENTRI. Se LIVELLO=4, vengono esaminati anche i tre valori di CENTRI mentre ciò non avviene per LIVELLO=1. In questo caso si dice che la prima parte dell'espressione decisionale "maschera" la seconda. È quindi consigliabile scomporre l'espressione e controllarne separatamente le varie parti.

Il diagramma di flusso della figura 3 presenta quattro blocchi decisionali a due uscite con $2^4=16$ possibili uscite. La seguente tabella mostra tutti i possibili stati di ciascun blocco decisionale per le uscite "vero" o "falso":

	1	2	3	4
Vero	LIVELLO>2	CENTRI=10	LIVELLO=6	BONUS>2000
Falso	LIVELLO=2 LIVELLO<2	CENTRI<10 CENTRI>10	LIVELLO<6 LIVELLO>6	BONUS=2000 BONUS<2000

La tabella viene poi usata per calcolare i valori per i dati di verifica significativi. Ad esempio, affinché il flusso del programma segua il percorso "adfi" (vedere fig. 3), LIVELLO deve essere maggiore di 2 e uguale a 6, CENTRI diverso da 10 e BONUS può avere qualsiasi valore (perché non risulta coinvolto). Ad esempio, i valori LIVELLO=6, CENTRI=10 e BONUS=600 verificano il percorso "adfi" mentre il percorso "abehi" può essere controllato con LIVELLO=4, CENTRI=10 e BONUS=600 (non si dimentichi che stiamo parlando del valore d'ingresso di BONUS, che solo successivamente può essere moltiplicato per LIVELLO).

Per un più corretto modo di procedere, i risultati per ciascun gruppo di valori di controllo andrebbero calcolati prima di eseguire il test di verifica, col quale si ottengono i valori per il confronto. I dati forniti in ingresso permettono di appurare se il programma funziona. Per accertarsi che esegua il suo compito in modo corretto, è necessario calcolare in precedenza, a mano, quali siano i risultati da ottenere. La serie completa di casi, per l'esempio in questione, è riportata nella tabella alla pagina precedente.

Avendo concepito un ragionevole sistema di analisi, occorre adesso un metodo per ridurre la complessità di programmi molto più complicati. Questa esigenza mette in luce un altro vantaggio della programmazione strutturata. Infatti, i programmi composti da una serie di moduli indipendenti, organizzati in modo gerarchico, permettono di verificare ciascun modulo individualmente. Si può iniziare da quello gerarchicamente più elevato e, procedendo verso il basso, adoperare i dati che si ottengono dai moduli già controllati per verificare quelli di gerarchia inferiore.

Ogni modulo sottoposto a verifica, quindi, avrà sopra di sé (a meno che non sia il primo) un altro modulo, chiamato "pilota". I moduli successivi, detti "tronconi" (non ancora verificati), non sono affidabili e spesso vengono simulati con brevi frammenti di programma che, richiamati dal modulo sottoposto a verifica, rispondono trasmettendo valori appropriati. Una simile struttura, che accoglie le routine del modulo per la verifica, è detta "imbrigliatura di controllo". La figura 4 ne chiarisce il funzionamento. I moduli 1, 2 e 3 sono già stati verificati, mentre i moduli 4, 5 e 6 sono simulazioni.

Per concludere, la fase di verifica è un momento importante del ciclo di vita di un programma e, come tale, ha bisogno di un'esauriente documentazione. Conviene quindi annotarsi i valori di controllo ottenuti dalle varie routine: se in seguito compare un errore, non si dovranno ripetere gli stessi test e la nuova verifica potrà limitarsi a quella parte di programma il cui funzionamento è dubbio.

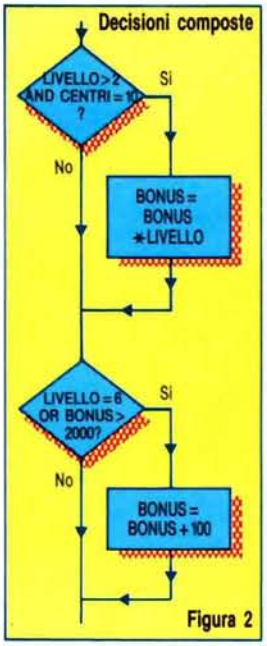


Figura 2

La "mascheratura"

La verifica è facilitata semplificando le decisioni complesse e contrassegnando i percorsi

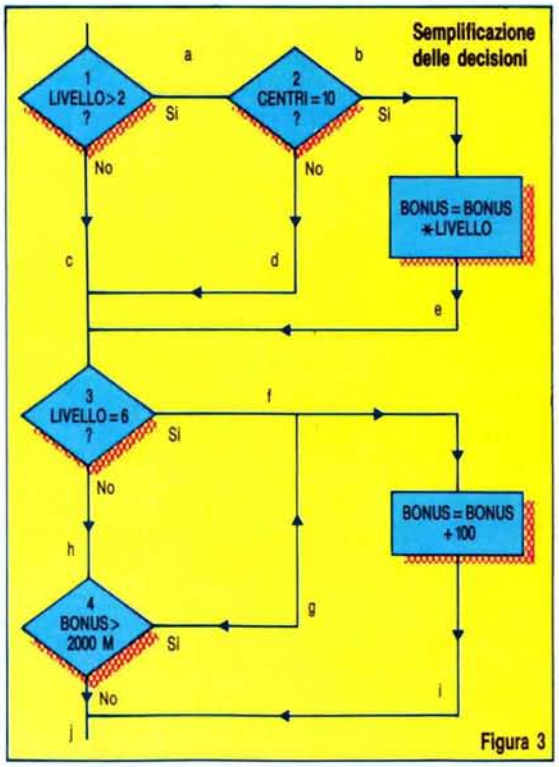


Figura 3

Verifica Top-Down

La tecnica Top-Down (dall'alto in basso) semplifica le operazioni di verifica, perché consente di esaminare ciascun modulo, durante la sua stessa stesura, individualmente o abbinato ad altri già controllati. I moduli non ancora verificati sono detti "tronconi", e spesso sono costituiti da frammenti di programma che generano semplicemente i valori attesi dai moduli già completati

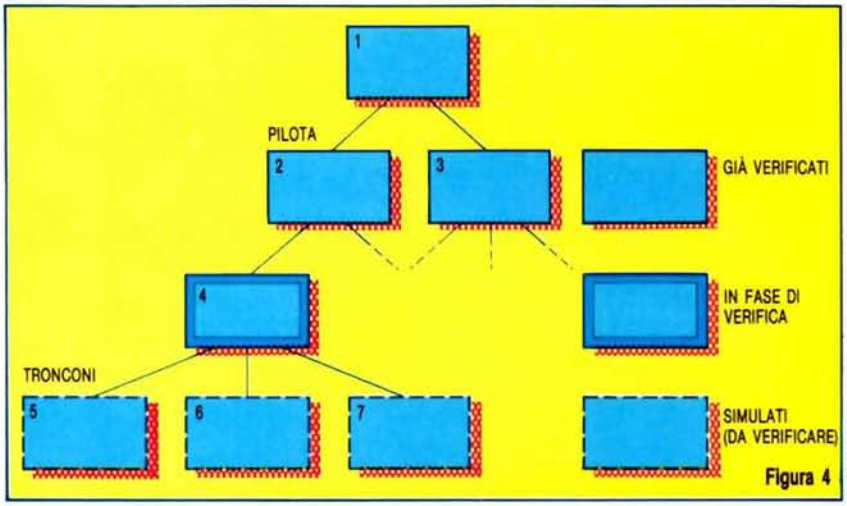


Figura 4

LIZ DIXON

DATABASE

Seconda scheda delle istruzioni per lo Z80 della Zilog
Gruppo delle istruzioni a 16 bit

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
LD dd, nn	dd←nn	•	•	X	•	X	•	•	•	00 dd0 001 ←n→ ←n→		3	3	10	dd Coppia 00 BC 01 DE 10 HL 11 SP
LD IX, nn	IX←nn	•	•	X	•	X	•	•	•	11 011 101 00 100 001 ←n→ ←n→	DD 21	4	4	14	
LD IY, nn	IY←nn	•	•	X	•	X	•	•	•	11 111 101 00 100 001 ←n→ ←n→	FD 21	4	4	14	
LD HL, (nn)	H←(nn + 1) L←(nn)	•	•	X	•	X	•	•	•	00 101 010 ←n→ ←n→	2A	3	5	16	
LD dd, (nn)	dd_A←(nn + 1) dd_B←(nn)	•	•	X	•	X	•	•	•	11 101 101 01 dd1 011 ←n→ ←n→	ED	4	6	20	
LD IX, (nn)	IX_A←(nn + 1) IX_B←(nn)	•	•	X	•	X	•	•	•	11 011 101 00 101 010 ←n→ ←n→	DD 2A	4	6	20	
LD IY, (nn)	IY_A←(nn + 1) IY_B←(nn)	•	•	X	•	X	•	•	•	11 111 101 00 101 010 ←n→ ←n→	FD 2A	4	6	20	
LD (nn) HL	(nn + 1)←H (nn)←L	•	•	X	•	X	•	•	•	00 100 010 ←n→ ←n→	22	3	5	16	
LD (nn), dd	(nn + 1)←dd_A (nn)←dd_B	•	•	X	•	X	•	•	•	11 101 101 01 dd0 011 ←n→ ←n→	ED	4	6	20	
LD (nn), IX	(nn + 1)←IX_A (nn)←IX_B	•	•	X	•	X	•	•	•	11 011 101 00 100 010 ←n→ ←n→	DD 22	4	6	20	
LD (nn), IY	(nn + 1)←IY_A (nn)←IY_B	•	•	X	•	X	•	•	•	11 111 101 00 100 010 ←n→ ←n→	FD 22	4	6	20	
LD SP, HL	SP←HL	•	•	X	•	X	•	•	•	11 111 001	F9	1	1	6	
LD SP, IX	SP←IX	•	•	X	•	X	•	•	•	11 011 101 11 111 001	DD F9	2	2	10	
LD SP, IY	SP←IY	•	•	X	•	X	•	•	•	11 111 101 11 111 001	FD F9	2	2	10	
PUSH qq	(SP—2)←qq_B (SP—1)←qq_A SP→SP—2	•	•	X	•	X	•	•	•	11 qq0 101		1	3	11	qq Coppia 00 BC 01 DE 10 HL 11 AF
PUSH IX	(SP—2)←IX_B (SP—)←IX_A SP→SP—2	•	•	X	•	X	•	•	•	11 011 101 11 100 101	DD E5	2	4	15	
PUSH IY	(SP—2)←IY_B (SP—1)←IY_A SP→SP—2	•	•	X	•	X	•	•	•	11 111 101 11 100 101	FD E5	2	4	15	
POP qq	qq_A←(SP + 1) qq_B←(SP) SP→SP + 2	•	•	X	•	X	•	•	•	11 qq0 001		1	3	10	
POP IX	IX_A←(SP + 1) IX_B←(SP) SP→SP + 2	•	•	X	•	X	•	•	•	11 011 101 11 100 001	DD E1	2	4	14	
POP IY	IY_A←(SP + 1) IY_B←(SP) SP→SP + 2	•	•	X	•	X	•	•	•	11 111 101 11 100 001	FD E1	2	4	14	

Note

dd indica una qualsiasi delle coppie di registri BC, DE, HL, SP
qq indica una qualsiasi delle coppie di registri AF, BC, DE, HL
nn indica un qualsiasi valore binario a 16 bit

$(COPPIA)_A$ e $(COPPIA)_B$ si riferiscono, rispettivamente, agli otto bit più significativi ed agli otto meno significativi del registro.
Ad esempio: AF_A corrisponde al registro A
BC_B corrisponde al registro C

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

PROVENIENZA

DESTINAZIONE		REGISTRO AF	BC	DE	HL	SP	IX	IY	EST. IMM. nn	IND. EST. (nn)	REG. IND. (SP)
REGISTRO	AF										F1
	BC								01 n n	ED 4B n n	C1
	DE								11 n n	ED 5B n n	D1
	HL								21 n n	2A n n	E1
	SP				F9		DD F9	FD F9	31 n n	ED 7B n n	
	IX								DD 21 n n	DD 2A n n	DD E1
	IY								FD 21 n n	FD 2A n n	FD E1
INDIRIZZO ESTERNO	(nn)		ED 43 n n	ED 53 n n	22 n n	ED 73 n n	DD 22 n n	FD 22 n n			ISTRUZIONI POP
REGISTRO INDIRETTO	(SP)	F5	C5	D5	E5		DD E5	FD E5	ISTRUZIONI PUSH		

NOTA: Dopo l'esecuzione delle istruzioni Push e Pop il valore dello Stack Pointer (SP) viene automaticamente aggiornato

per concessione della Zilog Inc.

Il PX-8

Anatomia di un microcomputer a diffusione europea: tecnologie a confronto



Il PX-8 ha le dimensioni di un elenco telefonico e pesa circa 2,3 chilogrammi; la custodia è in due tonalità di beige e possiede sul davanti una maniglia metallica estraibile. A prima vista, l'apparecchio somiglia molto poco ad un computer. La tastiera è protetta da un coperchio che va rimosso completamente prima dell'uso. Un'interruttore a slitta (evidenziato dalla dicitura "UNLOCK") sblocca il display mobile, consentendo di ruotarlo verso l'alto. Facendo ciò si accede anche ad un minuscolo registratore a microcassette. Il pannello che contiene il display può assumere 11 diverse posizioni ma solo 5 o 6 consentono una buona visione dello schermo.

La tastiera possiede 72 tasti e la loro funzione è indicata da diversi colori. Nei modelli per il mercato inglese e statunitense, i tasti alfanumerici di colore grigio scuro sono disposti secondo il formato QWERTY (nella tastiera inglese il simbolo della sterlina sostituisce il carattere # del modello americano). È disponibile anche una versione con tastiera di tipo francese. In realtà, modificando gli interruttori DIP del PX-8 (procedimento descritto nel manuale per l'utente), qualsiasi tastiera può utilizzare l'intero set dei caratteri "internazionali". Quattro tasti arancioni controllano i movimenti del cursore. Infine, ci sono i tasti Insert, Delete e Home, tre tasti di sistema (Escape, Pause e Help) e cinque tasti di funzione programmabili. La tastiera è ben costruita e di facile uso, soprattutto se l'apparecchio viene tenuto sulle ginocchia. L'uso è meno confortevole sul piano di una scrivania, per via dell'inclinazione dei tasti. Esistono due piedini retrattili che inclinano il PX-8, ma l'inconveniente rimane. I tasti Caps Lock, Number e Insert sono "a scatto" e vengono usati per cambiare i modi del display: tre piccoli LED rossi indicano il modo selezionato.

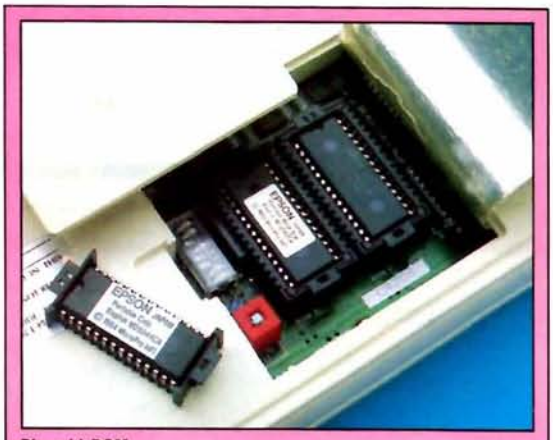
I due manuali Epson di corredo sono chiari ed esaurienti. Il primo è un manuale per l'utente che descrive la messa a punto dell'apparecchio, l'uso dell'hardware e del software e il funzionamento del CP/M; contiene

Ultimo modello

IL PX-8 è un computer da viaggio fabbricato dalla Epson, una casa nota per le sue stampanti ad aghi, per il portatile HX-20 e per il computer gestionale QX-10. Il PX-8 ha una RAM di 64 K e uno schermo a LCD, usa il diffusissimo sistema operativo CP/M, per il quale esiste una vasta scelta di software



Il display
Il PX-8 possiede uno schermo a LCD di 8 linee x 80 caratteri, alimentato a pile. La risoluzione grafica è di 480 x 64 pixel



Ricambi ROM
Sollevando un pannello, si accede ad uno zoccolo sul quale innestare le "capsule" contenenti il software su ROM. Su uno di questi è situato il sistema operativo CP/M. Per passare dal Wordstar al Calc, ad esempio, basta sostituire una capsula ROM

inoltre una serie di mappe della memoria, tabelle dei caratteri disponibili e dei loro codici, assieme ad un lungo programma in codice macchina per salvare e caricare da disco la grafica. Il secondo manuale è una guida completa alla programmazione in BASIC. Nella prima parte spiega come caricare ed usare il BASIC (fornito su di un chip ROM a "capsula"), quindi descrive le modalità di programmazione ed i vari modi di display del PX-8 e infine illustra nei dettagli tutti i comandi BASIC disponibili.
Nel PX-8 è usata una CPU CMOS compatibile con lo Z80. Questo chip CMOS consuma molta meno corrente dei chip standard e, abbinato al display LCD (Liquid Crystal Display) a basso consumo, permette di alimentare l'apparecchio a pile. Il PX-8 è corredato di due unità di alimentazione ricaricabili: una viene usata per fornire corrente al sistema, l'altra è di riserva. La batteria principale va messa sotto carica per circa 8 ore prima di essere collegata all'apparecchio. Ogni carica ha una durata di circa 15 ore e l'unità di alimentazione ha una "vita" media di 3 o 4 anni.
Dopo aver acceso l'apparecchio, occorre inizializzare il sistema operativo (la procedura è descritta in dettaglio nel manuale) inserendo la data e l'ora ed eseguendo poche altre operazioni di gestione interna, come la formattazione della "RAM disk". Il PX-8 ha la possibilità di accantonare un'area della RAM (da 9 a 24 Kbyte), che successivamente usa come memoria "disco". In altre parole, il sistema operativo tratta quest'area come se fosse una vera e propria unità a dischetti esterna al sistema. Prima dell'uso, quindi, questa zona di memoria va formattata, fissando anche la quantità di RAM che si intende usare. La Epson produce anche un "RAM disk" aggiuntivo da 120 Kbyte.
Terminata questa operazione, il PX-8 carica dalla ROM il sistema operativo CP/M e visualizza in forma di menu l'elenco (directory) dei programmi di utilità del CP/M e del software contenuto in ROM. L'apparecchio può utilizzare software su ROM, su disco o su cassetta. Il software ROM è immagazzinato in chip EPROM (detti "capsule"), che vengono innestati su appositi zoccoli posti in un alloggiamento interno all'apparecchio. Il gruppo di programmi applicativi offerti come "optional" (*Portable Wordstar, Portable Calc* e *Portable Scheduler*), come anche l'interprete BASIC di corredo, sono contenuti in queste capsule. Le diverse applicazioni vengono selezionate posizionando un cursore e battendo il tasto Return. Il programma scelto viene così prelevato dalla ROM (che il PX-8 "indirizza" come drive A e B) e caricato in RAM (indirizzata come drive A).
Lo schermo a LCD ha un formato di 8 linee x 80 colonne ed una risoluzione grafica di 480 x 64 pixel. Il maggior inconveniente è la lentezza di risposta del display: i caratteri compaiono appena vengono battuti sulla tastiera, ma la cancellazione di parole e frasi è piuttosto lenta.
Oltre ai programmi già ricordati, la Epson offre un programma per telecomunicazioni, da usare assieme ad un modem, ed un programma per trasferire i file dal PX-8 ad apparecchi di maggiori dimensioni, quale l'Epson QX10. Inoltre, sul PX-8 può girare gran parte del software CP/M in commercio. Il BASIC è una versione Microsoft perfezionata dalla Epson e comprende il comando AUTO, per la numerazione e rinumerazione dei programmi, un completo "screen editor", comandi per la grafica e il suono, istruzioni per la trasmissione attraverso l'interfaccia RS232 (incorporata) e comandi che consentono di adoperare il registratore a microcassette simulando il comportamento di un'unità a dischetti (memoria ad accesso diretto).
IL PX-8 è quindi l'apparecchio ideale per manager, giornalisti e per chiunque abbia bisogno di avere sempre con sé un piccolo e potente computer. Grazie alle sue prestazioni ed al prezzo contenuto, il PX-8 è oggi uno dei portatili di maggior successo.



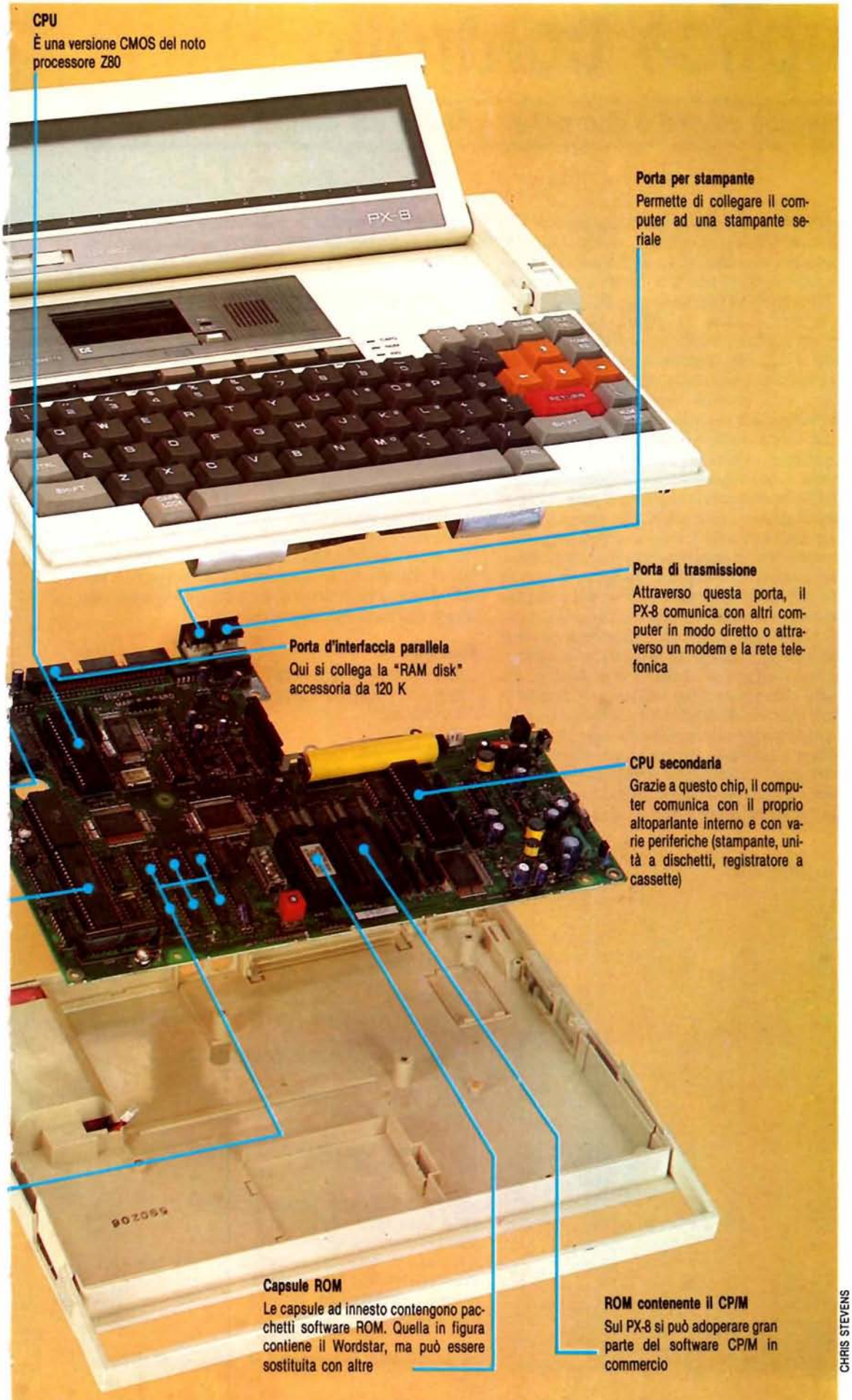
Uscita per l'altoparlante
Invia ad un altoparlante esterno i suoni generati dal PX-8

Porta A/D
Consente di collegare il computer a strumenti per misurazioni elettriche

Porta per "lettore di codici a barre"
Collegando un lettore di codici a barre si può usare il PX-8 per controllare, ad esempio, il prezzo o il codice degli articoli di un magazzino

CPU sussidiaria 7508
Trasforma le tensioni variabili che giungono alla porta A/D in segnali digitali

RAM
Il PX-8 contiene una RAM di 64 K, necessaria per far girare il CP/M. La batteria di riserva permette di conservare il contenuto della RAM anche quando l'apparecchio viene spento



CHRIS STEVENS

EPSON PX-8

DIMENSIONI

mm. 297 x 216 x 48

CPU

Versione CMOS dello Z80, funzionante a 2,4 MHz

MEMORIA

64 Kbyte di RAM, 32 Kbyte di ROM più 6 Kbyte di RAM video

VIDEO

Testuale: 8 righe di 80 caratteri. Grafico: 480 x 64 pixel

INTERFACCE

Seriale RS232C, per lettore di codici a barre, ingresso analogico

LINGUAGGI DISPONIBILI

Versione estesa del BASIC Microsoft, sotto il sistema operativo CP/M

TASTIERA

Tipo QWERTY, con 72 tasti, inclusi quelli per il cursore e 5 per funzioni programmabili. 12 tasti possono essere impiegati come tastierina numerica

DOCUMENTAZIONE

Vengono forniti due grossi manuali (uno per il sistema ed uno per il BASIC). Entrambi sono molto esaurienti e ben redatti

PRO

Il display a cristalli liquidi, il cui formato è di 80 caratteri per riga, è adatto alla maggior parte delle applicazioni. L'apparecchio offre un'ampia varietà di impieghi, grazie ad un sapiente impiego di RAM e ROM ed alla presenza del sistema operativo CP/M

CONTRO

Il display, come tutti quelli a cristalli liquidi, ha una certa lentezza nel presentare le scritte. Sebbene dotato di ampia documentazione in merito, il CP/M non è un sistema operativo di facile impiego per un principiante

Duplice controllo

Un semplice veicolo a due motori, pilotato dal computer

Con due motori di uguale potenza, un computer può controllare la direzione del veicolo, combinando i movimenti in avanti e all'indietro dei due motori; ma il veicolo può anche girare. Due sono le tecniche per far girare un veicolo a due motori: la prima consiste nel fermare un motore e azionare l'altro; in questo modo il veicolo compie una rotazione ad arco, facendo pernio sulle ruote ferme. Nella seconda, i due motori vengono azionati in senso inverso ed il veicolo esegue una rotazione che ha come centro il suo asse centrale.
Per ottenere il controllo bidirezionale di ciascun motore, dobbiamo usare le quattro prese rosse del "box di uscita" (vedere a pag. 1054) e collegare il motore di destra alle prese 0 e 1 e quello di sinistra alle prese 2 e 3 (che sono, rispettivamente, le uscite positive e negative). I due terminali di ciascun motore vengono abbinati ad una coppia di prese di uscita positive, permettendo di controllarne la direzione di rotazione in modo indipendente. Per far muovere il veicolo in avanti o indietro, oppure farlo girare a sinistra o a destra, va depositato un opportuno valore nel registro dei dati. Il motore di destra gira in avanti se la linea 0 viene posta a livello logico alto (ossia a 1) e la linea 1 a livello logico basso (ossia a 0). Gira invece all'indietro, se si inverte il livello logico delle linee (cioè linea 0 a 0 e linea 1 a 1). Lo stesso principio vale per le linee 2 e 3 del motore di sinistra. Combinando questi movimenti è possibile guidare il movimento del veicolo:

Movimento Veicolo	Motore Sinistro	Motore Destro	Config. dei bit	Valore in REGDAT
FERMO	FERMO	FERMO	0000	0
AVANTI	AVANTI	AVANTI	0101	5
INDIETRO	INDIETRO	INDIETRO	1010	10
ROTAZIONE	AVANTI	INDIETRO	0110	6
ROTAZIONE	INDIETRO	AVANTI	1001	9
ARCO SX	AVANTI	FERMO	0100	4
ARCO DX	FERMO	AVANTI	0001	1

Il seguente programma consente di controllare il veicolo direttamente da tastiera. Al tasto "T" corrisponde un movimento in avanti, a "B" indietro, a "F" una curva a sinistra e ad "H" una curva a destra. Se non viene premuto alcun tasto, il veicolo si ferma.

Micro BBC

```
  10 REM MOTORI GEMELLI — BBC
  20 DDR = &FE62:REGDAT = &FE60
  30 ?DDR = 255
  40 REPEAT
  50 A$ = INKEY$(10)
  60 PROCscansione_tastiera
  70 UNTIL A$ = "X"
  80 ?REGDAT = 0
  90 END
1000 DEF PROCscansione_tastiera
1010 IF A$ = "" THEN ?REGDAT = 0
1020 IF INKEY(—36) = —1 THEN ?REGDAT = 5
1030 IF INKEY(—101) = —1 THEN ?REGDAT = 10
1040 IF INKEY(—68) = —1 THEN ?REGDAT = 6
1050 IF INKEY(—85) = —1 THEN ?REGDAT = 9
1060 ENDPROC
```

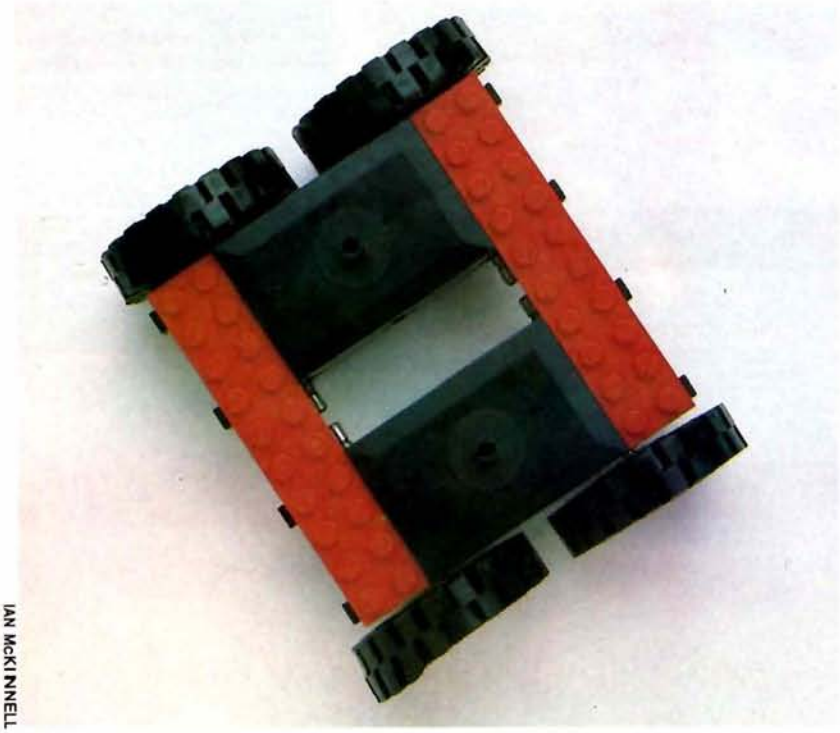
Commodore 64

```
  10 REM MOTORI GEMELLI — CBM 64
  20 DDR = 56579:REGDAT = 56577
  25 POKE650,128:REM RIPETIZ. AUTOM. TASTI
  30 POKE DDR,255
  40 GETA$
  50 GOSUB1000:GOTO70
  60 POKEREGDAT,0
  70 IF A$ > < "X"THEN FOR I = 1TO100:NEXT:GOTO40
  80 POKE REGDAT,0
  90 END
1000 REM SUBROUTINE PER ESAME DEL TASTO
1005 IF A$ = "" THEN POKE REGDAT,0
1010 IF A$ = "T" THEN POKE REGDAT,5
1020 IF A$ = "B" THEN POKE REGDAT,10
1030 IF A$ = "F" THEN POKE REGDAT,6
1040 IF A$ = "H" THEN POKE REGDAT,9
1050 RETURN
```

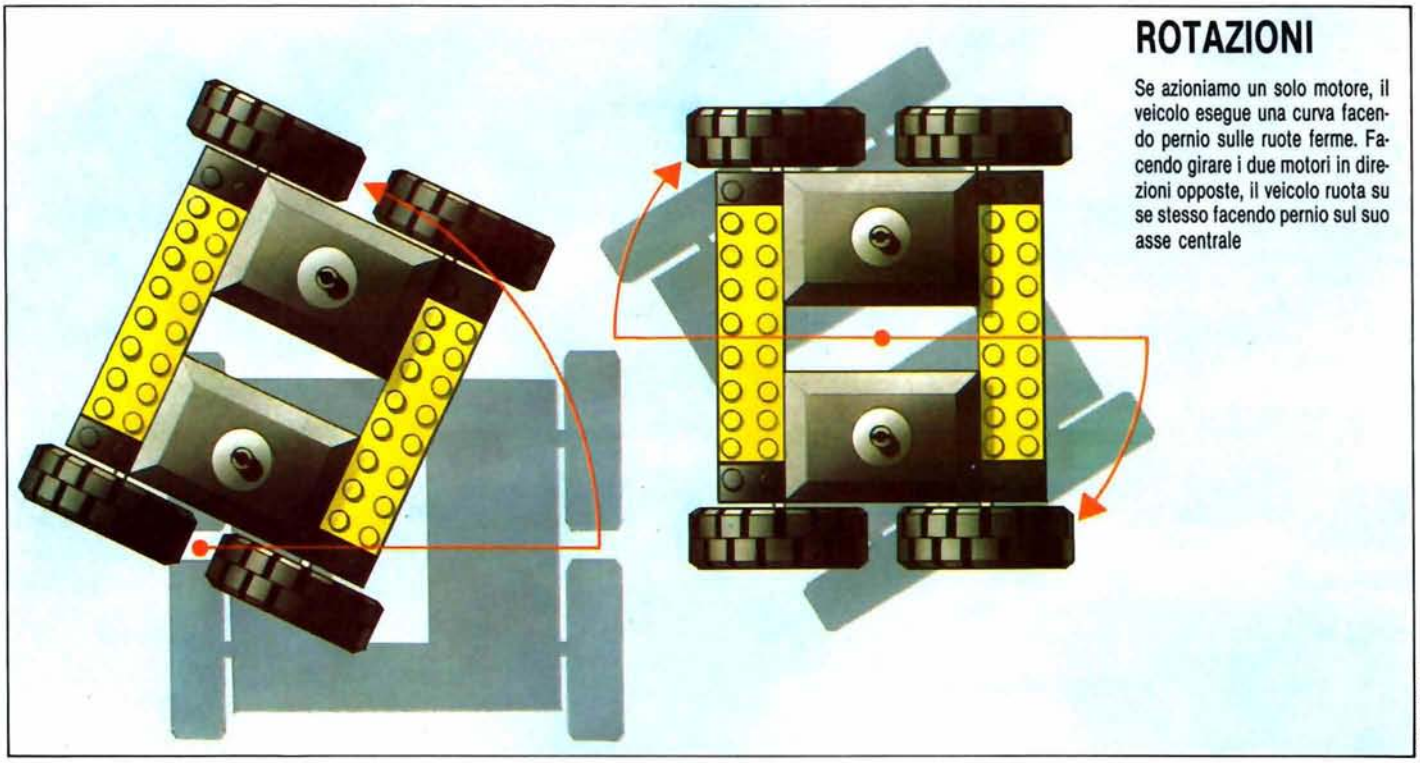
In entrambe le versioni il veicolo si muove

Motori in tandem

Questo piccolo veicolo è realizzato con due motorini che possono essere azionati in modo indipendente e permettono al veicolo di muoversi in qualsiasi direzione. I comandi di movimento possono essere impartiti da tastiera o da un programma



IAN McKINNELL



ROTAZIONI

Se azioniamo un solo motore, il veicolo esegue una curva facendo pernio sulle ruote ferme. Facendo girare i due motori in direzioni opposte, il veicolo ruota su se stesso facendo pernio sul suo asse centrale

KEVIN JONES

solo quando viene tenuto abbassato un tasto; appena questo viene lasciato, il programma inserisce uno zero nel registro dei dati ed i motori si fermano. Per uscire dal programma si deve battere il tasto X.
Nella versione BBC, la procedura SCANSIONE—TASTIERA usa INKEY per esaminare direttamente la tastiera, anziché il buffer dei caratteri immessi. Ciò consente un controllo più accurato del veicolo. Nella versione per il Commodore 64, prima viene abilitata la "ripetizione automatica" dei tasti, che sono poi inviati a gruppi, al buffer della tastiera, da dove poi vengono letti tramite il comando GET. Sfortunatamente non è possibile ispezionare direttamente la tastiera ed il controllo del veicolo risulta meno preciso che sul Micro BBC. È tuttavia possibile migliorare la precisione del controllo inserendo nel programma Commodore la seguente linea che svuota il buffer prima della lettura:

```
35 GET J$: IF J$<>"" THEN 35
```

Inoltre, occorre sostituire l'istruzione finale GOTO 40, alla linea 70, con GOTO 35.
La velocità di ripetizione del tasto può creare qualche problema in entrambi i programmi. Se il ciclo principale del programma viene eseguito più rapidamente del ritmo di ripetizione del tasto, può accadere che la routine di scansione della tastiera non riesca ad individuare tutte le ripetizioni del tasto. Ne consegue che il motore si accende e si spegne ad intermittenza, poiché i segnali ricevuti passano alternativamente dal valore che determina il movimento a quello di zero. A questo inconveniente si ovvia facilmente: basta inserire nei programmi un'istruzione che rallenti la velocità di esecuzione del ciclo principale.
Nella versione BBC, l'istruzione INKEY$(10) prolunga di 10 centesimi di secondo l'attesa del segnale prima di azionare il motore. Nella versione Commodore 64 viene invece inserita una breve routine di ritardo sulla linea 60. La durata di questi ritardi è stata appositamente calcolata e dipende dal tempo che il ciclo principale del programma impiega ad eseguire un singolo passaggio.
Adesso siamo in grado di controllare gli spostamenti del veicolo e possiamo provare a scrivere un programma che memorizzi e ripeta una serie di movimenti. Questo deve contenere una matrice bidimensionale DR() nella quale DR(C,1) registra la direzione e DR(C,2) la durata di ogni manovra. La prima parte del programma è identica alle due versioni appena viste, mentre la seconda invia al veicolo i dati immagazzinati. Ogni cambiamento di direzione, indicato da un cambiamento del contenuto del registro dei dati, viene immagazzinato in un nuovo elemento della matrice. La variabile C viene usata come contatore, per individuare gli elementi della matrice.

Micro BBC

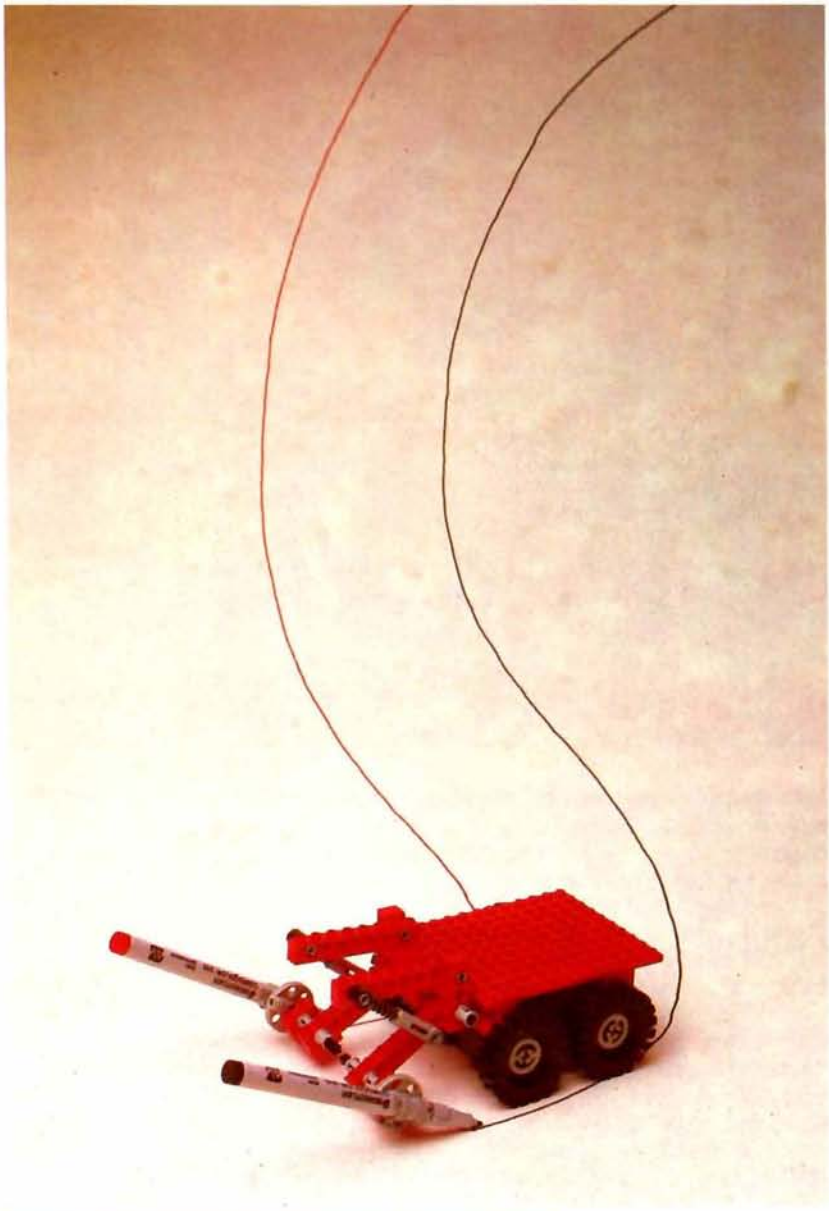
```
1000 REM MEMORIZZAZIONE DEL MOVIMENTO — BBC
1010 DDR = &FE62:REGDAT = &FE60
1020 DIM DR(100,2)
1030 ?DDR = 255:C = 1: REM INIZIALIZZA IL CONTATORE
1040 REPEAT
1050 A$ = INKEY$(10)
1060 PROCscansione_tastiera
1070 UNTIL A$b = "X"
1080 ?REGDAT = 0
1090 DR(C—1,2) = TIME
1100 REPEAT A$ = GET$
1110 UNTIL A$ = "C"
1120 REM RITRASMETTE I DATI MEMORIZZATI
1130 FOR I = 1TOC
1140 ?REGDAT = DR(I,1)
```

```
1150 TIME = 0
1160 REPEAT UNTIL TIME = DR(I,2)
1170 NEXT I
1180 END
1190 :
1200 PROCscansione_tastiera
1210 IFA$ = " "THEN ?REGDAT = 0
1220 IF INKEY(—36) = —1 THEN ?REGDAT = 5
1230 IF INKEY(—101) = —1 THEN ?REGDAT = 10
1240 IF INKEY(—68) = —1 THEN ?REGDAT = 6
1250 IF INKEY(—85) = —1 THEN ?REGDAT = 9
1260 PT = ?REGDAT
1270 IF PT < > DR(C—1,1) THEN PROCregistra_dati
1280 ENDPROC
1290 :
1300 DEF PROCregistra_dati
1310 DR(C—1,2) = TIME:REM ULTIMO TEMPO MISURATO
1320 TIME = 0:REM INIZIA NUOVA MISURAZIONE
1330 DR(C,1) = PT:REM REGISTRA CONTENUTO DELLA PORTA
1340 C = C + 1:REM INCREMENTA IL CONTATORE
1350 ENDPROC
```

Commodore 64

```
  10 REM MEMORIZZAZIONE DEL MOVIMENTO — CBM 64
  15 DIMDR(100,2):REM MATRICE DI DIREZIONE
  20 DDR = 56579:REGDAT = 56577
  25 POKE650,128:REM ABILITA RIPETIZIONE AUTOMATICA
  30 POKEDDR,255:REM TUTTE USCITE
  35 C = 1:REM INIZIALIZZA IL CONTATORE
  40 GETA$
  50 GOSUB1000:REM ESAMINA LA TASTIERA
  70 IF A$< > "X"THEN FOR I = 1 TO 200:NEXT:GOTO40
  80 POKE REGDAT,0:REM MOTORE SPENTO
  85 DR(C—12) = TI—T:REM REGISTRA L'ULTIMO TEMPO MISURATO
  90 STOP:REM DIGITARE "CONT" PER CONTINUARE
  95 REM RITRASMETTE I DATI
 100 FOR I = 1TOC
 110 POKE REGDAT,DR(I = 1)
 120 T = TI
 130 IF (TI—T) < DR(I,2)THEN130
 140 NEXT
 150 END
 999 :
1000 REM SUBROUTINE DI ESAME DELLA TASTIERA
1005 IF A$ = " "THEN POKEREGDAT,0
1010 IF A$ = "T"THEN POKEREGDAT,5
1020 IF A$ = "B"THEN POKEREGDAT,10
1030 IF A$ = "F"THEN POKEREGDAT,6
1040 IF A$ = "H"THEN POKEREGDAT,9
1045 PT = PEEK(REGDAT)
1050 IFPT < > DR(C—1,1)THENGOSUB1500
1498 RETURN
1499 :
1500 REM INSERISCE I DATI NELLA MATRICE
1510 DR(C—1,2) = TI—T:REM REGISTRA ULTIMO TEMPO
1520 T = TI:REM INIZIA LA NUOVA MISURAZIONE
1530 DR(C,1) = PT:REM REGISTRA CONTENUTO DELLA PORTA
1540 C = C + 1:REM AGGIORNA IL CONTATORE
1999 RETURN
```

Questo programma permette di controllare da tastiera i movimenti del veicolo. Poiché vengono registrate la direzione e la durata di ogni manovra, gli errori commessi nel definire la durata di ogni movimento compariranno anche nella fase di ripetizione. Questo programma illustra quindi i complessi problemi coinvolti nell'elaborazione in tempo reale, dove la struttura del programma ed il tempo di esecuzione diventano fattori di importanza vitale. Come sia possibile controllare un veicolo a due motori tramite joystick verrà spiegato in seguito.



ESERCIZI

I seguenti esercizi fanno tutti riferimento alle tecniche di controllo multidirezionale di un veicolo:

1) Si provi a tarare il veicolo. Quale numero dobbiamo inserire nel registro dei dati per ottenere che il veicolo si muova avanti o indietro di un metro, oppure per farlo ruotare di 90 gradi?

2) Progettiamo un percorso ad ostacoli e, basandoci sul programma listato in queste pagine, se ne scriva un altro che consenta di guidare il veicolo attraverso il percorso, usando la tastiera. A questo punto, il programma deve prendere il veicolo sotto il proprio controllo e, dopo averlo riportato al punto di partenza eseguendo il percorso in senso inverso, deve percorrerlo nuovamente in avanti.

3) Si colleghino quattro interruttori alle prese del "buffer box", in modo da poter controllare il veicolo tramite questi dispositivi esterni.

Movimenti in memoria

È abbastanza facile scrivere un programma che permetta di usare la tastiera di un computer per dirigere i movimenti di un veicolo a due motori. Una versione appena più complicata permette di memorizzare i comandi dell'operatore e ritrasmetterli in seguito al veicolo. Ovviamente, il programma opera in una sfera totalmente teorica e non tiene conto delle leggi che governano il movimento reale del veicolo sul terreno: l'inerzia, i rallentamenti dovuti all'attrito, le irregolarità del percorso ecc.. Nonostante ciò, i risultati che si ottengono con questa specie di "tartaruga" da pavimento sono più che soddisfacenti

Space Invaders



Il primo "classico" dei giochi in stile "arcade" che ha avuto un grandissimo successo

Quasi tutti gli home computer oggi in commercio dispongono di una versione del gioco *Space Invaders*. La sua popolarità è stata talmente vasta, che, come per "Thermos" e "Biro", il termine "Space Invaders" ha ormai assunto un significato più generale e viene usato per indicare un'intera famiglia di giochi del tipo "arcade".

Lo *Space Invaders* venne lanciato dalla Atarisoft nel 1978 e in brevissimo tempo riscosse un incredibile successo, soprattutto tra i ragazzi, facendo anche la fortuna di molte sale giochi.

Si può ragionevolmente affermare che lo *Space Invaders* abbia in qualche modo cambiato l'idea che la gente si era fatta dei computer. Prima che il gioco rivelasse le grandi potenzialità grafiche dei microprocessori, i computer erano spesso considerati ordigni pericolosi, come il "paranoico" elaboratore HAL del film *2001, Odissea nello Spazio*.

Lo *Space Invaders* è stato il progenitore di una fortunata serie di giochi "a fuoco rapido" per computer, nei quali il giocatore dispone di un solo pulsante di sparo e di tre vite per affrontare orde di marziani che calano dall'alto dello schermo.

Oggi però lo *Space Invaders* risente dei suoi anni: rispetto agli standard attuali, la sua struttura è estremamente semplice, anche se nessun altro gioco è più riuscito a stimolare l'immaginazione del pubblico in modo così totale. Il giocatore controlla una postazione laser e spara contro file compatte di invasori alieni, che calano verso la Terra a velocità crescente. Il giocatore perde una vita quando il laser viene centrato dai colpi degli alieni oppure quando gli invasori raggiungono il margine inferiore dello schermo.

Ci sono notevoli differenze tra il gioco arcade originale e le versioni per home computer. Invece di materializzarsi dallo spazio, oggi gli invasori si lanciano da una grande astronave posta sulla sinistra dello schermo. Gli stessi alieni hanno colori più brillanti e gli sprite che li compongono sono molto più complessi. Sono scomparse le barriere difensive, dietro le quali si muoveva il laser nella versione originale, e gli invasori devono percorrere un tragitto più breve per raggiungere il suolo.

Una caratteristica resta però immutata: il suono ritmico che accompagna gli alieni mentre compiono la loro discesa e cresce d'intensità via via che si avvicinano al suolo. Questa è forse la caratteristica che ha maggiormente contribuito al successo del gioco. Un altro particolare, già presente nella versione arcade, è il punteggio "a sorpresa" assegnato al giocatore quando colpisce uno dei "sigari volanti", che di tanto in tanto attraversano lo schermo.

I creatori dello *Space Invaders* pensano che il gioco manterrà inalterato il suo fascino ancora per molti anni e, nonostante la concorrenza di altri software molto più sofisticati, lo *Space Invaders* resta ancor oggi un classico piacevole ed eccitante.

Space Invaders: Per tutti i computer Atari
Editori: Atari Corporation
Autori: Atari
Joystick: Necessario
Formato: Cartuccia

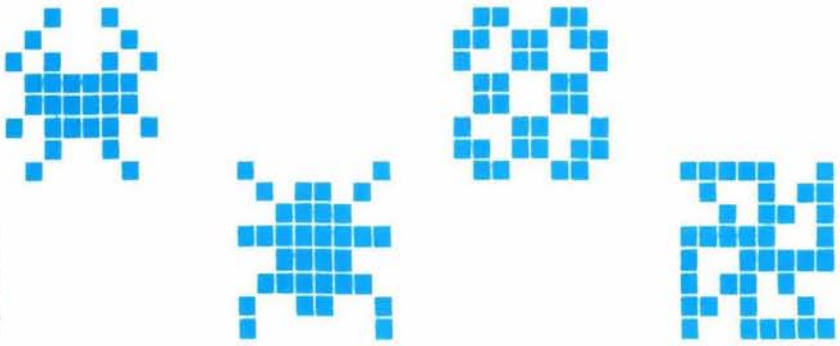


LIZ DIXON

Utili puntatori

Come completare il programma generatore di caratteri

Proviamo adesso a confrontare e, se possibile, migliorare le tre versioni del programma generatore di caratteri sviluppate in precedenza (vedere le pagg. 1052 e 1068). Per prima è stata concepita la versione Commodore, la più complessa da programmare; le altre due versioni sono direttamente derivate da questa. In parte per questo motivo e in parte per la ristrettezza dello spazio a disposizione, le versioni Spectrum e Micro BBC presentano una formattazione dello schermo piuttosto approssimativa e non usano comandi di suono, né di colore, né la grafica ad alta risoluzione. A questo proposito, vedremo in seguito quali miglioramenti sia possibile introdurre.
Tralasciando i problemi che riguardano l'efficienza della programmazione (dato che in questi programmi la velocità d'esecuzione ha un'importanza secondaria), concentriamoci sull'"interfaccia utente", ossia le pagine di spiegazioni e di assistenza, i tasti di comando e le routine di funzione.
Il listato presentato è privo di istruzioni, ma al loro posto si potrebbe visualizzare una pagina di spiegazioni all'inizio del programma, seguita da un breve promemoria, ad esempio: una succinta descrizione dei movimenti del cursore ed un sintetico elenco dei tasti di comando.
Anche la gamma dei tasti di comando può essere ampliata. Sul Micro BBC e sullo Spectrum il cursore della finestra viene controllato con gli abituali tasti per il cursore, mentre sul Commodore si usano i tasti di funzione (senza SHIFT). Questa particolarità semplifica la programmazione della versione Commodore, perché i codici ASCII degli 8 tasti di funzione sono consecutivi (tra 133 e 140). Tuttavia, i tasti di funzione del Commodore non sono dotati di "autoripetizione" (disponibile invece su quelli del BBC e dello Spectrum) e la loro disposizione sulla tastiera non è propriamente ergonomica. L'autoripetizione si può ottenere eseguendo una POKE 650,128, ma ciò non elimina il secondo inconveniente.



IAN McKINNELL

Un ulteriore perfezionamento potrebbe riguardare le modalità di movimento del cursore, che attualmente ignorano qualsiasi comando che lo porti fuori dalla finestra. In alternativa si può introdurre la funzione di movimento detta "ad avvolgimento", ossia la scomparsa del cursore da un lato della finestra e la sua immediata ricomparsa sul lato opposto (tanto in senso orizzontale che verticale). Questa funzione è facile da programmare, ma la relativa routine diventa notevolmente più lunga di quella alla linea 3500.
I comandi previsti sono quelli strettamente indispensabili, ma possono essere facilmente estesi. Nelle versioni Spectrum e BBC si potrebbero inserire come comandi il salvataggio ed il caricamento su nastro del set di caratteri, mentre tutte e tre le versioni potrebbero essere dotate della funzione per "copiare" la definizione di un carattere su un altro, ottenendo così due istruzioni, ad esempio CHR$(N) e CHR$(N+1), che visualizzano uno stesso carattere. Infine, si potrebbe introdurre un'opzione per la stampa su carta del nuovo set di caratteri. L'introduzione di tutti questi comandi viene facilitata dalla struttura modulare del programma.

Il "SAVE" del Commodore 64

Nel BASIC del Commodore 64, il comando SAVE opera esclusivamente sull'area di programma BASIC, mentre quello degli altri due apparecchi permette all'utente di specificare l'area da salvare. Il comando LOAD del Commodore permette di caricare i file in qualsiasi area della memoria: modificando il funzionamento del comando SAVE, potremmo dunque immagazzinare e richiamare senza problemi il nuovo set di caratteri.
Il SAVE del Commodore usa due puntatori per indirizzare l'area di programma BASIC: TXTTAB (locazioni 43 e 44) e VARTAB (locazioni 45 e 46). Il primo punta all'inizio dell'area di programma BASIC (dall'indirizzo 2048 in poi), il secondo all'inizio dell'area delle variabili BASIC. Dato che quest'ultima comincia dove termina la precedente, di fatto VARTAB punta alla fine dell'area di programma BASIC. Modificando i puntatori in modo che indichino l'inizio e la fine del nuovo set di caratteri, elimineremo la limitazione imposta dal comando SAVE.
Prima, però, occorre trasferire il set di caratteri in una conveniente zona di memoria. La subroutine alla linea 61000 (vedere a pag. 1053) trasferisce il set di caratteri residenti in ROM in un'area RAM di due Kbyte, che

inizia all'indirizzo 14336. La linea 50 modifica il puntatore di fine memoria per proteggere il set di caratteri da eventuali sovrascritture, riducendo di 2/3 la memoria a disposizione dell'utente. Questa operazione non crea problemi durante l'esecuzione del generatore di caratteri, ma può complicare le cose se il set di caratteri, a cui è stato assegnato quell'indirizzo, viene usato da un programma applicativo che occupa più di 12 Kbyte di memoria. Sfortunatamente, la meccanica del sistema operativo non consente di inserire il set di caratteri più in alto in memoria, altrimenti potremmo collocarlo nei due Kbyte più alti della memoria utente o in una speciale area di programma con indirizzo iniziale 49152. Come soluzione alternativa, possiamo modificare il contenuto dei puntatori di TXTTAB per inserire il set di caratteri più in basso possibile e spostare il BASIC sopra di esso. La modifica non può essere fatta dall'interno di un programma BASIC, ma va invece eseguita prima di caricare in memoria il programma generatore di caratteri.
La procedura è la seguente:

1) Caricare (LOAD) ed eseguire (RUN) il Programma 1. Questo visualizza i comandi di necessari che l'utente deve impartire in modo diretto, battendo il tasto Return.

2) Caricare il programma generatore di caratteri eapportarvi le seguenti modifiche:

```
61100 CGEN=53248: NCGEN=2048
61500 POKE PO,(PEEK(PO) AND 240) OR 2
```

e cancellare la linea 50.

3) Salvare questa versione modificata.

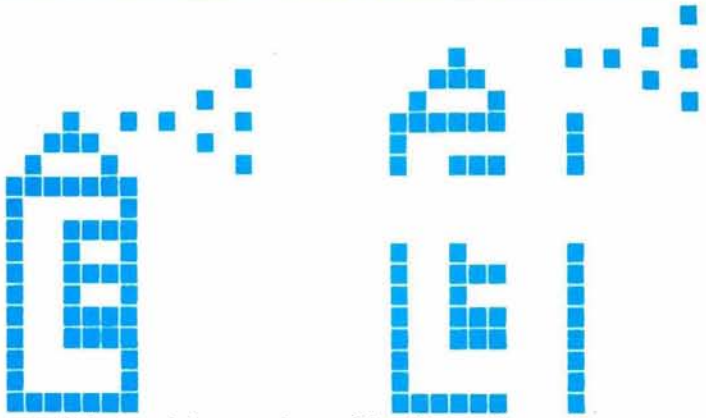
4) Caricare ed inviare in esecuzione nuovamente il programma generatore di caratteri.

5) Terminata la fase di lavoro, caricare ed eseguire il Programma 2 che, come il Programma 1, visualizza sullo schermo i comandi da inviare in esecuzione.

6) Il Programma 2 modifica i puntatori TXTTAB e VARTAB, cosicché l'istruzione SAVE "nome di file" salva l'intera area di 2 Kbyte del set di caratteri compresa tra 2048 e 4097. Da questo momento, per inviare in esecuzione il programma generatore di caratteri, si dovrà ripetere questa procedura con esclusione del punto 2.
Per richiamare il set di caratteri si deve caricare (LOAD) ed eseguire (RUN) il programma 1, per portare il BASIC più in alto in memoria, e caricare (LOAD) il set di caratteri con l'istruzione:

```
LOAD "nome di file",DN,1
```

dove DN (numero di periferica) è 1 per il registratore a cassette e 8 per l'unità a dischetti. Il numero 1 alla fine del comando è detto indirizzo secondario e sul Commodore ha la funzione di trasmettere i parametri di comando alle periferiche. Qui sta ad indicare che il file deve essere caricato nella stessa area di memoria dalla quale era stato salvato e non nell' area di programma BASIC indirizzata dal puntatore TXTTAB. Questa procedura è



possibile perché, quando un file viene salvato, il sistema operativo salva il suo indirizzo iniziale di RAM come primo dato del file. Se invece usiamo un comando LOAD privo di questa opzione, l'indirizzo iniziale del file viene ignorato ed il file viene caricato a partire dall'indirizzo indicato da TXTTAB.
Dopo aver trasferito il BASIC e caricato il nuovo set di caratteri, si deve indirizzare il sistema operativo sul nuovo set di caratteri. (Si veda la tabella in alto e la nuova versione della linea 61500).

```
199 REM ********************************
200 REM *                PROGRAMMA 1           *
201 REM *  ESEGUIRE IL PROGRAMMA E BATTERE DUE  *
202 REM *   VOLTE IL TASTO RETURN: IL BASIC VIENE  *
203 REM *   SPOSTATO DALLA LOCAZIONE 4096 IN POI  *
204 REM ********************************
300 PRINT CHR$(147):PRINT:PRINT
400 PRINT"POKE43,0:POKE44,16:POKE45,3:POKE46,16"
500 PRINT"POKE4096,0:POKE4097,0:POKE4098,0:CLR:NEW"
600 PRINT CHR$(19)
```

```
199 REM ********************************
200 REM *                PROGRAMMA 2           *
201 REM *   INVIARE IN ESECUZIONE IL PROGRAMMA   *
202 REM *   E BATTERE DUE VOLTE IL TASTO RETURN  *
203 REM * RIPORTA IL BASIC ALLA LOCAZIONE ORIGINALE *
204 REM ********************************
300 PRINT CHR$(147):PRINT:PRINT
400 PRINT"POKE43,0:POKE44,8:POKE45,1:POKE46,16"
500 PRINT"POKE4096,0:POKE4097,0:POKE4098,0:CLR"
600 PRINT CHR$(19)
```

valore di X =	bit 3,2,1,0	locazione puntata
0	0000	0
2	0010	2048
4	0100	4096
6	0110	6144
8	1000	8192
10	1010	10240
12	1110	12288
14	1110	14336

Tabella di valori

Il comando POKE 53272,(PEEK(53272) AND 240) OR X obbliga il chip di visualizzazione a esaminare l'area della RAM che contiene il nuovo set di caratteri. La tabella mostra i valori di x che selezionano l'indirizzo iniziale di questa area

Ricerca e abbinamento

Altri efficienti impieghi dell'indirizzamento indicizzato

Nell'indirizzamento indicizzato, sul processore 6809, l'effettivo indirizzo specificato, per esempio da OFFSET,X, è composto dalla somma dell'offset o scarto (che può essere una costante o il contenuto di una locazione, nel qual caso si scrive ,X oppure 0,X. In particolari applicazioni, al posto dell'offset può essere adoperato uno degli accumulatori A, B e D (ad esempio, B,X). Abbiamo già visto come uno dei più comuni impieghi dell'indirizzamento (la lettura sequenziale di una tabella) venga facilitato dall'uso del modo di autoincremento ed autodecremento, in cui il registro indice è aumentato di una o due unità, prima di eseguire l'istruzione (,X+ e ,X++) oppure è decrementato di una o due unità, ma dopo la sua esecuzione (,−Y e ,−−Y).
Vediamo adesso come sia possibile usare l'indirizzamento indicizzato abbinandolo al comando LEA (Load Effective Address), per eseguire alcune semplici operazioni aritmetiche sui valori contenuti nei registri d'indice. Le istruzioni aritmetiche comunemente applicate agli accumulatori, non hanno effetto sui valori contenuti nei registri. Anche se è possibile trasferire il contenuto di un registro d'indice nell'accumulatore D, eseguire l'operazione aritmetica e riportare il risultato sul registro, questa procedura si dimostra lenta e scomoda. L'istruzione LEA (che può essere applicata solo ai registri X, Y, S e U) consente invece di eseguire tutti i calcoli d'indirizzo necessari e di ricaricare nel registro, come di solito è necessario fare, il valore dell'effettivo indirizzo.
Ecco un esempio; l'istruzione:

```
LEA −1,X
```

calcola l'indirizzo effettivo come somma di −1 con il contenuto corrente del registro X e lo ricarica in X. L'effetto pratico è quello di decrementare il valore originale di X. L'istruzione LEA può anche essere usata, ad esempio, per calcolare un indirizzo una volta per tutte, salvando il risultato: in questo modo si evita di dover ripetere uno stesso calcolo più volte.
Un'altra istruzione usata per eseguire operazioni aritmetiche sul registro X è ABX (Add B to X), che somma il contenuto di B a quello di X, senza tener conto del segno positivo o negativo. Questa istruzione è però meno generica, quindi meno utile di LEA.

Subroutine

Una subroutine è una sezione autonoma del codice, che viene chiamata dal programma principale (o da un'altra subroutine) per eseguire uno specifico compito. Al termine, il controllo viene automaticamente restituito al programma principale (più precisamente all'istruzione immediatamente successiva a quella di chiamata). I motivi che consigliano l'uso delle subroutine sono essenzialmente tre:

1) Non si è costretti a scrivere più volte un'identica sezione di programma. Una subroutine può infatti essere richiamata quante volte occorre.

2) Le subroutine di uso più comune possono essere raccolte in una "libreria" ed impiegate in diversi programmi.

3) Il programma risulta suddiviso in sezioni di dimensioni ridotte e più facilmente controllabili.

Quando si usano le subroutine in linguaggio Assembly, occorre ricordare che il programma di chiamata e le subroutine si servono degli stessi registri. Uno degli errori più frequenti nella programmazione in codice macchina si verifica quando il programma, dopo aver immagazzinato un valore in un registro, chiama una subroutine che usa tale registro.
Il contenuto originale del registro viene ovviamente perduto. È quindi di vitale importanza conoscere ed annotare i registri usati dalla subroutine per poter salvare i valori in essi contenuti e recuperarli quando il controllo torna al programma di chiamata.
Vedremo in seguito come usare gli stack, sia per salvare questi contenuti, sia per trasferire alla subroutine valori ed indirizzi (ossia i "parametri"). Per il momento supponiamo che le subroutine utilizzino gli stessi dati del programma di chiamata (variabili globali) e che qualsiasi valore occorrente sia già contenuto nei registri.
Le istruzioni Assembly per chiamare una subroutine sono:

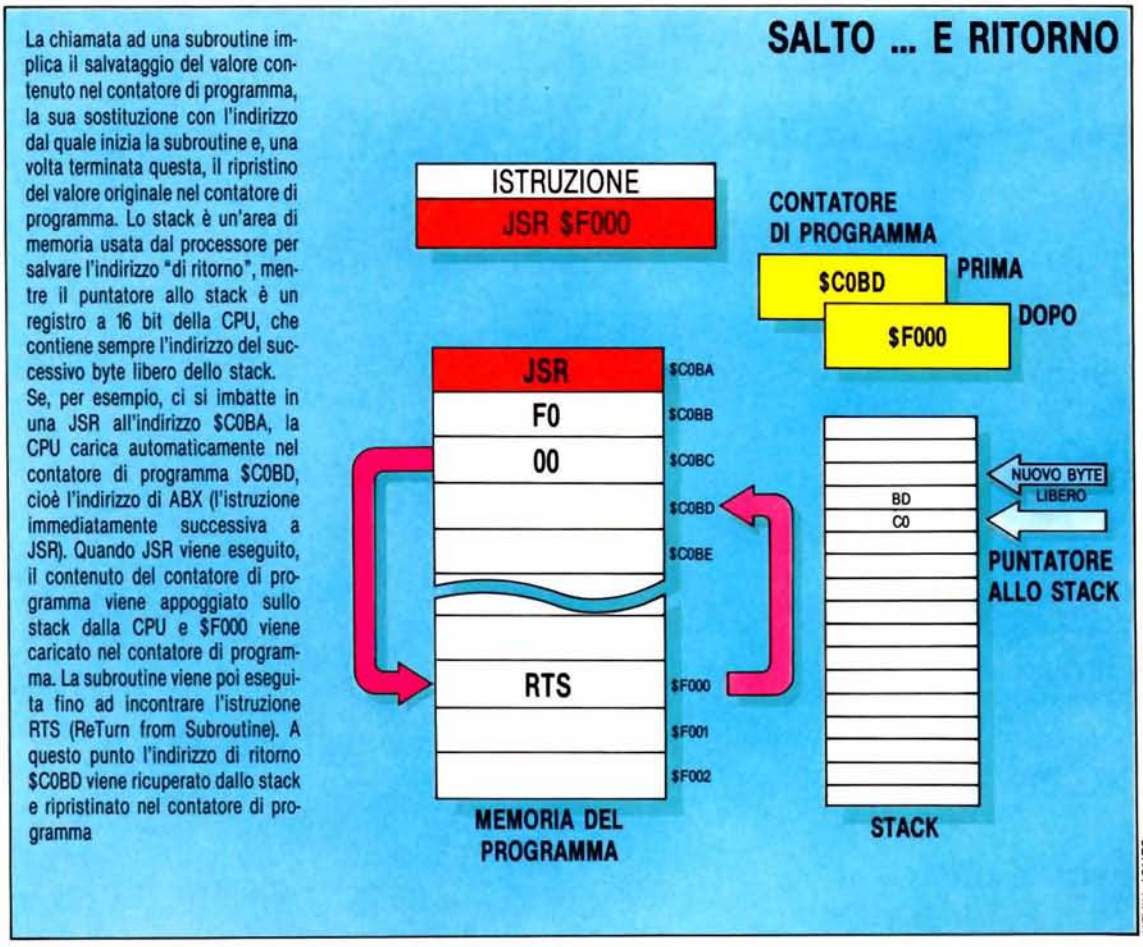
- BSR: Branch to SubRoutine (diramazione)
- JSR: Jump to SubRoutine (salto)

Il comando BSR provoca un salto relativo e trova la subroutine ad un dato offset dal corrente valore del contatore di programma. Viene di solito usato per le subroutine scritte appositamente per il programma.
Il comando JSR chiama una subroutine di

indirizzo specificato e dovrebbe essere usato per le subroutine residenti in ROMX o per quelle di libreria che occupano sempre la stessa posizione in memoria, ad esempio: le subroutine del sistema operativo per l'unità a dischetti.
Quando il processore incontra questi comandi, usa il registro S (puntatore allo stack) per "appoggiare" sullo stack il valore corrente del contatore di programma. Se la subroutine usa il registro S per un'operazione diversa da una successiva chiamata di subroutine, si deve controllare che in seguito ripristini nel registro il giusto valore. Nel caso dell'istruzione BSR, l'indirizzo della subroutine viene calcolato e caricato nel contatore di programma.
Perciò la prima istruzione ad essere eseguita sarà la prima della subroutine. Ovviamente la subroutine deve iniziare con un'istruzione e non con un byte di dati.
Ogni subroutine deve terminare con il comando RTS (ReTurn from Subroutine), che recupera dallo stack il vecchio valore del contatore di programma.
Il programma dimostrativo, listato alla pagina seguente, è più complesso di quelli finora esaminati, ma l'impiego di subroutine può renderlo più facilmente controllabile. Il programma scorre una tabella che contiene stringhe di diversa lunghezza ed estrae un valore associato ad una data stringa. Le stringhe sono registrate nel modo standard: iniziano con un byte che ne indica la lunghezza, seguito dai caratteri che compongono la stringa, e terminano con un indirizzo a 16 bit ad esse abbinato.
La fine della tabella è indicata da una stringa di lunghezza zero (il primo byte della stringa vale zero). Poniamo che l'indirizzo del primo byte della tabella sia contenuto in $10 e che l'indirizzo della stringa, di cui si cerca il valore associato, sia contenuto in $12. Se questo valore è presente in tabella, il suo indirizzo viene caricato in $14, altrimenti $12 e $14 vengono azzerati.

Confronto tra stringhe

Questo procedimento viene impiegato in molte situazioni, la più importante delle quali è la gestione degli accessi alle variabili stringa dell'interprete BASIC: ciascun nome di variabile deve essere sostituito dall'indirizzo nel quale è memorizzato il valore di quella particolare variabile.
Il problema presenta due diversi aspetti: gli elementi della tabella vanno letti, uno alla volta, finché si trova la stringa cercata o si raggiunge la fine della tabella. Dopo ogni lettura si devono confrontare le due stringhe (quella cercata e la stringa della tabella esaminata in quel momento) per vedere se corrispondono.
Programmare questo confronto tra stringhe

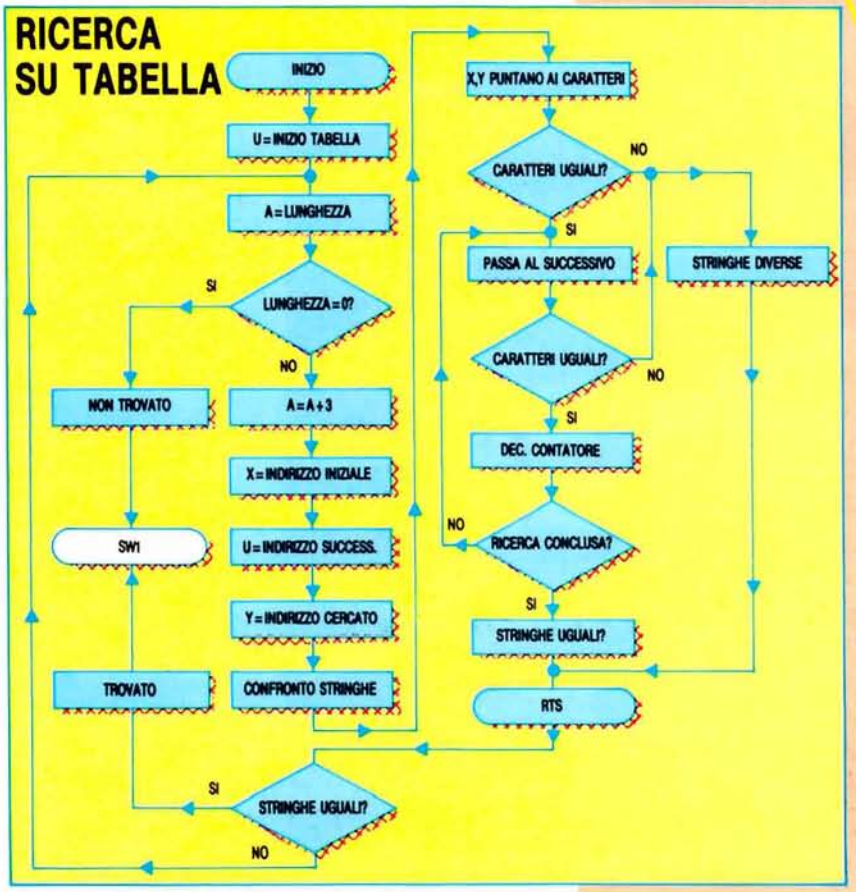


in forma di subroutine può essere un'ottima scelta, sia perché viene ripetuto più volte nel programma, sia perché consente di suddividere il problema in sezioni più facilmente controllabili. Inoltre, una simile subroutine può essere utilmente impiegata in altri programmi.
La subroutine deve ricevere dal programma di chiamata gli indirizzi delle due stringhe da confrontare. Poiché la subroutine legge le stringhe un byte alla volta, conviene caricare i due valori nei registri indice X e Y. La subroutine, a sua volta, deve restituire due valori: uno indica se il confronto è risultato positivo, l'altro comunica l'indirizzo (o valore) abbinato alla stringa trovata.

Vero o falso

Usando uno dei flag del registro di condizione del processore, è possibile trasmettere anche parametri dell'algebra Booleana (vero o falso).
Tuttavia, questa procedura richiede una precisa conoscenza degli effetti di ogni istruzione sui flag. Nel nostro programma, i valori restituiti alla routine di chiamata sono: $00 (tutti i bit a zero), se il confronto è risultato positivo oppure $FF (tutti i bit a 1) se è risultato negativo.
In effetti, per mantenersi più generica pF8TST (TeST). Questa influisce sui flag del registro di condizione, in base al valore contenuto nel registro sul quale opera.

```
TABEL   EQU   $10
STRNG   EQU   $12
INDIR   EQU   $14
        ORG   $1000      Inizio del programma principale
        LDU   TABEL      Inizio della tabella in U
CICL1   LDA   ,U         Estrae il byte della lunghezza
        BEQ   NONTR1     Se zero, la tabella è finita
        ADDA  #3         Per ottenere la lunghezza dell'elemento in tabella, sommare 1 (per il byte della lunghezza) e 2 (per l'indirizzo abbinato)
        TFR   U,X        Carica l'indirizzo della prima stringa nel registro X
        LEAU  A,U        Prepara U per l'elemento successivo
        LDY   STRNG      Y punta all'inizio della stringa di ricerca
        BSR   CONFR      Confronta le due stringhe
        TSTA             Controlla se sono uguali
        BEQ   TROV1      Se sono uguali salta a TROV1
        BRA   CICL1      Altrimenti passa all'elemento successivo
TROV1   LDD   ,X         Se la stringa è stata trovata, X dovrebbe puntare all'indirizzo del dato
        BRA   FINE1
NONTR1  LDD   #0         Se non è stata trovata, il contenuto dell'indirizzo è zero
FINE1   STD   INDIR      Salva l'indirizzo trovato
        SWI              Fine del programma principale
CONFR   LDB   ,X+        Inizio della subroutine
        CMPB  ,Y+        Estrae i byte della lunghezza e punta X e Y sui primi caratteri delle stringhe
        BNE   DIVER1     Se le stringhe hanno lunghezza diversa salta a DIVER
CICL2   LDA   ,X+        Estrae il carattere successivo dalla stringa in tabella
        CMPA  ,Y+        Lo confronta con il successivo della stringa di ricerca
        BNE   DIVER1     Salta a DIVER se non sono uguali
        DECB             Altrimenti sottrae 1 dal puntatore di posizione
        BGT   CICL2      Estrae il carattere successivo
        CLRA             Azzera A per indicare EQU e che le stringhe sono uguali
        BRA   FINE2      Inserisce tutti 1 se la stringa non è trovata nella tabella
DIVER1  LDA   #$FF       Ritorno al programma di chiamata
FINE2   RTS
        END
```



Un passo alla volta

I tre principali metodi per il movimento dei robot ed i migliori sistemi per controllarli

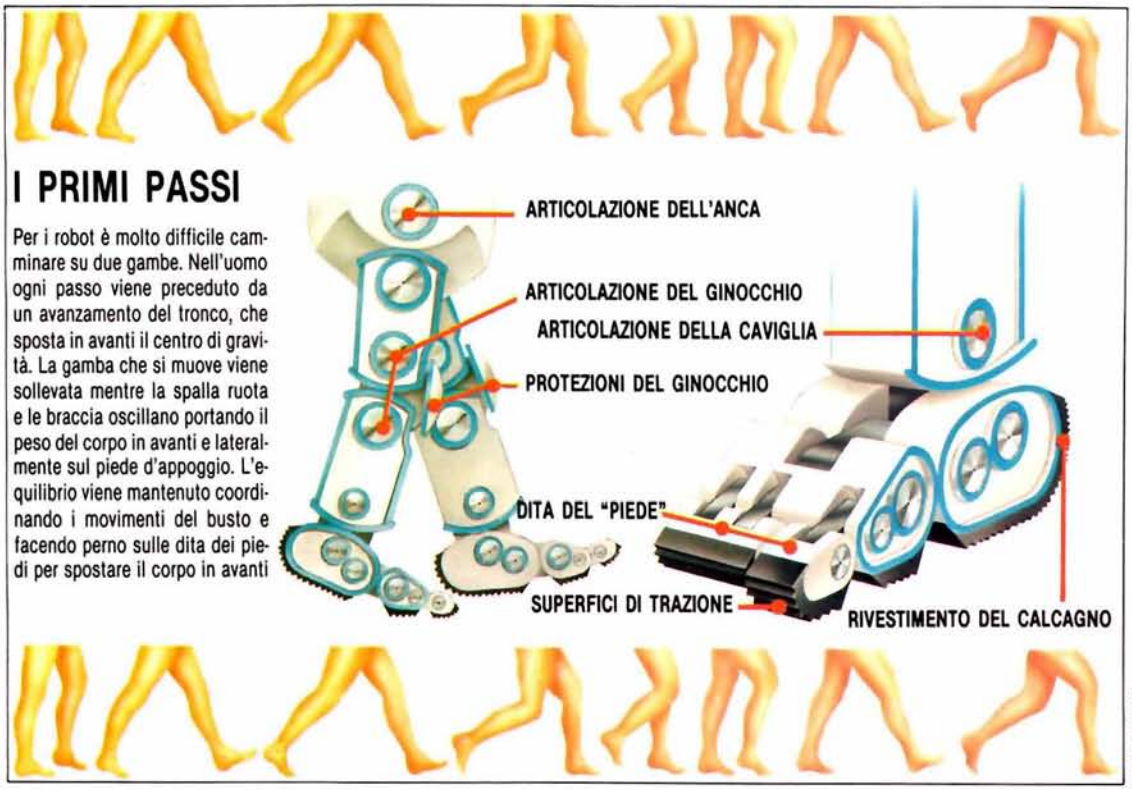
Ancor prima di riuscire a camminare, un bambino impara cose più semplici, come a prendere gli oggetti e a compiere altri movimenti. Infatti, camminare richiede molto tempo ed esercizio per diventare un'azione automatica.

Molti robot sono oggi in grado di "camminare", ma il loro movimento si basa su principi assai diversi da quelli umani; essi possono avere gambe che si spostano avanti e indietro, come quelle dell'uomo, ma poiché i loro piedi si muovono su ruote dotate di fermi non possono in realtà spostarsi all'indietro. Perciò, quando camminano, i robot di questo tipo eseguono una serie di movimenti predeterminati. La cosa più difficile da ottenere è un efficace sistema di controllo: di solito si hanno solo movimenti in avanti, piuttosto imprecisi.

Una soluzione più funzionale prevede la creazione di robot che, invece di poggiare costantemente su entrambi i piedi, camminino portando avanti una gamba dopo l'altra, come fanno gli uomini. In tal caso, la macchina deve essere capace di bilanciare il proprio peso su una sola gamba. I progettisti hanno cercato di risolvere questo problema in diversi modi. Uno di questi consiste nell'inclinare lateralmente il corpo del robot, in modo da far cadere il suo baricentro esattamente sulla gamba d'appoggio. In teoria, è possibile creare robot che salgano scale trasportando, ad esempio, una tazza di tè; in pratica, però, è molto difficile realizzare apparecchiature in grado di "vedere" dove le scale finiscono.

Altri tipi di robot presentano piedi dotati di "cingoli", che consentono loro di muoversi anche su terreni accidentati. L'esercito britannico, ad esempio, ha in dotazione robot cingolati per il trasporto di attrezzature particolarmente pericolose su terreni difficili e cosparsi di macerie.

I cingoli sono molto robusti e facili da guidare, ma il loro impiego in robotica presenta due grossi limiti. Poiché la maggior parte dei robot è di piccole dimensioni, le misure dei loro cingoli sono altrettanto ridotte e non consentono di superare grossi ostacoli. I moderni carri armati, ad esempio, possono superare dislivelli incredibili solo perché sono



dotati di cingoli grandi e pesanti. Nonostante ciò, anche i carri armati più potenti non riescono a superare ostacoli di dimensioni tali da spostare il loro baricentro al di fuori dell'area dei cingoli. Il secondo limite è dato dalla scarsa precisione di movimento dei cingoli.

Per far ruotare un robot lungo una data circonferenza, si deve infatti fermare un cingolo ed azionare l'altro; ovviamente, durante questa fase, anche il cingolo fermo subisce dei leggeri spostamenti, che spesso portano ad una posizione finale diversa da quella prevista. Nel caso dei carri armati l'errore viene facilmente corretto dal pilota, ma nei robot la rettifica del movimento implica difficoltà maggiori.

Il controllo di un robot dovrebbe infatti basarsi su una serie di istruzioni, che garantiscano di eseguire con assoluta precisione i movimenti programmati. Ecco perché la maggior parte dei robot si muove su ruote, sono più semplici, più efficienti e permettono di ottenere movimenti più veloci di quelli prodotti da vere e proprie gambe.

Vediamo adesso come funziona il controllo del movimento su ruote. Consideriamo, per ipotesi, un modellino d'automobile a molla: pur muovendosi su ruote non si tratta di un robot, perché non è possibile conoscerne la posizione in un dato momento. Occorre quindi un sistema di coordinate, che consenta di determinare l'esatta ubicazione dell'oggetto sul piano. Di solito si ricorre ad un sistema di assi cartesiani, col quale indicare l'esatta posizione del robot e i movimenti necessari per spostarlo in un altro punto del piano. Il meccanismo che genera il movimento deve però garantire una precisione pressoché assoluta.

Per muovere i robot vengono comunemente usati motori elettrici, anche se in qualche caso sono stati applicati apparecchi idraulici o pneumatici. Come abbiamo già visto nella sezione *Laboratorio* (vedere pagg. 1065 e 1092), un simile motore permette di controllare il movimento e, in parte, la direzione di un carrello. Ovviamente la precisione è limitata, perché un motore elettrico di semplice concezione compie, come minimo, una rotazione di 180 gradi prima di fermarsi e la sua inerzia spesso provoca leggeri incrementi nella rotazione.

Per questi motivi i robot vengono solitamente azionati da motori "passo passo" di vario tipo che, realizzati con un gran numero di avvolgimenti, consentono di eseguire piccoli e precisi movimenti di rotazione con un margine di errore (per difetto o per eccesso) estremamente ridotto.

Il mercato offre un gran numero di robot dotati di motori "passo passo". Su alcuni, le tartarughe, è montata una penna capace di tracciare linee sulla superficie su cui si muovono.

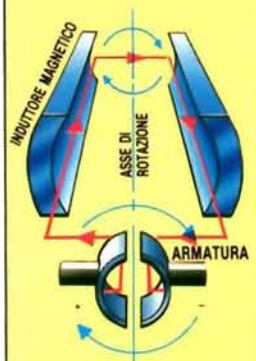
Ai disegni così ottenuti ci si riferisce talvolta col termine "turtle graphics" (grafica con la tartaruga). La precisione del movimento è elevata e può essere verificata tracciando figure chiuse, come rettangoli o stelle: il punto di arrivo della tartaruga deve corrispondere a quello di partenza.

I motori passo passo ed i sistemi di coordinate cartesiane permettono quindi di controllare con una certa precisione i movimenti di un robot.

Tuttavia, se il robot deve compiere operazioni più complesse del semplice movimento su una superficie piana (magari scontrandosi con eventuali ostacoli) deve essere in grado di rispondere con rapidità e precisione alle condizioni esterne.

Questo argomento verrà ripreso e approfondito più avanti.

MOTORI "PASSO PASSO"



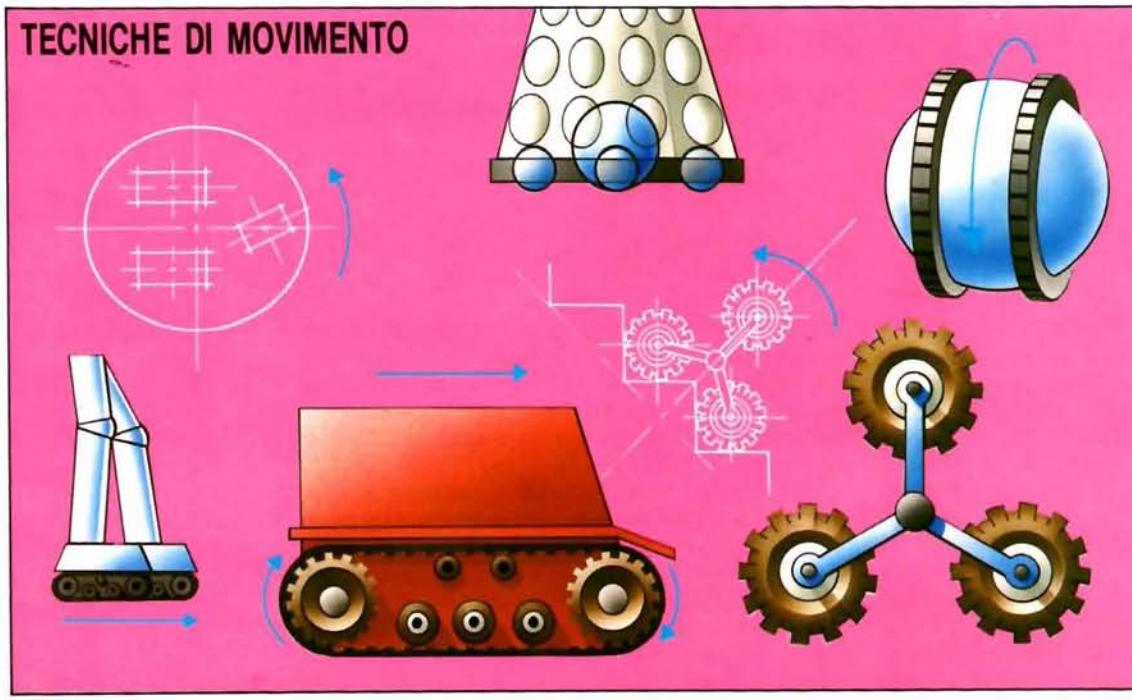
Nei motori elettrici più semplici, una corrente attraversa l'avvolgimento dell'indotto e crea un campo magnetico opposto a quello generato dall'induttore. Questo contrasto di forze fa girare il rotore all'interno del campo



I motori "passo passo" contengono molti avvolgimenti; il passaggio alternato della corrente su questi permette di controllare con precisione gli angoli di rotazione del rotore

KEVIN JONES

TECNICHE DI MOVIMENTO



Non sempre i robot possono muoversi su superfici piane come quelle di un laboratorio perciò vengono adottati alcuni meccanismi. I piedi cingolati hanno buona presa sul terreno, ma scarsa manovrabilità: consentono al robot un movimento "strascicato" ed eliminano molti dei problemi di equilibrio. I veicoli-robot cingolati sono comunemente usati dai reparti d'artiglieria e dalle spedizioni artiche. Il modello a tre assi è il solo sistema a ruote che permetta ad un robot di salire una serie di scalini. Una grossa sfera ruotante, dotata di cuscinetti a sfera, è facile da manovrare, ma si dimostra molto sensibile alle asprità del terreno. Per ottenere un minimo di stabilità, essa va abbinata ad almeno due ruote fisse. Una soluzione può essere sospendere il robot tra ruote motrici di grandi dimensioni, ma questo alza il baricentro riducendone la stabilità

KEVIN JONES

"Divide et impera"

Il LOGO può produrre sagome e disegni strani, dalle curiose proprietà

Il primo programma qui presentato traccia l'immagine di un albero. Per cominciare, possiamo disegnare un tronco con due diramazioni simmetriche; le ramificazioni successive sono più brevi, ma hanno la stessa struttura: uno stelo centrale ed un rametto per parte. Se il procedimento continua, si ottiene l'immagine di un albero con innumerevoli ramificazioni. Questo programma offre un buon esempio dell'uso della ricursione in LOGO.
La procedura per tracciare questo albero "binario" necessita di due parametri: uno per la lunghezza del tronco e l'altro per il numero di livello. In ciascun livello, la lunghezza delle diramazioni è la metà di quella del tronco.

```
TO RAMIFICAZIONE: LUNGHEZZA :LIVELLO
   IF :LIVELLO=0 THEN STOP
   FD :LUNGHEZZA
LT 45
RAMIFICAZIONE (:LUNGHEZZA/2) (:LIVELLO-1)
RT 90
RAMIFICAZIONE (:LUNGHEZZA/2) (:LIVELLO-1)
LT 45
BK :LUNGHEZZA
END
```

Si noti che la procedura è "trasparente". Se così non fosse, lo stato della tartaruga (posizione e direzione) risulterebbe diverso ogni volta che la procedura chiama se stessa, rendendo impossibile il disegno dell'albero.
La figura che si ottiene è decisamente stilizzata, ma la si può rendere più realistica modificando la procedura. La seguente versione traccia rami di diversa lunghezza in ogni livello.

```
TO RAMIFICAZIONE1 :LUNGHEZZA :LIVELLO
   IF :LIVELLO=0 THEN STOP
   FD :LUNGHEZZA
LT 30
RAMIFICAZIONE1 (:LUNGHEZZA/3) (:LIVELLO-1)
RT 40
RAMIFICAZIONE1 (:LUNGHEZZA/2) (:LIVELLO-1)
RT 50
RAMIFICAZIONE1 (:LUNGHEZZA/1.5) (:LIVELLO-1)
LT 60
BK :LUNGHEZZA
END
```

Modifiche di altro tipo permettono di creare alberi ancor più realistici.

Poligoni quadrettati

La seguente procedura disegna un quadrato, lo divide in quattro parti, quindi divide ciascun quadratino ancora in quattro e così via:

```
TO LAVAGNA :LUNGHEZZA :LIVELLO
   IF :LIVELLO=0 THEN REPEAT 4 [FD :LUNGHEZZA
   RT 90] STOP
   LAVAGNA (:LUNGHEZZA/2) (:LIVELLO-1)
   FD (:LUNGHEZZA/2)
   LAVAGNA (:LUNGHEZZA/2) (:LIVELLO-1)
   RT 90
   FD (:LUNGHEZZA/2)
   LT 90
   LAVAGNA (:LUNGHEZZA/2) (:LIVELLO-1)
   BK (:LUNGHEZZA/2)
   LAVAGNA (:LUNGHEZZA/2) (:LIVELLO-1)
   LT 90
   FD (:LUNGHEZZA/2)
   RT 90
END
```

Si provi a scrivere una procedura simile alla precedente, ma che suddivida un triangolo in quattro triangoli più piccoli, poi ciascuno di essi ancora in quattro e così via.

Fiocchi di neve

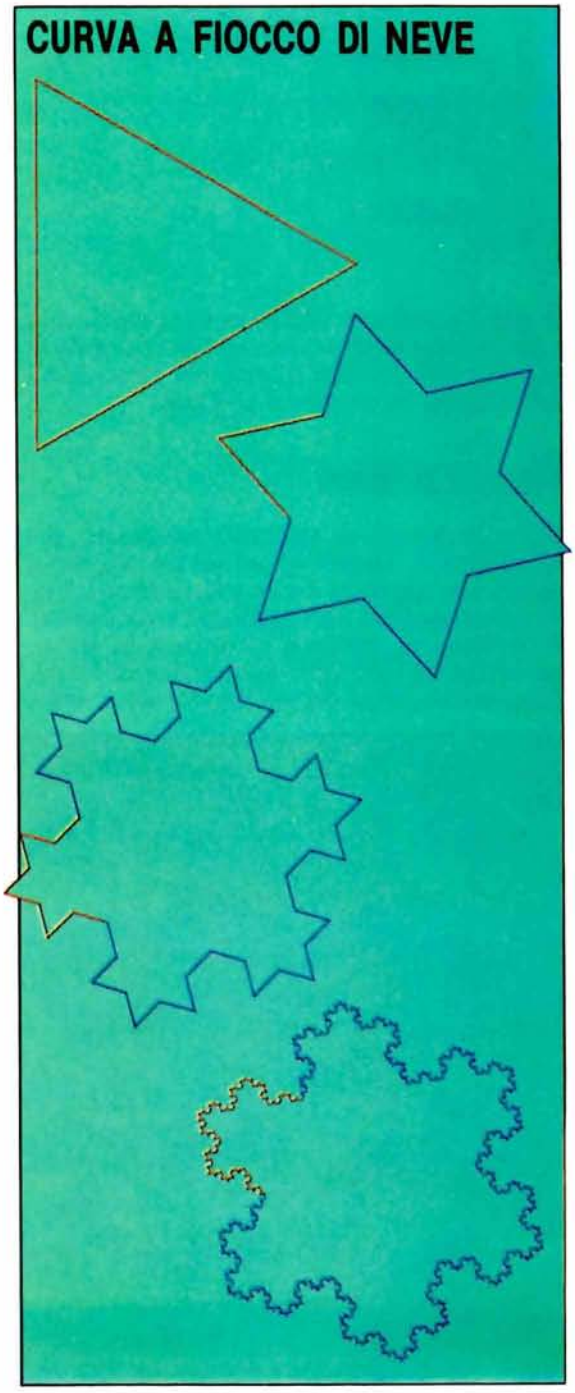
Prima si disegni un triangolo equilatero, poi si suddivida ciascun lato in tre segmenti uguali e si tracci un nuovo triangolo equilatero sul segmento centrale. Dopo aver cancellato le linee in comune, si esegua la stessa sequenza su ogni lato della figura ottenuta. Ripetendo questo procedimento molte volte



si ottiene una figura detta "curva a fiocco di neve".

```
TO NEVE :LUNGHEZZA :LIVELLO
   REPEAT 3 [LATO :LUNGHEZZA :LIVELLO RT 120]
END

TO LATO :LUNGHEZZA :LIVELLO
   IF :LIVELLO=0 THEN FD :LUNGHEZZA STOP
   LATO (:LUNGHEZZA/3) (:LIVELLO-1)
   LT 60
   LATO (:LUNGHEZZA/3) (:LIVELLO-1)
   RT 120
   LATO (:LUNGHEZZA/3) (:LIVELLO-1)
   LT 60
   LATO (:LUNGHEZZA/3) (:LIVELLO-1)
END
```



Si noti che la procedura LATO non è trasparente, tuttavia lascia la tartaruga nella posizione giusta per tracciare il successivo lato della figura. Se il procedimento venisse ripetuto all'infinito (i matematici direbbero "tendendo al limite") si otterrebbe una curva di lunghezza infinita, che racchiude una superficie di area finita! Si può dimostrare che questa curva non è ad una o a due dimensioni, ma una via di mezzo tra le due.
Si provi a scrivere una procedura che disegni una curva simile alla precedente, ma partendo da un quadrato, dividendo ciascun lato in tre segmenti uguali, tracciando un nuovo quadrato sui segmenti centrali e così via.

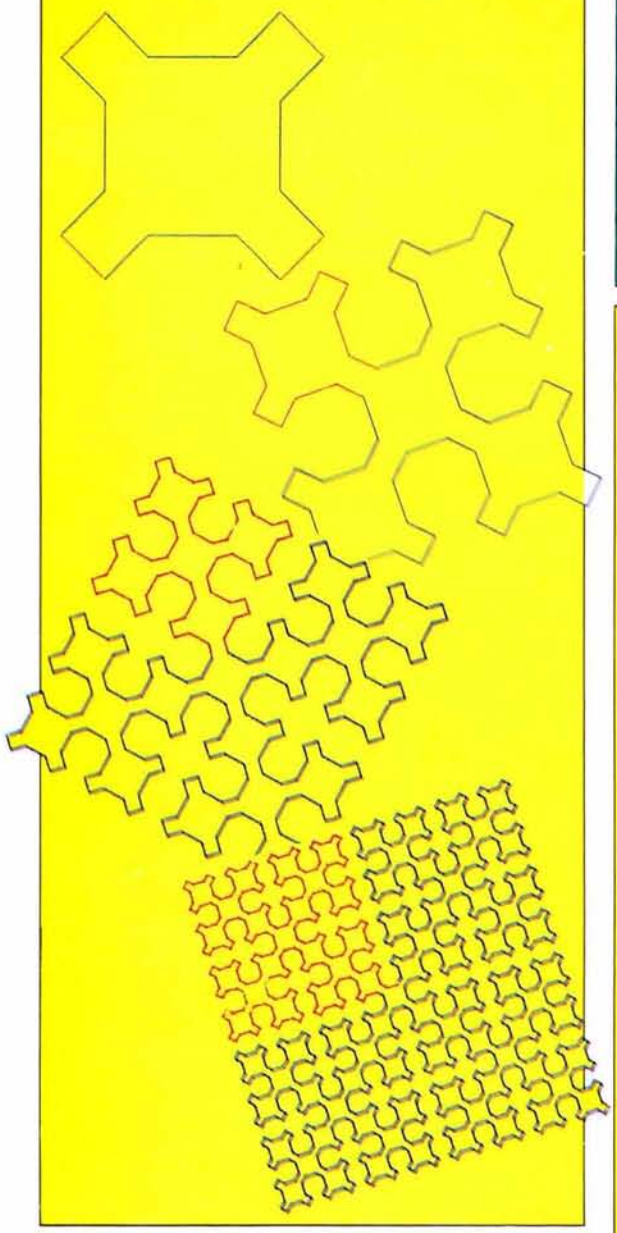
Curve di Sierpinski

Le serie di curve qui riprodotte sono state inventate dal matematico Sierpinski. Se questo procedimento continuasse all'infinito, si otterrebbe una curva (una "linea monodimensionale") che tocca tutti i punti del quadrato da essa circoscritto (una figura bidimensionale). Molte sono le "curve di riempimento" che presentano questo strano comportamento e le procedure sviluppate per tracciarle si dimostrano piuttosto complicate.
La curva di livello 1 è composta da quattro lati (in blu) collegati da quattro diagonali (in rosso). La procedura principale SIERP suddivide il procedimento in quattro parti, che la procedura UN.LATO esegue una alla volta.
Ogni lato è composto da tre segmenti: due diagonali ed uno orizzontale o verticale. Quando la procedura passa al livello 2, ciascuna diagonale viene sostituita da una serie di tre segmenti più piccoli ed ogni linea orizzontale o verticale da due serie di tre segmenti simili ai precedenti collegati da una linea. Lo stesso procedimento si ripete ad ogni livello.
Questa è la procedura che traccia la curva appena descritta. Si noti l'uso del comando MAKE del LOGO, per inizializzare DIAG:

```
TO SIERP :LATO :LIVELLO
   MAKE "DIAG :LATO/SQRT(2)
   REPEAT 4 [UN.LATO :LIVELLO RT 45 FD :DIAG RT 45]
END

TO  N.LATO :LIVELLO
   IF :LIVELLO=0 STOP
   UN.LATO (:LIVELLO-1)
   RT 45
   FD :DIAG
   RT 45
   UN.LATO (:LIVELLO-1)
   LT 90
   FD :LATO
   LT 90
   FD :LATO
   LT 90
   UN.LATO (:LIVELLO-1)
   RT 45
   FD :DIAG
   RT 45
   UN.LATO (:LIVELLO-1)
END
```

CURVA DI SIERPINSKI



RISPOSTA ALL'ESERCIZIO

Ecco una procedura ricursiva che traccia una "torre" di quadrati:

```
TO TORRE :LATO
   IF :LATO<5 THEN STOP
   QUADRATO :LATO
   SPOSTAMENTO :LATO
   TORRE (:LATO/2)
END
TO QUADRATO :LATO
   REPEAT 4 [FD :LATO RT 90]
END
TO SPOSTAMENTO :LATO
   FD :LATO
   RT 90
   FD (:LATO/4)
   LT 90
END
```

Logo & Logo

Le versioni LCSI del LOGO usano SETPOS per fissare la posizione della tartaruga. Poiché a tale istruzione occorrono due coordinate, queste vanno precedute dal comando LIST. Ad esempio:

SETPOS LIST 45 67

Si ricordi, inoltre, che nelle versioni LCSI, la primitiva IF ha un formato diverso. In un'istruzione di stop avremo:

IF LIVELLO=0 [STOP]

ESERCIZIO 4

Se la procedura che traccia queste curve viene eseguita all'infinito, la curva finale non presenterà alcun dislivello. La curva di livello 1 è composta da due linee, una ascendente e l'altra discendente. Al livello 2, la linea ascendente viene sostituita da una linea "spezzata" a sei segmenti che prima raggiunge metà dell'altezza della linea originaria, poi torna verso il basso quindi risale fino a raggiungere l'altezza massima (2 segmenti consecutivi). Infine, ridiscende per metà altezza e risale di nuovo fino all'altezza massima. La linea discendente si comporta allo stesso modo.

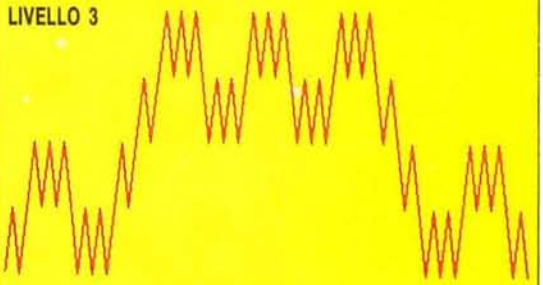
LIVELLO 1



LIVELLO 2



LIVELLO 3



Si provi a scrivere una serie di procedure che disegnino queste curve, usando SETXY al posto di FD e RT. La procedura iniziale dovrebbe suddividere il procedimento in due parti: una per la linea ascendente, l'altra per la discendente. Si devono quindi scrivere due distinte procedure per questo compito. Si rammenti che tali procedure possono chiamare se stesse oppure chiamarsi tra loro



La somma delle parti

I pro e i contro del software "integrato"

L'"integrazione" rappresenta una delle più interessanti aree di sviluppo del software. Nonostante oggi venga applicata essenzialmente ai sistemi gestionali, le sue tecniche cominciano ad essere impiegate anche sui micro di uso domestico. Ad esempio, il QL della Sinclair possiede 4 package di software integrati tra loro.

Se il software è "integrato" , l'operatore può passare da un programma applicativo ad un altro in modo quasi automatico, senza dover uscire ogni volta dal programma in uso per tornare al sistema operativo, cambiare il dischetto e chiamare il nuovo programma. Con un buon software integrato è sufficiente battere un solo tasto per cambiare applicazione: in alcuni package per microcomputer (come il *Lotus 1-2-3* ed il *Framework* della Aston Tate) l'integrazione è già una realtà!

Questa tecnica consente anche lo scambio di dati fra applicazioni diverse. Ad esempio, si può creare un tabulato del fatturato annuo di una ditta con un programma "spreadsheet" ed inserirlo nella relazione annuale scritta con un word processor, oppure trasferire i nomi e gli indirizzi contenuti in un database su un word processor per scrivere lettere personalizzate alle persone registrate nel file.

Sul Lisa e sul Macintosh è addirittura possibile creare disegni a mano libera con un programma grafico e copiarli in un documento elaborato tramite un word processor.

Inoltre, anche il funzionamento di tutti i programmi deve essere simile: ciò implica che tutti i formati del video, i tasti di comando, i messaggi guida e quelli d'errore, insomma tutte le caratteristiche dell'"interfaccia utente", siano uguali o per lo meno seguano le stesse convenzioni. In caso contrario, l'utente si troverebbe a disagio nel muoversi da un'applicazione all'altra e sarebbe costretto a fermarsi per riflettere sui cambiamenti.

Apprendere l'uso di cinque programmi di tipo diverso (alcuni controllati da menu e altri da comandi, ma tutti con comandi di formato diverso) può essere un'impresa ciclopica per chi non abbia una memoria di ferro! Al contrario, imparare ad usare il software integrato è molto facile, perché basta familiarizzarsi coi comandi di uno dei programmi; gli altri sono identici.

Questa caratteristica del software integrato è talvolta chiamata "comunanza" tra le componenti del package.

Si sono citati i tre principi che stanno alla base del software integrato: la possibilità di passare da un'applicazione ad un'altra, il libero scambio di dati e informazioni e la comunanza di funzionamento. L'integrazione facilita l'uso del computer all'utente medio che spesso deve lavorare con due o tre applicazioni diverse; nello stesso tempo accresce la popolarità dei personal computer che diventano sempre più efficienti e facili da usare.

Tuttavia esistono anche degli svantaggi, primo fra tutti che i package integrati implicano quasi sempre dimensioni notevoli della RAM. Si immagini di caricare un word processor, uno spreadsheet ed un database (i tre programmi applicativi più frequentemente integrati) in una RAM a 16 o 32 Kbyte: è probabile che non resti spazio per memorizzare i dati!

Questo è il problema che ha limitato l'impiego del software integrato ai computer con ampia memoria (da 128 Kbyte in su). Naturalmente, i programmi integrati possono utilizzare alcune routine in comune, come quelle dedicate all'archiviazione su disco, tuttavia ogni applicazione ha anche requisiti propri che necessitano di RAM.

C'è ancora un secondo svantaggio che deriva da problemi di memoria: per lasciare più spazio ai programmi dell'utente, i progettisti del software apportano piccoli "tagli" ai programmi applicativi. Quindi un word processor, integrato ad altre due o tre applicazioni, non sarà così dettagliato ed esauriente come un word processor autonomo. Infatti, un programma indipendente ha a disposizione mol-



Rassegna... integrata

I pacchetti integrati mostrati in figura sono: il "Lotus 1-2-3", l'"Open Access", il "Symphony" e il "Framework"

per concessione della MICROSCOPE

IAN McKINNELL

ta più memoria che non uno integrato e può quindi essere più completo.
Prendiamo come esempio due programmi che girano sul PC della IBM e su computer simili. *Multimate* è un programma che fa parte del software in dotazione allo Wang, un apparecchio dedicato al word processing. Possiede molti comandi, per la creazione e la redazione dei testi, assenti in programmi più piccoli e permette di creare con facilità lunghi e complessi documenti. Il *Multimate* occupa da solo ben 192 Kbyte di RAM! Il *Lotus 1-2-3* è invece un software integrato composto da word processor, spreadsheet e database ed occupa anch'esso 192 Kbyte. Come risultato si ha che il word processor del *Lotus 1-2-3* si limita alla semplice memorizzazione dei testi scritti.
Un terzo limite dipende proprio da una delle caratteristiche fondamentali del software integrato. Come si è visto, i programmi integrati devono funzionare in modo molto simile, affinché l'utente possa imparare ad usarli rapidamente; per ottenere questo risultato, i creatori del software devono però scendere a qualche compromesso. Se il funzionamento ideale per uno spreadsheet non corrisponde a quello di un database o di un word processor, si devono fare scelte che portino ad un rendimento medio accettabile. La Microsoft ha dovuto affrontare un simile problema durante lo sviluppo del *Microsoft Word*, un word processor autonomo, nel tentativo di rendere compatibili per una futura integrazione il formato del video ed il funzionamento del programma con il *Multiplan*, lo spreadsheet di maggior successo della Microsoft. Nel *Word* venne inizialmente inserito lo stesso menu adottato, con ottimi risultati, sul *Multiplan*. Ben presto i programmatori si accorsero che gli utenti del *Microsoft Word* erano infastiditi dalla continua presenza del menu sullo schermo.
Un software di buona qualità deve infatti adattarsi alle esigenze dell'utente. Il software integrato è lo strumento ideale per chi deve svolgere lavori di vario tipo come scrivere lettere, mailing list (indirizzari) e tenere una piccola contabilità. Gli utenti che vogliono usare il loro microcomputer per scrivere racconti o lunghe relazioni commerciali, trarranno maggiori vantaggi da programmi autonomi (quindi più completi) di word processor, spreadsheet e database. Tuttavia, grazie ai successi ottenuti nel comprimere le istruzioni del software di sistema in aree di memoria sempre più ridotte, e alla continua espansione delle memorie degli apparecchi domestici, il software integrato sta diventando sempre più importante sia per gli utenti di home computer sia per quelli di apparecchi gestionali.
Già oggi il mercato offre due macchine dotate di software integrato: il QL della Sinclair ed il Commodore Plus/4.
In seguito verranno esaminati alcuni dei programmi integrati che hanno maggiormente influenzato lo sviluppo del software, valutando due diversi tipi di approccio all'integrazione: il primo è quello impiegato nel *Lotus 1-2-3* (e affini), in cui le applicazioni, pur lavorando assieme, somigliano ancora a tipici programmi di computer. Il secondo è caratteristico del Lisa e del Macintosh dove l'intera area operativa è stata rivoluzionata a favore dell'integrazione.

REGOLE DEL GIOCO



Integrazione assoluta

Le illustrazioni di questa pagina sono state create con il Macintosh della Apple, un computer basato su software integrato. "Multiplan" (lo spreadsheet), "MacWrite" (il word processor) e "MacPaint" (il programma grafico) comunicano direttamente attraverso il sistema operativo: sono quindi tre applicazioni completamente integrate in un unico sistema

Codice comune

La comunanza, o identità di formato tra le diverse applicazioni integrate, traspare chiaramente da queste immagini video create con lo spreadsheet ed il word processor



Interscambio di dati

Il trasferimento diretto delle informazioni dallo spreadsheet al word processor dimostra l'importanza della compatibilità tra i dati, tipica del software integrato

IAN McKINNELL

DATABASE

Terza scheda delle istruzioni per lo Z80 della Zilog

Istruzioni di scambio

		INDIRIZZAMENTO IMPLICITO				
		AF'	BC', DE' e HL'	HL	IX	IY
IMPLICITO	AF	08				
	BC, DE e HL		D9			
	DE			EB		
REGISTRO INDIRETTO	(SP)			E3	DD E3	FD E3

Trasferimenti di blocchi

HL punta alla provenienza
DE punta alla destinazione
BC contiene il numero di byte

DESTINAZIONE: REG. IND.	PROVENIENZA: (DE)	PROVENIENZA: REG. IND. (HL)
		ED A0 'LDI': Carica (DE) con (HL), incrementa HL e DE, decrementa BC
		ED B0 'LDIR': Carica (DE) con (HL), incrementa HL e DE, decrementa BC Ripeti finché BC=0
		ED A8 'LDD': Carica (DE) con (HL) Decrementa HL e DE, decrementa BC
		ED B8 'LDDR': Carica (DE) con (HL) Decrementa HL e DE, decrementa BC Ripeti finché BC=0

Ricerca di blocchi

RICERCA LOCAZIONI

HL punta alla locazione di memoria da confrontare col contenuto dell'accumulatore
BC è il contatore

REG. IND. (HL)	
ED A1	'CPI' Incrementa HL, Decrementa BC
ED B1	'CPIR': Incrementa HL, Decrementa BC Ripeti finché BC=0 o trova corrispondenza
ED A9	'CPD': Decrementa HL e BC
ED B9	'CPDR': Decrementa HL e BC Ripeti finché BC=0 o trova corrispondenza

Scambio, trasferimento e ricerca di blocchi

Codice mnemonico	Operazione simbolica	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
EX DE, HL	DE↔HL	•	•	X	•	X	•	•	•	11 101 011	EB	1	1	4	
EX AF, AF	AF↔AF	•	•	X	•	X	•	•	•	00 001 000	08	1	1	4	
EXX	BC↔BC DE↔DE HL↔HL	•	•	X	•	X	•	•	•	11 011 001	D9	1	1	4	Scambio tra il gruppo principale di registri e quello alternativo
EX (SP), HL	H'↔(SP+1) L↔(SP)	•	•	X	•	X	•	•	•	11 100 011	E3	1	5	19	
EX (SP), IX	IX_A↔(SP+1) IX_B↔(SP)	•	•	X	•	X	•	•	•	11 011 101 11 100 011	DD E3	2	6	23	
EX (SP), IY	IY_A↔(SP+1) IY_B↔(SP)	•	•	X	•	X	•	•	•	11 111 101 11 100 011	FD E3	2	6	23	
LDI	(DE)←(HL) DE←DE+1 HL←HL+1 BC←BC−1	•	•	X	0	X	① ↕	0	•	11 101 101 10 100 000	ED A0	2	4	16	Carica (HL) in (DE), incremento dei puntatori e decremento del contatore BC
LDIR	(DE)←(HL) DE←DE+1 HL←HL+1 BC←BC−1 Ripeti finché BC=0	•	•	X	0	X	0	0	•	11 101 101 10 110 000	ED B0	2 2	5 4	21 16	se BC≠0 se BC=0
LDD	(DE)←(HL) DE←DE−1 HL←HL−1 BC←BC−1	•	•	X	0	X	① ↕	0	•	11 101 101 10 101 000	ED A8	2	4	16	
LDDR	(DE)←(HL) DE←DE−1 HL←HL−1 BC←BC−1 Ripeti finché BC=0	•	•	X	0	X	0	0	•	11 101 101 10 111 000	ED B8	2 2	5 4	21 16	se BC≠0 se BC=0
CPI	A←(HL) HL←HL+1 BC←BC−1	↕	② ↕	X	↕	X	① ↕	1	•	11 101 101 10 100 001	ED A1	2	4	16	
CPIR	A←(HL) HL←HL+1 BC←BC−1 Ripeti finché A=(HL) oppure BC=0	↕	② ↕	X	↕	X	① ↕	1	•	11 101 101 10 110 001	ED B1	2 2	5 4	21 16	se BC≠0 e A≠(HL) se BC=0 o A=(HL)
CPD	A←(HL) HL←HL−1 BC←BC−1	↕	② ↕	X	↕	X	① ↕	1	•	11 101 101 10 101 001	ED A9	2	4	16	
CPDR	A←(HL) HL←HL−1 BC←BC−1 Ripeti finché A=(HL) oppure BC=0	↕	② ↕	X	↕	X	① ↕	1	•	11 101 101 10 111 001	ED B9	2 2	5 4	21 16	se BC≠0 e A≠(HL) se BC=0 e A=(HL)

NOTE:
$(COPPIA)_A$ e $(COPPIA)_B$ si riferiscono, rispettivamente, agli otto bit più significativi ed agli otto meno significativi del registro.
Ad esempio: AF_A corrisponde al registro A
BC_B corrisponde al registro C

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
- ① = Il flag P/V vale 0 se BC −1=0 altrimenti P/V=1
- ② = Il flag di Z=1 se A=(HL) altrimenti Z=0

per concessione della Zilog Inc.

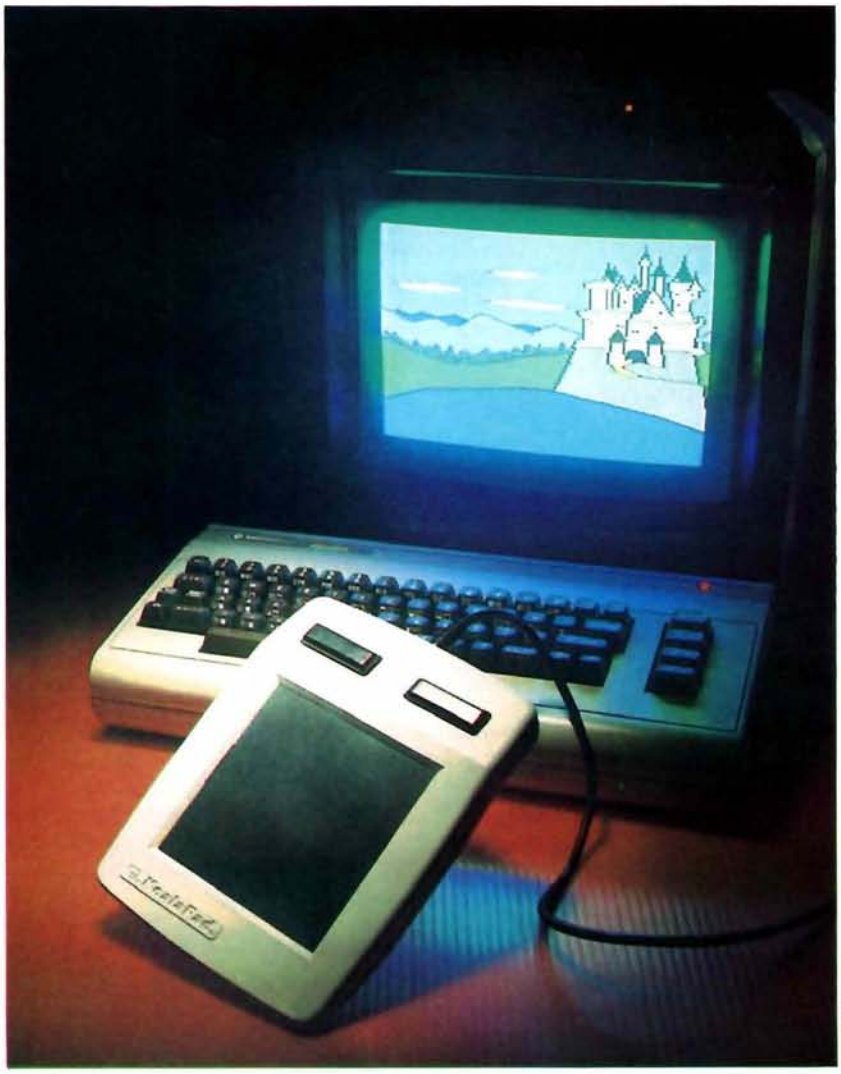
Il Koala-pad

Questo accessorio per il Commodore 64 facilita molto la creazione di immagini

Pur possedendo eccellenti capacità grafiche, il Commodore 64 non dispone ancora di periferiche grafiche di alta qualità, quali quelle adatte al Micro BBC. Infatti, le difficoltà che si incontrano nel creare immagini in alta risoluzione sul Commodore, hanno indotto molti costruttori a non fabbricare apparecchi di questo tipo. Nonostante ciò, la Koala Technology, una ditta americana, ha realizzato una tavoletta grafica che permette di utilizzare al meglio la grafica ad alta risoluzione del Commodore 64.
Diversamente da altre tavolette grafiche, quali la Grafpad (descritta a pag. 649), la Koala-pad è leggera, compatta e misura solo cm. 20,5 x 16. Il piano di lavoro (cm. 11 x 11) è costituito da una lamina in fibra di carbone, che ricopre una membrana sensibile al tatto simile a quella impiegata nella tastiera dello Spectrum. Mentre le tavolette grafiche più comuni usano uno speciale stilo per "chiudere" il circuito, sulla Koala-pad basta la leggera pressione di un dito o di una penna per muovere il cursore sullo schermo.
La membrana è composta da due strati di fili conduttori, orientati rispettivamente lungo l'asse orizzontale e verticale. Quando viene toccata, la tavoletta individua i punti di contatto tra i fili e trasmette al computer le loro coordinate. Sopra la membrana ci sono due grossi pulsanti: entrambi hanno la stessa funzione e servono per colorare i punti sullo schermo o per selezionare una delle opzioni grafiche disponibili.

Come funziona

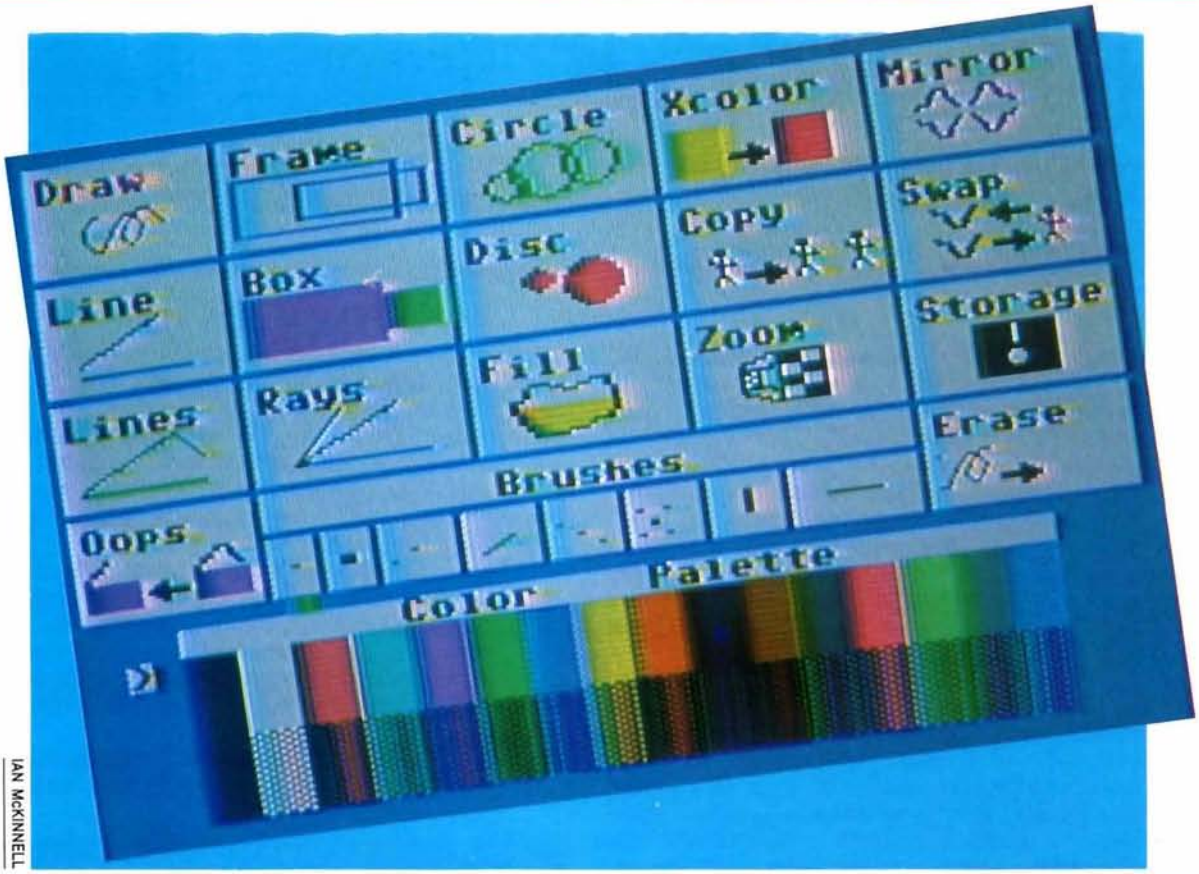
Una volta collegata la Koala-pad al computer, attraverso la porta del joystick, occorre caricare il Koala Painter (il software di controllo) da un dischetto. Avviato il programma, sul video compare un menu di apertura contenente le opzioni disponibili. Nella parte inferiore dello schermo c'è la "tavolozza" con 16 colori "veri" e 16 "sfumati". La sfumatura è ottenuta colorando i pixel alternativamente in diverse tonalità. Sopra la tavolozza ci sono 8 riquadri, corrispondenti ai "tratti" dei vari "pennelli": combinazioni diverse di punti e linee a singolo pixel. Il resto del video è occupato dalle opzioni per tracciare e colorare sagome di vario genere. Queste vengono selezionate posizionando su di esse il cursore a freccia, che l'utente muove agendo sulla membrana previa pressione di uno dei due pulsanti sulla tavoletta. Per conferma, il riquadro dell'opzione scelta lampeggia. La Koala-pad permette di tracciare linee singole, raggi (cioè segmenti con una stessa origine), figure chiuse e cerchi.
Le opzioni BOX e DISK visualizzano piccoli blocchi colorati, rispettivamente di forma quadrata e circolare. L'opzione FILL viene invece usata per colorare figure chiuse e XCOLOR per cambiare colore.
Selezionando l'opzione MIRROR si possono tracciare contemporaneamente più figure uguali. Lo schermo viene suddiviso in quattro settori ed il cursore viene posizionato su quello in alto a sinistra: le figure qui disegnate vengono immediatamente copiate sugli altri tre settori.
L'opzione ZOOM abilita la grafica su pixel ad alta precisione: l'utente può indicare una



IAN McKINNELL

Grafica a... due mani
Il software pilotato da menu permette di creare, in breve tempo, immagini grafiche complesse e di grande effetto. Le ridotte dimensioni della tavoletta Koala-pad permettono di tenerla in una mano e di disegnare con l'altra

Il menu grafico
Il menu principale è composto da una serie di riquadri che contengono il nome ed una figura esplicativa dell'opzione (alcune delle figure, però, possono trarre in inganno). Per selezionare un'opzione occorre posizionare il cursore sul relativo riquadro e premere uno dei due pulsanti di selezione della Koala-pad. Il nome dell'opzione scelta lampeggia per ricordare la funzione attiva



IAN McKINNELL

qualsiasi area del video, che viene poi ingrandita in forma di "finestra" nella parte inferiore dello schermo. Un pixel del video corrisponde ad una matrice di 8x8 pixel della finestra, ossia ad una normale cella di carattere. Si possono quindi progettare serie complete di caratteri e figure di tipo "sprite", che vengono poi copiate sullo schermo normale con l'opzione COPY.
Questa permette di definire una particolare area il cui contenuto viene copiato in un'altra parte dello schermo.
Selezionate le opzioni desiderate, il cursore a freccia va portato fuori dalla tabella e va premuto il pulsante di selezione: sullo schermo ricompare la "tela" sulla quale disegnare l'immagine. Agendo sul cursore a freccia e sul pulsante di selezione si possono tracciare linee e figure sul normale schermo grafico del Commodore 64.

Piccoli difetti

La qualità della grafica della Koala-pad è pari a quella ottenibile con i migliori programmi di grafica ad alta risoluzione in commercio. L'opzione DRAW (per tracciare a mano libera) presenta però alcuni limiti, cosa comune anche ad altri package grafici. La risoluzione della matrice di fili contenuta nella membrana non è paragonabile a quella dello schermo ad alta risoluzione del Commodore 64. Può quindi accadere che la penna non prema esattamente su un'intersezione della griglia, ma ne azioni due. Il computer, nel tentativo di interpretare i segnali ricevuti, traccerà un punto che non corrisponde con esattezza a quello indicato sulla tavoletta: si ottiene quindi una linea tremolante e confusa al posto di una retta. Un altro difetto è che, a parte lo ZOOM, non ci sono comandi per cambiare colore senza dover tornare al menu principale.
Tuttavia queste sono mancanze trascurabili, se si considera la velocità di esecuzione delle opzioni LINE e FILL.
Un altro limite del software è il sistema previsto per correggere gli errori: questi possono essere cancellati solo tornando al menu principale e selezionando l'opzione OOPS. Purtroppo, OOPS cancella tutto il lavoro eseguito dall'ultima volta che si è lasciato il menu. In alternativa si può usare l'opzione ZOOM e correggere l'errore un pixel alla volta, ma l'operazione richiede molto tempo. Una buona modifica potrebbe essere limitare l'azione di OOPS all'ultimo impiego del tasto di selezione e non all'ultima uscita dal menu principale.
La Koala-pad viene collegata alla porta per il joystick e può anche sostituirne l'impiego nei programmi scritti dall'utente. IL BASIC calcola la posizione del cursore esaminando (con PEEK) le locazioni 54297 e 54298 per le coordinate x e y.
Un'altra interessante caratteristica è la possibilità di salvare su disco intere immagini video (fino ad un massimo di 16 immagini da 8 Kbyte) e trasferirle ai programmi dell'utente: si possono quindi creare giochi d'avventura simili all'*Hobbit* composti da testi e immagini.
Purtroppo non è possibile caricare le immagi-

ni video dal dischetto direttamente in memoria, ma il manuale per l'utente del Koala Painter contiene un programma per trasferirle in memoria dall'area dello schermo in cui vengono caricate.
La risoluzione massima delle immagini video salvate è di 255x255 pixel per i programmi BASIC, ma risulta maggiore per quelli in codice macchina. Questi limiti possono creare qualche problema, dato che lo schermo ad alta risoluzione del Commodore 64 è di 320x200 pixel. La Koala-pad può trattare un massimo di 225 pixel perché questo è il numero massimo indirizzabile con un byte: per indirizzare l'intero schermo occorrerebbero valori a due byte (16 bit). Si corre quindi il rischio di perdere una parte dell'immagine perché, quando l'immagine da 8 Kbyte non è visualizzata sullo schermo, si trova immagazzinata all'interno della RAM per l'utente. Da qui viene trasferita, assieme all'informazione sui colori, nella memoria dello schermo di alta risoluzione (a partire dalla locazione 55269).
Nonostante questi piccoli difetti, la Koala-pad resta comunque una periferica grafica di grande utilità per chi desidera creare immagini grafiche ad alta risoluzione sul Commodore 64. Purtroppo il software di corredo è fornito soltanto su dischetto e questo riduce il numero dei possibili acquirenti dell'apparecchio.

KOALA-PAD

DIMENSIONI

mm. 210 x 165

VIDEO

Viene usato l'intero schermo a 320 x 200 pixel del Commodore, benché le immagini risultino limitate a 255 x 255 pixel qualora vengano incorporate in programmi BASIC

INTERFACCE

Il collegamento avviene tramite la porta per joystick del Commodore 64

DOCUMENTAZIONE

Tutto quanto concerne il funzionamento dell'apparecchio e del relativo programma è contenuto in un manuale. Tuttavia, gli esempi applicativi delle notevoli potenzialità sono scarsi

PRO

L'apparecchio consente la creazione di immagini in alta risoluzione sul Commodore 64, senza dover ricorrere alla programmazione

CONTRO

La precisione dei disegni riprodotti lascia talvolta a desiderare, a causa dell'approssimazione provocata dalla distanza degli elementi nel reticolo incorporato nel piano di lavoro

CREAZIONE DI UN'IMMAGINE

Questa serie di schermi illustra alcune delle procedure creative possibili sulla Koala-pad. Tramite il comando "ZOOM" si creano le singole lettere di una scritta. Per trasferirle nella parte inferiore dello schermo si usa il comando "COPY". Quindi, usando l'opzione "LINE", si traccia una linea sopra la scritta e si disegnano i contorni di una stella, successivamente colorata in diverse tonalità, mediante il comando "FILL", in modo da ottenere un effetto tridimensionale. Infine, si aggiungono i particolari di contorno con le opzioni "DISK" e "CIRCLE"



Una gara ... in moto

Un semplice ma divertente gioco realizzato con poche linee di BASIC

Il gioco, chiamato *Gara di moto*, trae spunto da una scena del famoso film *Tron*: consiste in un combattimento tra due avversari "motorizzati" all'interno di un'arena. Il gioco richiede una certa destrezza e buoni riflessi. Ogni giocatore guida una motocicletta che si muove a forte velocità senza possibilità di fermarla: il giocatore può solo compiere delle secche svolte ad angolo retto. Le motociclette lasciano una scia colorata e l'obiettivo del gioco è costringere l'avversario a schiantarsi in un labirinto che si stringe sempre più, delimitato dalle linee tracciate dalle motociclette.

Gara di moto è un gioco di movimento e si ottengono ottimi risultati anche sullo Spectrum ZX, che non e' certo famoso per la velocità del suo BASIC. Il programma qui presentato punta più sulla velocità che sull'eleganza; per questo motivo il listato è poco strutturato: le chiamate alle subroutine e alle altre procedure strutturate sono state eliminate per accelerare la velocità d'esecuzione. La parte iniziale del programma crea l'arena di gioco ed il quadro dei punteggi nel modo più semplice possibile per contribuire alla brevità del programma finale. La sola cosa da notare è che tra il perimetro dell'arena ed il margine dell'area video utilizzabile c'è lo spazio di un carattere. Questa sottile cornice evita che parte della grafica visualizzata dopo una collisione col bordo dell'arena scompaia dallo schermo:

```
 10 LET p=0:LET q=0
100 BORDER 0:PAPER 0:CLS
110 PRINT AT 0,1; INK 6; "Moto Uno=" q
120 PRINT AT 0,19; INK 5; "Moto Due=" p
130 INK 2
140 PLOT 8,8:DRAW 239,0:DRAW 0,159
150 DRAW -239,0:DRAW 0,-159
```

Il perimetro dell'arena viene tracciato in rosso, mentre le linee gialle e azzurre indicano rispettivamente le motociclette Uno e Due. Le variabili p e q contengono il punteggio corrente dei due avversari.

La fase successiva del programma riguarda l'inizializzazione di tutte le variabili e lo sviluppo dell'azione di gioco. Come si può vedere dal diagramma di flusso, il movimento delle motociclette è programmato in modo abbastanza semplice. Con il comando POINT possiamo controllare se la posizione corrente della motocicletta è già occupata. In caso affermativo, il programma esegue la routine di collisione altrimenti sposta la motocicletta nella nuova posizione (PLOT) e verifica che dalla tastiera non siano giunti segnali per un cambiamento di direzione. Infine viene incrementata di uno la posizione della moto, nella direzione corrente. Le variabili usate sono quattro: due per le coordinate x e y della posizione e due per la direzione lungo gli assi x e y.

Il problema del movimento contemporaneo delle moto poteva essere brillantemente risolto usando, ad esempio, quattro matrici a due elementi (x(2) e y(2)) per la posizione, ma questa scelta avrebbe notevolmente rallentato il gioco. Si è quindi preferito usare otto variabili indipendenti:

```
200 LET x=40:LET y=88
210 LET m=215:LET n=88
220 LET a=1:LET b=0
230 LET i=-1:LET j=0
```

Queste righe di programma selezionano la posizione iniziale delle motociclette e ne controllano gli spostamenti muovendole, a turno, avanti di un pixel. Le seguenti linee di programma controllano l'azione del gioco:

```
400 IF POINT(x,y)=1 THEN LET col=6:GOTO 700
410 IF POINT(m,n)=1 THEN LET col=5:LET x=m:LET
    y=n:GOTO 700
420 PLOT INK 6;x,y:PLOT INK 5;m,n
```

(Le linee da 500 a 570 contengono la routine

A TUTTO GAS!

L'idea di base di "Gara di moto" è semplice, ma il gioco richiede abilità e concentrazione. Nonostante la relativa velocità del BASIC dello ZX Spectrum, è piuttosto difficile tracciare trappole e barriere con la moto evitando, al tempo stesso, di farsi intrappolare dall'avversario



IAN McKINNELL

di controllo da tastiera, che fissa i nuovi valori per a, b, i e j).

```
600 LET x=x+a:LET y=y+b
610 LET m=m+i:LET n=n+j
620 GOTO 400
```

Forse l'unico punto poco chiaro è alla linea 410, dove le variabili della moto Uno vengono eguagliate a quelle della moto Due e viene introdotta la nuova variabile col. Ciò permette di usare un'unica routine per la collisione: x e y indicano semplicemente il punto in cui è avvenuta la collisione mentre col seleziona il colore.
La routine per la scansione della tastiera sarebbe molto rapida (è stato impiegato il veloce comando IN per leggere la tastiera), ma la presenza delle istruzioni IF...THEN ne rallenta molto l'esecuzione. I tasti per il controllo della direzione verso l'alto ed il basso sono: A e Q per la moto Uno, P e ENTER per la moto Due. I movimenti verso sinistra e verso destra sono invece guidati dai tasti X e C per la Uno e da N e M per la Due. (Una completa spiegazione delle correlazioni esistenti tra i tasti ed i byte in memoria è data a pag. 366.)

```
500 IF IN 64510 = 190 THEN LET a=0:LET b=1
510 IF IN 65022 = 190 THEN LET a=0:LET b=-1
520 IF IN 65278 = 187 THEN LET a=-1:LET b=0
530 IF IN 65278 = 183 THEN LET a=1:LET b=0
540 IF IN 57342 = 190 THEN LET i=0:LET j=1
550 IF IN 49150 = 190 THEN LET i=0:LET j=-1
560 IF IN 32766 = 187 THEN LET i=1:LET j=0
570 IF IN 32766 = 183 THEN LET i=-1:LET j=0
```

Esaminiamo infine la routine per la collisione e l'aggiornamento dei punteggi. Nel punto d'impatto, la routine visualizza una serie di cerchi concentrici con raggio di quattro, sei ed otto pixel:

```
700 FOR d=1 TO 3
710 CIRCLE BRIGHT 1;INK col;x,y,2+d*2
720 NEXT d
730 IF col=6 THEN LET p=p+1:GOTO 750
740 LET q=q+1
750 GOTO 100
```

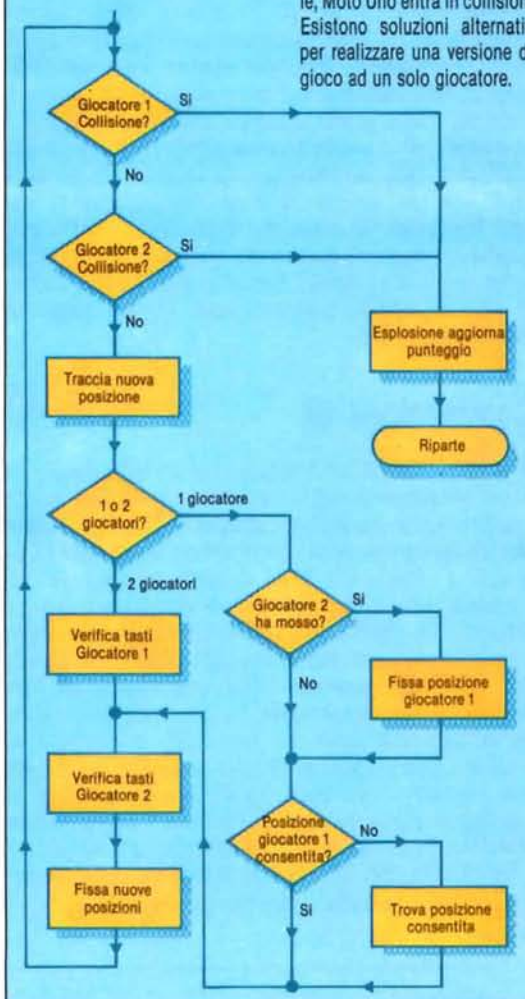
L'ultima istruzione del programma rimanda alla procedura di inizializzazione della linea 100. Il gioco va però completato con una procedura di avvio che ne faciliti l'uso e introduca una pausa tra due partite successive:

```
300 PRINT AT 10,5;INK 7;"Premere qualsiasi tasto per iniziare"
310 IF INKEY$ = "" THEN GOTO 310
320 PRINT AT 10,5;" "
```

Per salvare il gioco su cassetta si usi SAVE "Garamoto" LINE 10, cosicché il programma parte automaticamente appena viene caricato.
Gara di moto è forse un gioco un po' "spartano", ma può essere migliorato introducendo una o più pagine di spiegazione, l'opzione "Un solo giocatore" con una routine che controlla la seconda motocicletta, l'emissione di suoni ed una migliore grafica, ma tutto ciò rallenta l'azione.
Più avanti nel corso verrà offerta una versione del gioco in codice macchina che ne evidenzia tutte le potenzialità.

CONTRO IL COMPUTER

Nella versione a un solo giocatore il computer è "Moto Uno". Viene saltata la parte del programma che legge gli input da tastiera di Moto Uno, che continua a spostarsi in linea retta finché non compie un movimento illegale oppure si muove "Moto Due". Quest'ultimo movimento viene accettato, controllato ed eventualmente visualizzato. Ogni volta che Moto Uno compie un movimento illegale, il computer prevede un cambio di direzione verso destra. Se anche la nuova direzione è illegale, Moto Uno entra in collisione. Esistono soluzioni alternative per realizzare una versione del gioco ad un solo giocatore.



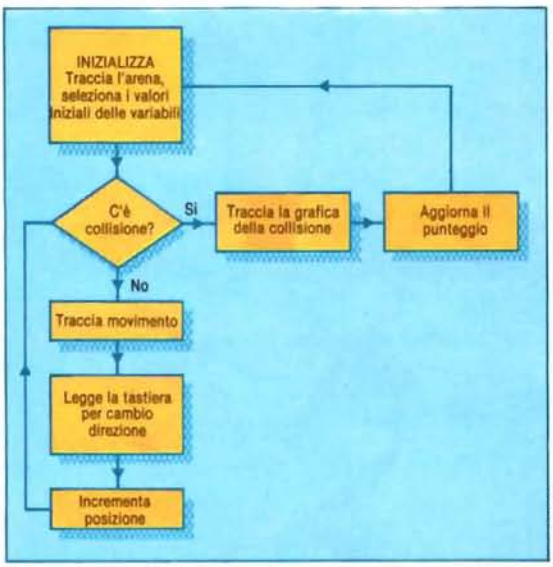
Ad esempio si può inserire, all'inizio del programma, un algoritmo per la scelta tra le versioni ad uno o due giocatori. Purtroppo, in BASIC, questa opzione tende a rallentare notevolmente il gioco. Si modifichi il programma nel seguente modo:

```
20 LET scelta=500: LET pt=-1: LET automat=0: RANDOMIZE
260 PRINT AT 10,5;"N. di Giocatori (1/2)?"
270 LET a$=INKEY$: IF a$<>"1" AND a$<>"2" THEN GOTO 270
280 IF INKEY$<>"2" THEN LET scelta=440
```

Queste linee permettono la scelta sul numero dei giocatori. La versione ad un giocatore è controllata dalle linee comprese tra 440 e 460, quella a due giocatori dalle linee comprese tra 500 e 570:

```
430 GOTO scelta
440 LET pt=SGN (RND-0.5): IF automat=pt THEN LET a=pt*j:LET b=pt*i:LET automat=0
450 IF POINT (x+a,y+b)<>1 THEN GOTO 540
460 LET pt=SGN (RND-5): LET a=a*pt:LET b=b*pt:LET d=a:LET a=b:LET b=d: IF POINT (x+a,y+b)<>1 THEN GOTO 540
490 LET a=-a:LET b=-b: GOTO 540
540 IF IN 57342=190 THEN LET i=0:LET j=1:LET automat=1
```

Infine aggiungere :LET automat=1 alle linee comprese tra 550 e 570



L'opzione "Un solo giocatore"

Questo diagramma di flusso semplificato mostra la struttura del programma modificato per un solo giocatore. Ovviamente, nella versione a due giocatori, ogni procedura è duplicata

Controllo del traffico

Usiamo un joystick per controllare il veicolo realizzato in "Laboratorio"

Il Commodore 64 ed il Micro BBC usano joystick di diversa concezione. Il primo adotta un joystick di tipo Atari con quattro interruttori interni ed un pulsante di fuoco. Il secondo usa invece un joystick analogico o "paddle". Il suo funzionamento non si basa sull'apertura e chiusura di una serie di contatti, ma sull'azione di due potenziometri uno per i movimenti destra/sinistra, l'altro per quelli avanti/indietro (è tuttavia possibile adattare un joystick digitale alla porta analogica del BBC). Esaminiamo i due joystick separatamente.

Commodore 64

Il joystick di tipo Atari per il Commodore 64 viene collegato ad una delle apposite porte situate accanto all'interruttore di accensione: nel programma qui presentato viene usata la porta 2. Spostando in avanti la cloche del joystick si chiude uno dei quattro contatti interni. La porta 2 è direttamente collegata alla locazione di memoria 56320 e la chiusura del contatto provoca l'azzeramento di uno dei bit di tale locazione; parimenti, la chiusura di un interruttore esterno, collegato al "buffer box", causa l'azzeramento di un bit del registro dei dati nella porta utente. Il seguente programma visualizza sullo schermo il valore contenuto nel registro dei dati ed i cambiamenti che si verificano quando si muove il joystick o si preme il pulsante di fuoco.

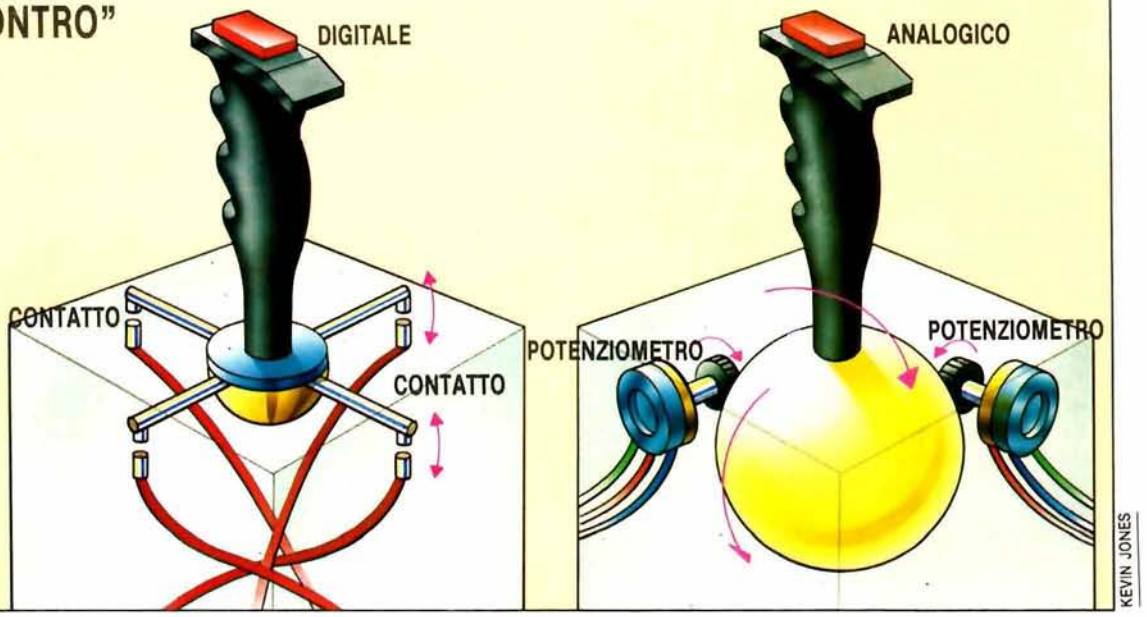
```
 10 REM ***LETTURA DEL JOYSTICK CBM 64***
 20 PORT2 = 56320
 30 JOY = PEEK(PORT2):GOSUB 500
 40 PRINT CHR$(145);JOY,B$
 50 GOTO30
 60 :
500 REM S/R DI CONVERSIONE IN BINARIO
510 B$ = " ":N = JOY
520 FOR D = 1 TO 8
530 N1 = INT(N/2):R = N-2*N1
540 B$ = B$ + STR$(R):N = N1
550 NEXT D
560 RETURN
```

Dopo pochi minuti d'esecuzione risulta chiaro che i bit contenuti nella locazione di memoria corrispondono ai quattro interruttori di direzione ed al pulsante di fuoco. Di solito, quando la cloche è in posizione centrale, il contenuto della locazione è 127, ossia 01111111 in binario. Spostando la cloche in avanti il valore passa a 126 (01111110) quindi il bit 0 è collegato al contatto anteriore del joystick (direzione in "avanti"). La seguente tabella descrive l'effetto degli interruttori del joystick sul contenuto della locazione di memoria:

Joystic	Decimale	Binario
Centrale	127	01111111
Alto	126	01111110
Basso	125	01111101
Sinistra	123	01111011
Destra	119	01110111
Fuoco	111	01101111

ANALOGICO "CONTRO" DIGITALE

Il joystick digitale o "a contatti" registra solo i movimenti della cloche nelle quattro direzioni principali, ma non ne misura l'entità. Il joystick analogico possiede due potenziometri, disposti perpendicolarmente tra loro, che misurano i movimenti della cloche e trasmettono l'informazione tramite due tensioni variabili. Queste vengono trasformate in valori numerici dal convertitore A/D



I movimenti diagonali chiudono due contatti contemporaneamente, ma qui non vengono presi in esame nel controllo del veicolo. Comunque, il loro effetto sulla locazione di memoria è il seguente:

Joystick	Decimale	Binario
Avanti/Sinistra	122	01111010
Avanti/Destra	118	01110110
Indietro/Sinistra	121	01111001
Indietro/Destra	117	01110101

Il prossimo programma usa il joystick per controllare i movimenti del veicolo a due motori, collegati al "box di uscita" nel modo visto a pag. 1066. Il joystick viene innestato sulla porta 2, posta sul lato destro del Commodore.

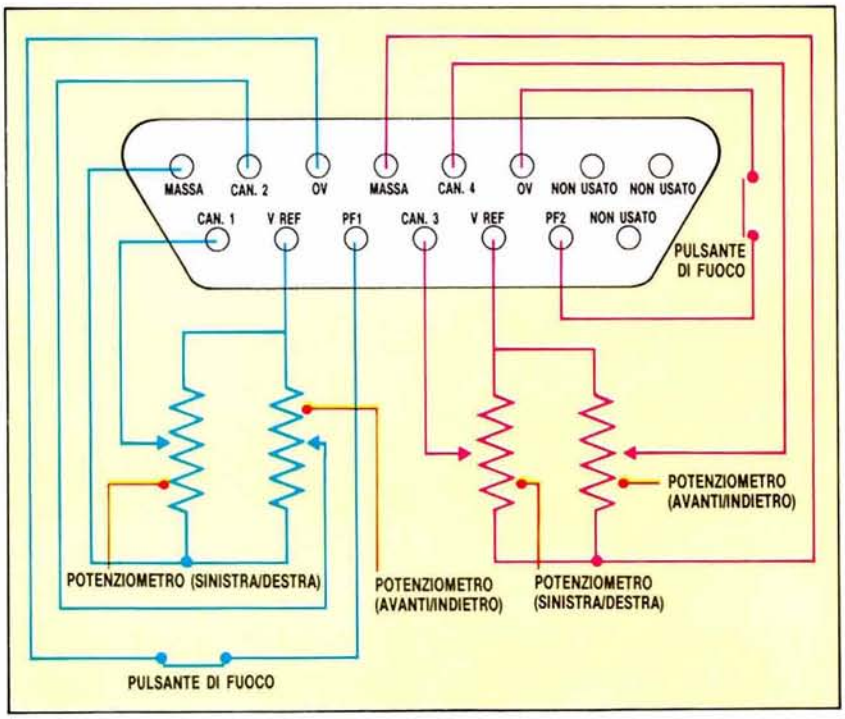
```
10 REM JOYSTICK CBM 64
20 DDR = 56579:REGDAT = 56577
30 POKEDDR,225 :REM TUTTE USCITE
40 JOY = PEEK (56320) :REM PORTA 2 PER JOYSTICK
50 GOSUB 1000: REM VERIFICA JOYSTICK
60 POKEREGDAT,0:GOTO40
90:
1000 REM S/R VERIFICA JOYSTICK
1005 IF JOY = 127 THEN POKEREGDAT,0
1010 IF JOY = 126 THEN POKEREGDAT,5
1020 IF JOY = 125 THEN POKEREGDAT,10
1030 IF JOY = 123 THEN POKEREGDAT,6
1040 IF JOY = 119 THEN POKEREGDAT,9
1050 IF JOY = 111 THEN POKEREGDAT,0:END
1060 RETURN
```

Micro BBC

Il joystick analogico del BBC si basa su due potenziometri, che trasmettono informazioni sui movimenti sinistra/destra e avanti/indietro. Diversamente da quello del Commodore, che fornisce solo informazioni sulla direzione del movimento, quello del BBC comunica anche le posizioni intermedie della cloche.
Un potenziometro non è che una resistenza dotata di tre terminali. A due di essi viene applicata una data tensione, mentre dal terzo è possibile prelevare una tensione che varia a seconda della posizione assunta dalla manopola di controllo.
Nei potenziometri lineari, se la manopola si trova a metà della corsa, la tensione prelevata è la metà di quella applicata, altrimenti essa varia in modo continuo tra 0 e la tensione applicata. I potenziometri sono comunemente usati per il controllo del volume nelle radio e nei registratori. Nei joystick analogici lo spostamento della manopola è controllato dal movimento della cloche.
Di solito i joystick per il Micro BBC vengono venduti a coppie; lo schema mostra i collegamenti della porta analogica dell'apparecchio per i joystick 1 e 2. Il computer applica ai due potenziometri del joystick una tensione fissa; quelle di controllo, variabili a seconda della posizione del joystick, vengono restituite al computer su due canali: il canale 1 per i segnali analogici del potenziometro relativo ai movimenti sinistra/destra, il canale 2 per quelli relativi ai movimenti avanti/indietro. Il pulsante di fuoco usa invece un comune interruttore.
I segnali analogici vengono quindi trasformati in digitali da un convertitore, interno al computer, che per ciascun joystick confronta le tensioni in arrivo con quella di controllo. Il tempo di conversione per ogni lettura di canale è di 10 millisecondi. Infine, i segnali digitali vengono usati per controllare il veicolo a due motori.
Per interpretare i segnali che raggiungono la porta analogica, il BASIC del BBC adopera l'istruzione ADVAL. Il valore ritrasmesso da ADVAL è un numero compreso tra 0 e 65520: il valore massimo corrisponde quindi ad un segnale in arrivo pari alla tensione di controllo, mentre il minimo si ottiene con una tensione zero. Tensioni intermedie producono valori di ADVAL compresi tra i due limiti. Nell'applicazione con il veicolo a due motori vengono presi in considerazione solo i due valori limite.
Il canale letto da ADVAL è indicato dal numero tra parentesi che segue la parola chiave, ad esempio: ADVAL(1) legge il canale 1.
L'istruzione ADVAL(0) svolge due diverse funzioni. I due bit meno significativi corrispondono, rispettivamente, ai pulsanti di fuoco dei joystick 1 e 2. Se viene premuto il pulsante di fuoco del joystick 1, X=ADVAL (0) AND 3 ritrasmette un valore di uno. Con X=ADVAL (0) DIV 256 si ottiene invece il numero del canale sul quale è stata eseguita l'ultima conversione A/D.
Poiché la conversione di ciascuna lettura di canale dura 10 millisecondi, per elaborare i segnali in arrivo sui quattro canali dei due joystick occorrono 40 millisecondi. Nell'applicazione con il veicolo a due motori vengono

Porta analogica del BBC

Di solito i joystick per il Micro BBC vengono venduti in coppia e si collegano ad un'unica porta. La disposizione dei piedini della porta analogica, posta sul retro dell'apparecchio, mostra che i collegamenti dei joystick 1 e 2 sono identici. Quindi i due joystick possono essere innestati l'uno al posto dell'altro



usati solamente i canali 1 e 2. L'istruzione *FX 16,2 li seleziona e, contemporaneamente, disattiva i canali 3 e 4.
Il seguente programma permette di controllare con i joystic del BBC il veicolo a due motori.

```
10 REM **CONTROLLO JOYSTICK + BBC**
 20 DDR = &FE62:REGDAT = &FE60
 30 ?DDR = 255: REM TUTTE USCITE
 40 REM SELEZIONA CANALI A—D 1_2
 50 *FX16,2
 60 REPEAT
 70 PROCverifica_ joystick
 80 UNTIL fuoco = 1
 90 END
 100 :
110 DEF PROCverifica_joystick
120 REPEAT
130 canale = ADVAL(0) DIV 256
140 UNTIL canale<>0: REM ATTESA PER CONVERSIONE
150 IF canale = 1 THEN PROCsinistra_ destra
160 IF canale = 2 THEN PROCavanti _indietro
170 ENDPROC
180 :
190 DEF PROCsinistra_destra
200 REPEAT
210 valjoy = ADVAL(1)
220 IF valjoy<100 THEN ?REGDAT = 9
230 IL valjoy<64000 THEN ?REGDAT = 6
240 fuoco = ADVAL(0) AND 3
250 PRINT ?REGDAT,canale,valjoy
260 UNTIL (valjoy>100 AND valjoy<6400) OR fuoco =
270 ?REGDAT = 0
280 ENDPROC
290 :
300 PROCavanti_indietro
310 REPEAT
320 valjoy = ADVAL(2)
330 IF valjoy<100 THEN ?REGDAT = 10
340 IF valjoy>64000 THEN ?REGDAT = 5
350 fuoco = ADVAL(0) AND 3
360 PRINT?REGDAT,canale,valjoy
370 UNTIL (valjoy>100 AND valjoy<6400) OR fuoco =
380 ?REGDAT = 0
390 ENDPROC
```

RISPOSTE AGLI ESERCIZI

1) La taratura del veicolo a due motori gemelli viene eseguita cronometrando il tempo impiegato a percorrere distanze diverse, ad esempio: 10, 20, 50, 100 e 150 centimetri.
Calcolando la velocità tenuta nei vari percorsi e facendone la media, si può ricavare la distanza media percorsa in un secondo. Lo stesso principio vale per i cambiamenti di direzione: in questo caso si fanno percorrere al veicolo angoli di diversa ampiezza e si cronometra il tempo impiegato. La verifica dei motori, eseguita accendendoli e spegnendoli di seguito per un certo lasso di tempo, presenta notevoli difficoltà, non ultima quella degli intervalli di tempo scanditi dal programma di controllo, che vanno misurati con precisione assoluta. Differenze di pochi centesimi di secondo possono causare notevoli differenze nelle distanze e negli angoli percorsi. Simili problemi possono essere limitati introducendo demoltipliche tra i motori e le ruote motrici.

2) Il listato di programma dato a pag. 1093 permette di guidare il veicolo attraverso un percorso ad ostacoli: ripercorrerlo all'indietro è un po' più complesso. Occorre usare una coppia di variabili per ciascuna direzione e per il suo opposto (ad esempio, avanti e indietro).

Commodore 64

```
17 A(1) = 5:B(1) = 10:A(2) = 10:B(2) = 5
18 A(3) = 6:B(3) = 9:A(4) = 9:B(4) = 6
```

Per ottenere la sequenza invertita dei movimenti si possono usare queste routine:

```
  92 GOSUB 2000: REM SEQUENZA INVERTITA
2000 REM S/R PER SEQUENZA INVERTITA
2010 FOR I = C TO 1 STEP —1
2020 FOR J = 1 TO 4
2030 IF DR(I,1) = A(J) THEN POKE REGDAT,B(J):J = 4
2040 NEXT J
2050 T = TI
2060 IF(TI—T) <DR(I,2) THEN 2060
2070 NEXT I
2080 STOP
2090 RETURN
```

Micro BBC

```
1020 DIM DR(100,2),A(10),B(10)
1025 A(1) = 5:B(1) = 10:A(2) = 10:B(2) = 5
1026 A(3) = 6:B(3) = 9:A(4) = 9:B(4) = 6
1115 PROCinverte_sequenza
2000 DEF PROCinverte_sequenza
2010 FOR I = C TO 1 STEP —1
2020   FOR J = 1 TO 4
2030      IF DR(I,1) = A(J) THEN?REGDAT = B(J):J = 4
2040   NEXT J
2042   TIME = 0
2045   REPEAT UNTIL TIMEDR(I,2)
2047   ?REGDAT = 0
2050 NEXT I
2055 PRINT "PREMI C PER CONTINUARE"
2060 REPEAT A$ = GET$
2070 UNTIL A$ = "C"
2080 ENDPROC
3000 FORI = 1TOC:PRINTDR(I,1),DR(I,2)
3010 NEXT
```

3) Poniamo che la linea 7 sia "avanti", la 6 "indietro", la 5 "sinistra" e la 4 "destra":

Micro BBC

```
 10 REM INTERRUTTORI ESTERNI — BBC
 20 DDR = &FE62:REGDAT = &FE60
 30 ?DDR = 15:REM LINEE 4—7 IN INGRESSO
 40 ?REGDAT = 0
 50 PROC verifica—input
 60 GOTO50
 70 :
 80 DEF PROC verifica—input
 90 IF(?REGDAT AND 240) = 240 THEN ?REGDAT = 0
100 IF(?REGDAT AND 128) = 0 THEN ?REGDAT = 5
110 IF(?REGDAT AND 64) = 0 THEN ?REGDAT = 10
120 IF(?REGDAT AND 32) = 0 THEN ?REGDAT = 6
130 IF(?REGDAT AND 16) = 0 THEN ?REGDAT = 9
140 ENDPROC
```

Commodore 64

```
  10 REM INTERRUTTORI ESTERNI — CBM 64
  20 DDR = 56579:REGDAT = 56577
  30 POKE DDR: REM LINEE 4—7 IN INGRESSO
  40 POKE REGDAT,0: REM MOTORI SPENTI
  50 REM L7 = AVANTI, L6 = IND., L5 = SIN., L4 = DES.
  55 REM UNA DELLE LINEE È A LIVELLO 0?
  60 IF(PEEK(REGDAT)AND240)<>240THENGOSUB1000
     :GOTO60
  70 POKEREGDAT,0:GOTO60
  80 :
1000 REM S/R SCANSIONE LINEE IN INGRESSO
1005 IF(PEEK(REGDAT)AND128) = 0THENPOKEREGDAT,5
1010 IF(PEEK(REGDAT)AND64) = 0THENPOKEREGDAT,10
1020 IF(PEEK(REGDAT)AND32) = 0THENPOKEREGDAT,6
1030 IF(PEEK(REGDAT)AND16) = 0THENPOKEREGDAT,9
1040 RETURN
```

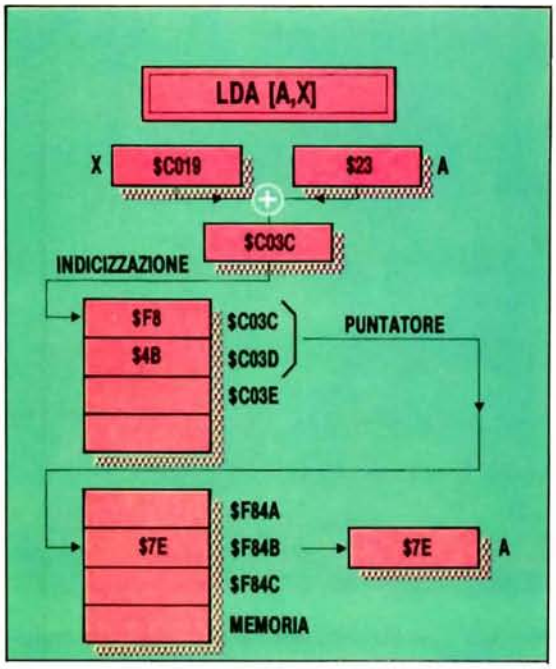
Salti ad effetto

L'indirizzamento indiretto del 6809 in una routine per visualizzare caratteri

L'indirizzamento indiretto è una caratteristica applicabile a molti altri metodi di indirizzamento. In pratica, si tratta di un ulteriore passo avanti nel calcolo dell'indirizzo effettivo (l'indirizzo dal quale o nel quale viene effettivamente estratta o depositata un'informazione). L'indirizzo effettivo viene calcolato con varie tecniche già descritte (ad accesso diretto, con indirizzamento indicizzato ecc.), ma nel caso dell'indirizzamento indiretto, il contenuto dell'indirizzo calcolato e quello della locazione di memoria immediatamente successiva vengono trattati come un indirizzo: quello effettivo dal quale viene caricata (o nel quale è depositata) l'informazione.
Consideriamo, ad esempio, i seguenti valori:

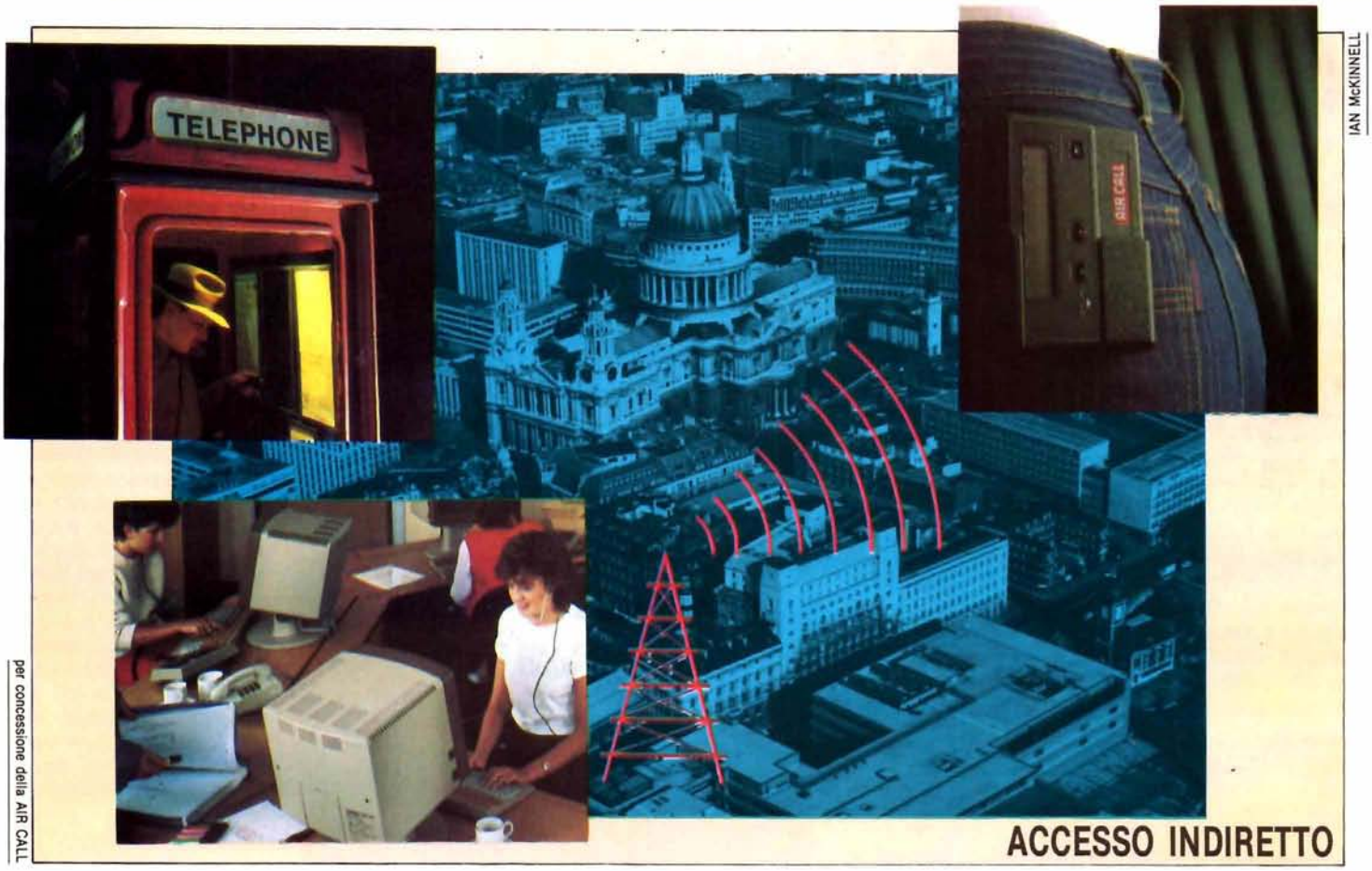
INDIRIZZO	CONTENUTO
3000	40
3001	0A
400A	F2

L'istruzione LDA $3000 carica il valore $40 nell'accumulatore A e l'indirizzo effettivo è dunque $3000. L'indirizzamento indiretto viene sempre specificato facendo precedere e seguire l'operando da parentesi quadre, pertanto LDA [$3000] carica in A il valore $F2, poiché l'indirizzo effettivo ($400A nel nostro esempio) è il valore memorizzato nelle locazioni $3000 e $3001. I valori contenuti in $3000 e $3001 costituiscono quindi un "puntatore" o "vettore" all'indirizzo effettivo $400A. Ricordiamo che sul 6809 gli indirizzi vengono immagazzinati con il byte alto (hi) che precede il byte basso (lo), quindi $40 viene registrato in $3000 e $0A in $3001. Questa regola viene chamata "convenzione hi-lo". I processori Zilog Z80 e MOS Tech 6502 usano invece la convenzione opposta: $0A (il byte basso dell'indirizzo) verrebbe immagazzinato in $3000 e $40 (il byte alto) in $3001.
L'indirizzamento indiretto è particolarmente utile se abbinato all'indirizzamento indicizzato. L'istruzione LDA [X,A] ricorre all'indirizzamento indicizzato indiretto. Nel calcolo dell'indirizzo effettivo, prima somma il contenuto di A e X, quindi usa il valore a 16 bit immagazzinato nella locazione risultante e nella successiva, come indirizzo effettivo. Il contenuto di quest'ultimo viene infine caricato in A. Sul 6809 l'indirizzamento indiretto è impiegato molto meno che non su altri processori (ad esempio, sul 6502 o sullo Z80), poiché possiede efficaci metodi di indirizzamento indicizzato. Tuttavia, in particolari casi, l'indirizzamento indiretto si dimostra molto vantaggioso, come, ad esempio, nel controllo delle periferiche. Diversamente dalle "famiglie" 8080 e 8086 dell'Intel, i processori Motorola possiedono particolari locazioni di memoria riservate all'input/output: i registri di comunicazione delle periferiche si trovano nella "mappa di memoria" principale del sistema e i valori vengono memorizzati o caricati da questi registri come se si trattasse di normali locazioni di memoria. Alle routine per il controllo delle periferiche (ad esempio, una routine di stampa) occorre l'indirizzo del registro d'interfaccia per quella periferica. Supponiamo che questo indirizzo venga rilocato (ossia cambiato di posizione) nella mappa di memoria oppure che il sistema sia collegato a più periferiche dello stesso tipo: in simili casi il controllo della periferica risulta molto più agevole se modifichiamo la locazione di memoria che contiene l'indirizzo del registro di comunicazione della periferica, anziché essere costretti ad individuare e modificare l'indirizzo della periferica tutte le volte che compare in un programma. Tramite il puntatore, la routine "dialoga" direttamente con la periferica di sua competenza.
L'esempio precedente illustra l'uso generico dell'indirizzamento indiretto: quando è possibile una variazione degli indirizzi ai quali un programma fa riferimento, conviene usare appositi puntatori (situati in locazioni fisse) a tali indirizzi: se cambia l'indirizzo effettivo, basta modificare il contenuto del puntatore.
Questa tecnica viene comunemente usata nelle strutture dette "tabelle di salto" (jump table), che contengono una lista di puntatori. I



Indirizzamento indiretto indicizzato

I parametri [A,X] dell'istruzione LDA sono tra parentesi quadre, quindi il contenuto di X ($C019) deve essere sommato a quello di A ($23) per ottenere un indirizzo a 16 bit ($C03C). I due byte in sequenza formano un vettore che punta all'indirizzo effettivo ($F84B) il cui contenuto ($7E) viene infine caricato in A. Poiché X viene sommato ad A prima dell'accesso indiretto, questa procedura è detta "indirizzamento indiretto pre-indicizzato". Il suo equivalente opposto, o "post-indicizzato", richiede che l'indirizzamento indiretto venga calcolato prima dell'indicizzazione



IAN McKINNELL

per concessione della AIR CALL

ACCESSO INDIRETTO

Sebbene l'indirizzamento indiretto sia di importanza vitale per il funzionamento del computer, è difficile trovare esempi di questa tecnica nella vita reale. Una buona analogia può essere quella con i servizi di "chiamata telefonica via radio". Quando un utente vuol telefonare ad un altro abbonato, non può chiamarlo direttamente perché quest'ultimo può trovarsi dovunque: deve invece rivolgersi al servizio di chiamata che, a sua volta, chiama l'abbonato via radio. Questo semplice e flessibile sistema si basa sul principio dell'accesso (o indirizzamento) indiretto dei propri abbonati

sistemi operativi possiedono numerose routine che eseguono operazioni elementari del sistema (la lettura di caratteri battuti sulla tastiera o la loro visualizzazione su schermo, ad esempio), spesso utilizzate anche dai programmi in codice macchina. Per accedere ad esse, i programmi si servono di una tabella di salto: l'indirizzo delle routine "puntate" dai vettori della tabella può essere tranquillamente modificato, senza modificare necessariamente i programmi che le chiamano ma solo la tabella di salto. In altre parole, queste routine vengono sempre raggiunte in modo indiretto, attraverso i puntatori della tabella di salto. Quando viene sviluppata una nuova versione di un sistema operativo o viene prodotta una ROM "aggiornata", spesso le routine del sistema operativo cambiano di posizione. Tuttavia, se la collocazione della tabella di salto resta immutata ed i puntatori vengono corretti in modo da puntare alla nuova disposizione delle routine, qualsiasi programma scritto per il vecchio sistema operativo funziona senza modifiche anche sul nuovo.
Una tecnica comunemente usata in molti sistemi operativi prevede l'impiego di un solo punto d'ingresso al sistema attraverso cui passano tutte le chiamate di subroutine. In tal caso, in uno dei registri della CPU viene depositato un "codice di funzione", corrispondente alla subroutine desiderata. Il codice di funzione si comporta come un indice o un offset per ricavare il puntatore dalla tabella di salto. Tramite il puntatore, il controllo viene infine trasferito alla routine chiamata.

Supponiamo di avere 4 Kbyte di ROM con indirizzo iniziale $F000: i primi 256 byte contengono una tabella con un massimo di 128 indirizzi di subroutine archiviate in ROM. La routine d'ingresso, ossia l'indirizzo mediante il quale vengono chiamate tutte le routine del sistema operativo, si trova alla locazione $F100. Quando nell'accumulatore B viene caricato un codice di funzione compreso tra 0 e 127, la routine d'ingresso lo usa per trasferire il controllo alla subroutine corrispondente e per riportarlo al programma di chiamata al termine della subroutine. La routine di chiamata per il numero di funzione 1 potrebbe essere:

```
LDB #1      inserisce il codice di funzione in B
JSR $F100   chiama la routine d'ingresso
```

Questa è invece la routine d'ingresso:

```
LDX $F000   primo indirizzo della tabella di salto
LSLB        sposta B di una posizione (1 byte) a sinistra (il che
            equivale a moltiplicarlo per 2) dato che ogni indi-
            rizzo in tabella è composto da due byte: in $F002 e
            $F003 si trova il puntatore corrispondente al codice
            di funzione 1, in $F004 e $F005 il puntatore per quello
            di funzione 2 ecc.
BRA [B,X]   trasferisce il controllo all'indirizzo trovato nel
            puntatore della tabella, indicato dal codice di fun-
            zione
```

Si noti che è usato BRA (o JMP) e non BSR (o JSR) per chiamare la routine di sistema: in questo modo l'istruzione RST, che chiude la routine chiamata, riporta il controllo direttamente alla routine del programma applicativo, invece che alla routine d'ingresso.
L'esempio seguente mostra un ulteriore impiego dell'indirizzamento indiretto, stavolta abbinato alla memoria video. Su molti micro

questa occupa una parte della memoria principale e può essere indirizzata direttamente se è richiesta una maggiore velocità di visualizzazione.
Supponiamo che il video, di 16 linee x 64 caratteri, occupi un'area di memoria compresa tra $E000 e $E3FF e che la posizione del cursore sia un valore a 16 bit, interno a quest'area, contenuto nell'indirizzo $E400. La prima subroutine cancella lo schermo inserendo uno "spazio" (codice ASCII 32) in tutte le posizioni di carattere. La seconda stampa il carattere caricato in A nella cella occupata dal cursore, a meno che non si tratti di un "ritorno-carrello" (codice ASCII 13): in tal caso la routine cancella la parte della linea alla destra del cursore e posiziona quest'ultimo all'inizio della linea successiva. Nell'esempio il cursore è rappresentato dal carattere "sottolineatura" (_).

```
SPAZIO  EQU 32          Cod. ASCII per "spazio"
CR      EQU 13          Cod. ASCII per "ritorno-carrello"
INIZIO  EQU $E000       Inizio della memoria video
LUNGH   EQU 1024        Dimensioni della memoria video (16 li-
                        nee x 64 caratteri = 1024)
CURSOR  EQU $E400       $E400 e $E401 puntano all'indirizzo cor-
                        rente del cursore nell'area di memoria
                        video
        ORG $1000
CARCUR  FCB 95          Carattere sottolineat. (cod. ASCII 95)
```

Subroutine per cancellare lo schermo

```
        LDA #SPAZIO     Carica il carattere "spazio" in A
        LDX #INIZIO     Punta cursore a inizio schermo
        STX CURSOR      Memorizza in CURSOR ($E400, $E401) l'at-
                        tuale posizione del cursore
        LDB #LUNGH      Carica le dimensioni schermo in B
CICL1   STA [CURSOR]    Deposita uno spazio nella posizione
                        corrente del cursore
        INC CURSOR      Incrementa la posizione del cursore
        DEC B           Decrementa la quantità di memoria
                        video compresa tra la posizione del
                        cursore e la fine della memoria video
        BGT CICL1       Spazio successivo, a meno che la me-
                        moria video non sia finita
        STX CURSOR      Riporta il cursore in posizione iniziale
        LDA CARCUR      Carica in A il cod. ASCII del carattere
                        per il cursore
        STA [CURSOR]    Deposita il carattere del cursore nella
                        posizione corrente del cursore
        RTS
```

Subroutine che visualizza il carattere in A, se visualizzabile

```
        CMPA SPAZIO     "Spazio" è il primo carattere stampa-
                        bile in ASCII
        BLT NONST       Se l'accumulatore contiene un cod.
                        ASCII minore di 32, il carattere non è
                        stampabile, quindi: GOTO NONST
        STA [CURSOR]    Deposita il carattere nella posizione
                        del cursore
        INC CURSOR      Incrementa la posizione del cursore
CHKEOS  LDX #INIZIO     Verifica per fine dello schermo
        LEAY #LUNGH,X   Fine dello schermo in Y
        CMPY CURSOR     Se la posizione del cursore supera la
                        fine dello schermo...
        BGT FINITO      ... abbiamo raggiunto la fine, quindi:
                        GOTO FINITO
```

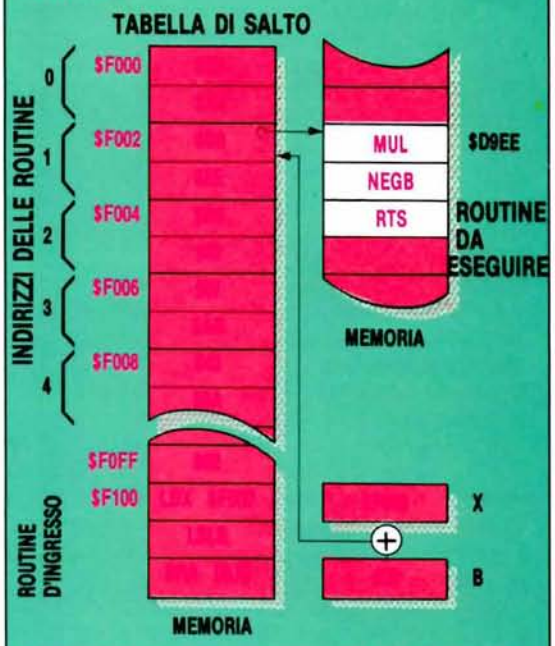
Subroutine per lo scrolldello schermo

```
SCROLL  LEAY 64,X Y     dista una linea da X (fine della area di
                        memoria)
        LDB #LUNGH      Calcola le dimensioni dello "scroll"
        SUBB £64        Sottrae 64
CICL2   LDA,Y+          Sposta i caratteri indietro di una linea
                        (notare l'auto-incremento: pag. 1098)
        STA,X+
        DECB
        BGT CICL2       Esegue uno "scroll" completo
        LDD CURSOR      Cursore all'inizio dell'ultima linea
        SUBD #64
        STD CURSOR
        BRA FINITO
```

Subroutine di verifica per il ritorno-carrello

```
NONST   CMPA #CR        Il carattere non stampabile è un "Ri-
                        torno-carrello"?
        BNE FINITO      Se "no", lo ignora
        LDD CURSOR      Si provi a calcolare come questa istru-
                        zione ricavi l'inizio della linea succes-
                        siva (notare l'abbinamento AND-ma-
                        schera)
                        ANDB #%11100000
        ADDD #64
        STD CURSOR
        BRA CHKEOS      Verifica se lo schermo è finito
FINITO  LDA CARCUR      Carica il carattere del cursore in A
        STA CURSOR      Lo deposita nella posizione corrente
                        del cursore
        RTS
```

LA TABELLA DI SALTO



Questa tabella di salto è composta da una lista di 128 puntatori d'indirizzo a due byte, situati tra $F000 e $F0FF. Ogni puntatore contiene l'indirizzo iniziale di una particolare routine in memoria. Per eseguire una routine (che nel nostro esempio si trova alla locazione $D9EE) si deve caricare nell'accumulatore B il codice di funzione (ad esempio, $01) che la identifica e chiamare (JSR) la routine d'ingresso, il cui indirizzo iniziale è $F0FF. Se le routine si trovano in ROM (cioè fanno parte di un software residente in ROM come il sistema operativo), può darsi che nel manuale si trovi il codice di funzione e l'indirizzo iniziale della routine d'ingresso. Questa moltiplica il codice di funzione per 2 e lo usa come offset (scarto) da sommare al primo indirizzo della tabella per trovare il puntatore d'indirizzo della routine chiamata: il puntatore alla routine $01 si trova all'indirizzo $F002 (=$F000+2x$01), quello alla routine $02 all'indirizzo $F004 (=$F000+2x$02) e così via. Il puntatore viene quindi usato dalla routine d'ingresso in un'istruzione di salto indiretto, per trasferire il controllo alla routine situata a $D9EE. Si noti che la routine d'ingresso esegue un "salto" (anziché una chiamata) fino alla routine desiderata; perciò quando l'esecuzione incontra un'istruzione RTS essa torna al punto del programma dal quale era stata inizialmente chiamata la routine d'ingresso. La tabella di salto (soprattutto se abbinata ad una routine d'ingresso) consente al programmatore di modificare e rilocare le routine che indirizza. Inoltre permette di eseguire, con la nuova versione di un sistema operativo, anche i programmi scritti prima della revisione: i codici di funzione e l'indirizzo della routine d'ingresso restano tuttavia immutati, anche se possono cambiare i contenuti dei puntatori in tabella.

Pacman

Concepito dalla Atarisoft, questo gioco è stato uno dei primi disponibili su home computer

Il *Pacman* è stato il primo gioco arcade di tipo "labirinto". La sua formula è stata poi imitata da molti altri giochi per home computer, quali l'*AticAtac* ed il *Jet Set Willy*. In essi il "protagonista" deve aggirarsi lungo una fitta rete di percorsi, raccogliendo i tesori ed evitando i pericoli che gli si parano davanti. In particolari condizioni invece, il giocatore può cambiare tattica ed assalire a sua volta i nemici che lo stanno inseguendo.
Per esempio, nel *Sabre Wulf*, il giocatore diventa invisibile ad alcuni animali della giungla, se tocca particolari orchidee. Questa idea è stata tratta dai giochi di tipo Adventure, dove il possesso di una spada magica o di un altro talismano conferisce al giocatore particolari vantaggi.
Nel *Pacman*, il protagonista si trova al centro di un labirinto ed il giocatore deve guidarlo sullo schermo, cercando di fargli inghiottire il maggior numero di punti lungo il cammino. Contemporaneamente, deve guardarsi dai fantasmi che lo inseguono e cercano di intrappolarlo per ucciderlo. Se riesce ad inghiottire una "pillola d'energia", la figurina può dare la caccia ai fantasmi e mangiarseli ottenendo un maggior punteggio. Altri punti si guadagnano mangiando i frutti che ogni tanto compaiono nel labirinto.
Le varie versioni del gioco spesso si dimostrano inferiori alle aspettative, perché un gioco progettato per un particolare microcomputer esalta le caratteristiche proprie di quella macchina.
Questo principio è vero soprattutto per il software degli apparecchi Atari, che sono relativamente costosi, ma presentano svariati giochi Arcade di elevata qualità. Tuttavia, le nuove versioni di *Pacman* (per lo Spectrum e per il Vic 20), sono notevolmente diverse dall'originale.
Il *Pacman* per gli Atari visualizza un grande labirinto con buone caratteristiche grafiche e sonore, sprite che si muovono con rapidità e vari livelli di difficoltà. Considerando poi che gli home computer Atari non sono, in effetti, delle vere e proprie consolle da gioco, la versione si dimostra abbastanza fedele a quella originale. La stessa cosa non si può dire per le versioni Spectrum e Commodore. Sul Vic-20 i programmatori hanno dovuto ridurre di 3/4 le dimensioni del labirinto mostrato sull'Atari, mentre gli sprite sono il doppio degli originali: è quindi molto difficile sfuggire ai fantasmi. In compenso la definizione grafica è eccellente, i movimenti sono rapidi e la qualità del suono pari a quella Atari.
Il *Pacman* per lo Spectrum possiede istruzioni più complete e permette di controllare il gioco da tastiera, mentre le versioni per Atari e Commodore usano solo il joystick. La scarsa qualità della "colonna sonora" dipende dalle limitate capacità di suono dell'apparecchio. Anche la grafica è deludente: il labirinto è simile all'originale, ma gli sprite si muovono a scatti. Mentre la mediocrità della versione Vic-20 è dovuta ai limiti imposti dall'hardware, la povertà del *Pacman* per lo Spectrum non ha alcuna concreta giustificazione.
Alla sua comparsa, nel 1980, il gioco ottenne un successo travolgente, anche se oggi appare superato. La Atarisoft ha intrapreso varie azioni legali contro case concorrenti, che hanno sviluppato giochi simili al *Pacman*, ma ha lasciato trascorrere troppo tempo prima di pubblicare versioni "ufficiali" per altre macchine. Nei primi anni l'Atarisoft vendeva migliaia di copie di *Pacman*, ma in seguito il mercato ha offerto giochi arcade migliori. Dunque, chi acquista oggi una versione del *Pacman* prodotta dall'Atarisoft per il Vic-20 o per lo Spectrum deve sapere che non si tratta di una versione fedele a quella del gioco originale.

Pacman: Per tutti i computer Atari, per il Vic-20 della Commodore e per lo Spectrum.

Editori: Atari Corporation

Autori: Atari

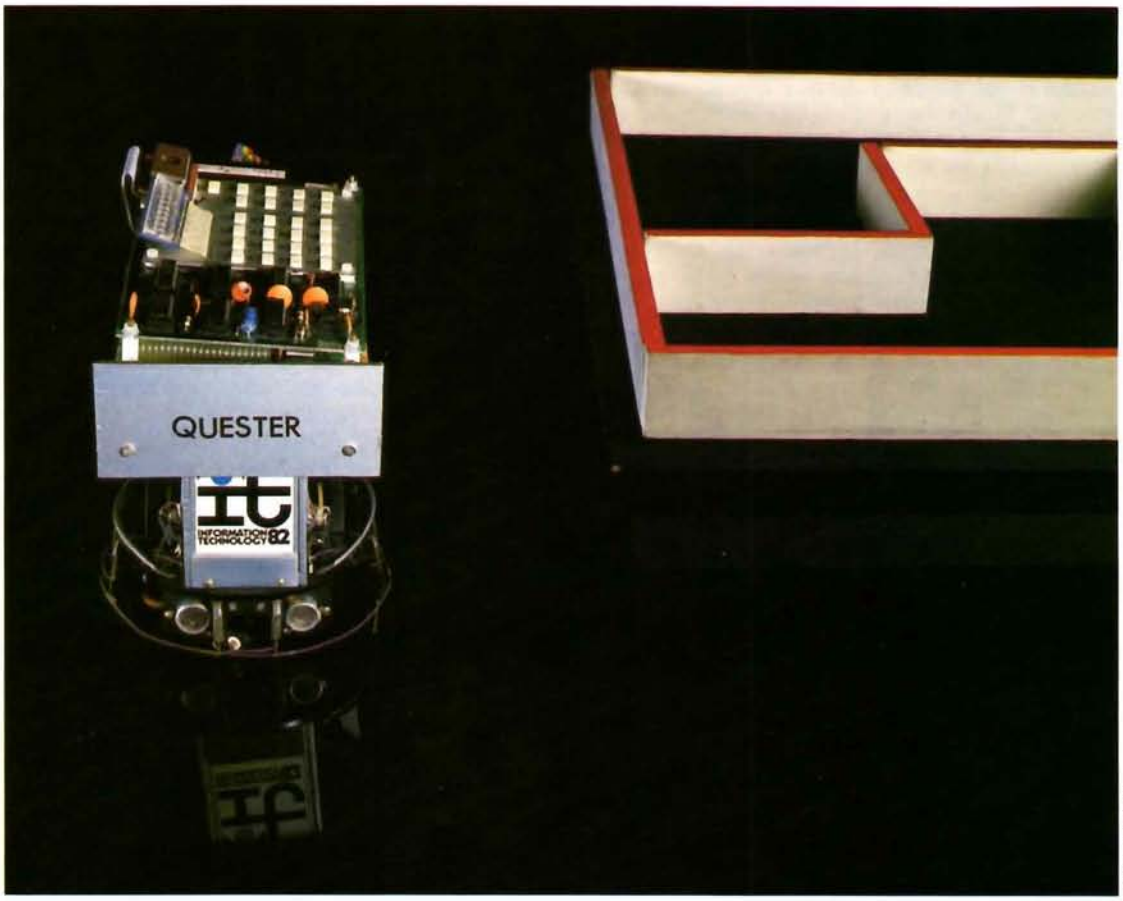
Joystick: necessario (opzionale sullo Spectrum)

Formato: Atari e Vic-20, cartuccia; Spectrum, cassetta

LIZ DIXON

Topi e labirinti

Come controllare i movimenti di un meccanismo robotizzato ?



MARCUS WILLY

TOPI...

Nelle gare di Micro Mouse, ormai diffuse in moltissimi paesi, piccoli robot (chiamati "topi") competono tra loro nella ricerca dell'uscita da un labirinto. Queste gare offrono inesauribili spunti per apprendere ed approfondire i principi della robotica. I topi in gara, generalmente realizzati da semplici appassionati, possiedono spesso una completa gamma di sensori ottici, audio, ecc.

... E LABIRINTI

In gara, ai topi viene concesso qualche giro di prova per "imparare", con qualunque metodo, il percorso del labirinto, purché non vi siano comunicazioni con l'esterno (ossia con l'operatore). Scopo delle gare è quello di far uscire il topo dal labirinto nel minor tempo possibile

Uno dei più semplici metodi di controllo del movimento di un robot consiste nell'impiego di un dispositivo meccanico, in grado di leggere una specie di scheda sagomata. Il contorno della scheda è interpretato dal meccanismo di lettura che genera i movimenti e i cambiamenti di direzione. Esistevano, in passato, piccole macchine a molla che funzionavano così e bastava qualche colpo di forbice per creare un nuovo "percorso".

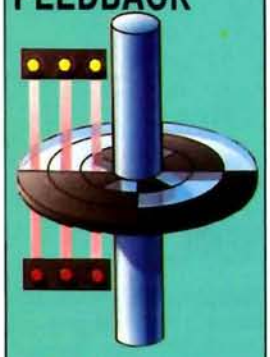
Un altro sistema è basato sull'uso di svariati relais elettromeccanici, azionati secondo una sequenza ben precisa. Tuttavia, questi metodi di controllo meccanico hanno un limitato campo applicativo, a causa dell'elevato costo e della scarsa precisione delle componenti meccaniche. Nonostante ciò, simili apparecchi hanno creato un precedente per le apparecchiature odierne.

Oggi, uno dei più diffusi sistemi di controllo ricorre all'impiego di una speciale "pista", tracciata sul suolo, che il robot riesce a individuare e seguire. Il metodo è simile a quello delle piste per macchinine giocattolo che vengono agganciate alla pista e percorrono un solco. Per i robot, si ricorre invece a due tipi di piste: una semplice striscia tracciata sul suolo oppure un filo percorso da corrente elettrica.

Per seguire il primo tipo di pista, il robot viene dotato di un fotosensore (talvolta una cellula sensibile ai raggi infrarossi) che stabilisce se il robot si trova sopra una zona "scura" oppure "luminosa"; in genere il pavimento è di colore scuro, mentre la pista è colorata in bianco. Il sensore produce un segnale elettrico diverso a seconda dei due casi, cosicché il robot segue sempre il percorso che provoca il più alto segnale elettrico nel sensore.

I problemi sorgono quando il segnale indica che il robot si è allontanato dalla pista: con un solo sensore, l'unica soluzione è vagare a caso, sperando di incrociare nuovamente la pista! Questo metodo appare molto meno affidabile di quanto sia in realtà. Infatti, se il segnale cade mentre il robot si sta muovendo verso sinistra, è logico che l'azione immediatamente successiva sia una rotazione a destra, fino a ritrovare la pista. Per ridurre questi erratici sbandamenti, si può dotare il robot di due fotosensori, orientati verso i due

FEEDBACK



Tramite un codificatore assiale, il robot può conoscere l'ampiezza di un movimento, misurando la rotazione dell'asse. Il codificatore consiste in un semplice disco calibrato, fissato all'asse di rotazione, e suddiviso in settori circolari concentrici diversamente colorati (opachi o trasparenti), che si trova interposto tra una serie di minuscole sorgenti luminose e una serie di cellule fotosensibili, che intercettano o meno il passaggio di luce.
L'accuratezza di un codificatore assiale dipende dalla quantità di anelli: più sono, maggiore è la precisione. Nell'illustrazione un codificatore con tre anelli: il codice risultante è composto da tre bit e varia da 000 a 111 (da 0 a 7 in decimale). Ciò determina un'accuratezza di 360/8 = 45°. Otto settori concentrici offrono un'accuratezza di 1,41°

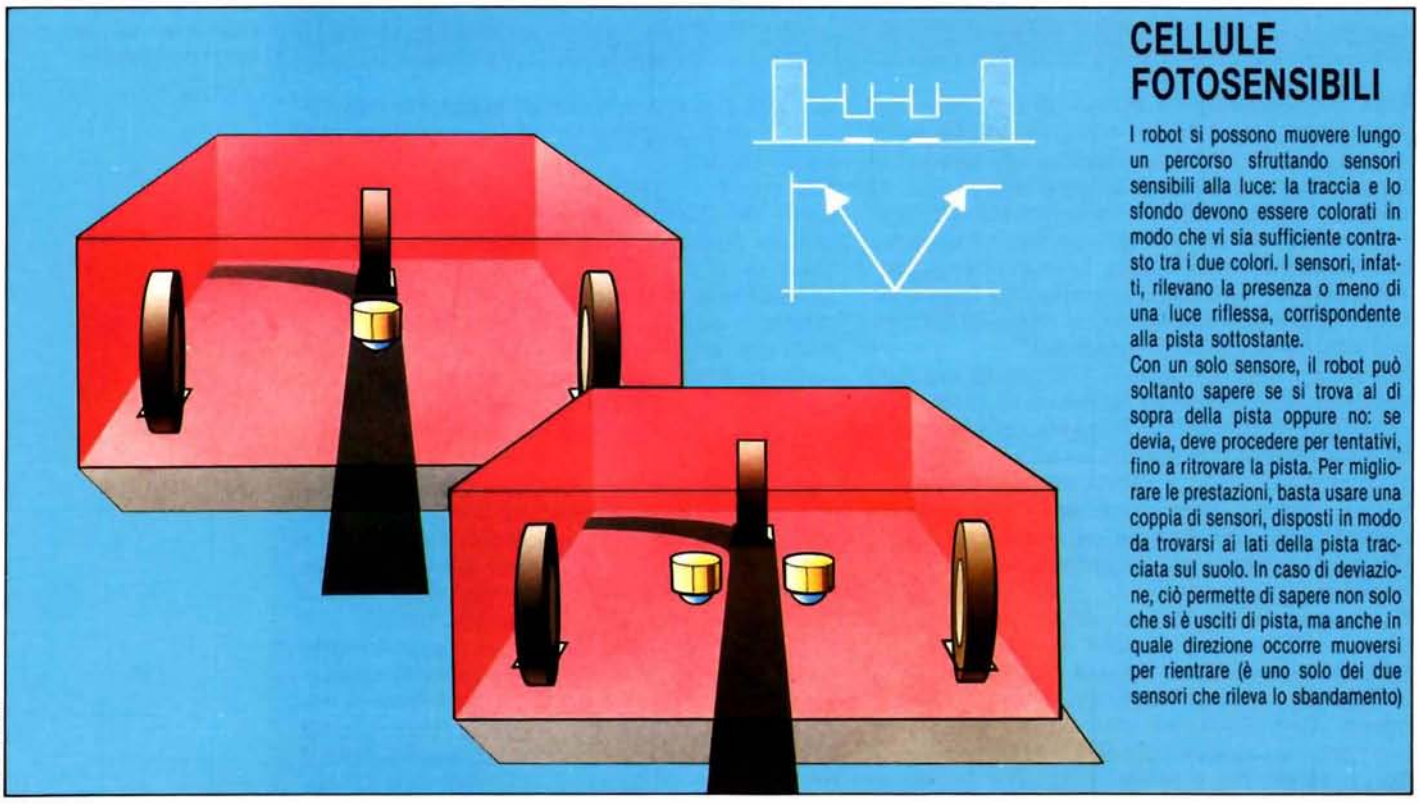
margini della pista; finché il robot si muove nella giusta direzione, entrambi i sensori provocano un segnale di livello basso. Appena c'è uno sbandamento, uno dei due sensori passa ad un livello alto e il robot si accorge non solo di essere uscito dal percorso, ma anche verso quale direzione: se è il sensore sinistro a segnalare la deviazione (avendo incrociato la striscia di riferimento) basta una leggera rotazione a sinistra, finché il relativo segnale non torna a livello basso. Per quanto riguarda il computer, non ha importanza il colore della pista e quello del suolo, purché vi sia un sufficiente contrasto tra i due. Il programma di controllo, provvede ad aggiustare la direzione in base ai segnali ricevuti dai sensori.
Il secondo metodo accennato ricorre all'impiego di una pista formata da un cavetto elettrico, lungo il quale viene fatta passare una corrente. Quest'ultima produce un piccolo campo magnetico attorno al cavetto, che viene rilevato da un apposito sensore. Non occorrono sensori estremamente complicati, anzi basta un piccolo avvolgimento su di un nucleo ferromagnetico: il segnale rilevato viene poi amplificato e utilizzato come nel caso dei fotosensori. Molti robot industriali lavorano in base a questo principio e il cavetto viene interrato lungo i percorsi, che in genere sono obbligati e ripetitivi. Una striscia dipinta sul suolo, in questo caso, sarebbe poco affidabile, poiché eventuali tracce di sporco confonderebbero il robot.

Controllo a distanza

Ovviamente, il robot può essere controllato a distanza da un operatore. Questo sistema è particolarmente adatto in quelle circostanze in cui le pericolose condizioni ambientali non permettono che sia un uomo a svolgere un determinato lavoro; ad esempio nella manipolazione di reagenti chimici o materiali radioattivi oppure in ambienti troppo caldi o troppo freddi o comunque dannosi all'organismo umano. Un noto robot per tali applicazioni è quello russo, il Lunokhod 1, depositato sulla Luna nel 1970. Il controllo di questo apparecchio semovente avveniva tramite segnali radio inviati dalla Terra.
Simili apparecchiature vengono controllate in modo abbastanza diverso da quello dei piccoli modelli radiocomandati. Il segnale radio può essere formato da impulsi analogici, proporzionali alla quantità di movimento del robot, oppure digitali, in cui è la combinazione di bit che stabilisce come e di quanto muovere il robot. A causa delle interferenze, le comunicazioni di tipo analogico tendono ad esser meno affidabili di quelle digitali. Si pensi a quanto sia poco stabile, su una normale radio, la ricezione di emittenti lontane: scariche e improvvisi "oscuramenti" rendono talvolta impossibile l'ascolto.
Le comunicazioni digitali sono più sicure, anche se non esenti da interferenze che possono far "perdere" un bit o modificarne il valore. Per sicurezza i messaggi digitali diretti ai robot vengono spesso ripetuti più volte ed il robot non agisce se non dopo aver ricevuto due sequenze identiche tra loro.

Sistemi di feedback

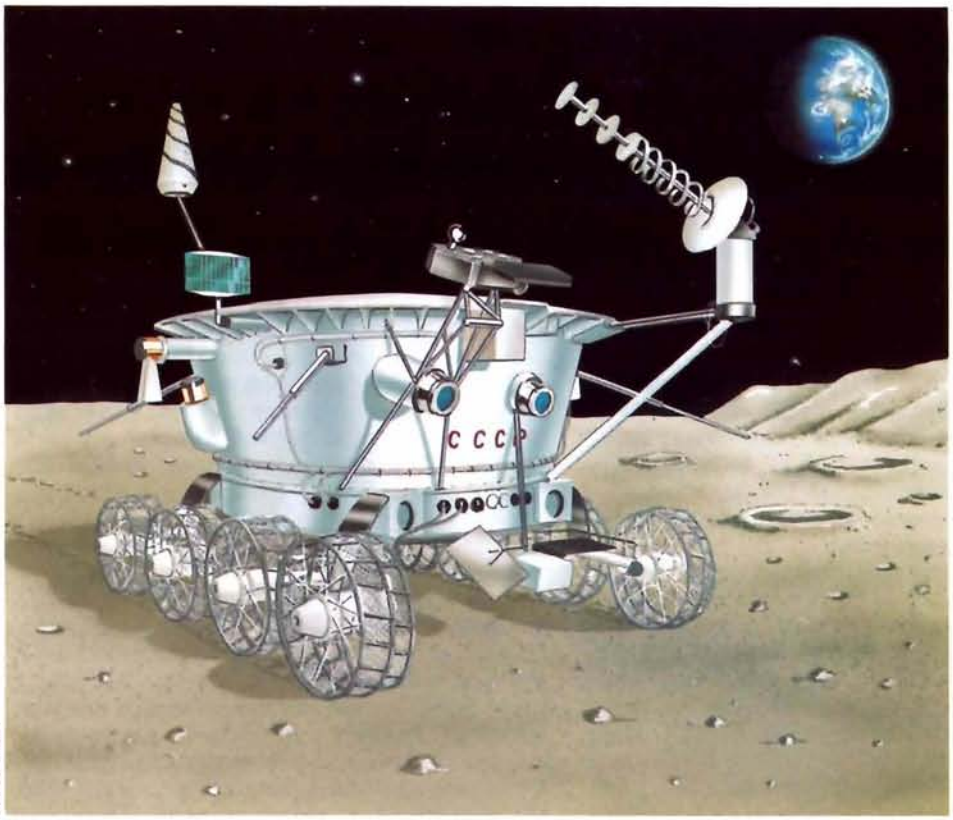
Un sistema più sofisticato si basa su di un



KEVIN JONES

CELLULE FOTOSENSIBILI

I robot si possono muovere lungo un percorso sfruttando sensori sensibili alla luce: la traccia e lo sfondo devono essere colorati in modo che vi sia sufficiente contrasto tra i due colori. I sensori, infatti, rilevano la presenza o meno di una luce riflessa, corrispondente alla pista sottostante.
Con un solo sensore, il robot può soltanto sapere se si trova al di sopra della pista oppure no: se devia, deve procedere per tentativi, fino a ritrovare la pista. Per migliorare le prestazioni, basta usare una coppia di sensori, disposti in modo da trovarsi ai lati della pista tracciata sul suolo. In caso di deviazione, ciò permette di sapere non solo che si è usciti di pista, ma anche in quale direzione occorre muoversi per rientrare (è uno solo dei due sensori che rileva lo sbandamento)



STEVE CROSS

PASSI DA GIGANTE PER I ROBOT

Il modulo sovietico Lunokhod 1 è stato depositato sul suolo lunare nel 1970 per raccogliere informazioni sulla natura della crosta e dell'atmosfera del nostro satellite naturale. Pur non trattandosi, in senso stretto, di un vero e proprio robot (essendo controllato via radio dalla Terra), questo modulo ha permesso di eseguire innumerevoli esperimenti: molti più di quelli possibili se fosse stata impiegata una equipe umana.
Come tutte le apparecchiature controllate a distanza nello spazio, il Lunokhod 1 agiva con un certo ritardo (per la precisione tre secondi, trattandosi della Luna) rispetto al segnale di comando

funzionamento "ad anello", nel quale il robot ritrasmette, in qualche modo, informazioni all'emittente. In sostanza, si instaura una sorta di dialogo, chiamato "feedback", tra trasmittente e robot. Tanto per fare un esempio, il messaggio trasmesso potrebbe essere: «Muoviti in avanti». Ad esso il robot potrebbe reagire chiedendo, a sua volta: « Mi hai detto di muovermi in avanti?» e il trasmittente, infine, potrebbe confermare il comando. Questa specie di dialogo, anche se può sembrare infantile, consente di evitare al robot pericolosi errori nel maneggiamento di sostanze radioattive oppure la caduta in un cratere lunare.
Per estensione, a questo sistema appartengono anche le comunicazioni tramite raggi infrarossi (come quelli adoperati da molti comandi a distanza per televisori) o tramite ultrasuoni o anche normali suoni quali un battito di mani. Qualunque sia la forma di trasmissione adottata, la tecnica di comando/conferma rimane sempre la stessa.
Se non vi è molta distanza tra operatore e robot, le comunicazioni possono avvenire tramite un collegamento elettrico e senza dover ricorrere a sofisticati sistemi di conferma (basta quella visiva). Inoltre, il formato dei segnali può essere seriale o parallelo, vale a dire che i bit vengono trasmessi, uno alla volta in sequenza, lungo un singolo cavetto oppure, tutti assieme, lungo più cavetti. La comunicazione di tipo parallelo consente una maggiore rapidità nel controllo del robot. Inoltre, considerato che molti computer possiedono porte parallele, è possibile guidare il robot comodamente attraverso l'uso della tastiera del computer.

Come accennato, la verifica sulla corretta esecuzione dei comandi può essere fatta "a vista" dall'operatore, riducendo la complessità dell'intero sistema. Ma se l'operatore non ha un diretto controllo visivo (ad esempio, il robot si trova sulla superficie della Luna) oppure l'apparecchiatura è controllata dal programma di un computer, allora il robot deve esser dotato di una maggiore "intelligenza".
Ciò che occorre, essenzialmente, è una qualche forma di feedback, ossia un circuito di conferma. Se l'operatore controlla a vista i movimenti di un robot, può facilmente valutare se la sua direzione e velocità sono quelle desiderate: il feedback, in tal caso, è costituito dallo stesso contatto visivo.
In situazioni diverse, è lo stesso robot che deve inviare opportune segnalazioni di feedback ed il metodo più semplice consiste nel dotare i meccanismi del robot di speciali "codificatori assiali". Questi non sono altro che dei dischetti fissati a ciascun asse in rotazione, sui quali sono riportati dei codici, letti da appositi strumenti. Ciò consente di misurare con precisione la rotazione compiuta da ciascun motore del robot: se il comando trasmesso comporta un movimento in avanti di 90 centimetri, durante lo spostamento viene tenuto sotto controllo il codificatore assiale fissato sulle ruote, finché esso non segnala un valore equivalente al percorso assegnato. Questo genere di feedback può essere ritrasmesso al computer di origine oppure, se nel robot è incorporato un microprocessore, è quest'ultimo che compie tutte le verifiche necessarie ad assicurare l'esatta esecuzione dei comandi.

Una sinfonia corale

Valutiamo pro e contro di tre diffusi package "integrati" per applicazioni commerciali

Come esposto precedentemente, i requisiti di un sistema integrato di programmi prevedono l'omogeneità dell'ambiente operativo (il tipo ed il formato dei comandi devono essere coerenti in tutti i moduli) e il libero scambio di dati tra i vari moduli del sistema. Esistono più modi per raggiungere tali obiettivi.

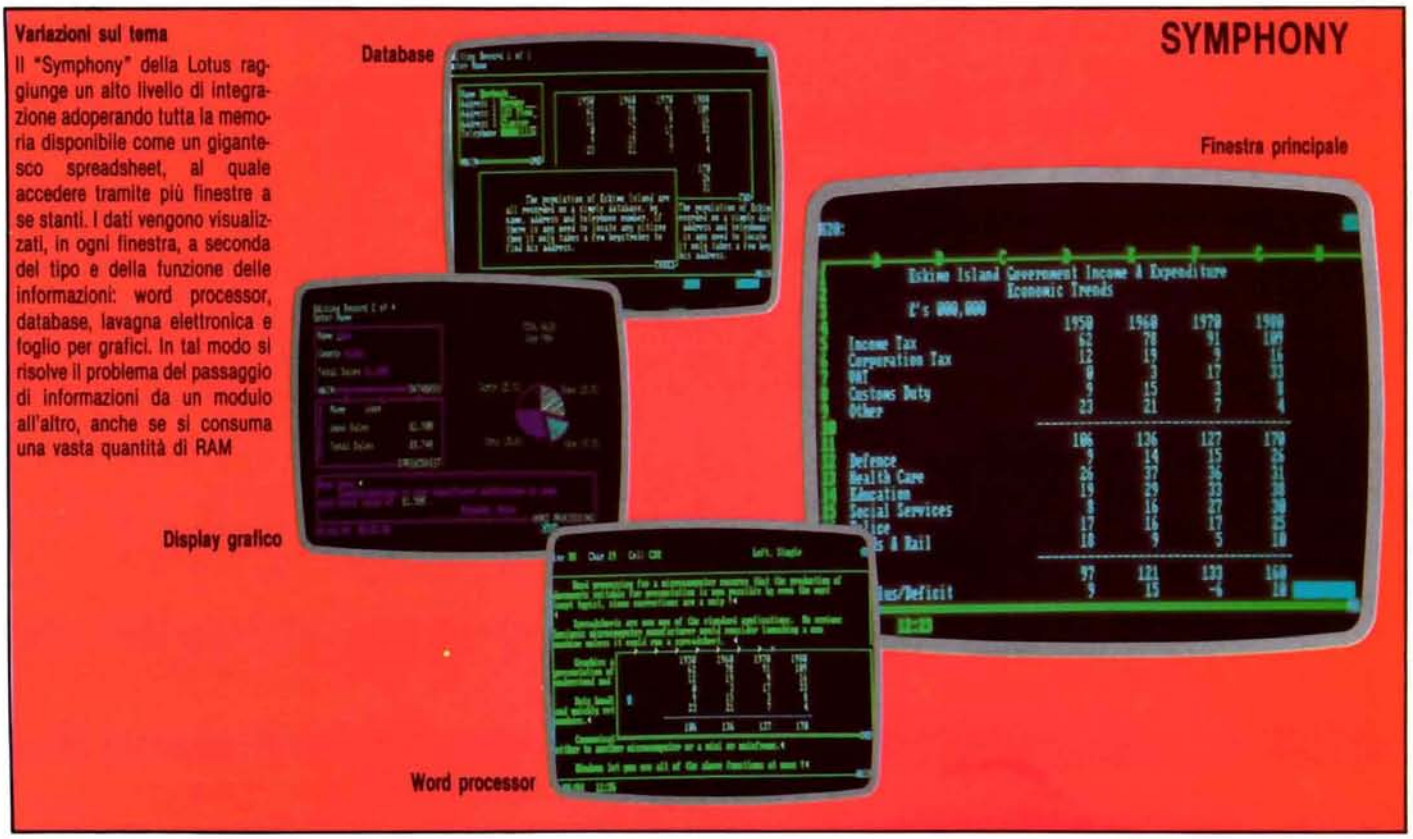
Il *Lotus 1-2-3* ricorre all'ormai consueto formato degli spreadsheet, in cui le cifre e le formule, immesse in una matrice di "celle", possono essere istantaneamente modificate o ricalcolate. Tuttavia, il *Lotus 1-2-3* offre molte possibilità applicative, non limitate alla preparazione di bilanci o di analisi finanziarie preventive. Le celle possono, ad esempio, contenere scritte, nominativi o numeri telefonici, cosicché una parte dello schermo può contenere un tabella di informazioni pertinenti alla particolare applicazione (i nominativi dei clienti di una ditta, assieme ai codici loro assegnati, per esempio). Poiché il *Lotus 1-2-3* offre anche opzioni per la ricerca e per il riordino di dati, questa tabella può essere efficacemente adoperata come un piccolo database.

Inoltre, esiste la possibilità di estrarre una serie di valori dalle celle, per visualizzarli sottoforma di grafico, evitando così di dover utilizzare un programma a parte per ottenere una rappresentazione efficace dei dati. Infine, il package prevede anche un certo numero di funzioni di editing, grazie alle quali è possibile redigere appunti e promemoria, benché la limitata memoria a disposizione non consenta veri e propri interventi di word processing.

Grazie a questa felice combinazione di prestazioni, molti possessori di computer possono limitarsi all'acquisto del solo *Lotus 1-2-3*, per le proprie esigenze di lavoro: l'accesso a tutte le informazioni memorizzate avviene tramite il formato spreadsheet, un risultato, non certo ottenibile con programmi più conformi alla tradizione. Supponiamo che l'utente del *Lotus 1-2-3* sia il proprietario di una catena di negozi. Per ciascun negozio, è utile conoscere l'ammontare settimanale, mensile, trimestrale e annuale delle vendite. Il formato offerto da uno spreadsheet è ideale: le formule vengono immesse nelle celle in modo tale che le uniche cifre da immettere, di volta in volta, siano quelle relative alle vendite

Variazioni sul tema

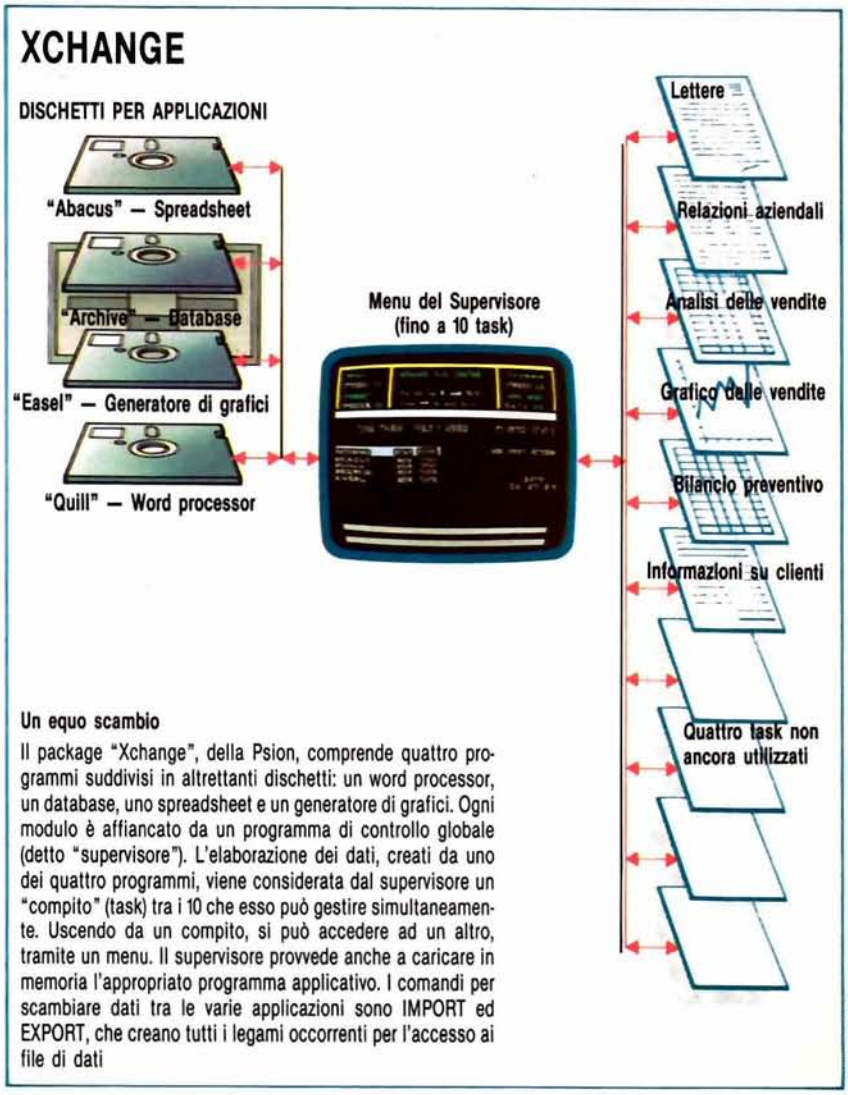
Il "Symphony" della Lotus raggiunge un alto livello di integrazione adoperando tutta la memoria disponibile come un gigantesco spreadsheet, al quale accedere tramite più finestre a se stanti. I dati vengono visualizzati, in ogni finestra, a seconda del tipo e della funzione delle informazioni: word processor, database, lavagna elettronica e foglio per grafici. In tal modo si risolve il problema del passaggio di informazioni da un modulo all'altro, anche se si consuma una vasta quantità di RAM



settimanali (tutte le altre vengono calcolate automaticamente).
Fin qui, niente di eccezionale. Ma, se il proprietario intendesse riorganizzare l'ordine di visualizzazione in base all'ammontare delle vendite, occorrerebbe riordinare settimanalmente tutte le informazioni e questa funzione, in una normale lavagna elettronica, non è prevista.
Con il *Lotus 1-2-3* ciò si ottiene facilmente e in breve tempo. Premendo pochi tasti si può anche ottenere una tabella sul rendimento settimanale di ciascun negozio oppure un grafico sull'andamento globale, mese per mese: le possibilità sono limitate soltanto dalla fantasia.

Una sinfonia per il Lotus

Symphony è il successore del *Lotus 1-2-3*, di cui ricalca il formato a spreadsheet. Tuttavia, consente una suddivisione dello schermo in comode finestre, ciascuna delle quali evidenzia una parte dell'intera lavagna elettronica. Il formato di ogni finestra varia a seconda del tipo di informazioni visualizzato.
Se si tratta di testi, la finestra assomiglia ad un piccolo schermo per word processing, evidenziando i margini ed i tabulatori. Se invece si tratta di un grafico, nella finestra compaiono gli assi di riferimento, adeguatamente suddivisi in scala. Per le informazioni tratte dal database vengono adoperate apposite finestre, in modo da far apparire i dati raccolti come in una scheda tradizionale. Insomma, pur trattandosi di uno spreadsheet cresciuto a dismisura, il *Symphony* dà l'impressione di possedere quattro applicazioni distinte ma simultanee.
Come il *Lotus 1-2-3*, anche il *Symphony* è capace di "imparare" sequenze di comandi: ciò facilita l'esecuzione di procedure che ricorrono con frequenza. Tali sequenze, veri e propri minuscoli programmi, sono talvolta indicate col termine "macrocomandi". Il *Symphony* possiede anche un proprio linguaggio di programmazione ad alto livello e ciò permette di creare sequenze di istruzioni per organizzare convenientemente l'accesso alle informazioni. I programmi vengono conservati nello stesso formato riservato ai dati. Così, la gestione di un magazzino, ad esempio, può essere interamente programmata nel linguaggio del *Symphony* e divenire parte integrante del package, accessibile per ogni sorta di applicazioni. Acquisita una certa familiarità col *Symphony*, programmare nel linguaggio incorporato diviene molto più semplice e rapido che non ricorrere a linguaggi esterni, quali il BASIC: tutte le funzioni relative alla produzione di grafici, alla ricerca e al riordinamento di dati sono già predisposte!
Symphony è soltanto uno dei tanti package oggi disponibili sul mercato: il *Framework* della Ashton Tate è il più forte concorrente. La gamma di funzioni è pressoché identica, ma la struttura interna dei dati è di gran lunga più ampia. Entrambi i package, però, sono piuttosto costosi ed occupano una porzione di memoria non certo indifferente. Infatti, il *Symphony* richiede 320 Kbyte di RAM, e per applicazioni di un certo livello si arriva fino a 512 Kbyte. Al *Framework* occorrono come minimo 256 Kbyte. Va da sé, dunque, che i due programmi sono adatti soltanto a computer con microprocessori a 16 bit.
Un'occupazione di memoria così ampia deriva dal fatto che raramente i programmi vengono suddivisi in moduli e letti a turno dai dischetti. In effetti, considerato che almeno in teoria il costo della RAM è in continuo calo, non è tanto insensato sviluppare programmi impostati su una vasta occupazione di memoria.
Tuttavia, ai fini pratici, occorrerà molto tempo prima che la maggioranza dei computer possieda così vaste porzioni di RAM. Programmi sul genere di *Symphony*, dunque, sono fortemente legati alle limitazioni hardware degli apparecchi e le loro prestazioni sono il frutto di un'attenta, ma non certo economica, programmazione. Esistono ancora altri aspetti del software integrato, sviluppati negli ultimi 20 anni, che saranno l'oggetto di una trattazione successiva.



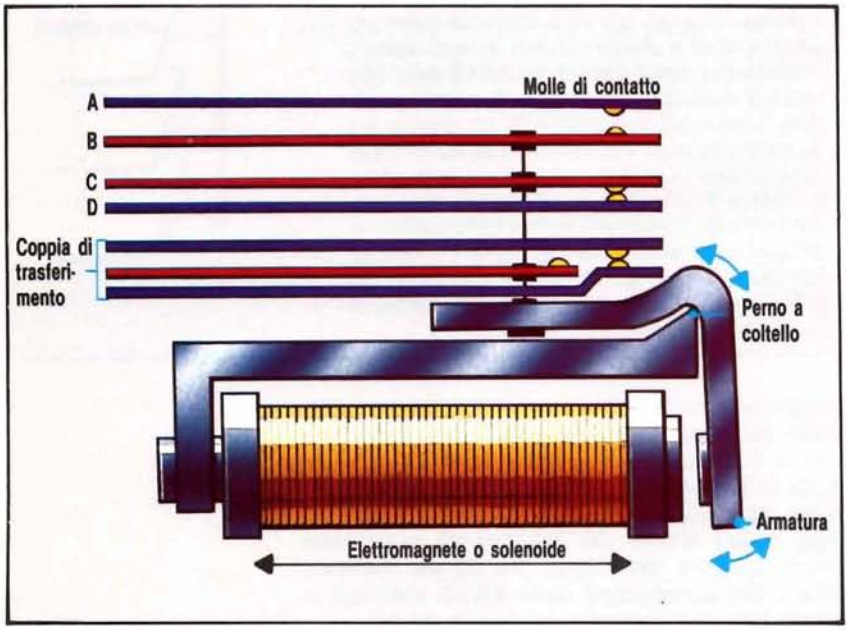
Un equo scambio

Il package "Xchange", della Psion, comprende quattro programmi suddivisi in altrettanti dischetti: un word processor, un database, uno spreadsheet e un generatore di grafici. Ogni modulo è affiancato da un programma di controllo globale (detto "supervisore"). L'elaborazione dei dati, creati da uno dei quattro programmi, viene considerata dal supervisore un "compito" (task) tra i 10 che esso può gestire simultaneamente. Uscendo da un compito, si può accedere ad un altro, tramite un menu. Il supervisore provvede anche a caricare in memoria l'appropriato programma applicativo. I comandi per scambiare dati tra le varie applicazioni sono IMPORT ed EXPORT, che creano tutti i legami occorrenti per l'accesso ai file di dati

Fonte d'energia

Accendere luci o apparecchi elettrici tramite il computer

I relais sono interruttori elettromagnetici azionati da un segnale elettrico. In questo progetto i relais vengono usati per accendere e spegnere apparecchi ad alta tensione (quella di rete), mediante segnali a bassa tensione (quelli emessi dal computer). Esistono più tipi di relais, ma i più comuni sono quelli "ad armatura", che usano un solenoide per aprire e chiudere circuiti.
L'apertura e la chiusura dei contatti viene provocata da piccoli spostamenti dell'armatura del relais. Una tensione di valore appropriato, applicata all'avvolgimento del solenoide, genera un campo magnetico che attira l'armatura.



Questo movimento provoca uno spostamento verso l'alto delle molle di contatto fissate all'altra estremità dell'armatura.
La disposizione che viene mostrata in figura è quella "di riposo", in cui il solenoide non riceve corrente: il contatto AB è aperto, mentre CD è chiuso. Quando al solenoide viene applicata tensione, le asticelle B e C vengono spinte verso l'alto, chiudendo così il contatto AB ed aprendo CD. Questo relais può avere due funzioni: chiudere un circuito e contemporaneamente aprirne un altro oppure, più semplicemente, aprire e chiudere un solo circuito.
Oltre a questo tipo di funzionamento, un relais può comportarsi come un meccanismo di trasferimento. Le tre asticelle inferiori sono disposte in modo che la prima sia in contatto con la terza se il relais è in posizione di riposo. Quando il solenoide riceve corrente, l'asticella centrale si sposta verso l'alto, entra in contatto con quella superiore ed interrompe il contatto tra quest'ultima e la molla inferiore.

ELENCO DEI COMPONENTI

Quantità	Descrizione
2	Relais a 8-15V, contatti a 220V da 10 Ampere
2	Prese di corrente
2	Scatolette di derivazione per circuiti elettrici
2	Metri di cavo a 3 conduttori interni da 6 Ampere
2	Spine di alimentazione
2	Fusibili (con portafusibile) da 5 Ampere
*	Spinotti da 4 mm.
*	0,5 m. di cavo a 2 linee
*	Piccole schede di veroboard

N.B. I componenti segnati con un asterisco dovrebbero essere avanzati da un progetto precedente (vedere a pag. 1004). Con questi componenti si possono costruire due scatole a relais a presa singola. Il box di uscita può controllarne fino a quattro

DENTRO LA SCATOLA

Controllare che tutte le connessioni siano ben salde ed affidabili. Assicurarsi che non ci siano contatti tra le linee ad alta tensione e quelle dei segnali di controllo.
Incollare la scheda del circuito in un angolo della scatoletta di derivazione, usando una colla epossidica. Non si adoperino colle generiche, perché possono essere conduttrici di corrente. (Se non si conoscono le proprietà di conduzione della colla, se ne spalmi un po' su di una striscia di carta, la si lasci seccare e si applichi un ohmmetro alle due estremità: se l'apparecchio segnala un passaggio di corrente, occorre cambiare colla!)
Quando la colla è asciutta, fissare il coperchio sulla scatola e fissare gli spinotti da 4 mm. sui cavetti per i segnali (questi possono essere anche dello stesso colore, perché il relais funziona indipendentemente dalla direzione della corrente). Collegare la spina al cavetto di alimentazione, interponendo un fusibile da 5 Ampere. Il relais consente un massimo carico di 10 A ma, per sicurezza, conviene non applicare correnti superiori a 5 A; inseriamo quindi nella spina un fusibile da 5 A. In questo modo è possibile controllare apparecchi da 1 kW

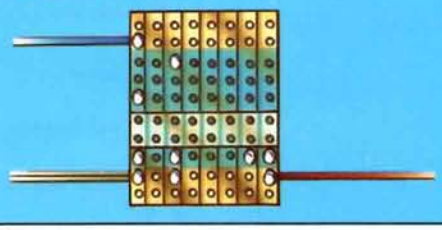
ATTENZIONE!

L'apparecchio è molto semplice, ma operando con la corrente di rete (220 V), occorre prestare la massima attenzione:

- **Staccare tutte le prese di corrente, prima di lavorare sui circuiti della scatola.**
- **Controllare con un multimetro tutti i contatti e gli isolamenti prima di dare corrente per la prima volta.**
- **Attenersi nel modo più assoluto allo schema del progetto. Si rammenti che LA CORRENTE PUÒ UCCIDERE!**

LA SCHEDA

Tagliare una lastrina di veroboard, che verrà poi inserita in un angolo del contenitore, uguale a quella in figura. Tagliare le piste e saldare il relais secondo lo schema mostrato in basso.
Controllare attentamente la scheda prima di procedere.
Usare un multimetro per verificare che non vi siano contatti tra le piste: un corto circuito potrebbe avere conseguenze mortali!
Saldare sulla scheda uno dei capi del cavetto di alimentazione e il cavo a due linee. Incidere una delle fessure presagomate della scatola di derivazione, per farci passare i cavetti. Fare un nodo ai fili prima di inserirli nella fessura per evitare che un involontario strattone danneggi la scheda. Saldare un breve cavetto isolato alla scheda e fissarlo al serrafilo di una delle due prese di corrente. Collegare i cavetti di alimentazione agli altri due serrafili, facendo attenzione a collegare a quello centrale il cavetto di massa (o di "terra")



PROGRAMMA DI CONTROLLO

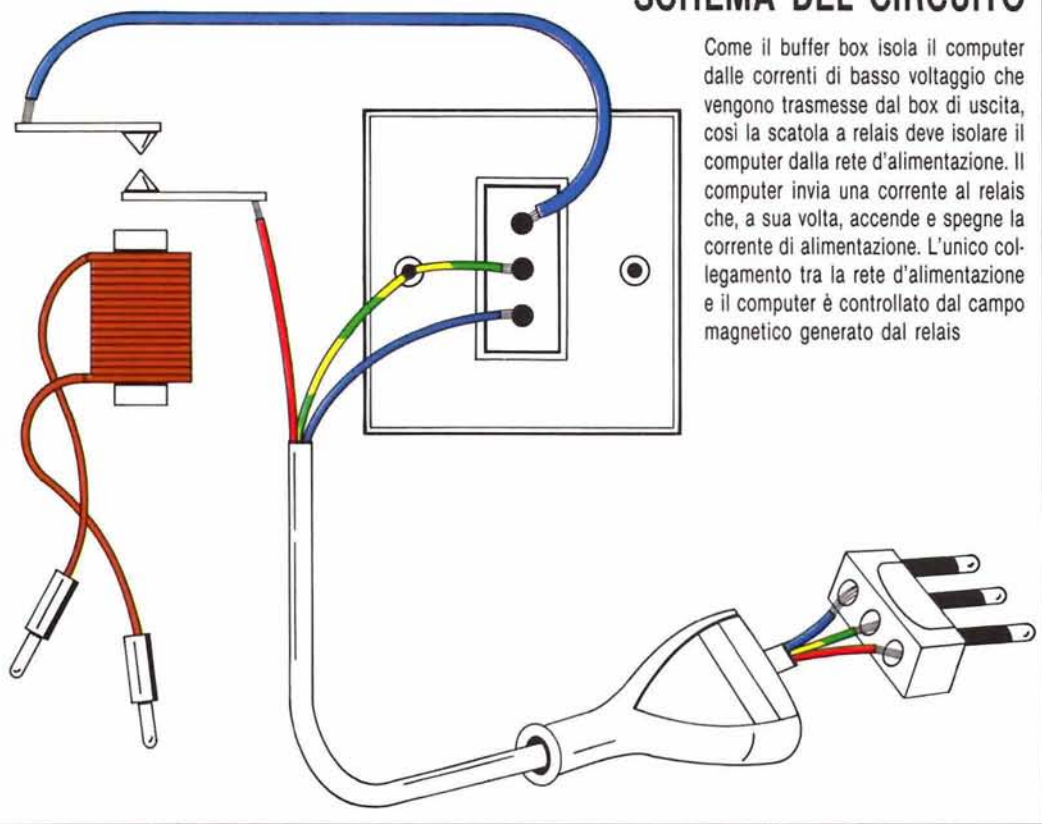
Dopo aver costruito la scatola a relais e controllato tutti i collegamenti, possiamo provarne il funzionamento con un breve programma che accenda e spenga un apparecchio elettrico, ad esempio una lampada da tavolo. La spina di questa viene collegata alla presa di alimentazione della scatola a relais, mentre gli spinotti delle linee di segnale vengono innestati sui poli positivo e negativo della linea 0 del box di uscita a basso voltaggio. Gli spinotti possono essere innestati indifferentemente all'uno o all'altro dei due poli, senza influire sul funzionamento del relais. Il cavetto di alimentazione proveniente dalla scatola a relais viene infine collegato ad una presa a muro.
Eseguite queste operazioni, scrivere il seguente programma che accende la lampada per 5 secondi e poi la spegne.

```
10 REM VERIFICA DEL RELAIS - BBC
20 DDR=&FE62:REGDAT=&FE60
30 ?DDR=255:REM TUTTE LINEE IN USCITA
40 ?REGDAT=1:REM ACCENDE LA LUCE
50 TIME=0:REM AVVIA IL TEMPORIZZATORE
60 REPEAT
70 UNTIL TIME500
80 ?REGDAT=0:REM SPEGNE LA LUCE
```

```
10 REM VERIFICA DEL RELAIS - CBM 64
20 DDR=56579:REGDAT=56577
30 POKE DDR,255:REM TUTTE LINEE IN USCITA
40 POKE REGDAT,1:REM ACCENDE LA LUCE
50 T=TI:REM LEGGE IL TEMPORIZZATORE
60 IF (TI—T)
70 POKE REGDAT,0:REM SPEGNE LA LUCE
```

Se, eseguito il programma, la lampada non si è accesa, staccare la scatola a relais e controllare tutti i collegamenti

SCHEMA DEL CIRCUITO



Come il buffer box isola il computer dalle correnti di basso voltaggio che vengono trasmesse dal box di uscita, così la scatola a relais deve isolare il computer dalla rete d'alimentazione. Il computer invia una corrente al relais che, a sua volta, accende e spegne la corrente di alimentazione. L'unico collegamento tra la rete d'alimentazione e il computer è controllato dal campo magnetico generato dal relais

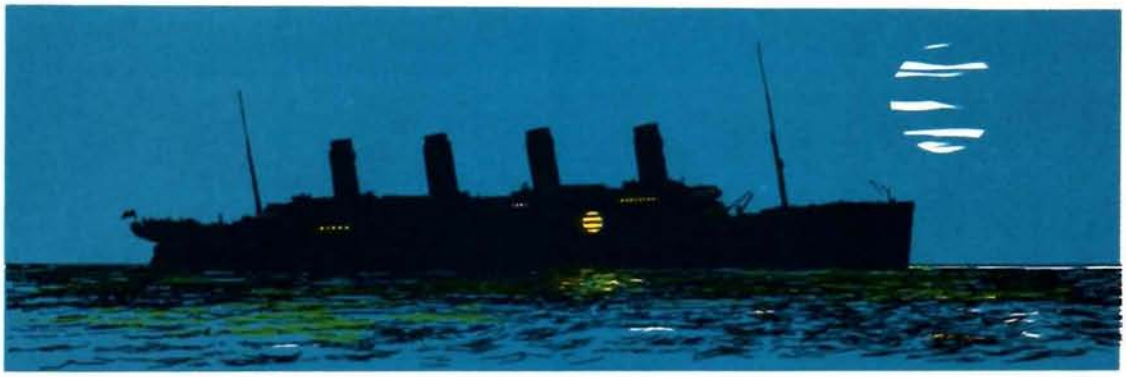
PROGETTO DI ALFABETO MORSE

```
10 REM*********************************
11 REM*       C64 ESERCIZIO MORSE      *
12 REM*  COLLEGARE UNA LAMPADA ALLA    *
13 REM*  SCATOLA A RELAIS: SCRIVERE    *
14 REM*  UNA STRINGA ALFANUMERICA.     *
15 REM* IL MESSAGGIO VIENE RIPRODOTTO *
16 REM* IN MORSE DALLA LAMPADA E DAI   *
17 REM*       "BEEP" DEL COMPUTER      *
20 REM*********************************
100 GOSUB 2000:  REM INIZIALIZZAZIONE
150 FOR L=0 TO 1 STEP 0
200 PRINT "IMMETTERE UN MESSAGGIO"
220 PRINT "DIGITARE 'CIAO' PER FINIRE"
240 INPUT "MESSAGGIO ";MS$
300 ML=LEN(MS$):M$=""
320 FOR K=1 TO ML
330 C$=MID$(MS$,K,1)
340 IF C$=>"A"ANDC$<="Z"THEN M$=M$+C$
350 IF C$=" " THEN M$=M$+C$
360 NEXT K:IF M$="" THEN NEXT L
400 ML=LEN(M$)
420 FOR K=1 TO ML
440 CH$=MID$(M$,K,1):CH=ASC(CH$)-64
450 IF CH=-32 THEN FORPP=1TO6*DE:NEXTPP
460 IF CH<>-32 THEN GOSUB 3000
480 FOR PP=1 TO 3*DE:NEXT PP
500 NEXT K
550 IF M$="CIAO" THEN L=1
600 NEXT L
900 END
2000 REM****** INIZIALIZZAZIONE ******
2100 DIM M$(26)
2110 DDR=56579:REGDAT=56577:POKE DDR,255
2120 DE=25:RX=2*DE
2130 V=54296:LF=54272:HF=54273:W=54276
2140 A=54277:S=54278
2150 FOR K=LF TO LF+24:POKEK,0:NEXT K
2160 POKEA,24:POKES,129:POKEV,15
2200 DATA ".-","-...","-.-.","-..","."
2220 DATA "..-.","--.","....",".."
2240 DATA ".---","-.-","-...","--"
2260 DATA "-.","---",".--.","--.-"
2280 DATA ".-.","...","-","..-","...-"
2300 DATA ".--","-..-","-.--","--.."
2400 FOR K=1 TO 26:READ M$(K):NEXT K
2990 RETURN
3000 REM****** LAMPEGGIO & BEEP ******
3050 PRINT CH$,M$(CH)
3100 N=LEN(M$(CH))
3200 FOR C=1 TO N
3220 D=DE-(ASC(MID$(M$(CH),C,1))-46)*RX
3240 POKE DATRG,1            : REM
     LAMPEGGIO
3250 POKE LF,172:POKE HF,57: REM BEEP
3260 POKE W,33:FOR PP=1 TO D:NEXT PP
3270 POKE W,32
3280 POKE LF,0:POKE HF,0   : REM NO BEEP
3290 POKE REGDAT,0           : REM NO
     LAMPEGGIO
3300 FOR PP=1 TO DE:NEXT PP
3320 NEXT C
3490 RETURN
```

```
10 REM*********************************
11 REM*       BBC ESERCIZIO MORSE      *
12 REM*  COLLEGARE UNA LAMPADA ALLA    *
13 REM*  SCATOLA A RELAIS: SCRIVERE    *
14 REM*  UNA STRINGA ALFANUMERICA.     *
15 REM* IL MESSAGGIO VIENE RIPRODOTTO *
16 REM* IN MORSE DALLA LAMPADA E DAI   *
17 REM*       "BEEP" DEL COMPUTER      *
20 REM*********************************
100 PROCinizializza
120 ALLOVER=FALSE
150 REPEAT
200 PRINT "IMMETTERE UN MESSAGGIO"
220 PRINT "DIGITARE 'CIAO' PER FINIRE"
240 INPUT "MESSAGGIO ",MS$
300 ML=LEN(MS$):M$=""
320 FOR K=1 TO ML
330 C$=MID$(MS$,K,1)
340 IF C$>="A"ANDC$<="Z"THEN M$=M$+C$
350 IF C$=" " THEN M$=M$+C$
360 NEXT K:IF M$="" THEN UNTIL FALSE
400 ML=LEN(M$)
420 FOR K=1 TO ML
440 CH$=MID$(M$,K,1):CH=ASC(CH$)-64
450 IF CH=-32 THEN PROCritardo(6*DE*IX)
460 IF CH<>-32 THEN PROCbipluce
480 PROCritardo(3*DE)
500 NEXT K
550 IF M$="CIAO" THEN FINITO=TRUE
600 UNTIL ALLOVER
900 END
2000 REM****** INIZIALIZZAZIONE ******
2050 DEF PROCinizializza
2100 DIM M$(26)
2110 DDR=&FE62:REGDAT-&FE60:?DDR=255
2120 DE=3:RX=2*DE:IX=30
2200 DATA ".-","-...","-.-.","-..","."
2220 DATA "..-.","--.","....",".."
2240 DATA ".---","-.-","-...","--"
2260 DATA "-.","---",".--.","--.-"
2280 DATA ".-.","...","-","..-","...-"
2300 DATA ".--","-..-","-.--","--.."
2400 FOR K=1 TO 26:READ M$(K):NEXT K
2990 ENDPROC
3000 REM****** BEEP&LAMPEGGIA *******
3020 DEF PROCbeep_luce
3050 PRINT CH$,M$(CH)
3100 N=LEN(M$(CH))
3200 FOR C=1 TO N
3220 D=DE-(ASC(MID$(M$(CH),C,1))-46)*RX
3240 ?REGDAT=1               : REM LAMPEGGIO
3260 SOUND 1, -15, 200, D    : REM BEEP
3270 PROCritardo(IX*D)
3290 ?REGDAT=0               : REM NO
     LAMPEGGIO
3300 PROCritardo(DE*IX)
3320 NEXT C
3490 ENDPROC
4000 REM********** RITARDO **********
4100 DEF PROCritardo(tempo)
4200 FOR DD=1 TO tempo: NEXT DD
4490 ENDPROC
```

ALFABETO MORSE

A	•–
B	–•••
C	–•–•
D	–••
E	•
F	••–•
G	––•
H	••••
I	••
J	•–––
K	–•–
L	•–••
M	––
N	–•
O	–––
P	•––•
Q	––•–
R	•–•
S	•••
T	–
U	••–
V	•••–
W	•––
X	–••–
Y	–•––
Z	––••
.	•–•–•–
,	––••––



DATABASE

Quarta scheda delle istruzioni per lo Z80 della Zilog. Istruzioni aritmetiche e logiche ad 8 bit

PROVENIENZA

	INDIRIZZAMENTO DEI REGISTRI							REG IND.	INDICIZZATO		IMM.
	A	B	C	D	E	H	L	(HL)	(IX+d)	(IY+d)	n
SOMMA, 'ADD'	87	80	81	82	83	84	85	86	DD 86 d	FD 86 d	C6 n
SOMMA CON CARRY, 'ADD'	8F	88	89	8A	8B	8C	8D	8E	DD 8E d	FD CE d	CE n
SOTTRAZIONE, 'SUB'	97	90	91	92	93	94	95	96	DD 96 d	FD 96 d	D6 n
SOTTR. CON CARRY, 'SBC'	9F	98	99	9A	9B	9C	9D	9E	DD 9E d	FD 9E d	DE n
AND	A7	A0	A1	A2	A3	A4	A5	A6	DD A6 d	FD A6 d	E6 n
XOR	AF	A8	A9	AA	AB	AC	AD	AE	DD AE d	FD AE d	EE n
OR	B7	B0	B1	B2	B3	B4	B5	B6	DD B6 d	FD B6 d	F6 n
CONFRONTO, 'CP'	BF	B8	B9	BA	BB	BC	BD	BE	DD BE d	FD BE d	FE n
INCREMENTO, 'INC'	3C	04	0C	14	1C	24	2C	34	DD 34 d	FD 34 d	
DECREMENTO, 'DEC'	3D	05	0D	15	1D	25	2D	35	DD 35 d	FD 35 d	

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
ADD A, r	A←A+r	↕	↕	X	V	0	↕	10 [000] r		1	1	4	r Reg. 000 B 001 C 010 D 011 E 100 H 101 L 111 A
ADD A, n	A←A+n	↕	↕	X	V	0	↕	11 [000] 110 ←n→		2	2	7	
ADD A,(HL)	A←A+(HL)	↕	↕	X	V	0	↕	10 [000] 110		1	2	7	
ADD A,(IX+d)	A←A+(IX+d)	↕	↕	X	V	0	↕	11 011 101 10 [000] 110 ←d→	DD	3	5	19	
ADD A,(IY+d)	A←A+(IY+d)	↕	↕	X	V	0	↕	11 111 101 10 [000] 110 ←d→	FD	3	5	19	
ADC A, s	A←A+s+CY	↕	↕	X	V	0	↕	[001]					s corrisponde ad una qualsiasi delle combinazioni r oppure n, oppure a: (HL), (IX+d), (IY+d) come illustrato per le istruzioni di somma (ADD). Il gruppo di bit indicato sostituisce quello [000] delle istruzioni ADD.
SUB s	A←A−s	↕	↕	X	V	1	↕	[010]					
SBC A, s	A←A−s−CY	↕	↕	X	V	1	↕	[011]					
AND s	A←A s	↕	↕	X	P	0	0	[100]					
OR s	A←A s	↕	↕	X	P	0	0	[110]					
XOR s	A←A⊕s	↕	↕	X	P	0	0	[101]					
CP s	A−s	↕	↕	X	V	1	↕	[111]					
INC r	r←r+1	↕	↕	X	V	0	•	00 r [100]		1	1	4	
INC (HL)	(HL)←(HL)+1	↕	↕	X	V	0	•	00 110 [100]		1	3	11	
INC (IX+d)	(IX+d)← (IX+d)+1	↕	↕	X	V	0	•	11 011 101 00 110 [100] ←d→	DD	3	6	23	
INC (IY+d)	(IY+d)← (IY+d)+1	↕	↕	X	V	0	•	11 111 101 00 110 [100] ←d→	FD	3	6	23	m corrisponde ad una qualsiasi delle combinazioni r, oppure a: (HL), (IX+d), (IY+d), come mostrato per INC. DEC ha lo stesso formato di INC, ma al gruppo [100] va sostituito [101].
DEC m	m←m−1	↕	↕	X	V	1	•	[101]					

NOTE:

La lettera V, nella colonna del flag P/V, indica che detto flag contiene l'eventuale Overflow risultante dall'operazione. Analogamente, la lettera P si riferisce alla Parità

V = 1 indica Overflow.
V = 0 nessun Overflow.
P = 1 indica Parità pari.
P = 0 indica Parità dispari.

Notazione per i flag:

- • nessun effetto
- 0 flag azzerato
- 1 flag posto a 1
- X effetto non definito
- ↕ flag rispecchia il risultato operazione

per concessione della Zilog Inc.

Un trio da viaggio

Il Tandy Model 100 a confronto con i suoi "gemelli"

Una prassi ormai consolidata, tra i fabbricanti di apparecchiature elettroniche, consiste nell'acquistare un prodotto già "finito", apportarvi qualche modifica di lieve entità (per personalizzarlo) e poi commercializzarlo col proprio marchio di fabbrica. Tipica del mercato dei televisori e delle radio, questa tendenza si è affermata anche nel settore dei computer.

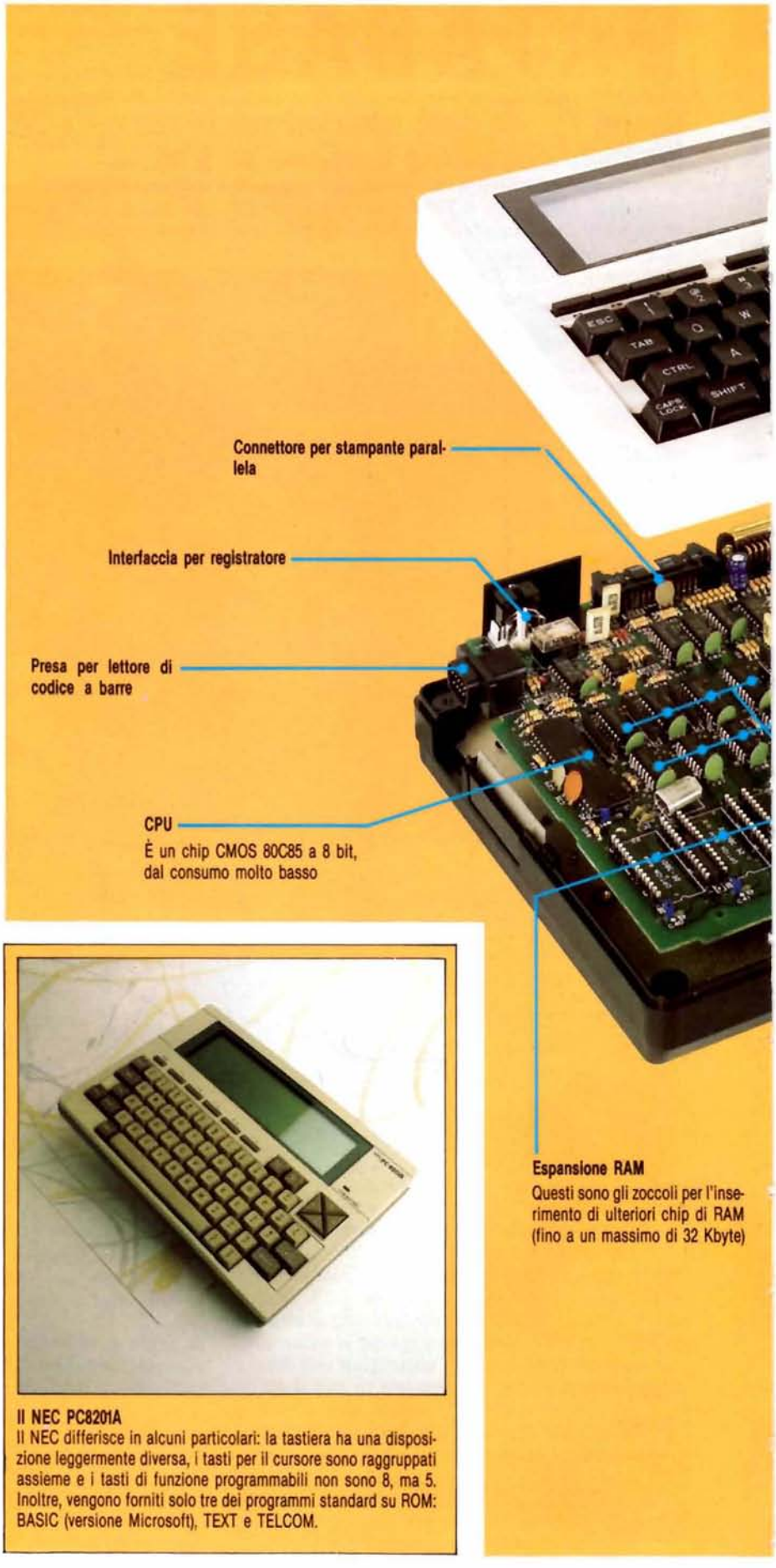
Il Tandy Model 100, l'Olivetti M10 e il NEC PC8201A sono il risultato di una simile operazione di "cosmesi": tutte e tre le macchine sono costruite da una ditta giapponese, la Kyocera, e poi vendute alla Tandy, alla Olivetti e alla NEC, che racchiudono gli apparecchi in contenitori leggermente diversi e li vendono con il proprio nome. Qui vengono prese in esame le caratteristiche del Tandy Model 100, confrontandole con quelle degli altri due gemelli.

Pesando poco meno di 2 kg., i tre computer appartengono alla categoria dei computer "da viaggio" (si usano comodamente sulle ginocchia). Il Model 100 possiede una tastiera QWERTY completa, diversi programmi applicativi su ROM e uno schermo a LCD (cristalli liquidi). Il computer può anche essere alimentato a pile e, grazie speciali pile "in tampone", il contenuto della RAM non si perde spegnendo l'apparecchio. I file possono essere quindi mantenuti in RAM, e l'accesso ad essi è immediato e non comporta alcuna difficoltà.

Il Model 100 prevede anche il collegamento ad un registratore o ad una unità a dischetti, ma per lavorare "in movimento", basta in genere la memoria interna, di tipo non volatile.

Il display visualizza 8 righe di 40 caratteri, con possibilità di uso combinato di testo e grafica. Il numero complessivo di punti, sul display, è 15.360, ciascuno dei quali è indirizzabile individualmente. I caratteri, maiuscoli e minuscoli, sono formati in una matrice di 6 × 8 punti.

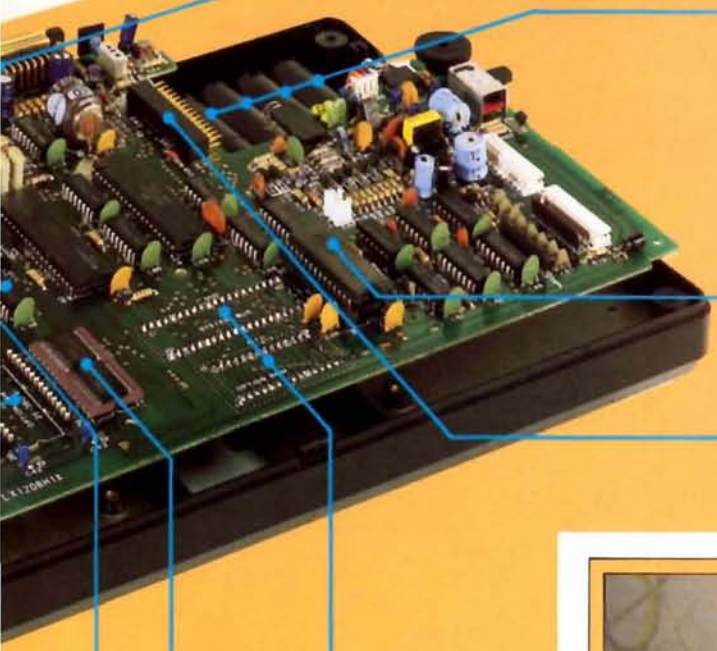
Il Model 100 possiede un set completo di caratteri internazionali ed uno di simboli grafici (mentre nel NEC esistono solo tre simboli grafici). Sia nel Model 100 che nel NEC, il display a cristalli liquidi è fisso, mentre invece nell'Olivetti può essere inclinato a piacimento, facilitando la lettura. Tutti



Il NEC PC8201A
Il NEC differisce in alcuni particolari: la tastiera ha una disposizione leggermente diversa, i tasti per il cursore sono raggruppati assieme e i tasti di funzione programmabili non sono 8, ma 5. Inoltre, vengono forniti solo tre dei programmi standard su ROM: BASIC (versione Microsoft), TEXT e TELCOM.



Connettore per modem
Questa è una normale porta seriale RS232. Il programma di telecomunicazione, incorporato, consente il collegamento ad altri computer



Alimentazione
Con quattro pile alcaline, formato AA, si ha un'autonomia di circa 20 ore di uso continuo. La memoria interna è alimentata da apposite pile ricaricabili al nichel-cadmio, automaticamente poste sotto carica durante il funzionamento

ROM
Questo chip contiene i programmi di corredo

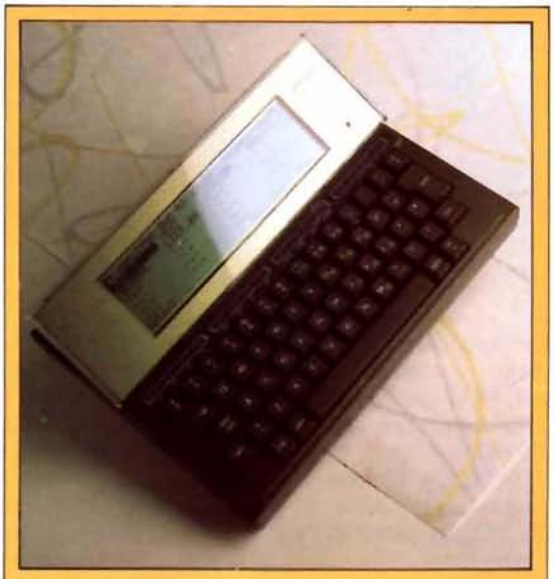
Connettore del display
Un cavo piatto collega il visualizzatore al computer

Bus di sistema e zoccoli ROM
Questi zoccoli servono per ampliare la capacità ROM dell'apparecchio

RAM di corredo, da 8 Kbyte

Unità di controllo per tastiera
Questi chip controllano le funzioni di ingresso/uscita della tastiera, oltre a contenere la definizione del set di caratteri

per concessione della CL PROJECTS



Viva la differenza!
Benché commercializzati da tre case diverse, il Tandy Model 100, l'Olivetti M10 e il NEC PC8201A sono tutti fabbricati dalla giapponese Kyocera. Del resto, anche la forte rassomiglianza esteriore fa intuire la loro origine comune. Le differenze, specialmente tra il Model 100 e l'M10, sono minime

IAN McKINNELL

TANDY MODEL 100

DIMENSIONI

mm. 300 x 215 x 50 (tastiera)

CPU

80C85 (chip CMOS a 8 bit). Frequenza di lavoro: 2,4 MHz

MEMORIA

8 o 24 Kbyte di RAM (espandibile fino a 32 Kbyte); 32 Kbyte di ROM, contenenti il BASIC Microsoft e altri programmi applicativi

VIDEO

Display LCD (a cristalli liquidi) con 8 righe di 40 caratteri. 240 x 64 punti individualmente indirizzabili. Set di caratteri ASCII internazionale e 39 simboli grafici

INTERFACCE

Per stampante parallela, per registratore. Porta seriale RS232, connettore per lettore di codici a barre, bus di sistema

LINGUAGGI FORNITI

BASIC versione Microsoft

TASTIERA

Tipo QWERTY, con 56 tasti. Opzione per tastierino numerico incorporato. 8 tasti di funzione programmabili, 4 per funzioni interne e 4 per il cursore

DOCUMENTAZIONE

Vengono forniti due pubblicazioni: una guida rapida al BASIC e un denso manuale d'uso (circa 200 pagine)

PRO

Le prestazioni sono paragonabili ad apparecchi di dimensioni molto maggiori, pur essendo un vero portatile in tutti i sensi. Il vantaggio di possedere una RAM non volatile è enorme

CONTRO

La quantità massima di memoria disponibile, anche dopo le espansioni, non consente applicazioni molto complesse. Il display a 8 righe può essere scomodo nella redazione di testi

DIFFERENZE

Olivetti M10: tastiera con 57 tasti, 47 simboli grafici, display orientabile. Un manuale ed una guida sintetica
NEC PC8201A: tastiera con 57 tasti, tasti cursore raggruppati a parte, 5 tasti di funzione programmabili. La RAM da 16 Kbyte può essere espansa fino a 96 Kbyte. Tre soli simboli grafici

possiedono un comando per regolare il contrasto, per adattarlo alle condizioni luminose ambientali.
Le tastiere del Model 100 e dell'M10 sono dotate di due tasti speciali: uno per accedere ai simboli grafici, l'altro per utilizzare parte della tastiera come "pad" numerico (per immettere cifre come su di una calcolatrice): il tasto M diventa lo 0; J, K, e L, rispettivamente, 1, 2 e 3; U, I e O diventano 4, 5 e 6, mentre i tasti 7, 8 e 9 mantengono la loro funzione originale.
Tutte e tre le macchine possiedono tasti per il controllo del cursore, ma la loro posizione varia: quattro tasti affiancati nel Model 100 e nell'M10, uno speciale pad di forma quadrata sul NEC.
Esistono anche tasti di funzione programmabili, che vengono sfruttati dal software residente, ma utilizzabili anche tramite i propri programmi.
Anche in questo caso, esistono differenze: il Model 100 possiede, come l'Olivetti, 8 tasti di funzione, più 4 riservati a funzioni speciali: PASTE (per spostare blocchi di dati), LABEL (per visualizzare i comandi abbinati ai tasti di funzione), PRINT (per attivare la stampante) e BREAK (per interrompere l'esecuzione di un programma).
Il NEC possiede invece soltanto 5 tasti di funzione programmabili, per un totale di 10 funzioni, più un tasto di "pausa".



IAN McKINNELL

Olivetti M10
Questo computer è identico al Tandy Model 100, eccezion fatta per un particolare degno di nota: il display a cristalli liquidi è orientabile e ciò ne facilita enormemente la lettura. La configurazione della tastiera ed i programmi forniti su ROM sono identici a quelli del Model 100

Capacità della memoria

Il Tandy Model 100 e l'Olivetti M10 vengono venduti sia nella versione a 8 Kbyte che in quella a 24 Kbyte (entrambe espandibili fino a 32 Kbyte con moduli aggiuntivi). Il NEC è leggermente diverso dagli altri: venduto nella versione di base, con 16 Kbyte, si può ampliarne internamente la RAM fino a 64 Kbyte oppure, ricorrendo a unità esterne, fino a 96 Kbyte.
Tutti e tre i modelli sono corredati da un BASIC versione Microsoft e da alcuni programmi applicativi: TEXT, TELCOM e (solo su Tandy e Olivetti) ADDRESS e SCHEDULE. TEXT è un word processor molto elementare, dotato di alcune funzioni (ricerca, spostamento o cancellazione di blocchi, ecc.). TELCOM è un programma di telecomunicazioni, che consente lo scambio di dati con altri computer, (direttamente o tramite modem) mediante la porta seriale RS232. SCHEDULE è un programma per database, particolarmente orientato alla gestione di appuntamenti, scadenze "cose da fare" e via dicendo.
Una funzione interna di ricerca facilita l'estrazione selezionata delle informazioni memorizzate. ADDRESS è un programma per gestire un indirizzario.
All'accensione, tutti e tre gli apparecchi visualizzano l'elenco dei file memorizzati.
Le interfacce non mancano: oltre alla citata RS232, i tre computer possiedono una porta parallela per stampante, l'interfaccia per registratore e una presa per lettore di codici a barre. Sia nel Tandy che nell'Olivetti è accessibile il bus di sistema, mentre nel NEC sono previste due ulteriori porte seriali.
Considerata la facilità d'uso (sebbene i display ad 8 righe siano una forte limitazione) e la capacità di comunicare via modem con altri apparecchi, questi tre computer da viaggio sono allettanti per chi si trovi a dover lavorare durante frequenti spostamenti.

Un altro giro in moto

Ecco le versioni del gioco per Commodore 64 e per il Micro BBC

A differenza dei dialetti BASIC dello Spectrum e del BBC, il BASIC del Commodore 64 non possiede comandi (tipo PLOT) per accendere individualmente i pixel. Nella versione Commodore del gioco *Gara di Moto*, per tracciare le linee luminose lasciate dalle moto, si adoperano caratteri in bassa risoluzione. È usato il carattere "spazio" inverso, visualizzato sullo schermo depositando con POKE il suo codice (160) nella mappa della memoria video. Il codice di colore è depositato nella corrispondente locazione della mappa di colore. Come sullo Spectrum, la versione Commodore non è strutturata: ciò al fine di ottenere una maggior velocità d'esecuzione.

Tuttavia, nelle fasi di gioco in cui la velocità ha importanza secondaria (ad esempio dopo le collisioni) è stata introdotta una certa strutturazione nelle chiamate di subroutine per incrementare il punteggio e far lampeggiare lo schermo.

La versione per il BBC ha invece una struttura ben più chiara: ciò dipende dalla maggior velocità del BASIC BBC, rispetto a quello dello Spectrum o del Commodore, e dalla possibilità di eseguire chiamate alle procedure. La maggior parte dei dialetti BASIC consente un certo grado di strutturazione ricorrendo alle subroutine, ma questa tecnica rallenta la velocità d'esecuzione, in quanto dopo ogni chiamata di subroutine il computer esegue una ricerca. Al contrario, quando il BASIC BBC chiama una procedura per la prima volta, ne registra l'indirizzo in una tabella di riferimento.

COMMODORE 64

```
10 REM "GARA DI MOTO" - CBM 64
15 POKE53281,0:POKE53280,4:REM SCR/BORD COLOUR
20 SC=1024:CO=55296
25 PRINT CHR$(147);: REM CANCELLA LO SCHERMO
30 X1=2:Y1=12:X2=37:Y2=12
35 DX=1:DM=-1:DN=0:DY=0
40 PRINT CHR$(19) CHR$(158) "GIOCATORE 1: "S1
50 PRINT CHR$(19) TAB (27) "GIOCATORE 2: "S2
52 FORI=1TO8:PRINTCHR$(17);:NEXT
54 PRINT TAB(14) "PREMI UN TASTO"
55 GETJ$:IFJ$<>""THEN55
56 GETA$:IFA$=22THEN56
57 PRINTTAB(14)CHR$(145)"
60 REM CICLO PRINCIPALE
70 GETA$
80 IFA$="W"THENDY=-1:DX=0
90 IFA$="X"THENDY=1:DX=0
100 IFA$="A"THENDX=-1:DY=0
110 IFA$="D"THENDX=1:DY=0
120 IFA$="O"THENDN=-1:DM=0
130 IFA$="M"THENDN=1:DM=0
140 IFA$="J"THENDM=-1:DN=0
150 IFA$="L"THENDM=1:DN=0
155 Y1=Y1+DY
156 IFY1<1ORY1>24THENF=0:GOSUB1000:GOTO25
157 X1=X1+DX
158 IFX1<0ORX1>39THENF=0:GOSUB1000:GOTO25
160 Y2=Y2+DN
162 IFY2<1ORY2>24THENF=1:GOSUB1000:GOTO25
164 X2=X2+DM
166 IFX2<0ORX2>39THENF=1:GOSUB1000:GOTO25
167 P1=X1+40*Y1
168 P2=X2+40*Y2
170 IFPEEK(SC+P1)=160THENF=0:GOSUB1000:GOTO25
180 IFPEEK(SC+P2)=160THENF=1:GOSUB1000:GOTO25
190 POKE SC+P1,160
200 POKE CO+P1,14
210 POKE SC+P2,160
220 POKE CO+P2,5
230 GOTO 70: REM RILANCIA IL CICLO PRINCIPALE
240 :
1000 REM S/R PUNTEGGIO
1020 IF F=1THENS1=S1+1:GOSUB2000:RETURN
1030 S2=S2+1:GOSUB2000:RETURN
1999 :
2000 REM S/R LAMPEGGIAMENTO SCHERMO
2010 FORJ=1TO10
2020 FORI=0TO15
2030 POKE53281,I
2040 NEXTI,J
2045 POKE53281,0
2050 RETURN
```

BBC MICRO

```
10 REM BBC
20 MODE1: moto1=0: moto2=0
30 PROCinizializza
40 PROCtasto
50 PROCbordo
60 PROCpunteggio
70 PROCtraccia_pixel
80 PROCtastiera
90 PROCverifica_punto
100 GOTO70
110 END
120 DEF PROCbordo
130 GCOL0,bordo
140 MOVE0,0
150 DRAW1279,0
160 DRAW1279,980
170 DRAW0,980
180 DRAW0,0
190 ENDPROC
200 DEF PROCtastiera
210 REM GIOCATORE UNO
220 IF INKEY(-51)=-1THEN dx=4:dy=0
230 IF INKEY(-87)=-1THEN dm=4:dn=0
240 IF INKEY(-66)=-1THEN dx=-4:dy=0
250 IF INKEY(-70)=-1THEN dx=0:dy=-4
260 IF INKEY(-67)=-1THEN dx=0:dy=4
270 IF INKEY(-34)=-1THEN dm=0:dn=4
280 IF INKEY(-55)=-1THEN dm=0:dn=-4
290 IF INKEY(-102)=-1THEN dm=0:dn=-4
300 x=x+dx:y=y+dy:m=m+dm:n=n+dn
310 ENDPROC
320 DEF PROCinizializza
330 x=100:y=490:m=1179:n=490
340 dx=4:dy=0:dm=-4:dn=0
350 col1=1:col2=2:bordo=3
360 ENDPROC
370 DEF PROCverifica_punto
380 pixel1=POINT(x,y)
390 pixel2=POINT(m,n)
400 IF pixel1 THEN PROCesplosione(x,y,1)
410 IF pixel2 THEN PROCesplosione(m,n,2)
420 ENDPROC
430 DEF PROCtraccia_pixel
440 GCOL0,col1:PLOT69,x,y
450 GCOL0,col2:PLOT69,m,n
460 ENDPROC
470 DEF PROCesplosione(ex,ey,chi)
480 FOR1=1TO100
490 MOVEex,ey
500 GCOL0,RND(3)
510 PLOT1,RND(50)-25,RND(50)-25
520 NEXT
530 IF chi=2 THEN moto1=moto1+1
540 IF chi=1 THEN moto2=moto2+1
550 PROCinizializza
560 CLS
570 PROCtasto
580 PROCbordo
590 PROCpunteggio
600 ENDPROC
610 DEF PROCpunteggio
620 PRINT TAB(1,0) "Moto Uno=";moto1
630 PRINT TAB(28,0) "Moto Due=";moto2
640 ENDPROC
650 DEF PROCtasto
660 PRINT TAB(8,12) "PREMI UN TASTO QUALUNQUE
    PER PARTIRE"
670 *FX21
680 A$=GET$
690 PRINTTAB(8,12)"
700 ENDPROC
```

Una tartaruga spaziale

Sviluppo di un gioco in cui la tartaruga si perde nello spazio profondo

Nel gioco qui presentato, la tartaruga si trova in difficoltà nelle profondità dello spazio, molto lontano dalla "base" alla quale deve tornare. Il gioco richiede l'emissione su schermo di alcuni messaggi: ovviamente il comando da usare è PRINT. Dopo la stampa di un messaggio, il cursore si sposta all'inizio della linea successiva. Il comando PRINT, seguito dal segno " e da una sola parola, visualizza tale parola: PRINT "SALVE, quindi, visualizza la parola SALVE. La forma PRINT" viene invece usata per visualizzare una "parola nulla", ossia una parola che non contiene alcun carattere: come risultato, si ha una linea vuota (ossia un "a capo"). Volendo visualizzare un'intera frase, occorre racchiudere la scritta tra parentesi quadre per indicare che essa forma una "lista":

```
PRINT [IL TEMPO A TUA DISPOSIZIONE È TERMINATO]
```

PRINT viene usato anche per visualizzare il contenuto di una variabile, come ad esempio in: PRINT :PUNTEGGIO. Un'istruzione PRINT può anche contenere simultaneamente messaggi e valori di variabili, ma in tal caso l'intera istruzione deve essere racchiusa tra parentesi tonde:

```
(PRINT [IL TUO PUNTEGGIO È]:PUNTEGGIO)
```

PRINT1 si comporta esattamente come PRINT, ma il cursore si ferma alla fine di quanto visualizzato, senza passare alla riga successiva. PRINT1 viene comunemente usato in istruzioni del tipo:

```
PRINT1 [COME TI CHIAMI?]
```

Operazioni di output

I comandi LOGO come HIDETURTLE o PRINT producono sempre qualche risultato, ossia interagiscono in qualche modo con la tartaruga. Tuttavia, altre "primitive" LOGO, ad esempio XCOR, non hanno effetti visibili bensì inviano un valore in uscita (output) che viene poi usato come valore in ingresso (input) per un comando. Ad esempio, nell'istruzione:

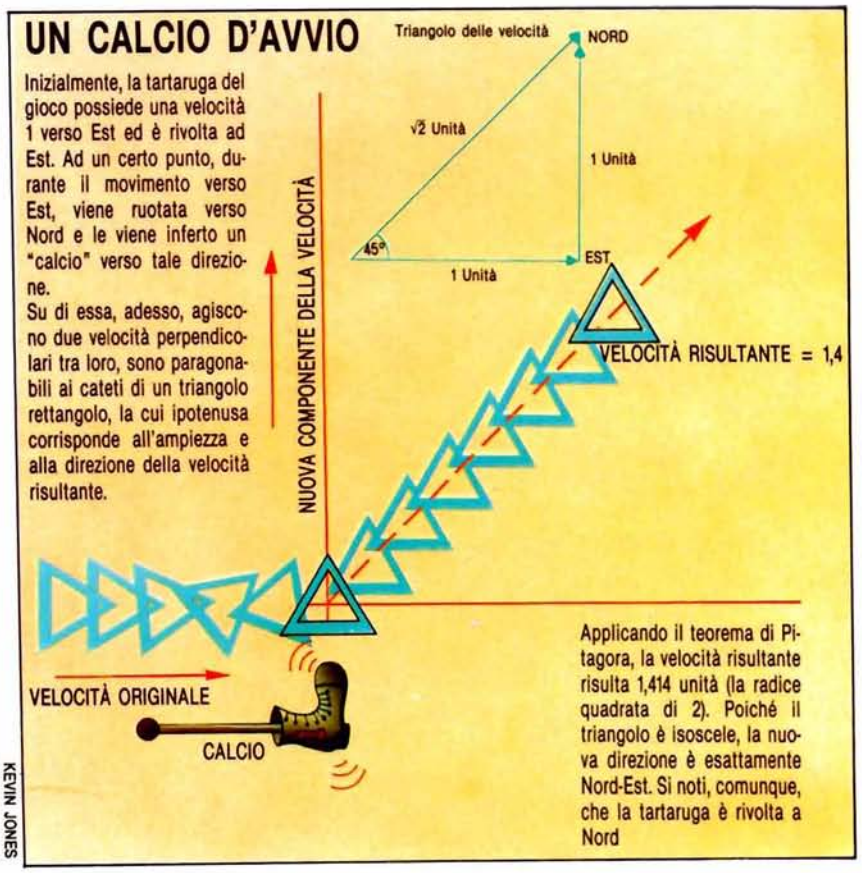
```
PRINT XCOR
```

XCOR trasmette il valore corrente della coordinata x della tartaruga a PRINT che lo visualizza. Quindi se il valore corrente di XCOR è 20, l'istruzione PRINT XCOR visualizza il numero 20. Se XCOR viene battuto da solo, sullo schermo appare il messaggio RESULT:20 che, in realtà, è un messaggio d'errore (le versioni della LCSI sono un po' meno "educate", e visualizzano un messaggio del tipo «NON HAI SPECIFICATO CHE FARE COL VALORE 20»).
Tutte le procedure fin qui sviluppate corrispondono a comandi: per creare delle "operazioni", si ricorre alla primitiva OUTPUT del LOGO. Ecco, ad esempio, una semplice procedura che emette la distanza della tartaruga dall'origine, in cui è adoperata SQRT, per calcolare la radice quadrata di un numero:

```
TO DISTANZA
   OUTPUT SQRT (XCOR*XCOR + YCOR*YCOR)
END
```

Si provi a muovere la tartaruga per lo schermo, usando poi DISTANZA per valutare quanto disti dall'origine. Per esempio, con SETXY 30 40 PRINT DISTANCE, la risposta dovrebbe essere 50. Quando esegue una OUTPUT, il LOGO interrompe l'esecuzione di una procedura, per tornare a quella chiamante. Ciò si può meglio apprezzare in una procedura MAX, che emette il maggiore tra due numeri:

```
TO MAX:X:Y
   IF :X > :Y THEN OUTPUT :X
   OUTPUT :Y
END
```



Cosicché PRINT MAX 6 2 dà come risultato 6. Si provi a scrivere una procedura che indichi il valore assoluto di un numero, in modo che impartendo sia PRINT ABS 4 sia PRINT ABS (—4) si abbia come risultato 4.
Il gioco prevede l'immissione di un nome, seguita da un Return. Ecco la procedura:

```
TO ACC.NOME
   SPLITSCREEN
   PRINT1 [COME TI CHIAMI?]
   MAKE "NOME FIRST REQUEST
   (PRINT "CIAO :NOME)
END
```

REQUEST attende l'immissione di una riga di testo, terminata da un Return, poi visualizza la riga sotto forma di lista. FIRST seleziona il primo elemento della lista. Si provi ad usare ACC.NOME, immettendo come nome Enrico. Poi si riprovi immettendo Enrico Fermi.
I tasti di controllo, nel gioco, sono: D, S e C. D e S fanno ruotare la tartaruga, rispettivamente, a destra o a sinistra di 30 gradi. C è usato per dare un "calcio" alla tartaruga, facendole aumentare la velocità nella direzione verso la quale è rivolta. Ovviamente, la tartaruga può muoversi sullo schermo e reagire immediatamente alla pressione di detti tasti. Una primitiva LEGGITASTO, che producesse il codice del tasto premuto, tornerebbe molto utile e potremmo scrivere:

```
TO COMANDI
   MAKE "COM LEGGITASTO
   IF :COM = "D THEN RIGHT 30
   IF :COM = "S THEN LEFT 30
   IF :COM = "C THEN CALCIO
END
```

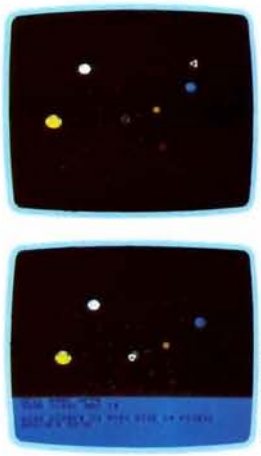
Disgraziatamente, tale primitiva non esiste! Si può tuttavia creare una procedura:

```
TO LEGGITASTO
   IF RC? THEN OUTPUT READCHARACTER
   OUTPUT "
END
```

Il codice del tasto premuto viene depositato nel buffer della tastiera: READCHARACTER emette semplicemente l'ultimo carattere del buffer. Se questo è vuoto, il comando attende la pressione di un tasto. RC? è vera se il buffer contiene almeno un carattere, altrimenti è falsa. LEGGITASTO, quindi, emette il carattere premuto oppure una parola nulla, se il buffer è vuoto.

La tartaruga dinamica

La tartaruga del gioco non possiede soltanto una posizione e una direzione, ma anche una velocità (come del resto le tartarughe vere!). Tuttavia, poiché si muove nello spazio, la tartaruga non è soggetta ad attrito né alla forza di gravità, pur rispettando le leggi di Newton sul moto. Supponiamo che la tartaruga si stia muovendo da sinistra a destra sullo schermo con una velocità 1. Se si preme il tasto S, la tartaruga si rivolge verso la parte superiore dello schermo, ma l'inerzia la fa muovere ancora lungo l'asse orizzontale. Premendo C, la tartaruga riceve una spinta nella direzione verso la quale è rivolta. Come risultato si ha che la tartaruga si muove verso l'alto con velocità 1 e in diagonale con velocità 1,4. Con questo semplice gioco si possono sperimentare e comprendere intuitivamente le leggi di Newton sul moto dei corpi, senza ricorrere a complesse formule.
Nel programma, la velocità della tartaruga è considerata come formata da due componenti: una orizzontale (asse x) e una verticale (asse y). Entrambe vengono calcolate tramite le funzioni trigonometriche SIN e COS. I comandi sono soltanto i tre menzionati e per iniziare il gioco basta digitare START. C'è un limite massimo al tempo impiegato per raggiungere la base e viene tenuto un conto del punteggio massimo.



IAN McKINNELL

Tartaruga extraterrestre

Il programma presentato visualizza semplicemente la tartaruga e la base spaziale (l'obiettivo). I pianeti e le stelle qui raffigurate sono state aggiunte usando semplici procedure per tracciare cerchi

LOGO & LOGO

LOGO del MIT	LOGO della LCSI
DRAW	CS
PRINT1	TYPE
RC?	KEYP
READCHARACTER	RC
REQUEST	RL
SETHEADING	SETH
SETXY	SETPOS (seguito da una lista)
FULLSCREEN	FS (non disponibile su Spectrum)
SPLITSCREEN	SS (non disponibile su Spectrum)

La IF ha un formato leggermente diverso nel LOGO della LCSI, ad esempio:

IF DISTANZA <5 [DONE STOP]

PROPOSTA DI SVILUPPO

Si scriva un programma per giocare all'atterraggio lunare. In breve, ecco gli scopi del gioco:
Si è alla guida di un veicolo spaziale, che si trova ad una certa distanza dalla superficie lunare, con una quantità di carburante limitata. La gravità esercita una costante attrazione verso il suolo lunare, facendo aumentare costantemente, ad ogni secondo trascorso, la velocità verso il basso del veicolo. La pressione del tasto G aziona i getti direzionali che bilanciano la velocità di caduta, ma ciò comporta un notevole consumo di carburante. Scopo ultimo del gioco è quello di adoperare abilmente il tasto F, per assicurarsi un atterraggio "morbido".

ABBREVIAZIONI

OUTPUT	OP
PRINT	PR
READCHARACTER	RC
REQUEST	RQ
SETHEADING	SETH



RISPOSTE AGLI ESERCIZI

1. Triangoli nidificati

```
TO TRI :MISURA :LIVELLO
   IF: LIVELLO =0 THEN REPEAT 3
   [FD :MISURA RT 120]
   STOP
   TRI (:MISURA/2)(:LIVELLO —1)
   FD (:MISURA/2)
   TRI (:MISURA/2)(:LIVELLO —1)
   RT 60
   TRI (:MISURA/2)(:LIVELLO —1)
   FD (:MISURA/2)
   RT 60
   TRI (:MISURA/2)(:LIVELLO —1)
   LT 60
   BK (:MISURA/2)
   LT 60
   BK (:MISURA/2)
END
```

2. Fiocco di neve quadrato

```
TO NEVE1 :MISURA :LIVELLO
   REPEAT 4 [LATO1 :MISURA :LIVELLO RT 90]
END
TO LATO1 :MISURA :LIVELLO
   IF :LIVELLO = 0 THEN FD :MISURA STOP
   LATO1 (:MISURA/3)(:LIVELLO —1)
   LT 90
   LATO1 (:MISURA/3)(:LIVELLO —1)
   RT 90
   LATO1 (:MISURA/3)(:LIVELLO —1)
   RT 90
   LATO1 (:MISURA/3)(:LIVELLO —1)
   LT 90
   LATO1 (:MISURA/3)(:LIVELLO —1)
END
```

3. Curva senza gradienti

```
TO W :PASSOX :PASSOY :LIVELLO
   WSU :PASSOX :PASSOY :LIVELLO
   WGIU :PASSOX :PASSOY :LIVELLO
END

TO WSU :PASSOX :PASSOY :LIVELLO
   IF: LIVELLO = 0 THEN SETXY (XCOR +:PASSOX)
   (YCOR +:PASSOY) STOP
   WSU (:PASSOX/6)(:PASSOY/2)(:LIVELLO—1)
   WGIU (:PASSOX/6)(:PASSOY/2)(:LIVELLO —1)
   WSU (:PASSOX/6)(:PASSOY/2)(:LIVELLO —1)
   WSU (:PASSOX/6)(:PASSOY/2)(:LIVELLO —1)
   WGIU (:PASSOX/6)(:PASSOY/2)(:LIVELLO —1)
   WSU (:PASSOX/6)(:PASSOY/2)(:LIVELLO —1)
END

TO WGIU :PASSOX :PASSOY :LIVELLO
   IF: LIVELLO = 0 THEN SETXY (XCOR +:PASSOX)
   (YCOR —:PASSOY) STOP
   WGIU (:PASSOX/6)(:PASSOY/2)(:LIVELLO —1)
   WSU (:PASSOX/6)(:PASSOY/2)(:LIVELLO —1)
   WGIU (:PASSOX/6)(:PASSOY/2)(:LIVELLO —1)
   WGIU (:PASSOX/6)(:PASSOY/2)(:LIVELLO —1)
   WSU (:PASSOX/6)(:PASSOY/2)(:LIVELLO —1)
   WGIU (:PASSOX/6)(:PASSOY/2)(:LIVELLO —1)
END
```

TARTARUGA SPAZIALE

```
TO START
   MAKE "MAX 0
   MAKE "MIGLIORE"
   DRAW
   HT
   OBIETTIVO
   GIOCO
END

TO OBIETTIVO
   PU SETXY 0 5 PD
   RT 90
   REPEAT 36 [FD 31.4/36 RT 10]
   PU
END

TO GIOCO
   ACC.NOME
   INIZ
   PILOTA
END

TO ACC.NOME
   SPLITSCREEN
   PRINT1 [COME TI CHIAMI?]
   MAKE "NOME FIRST REQUEST
END

TO INIZ
   MAKE "PUNTI 200
   SETXY 100 100
   SETH 270
   MAKE "XVEL 0
   MAKE "YVEL 0
   FULLSCREEN
   ST
END

TO PILOTA
   COMANDO
   MUOVI.TART
   IF DISTANZA <5 THEN DONE STOP
   MAKE "PUNTI :PUNTI —1
   IF :PUNTI = 0 THEN TEMPO.SCADUTO
   STOP
   PILOTA
END

TO COMANDO
   MAKE "COM LEGGITASTO
   IF :COM = "D THEN RIGHT 30
   IF :COM = "S THEN LEFT 30
   IF :COM = "C THEN CALCIO
END

TO LEGGITASTO
   IF RC? THEN OUTPUT READCHARACTER
   OUTPUT "
END

TO CALCIO
   MAKE "VELX + 3 * SIN HEADING
   MAKE "VELY + 3 * COS HEADING
END

TO MUOVI.TART
   SETXY XCOR + :VELX YCOR +:VELY
END

TO DISTANZA
   OUTPUT SQRT (XCOR * XCOR + YCOR *
   YCOR)
END

TO FATTO
   PRINT "
   SPLITSCREEN
   (PRINT [COMPLIMENTI]:NOME)
   (PRINT [HAI FATTO] :PUNTI)
   RAPPORTO
   ANCORA
END

TO RAPPORTO
   IF :PUNTI >:MAX THEN MAKE
   "MAX :PUNTI MAKE" MIGLIORE :NOME
   PRINT "
   (PRINT [MIGLIOR GIOCATORE]:NOME
   [CON] :MAX [PUNTI])
END

TO ANCORA
   PRINT1 [ALTRA PARTITA?]
   MAKE "RISP FIRST REQUEST
   IF :RISP = "SI THEN RIGIOCA
   STOP
   IF :RISP = "NO THEN STOP
   PRINT [RISPONDI SI O NO PER FAVORE?]
   ANCORA
END

TO TEMPO.SCADUTO
   PRINT "
   SPLITSCREEN
   PRINT [TEMPO SCADUTO!!!]
   ANCORA
END

TO RIGIOCA
   HT
   ACC.NOME
   INIZ
   PILOTA
END
```

Uso degli stack

Dopo aver esaminato i modi di indirizzamento, vediamo altre caratteristiche del 6809

Fino a questo punto, abbiamo esaminato l'uso dei due registri S e U come indici nell'indirizzamento. È il momento di approfondire la loro funzione principale: quella di puntatori agli stack, cui si è accennato solo di passaggio.

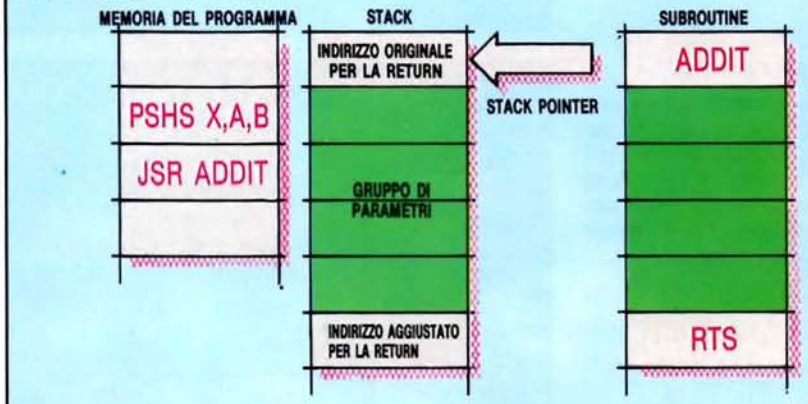
Uno stack non è che un caso particolare di una struttura di dati più generale, detta "lista". Questa struttura dovrebbe già esser nota anche a chi abbia poca familiarità coi linguaggi LOGO e LISP, in cui le liste assumono un ruolo fondamentale. In sostanza, una lista è semplicemente una sequenza di dati, che può essere organizzata in base ad una delle proprietà degli stessi dati (ad esempio, una serie di numeri ordinati per valori crescenti o una stringa di caratteri ordinata alfabeticamente) oppure semplicemente in base all'ordine d'immissione dei dati nella lista. Nell'impiego delle liste esistono alcune definizioni molto importanti: quella di "elemento precedente" e di "elemento successivo" e, in particolare, quella di "primo elemento" (detto anche "testa") e di "ultimo elemento" (detto anche "coda").

Ciò che caratterizza una lista è la dinamicità della struttura dei dati, che permette di aggiungere o togliere a piacimento gli elementi che la compongono. In alcuni tipi di liste, ciò può avvenire qualunque sia la posizione dell'elemento. Uno stack è contraddistinto dal fatto di possedere una sola estremità dalla quale si estraggono o si aggiungono i dati. Ogni nuovo elemento aggiunto diviene automaticamente la testa della lista e solo questa può essere estratta dallo stack. L'uso dello stack sui microprocessori Z80 e 6502 è stato esaminato a pag. 737.

Nel lavorare con uno stack, si possono presentare due situazioni estreme: lo stack può vuotarsi (condizione di "underflow") oppure superare la massima capacità consentita (condizione di "overflow"). Nel primo caso non si hanno conseguenze soltanto se la successiva operazione è l'immissione di un elemento, altrimenti sorgono dei problemi. Nel secondo caso le eccessive dimensioni dello stack potrebbero causare una sovrascrittura in aree di memoria riservate al programma: le conseguenze possono essere catastrofiche!

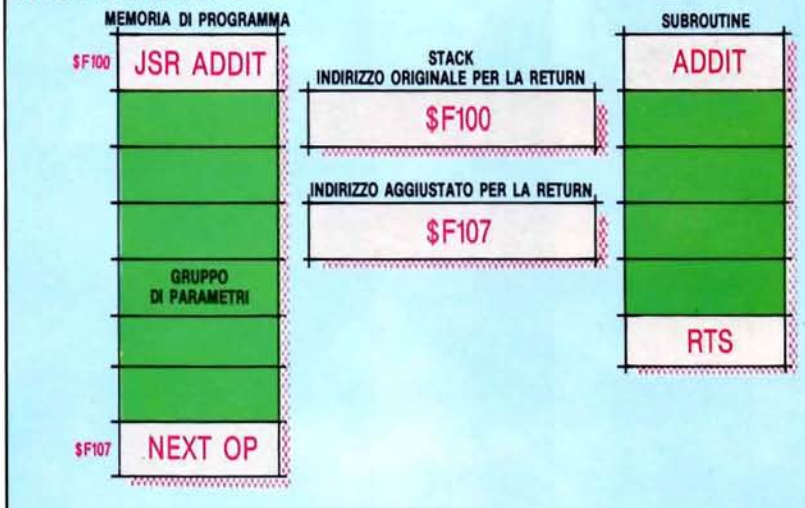
Le istruzioni di immissione e di estrazione di elementi dallo stack vengono indicate, rispettivamente, coi termini push e pop (o pull). Nel gergo si usano spesso anche le espressioni "appoggiare un valore sullo stack" e "recuperare un valore dallo stack".

PASSAGGIO DI PARAMETRI TRAMITE STACK



I parametri possono essere passati ad una subroutine caricandoli nei registri e depositandoli poi sullo stack. La subroutine chiamata estrae i parametri dallo stack, ma deve provvedere a calcolare e depositarvi l'esatto indirizzo usato dall'istruzione di ritorno (RTS) al termine della subroutine.

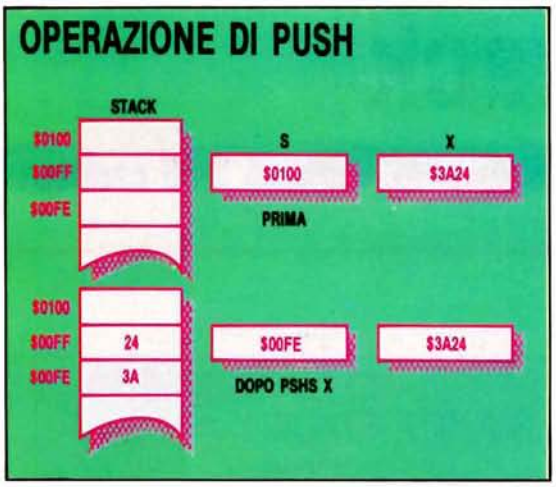
PASSAGGIO DI PARAMETRI TRAMITE INSERIMENTO



Un metodo più diffuso, per passare parametri, è quello di depositarli in locazioni opportunamente riservate, nel programma, subito dopo la chiamata alla subroutine (JSR). La subroutine estrae i parametri utilizzando il valore dell'indirizzo originale per la return, che punta al primo dei parametri. L'indirizzo da usare per l'istruzione di ritorno (RTS), prima di essere appoggiato sullo stack, va opportunamente aggiustato, affinché corrisponda all'indirizzo dell' istruzione seguente nel programma

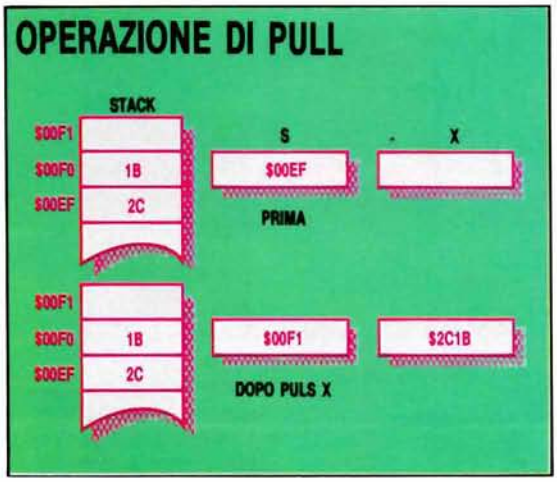
Appoggio sullo stack

Il puntatore di stack del 6809 punta sempre alla cima della lista, ossia all'ultimo dato depositato. Pertanto, quando viene eseguita una PSHS X, l'indirizzo in S è decrementato di 2 (per puntare alla nuova "testa") e il contenuto di X (a 16 bit) viene depositato in tale posizione, secondo il formato "alto-basso". Si noti che, in realtà, lo stack cresce verso la parte bassa della memoria



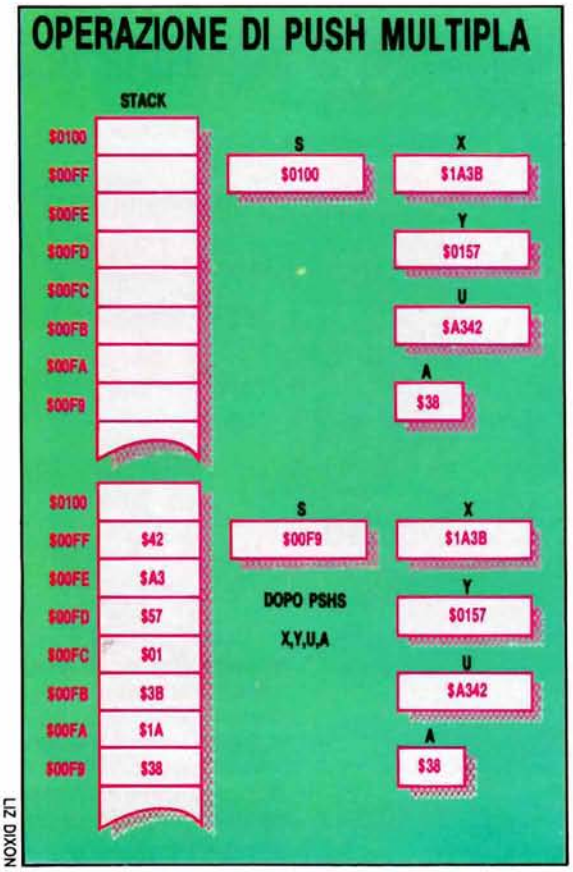
Recupero dallo stack

Quando viene eseguita una PULS X, il contenuto dei due byte individuati dall'indirizzo in S viene ricopiato nel registro X. Poi viene incrementato di due il valore nel puntatore S, per puntare alla nuova testa



Appoggio collettivo

Quando viene eseguita un'istruzione di push multipla, l'ordine di appoggio è ben preciso: PC, U (oppure S), Y, X, DP, B, A e CC. Pertanto, specificando PSHS X, Y, U, A, l'effettivo ordine di appoggio è: U, seguito da Y, X ed A



LIZ DIXON

Uno stack può essere implementato in vari modi (in BASIC, ad esempio, tramite l'impiego di una matrice) ma per quanto concerne il codice macchina, per creare uno stack si adopera un blocco di memoria libera ed un registro, denominato "stack pointer", ossia "puntatore allo stack". Questo puntatore è indispensabile per conoscere l'esatta posizione in cui si trova la testa della lista. Benché l'espressione "appoggiare un valore sullo stack" renda bene l'idea di un deposito temporaneo (si immagini una pila di vassoi in un self-service) va chiarito che in effetti dati e indirizzi non vengono realmente caricati o depositati nelle locazioni di memoria, ma soltanto copiati e che a "muoversi" è soltanto il puntatore che indica la testa della lista.
Per esser più precisi, esistono due varianti circa il contenuto del puntatore di stack: esso può indicare l'indirizzo della "prossima" posizione libera, oppure quello della locazione in cui è stato copiato l'ultimo elemento depositato. Sul 6809 è stato adottato questo secondo sistema, benché non vi siano particolari vantaggi nell'adoperare l'uno o l'altro metodo.
Una particolarità, nello stack del 6809, è che esso non cresce verso l'alto, ma verso la parte bassa della memoria. Vale a dire che l'indirizzo nel puntatore, ad ogni nuova immissione, viene decrementato.

Operazioni di stack

Nel linguaggio Assembly per il 6809, i codici mnemonici corrispondenti alle operazioni di appoggio e di recupero sono, rispettivamente: PSH e PUL. Tali istruzioni si applicano ad uno dei due registri S e U, cosicché si ottiene: PSHS, PULS, PHSU e PULU. Inoltre, dato che l'elemento da depositare o estrarre proviene da un registro o gli è destinato, va specificato anche il nome del registro.
Riassumendo, l'istruzione PSHS X, ad esempio, provoca due azioni ben precise. Prima di tutto viene decrementato il puntatore di stack S, per ottenere l'indirizzo della prossima locazione libera; (il decremento è di 1 o di 2 a seconda che il registro da appoggiare sia di 8 o di 16 bit); poi il valore contenuto in X viene copiato nella locazione individuata dall'indirizzo in S. Come si può notare dai diagrammi qui riportati, il 6809 ricorre alla solita convenzione "alto-basso" nel depositare valori a 16 bit in memoria: il byte alto di X($3A) è depositato in $00FE, ossia ad un indirizzo inferiore a $00FF, in cui viene messo il valore basso ($24). Lavorando con un assembler, tutti questi dettagli possono anche essere ignorati, in quanto il processore provvede a tutti gli adattamenti del caso.
L'istruzione PULS X ha l'effetto opposto: il valore a 16 bit depositato all'indirizzo contenuto in S viene copiato in X, poi S è incrementato di 2. Il secondo diagramma illustra questa operazione.
Esiste anche la possibilità di appoggiare o recuperare più registri in un sol colpo. Si consideri l'istruzione:

PSHS X,Y,U,A

Indipendentemente dall'ordine specificato in questa istruzione multipla, i registri vengono sempre depositati secondo la seguente progressione: PC (il contatore di programma), U oppure S, Y, X, DP (il registro di pagina diretta), B, A e CC (il registro di condizione del processore). Nell'operazione di recupero, l'ordine è quello inverso.
L'unica reale limitazione è che non è consentito operare una PSH o PUL del registro (S o U) usato come puntatore.
Nella redazione di programmi, lo stack viene comunemente adoperato come comodo deposito temporaneo per i valori dei registri. Tuttavia, esso gioca un ruolo estremamente importante nella gestione delle "interrupt" (esaminate più a fondo in seguito) e delle chiamate alle subroutine. Abbiamo già osservato come il contenuto del contatore di programma venga automaticamente depositato sullo stack quando il programma incontra una chiamata ad una subroutine e come, al termine di questa, il valore venga recuperato dallo stack (RTS, in fondo, equivale a PULS PC). Inoltre, uno dei due stack (ma quello S in particolare) può essere usato per passare parametri ad una subroutine.
Il passaggio di parametri tramite i registri (come quello descritto nell'uso della tabella dei salti, a pag. 1119) ha infatti due punti deboli.
In primo luogo il numero di parametri può esser superiore a quello dei registri disponibili; secondariamente, la subroutine può modificare uno o più parametri, che invece servono ancora nella routine principale.
Esistono, comunque, altre due tecniche per il passaggio di parametri:

1) I parametri possono essere depositati in apposite locazioni interne al programma (riservate tramite FCB, FDB o FCC), subito dopo la chiamata alla subroutine. Il valore del contatore di programma, automaticamente appoggiato sullo stack dall'istruzione JSR, coincide con l'indirizzo del primo parametro (PC infatti contiene sempre l'indirizzo della "prossima" istruzione) e può quindi essere usato per estrarre i vari parametri. Il primo esempio di programma ricorre al metodo appena descritto. Occorre prestare molta attenzione nel far eseguire correttamente la RTS, affinché riprenda l'esecuzione da un'istruzione e non da un dato dei parametri.
2) I parametri, tramite i registri, possono essere depositati sullo stack prima della chiamata alla subroutine, che poi provvede ad estrarli. Anche in questo caso, è necessario accertarsi che l'indirizzo in seguito usato dalla RTS non vada perso. Un esempio di questa tecnica, usata più frequentemente della precedente, è illustrato nel secondo programma.
Si noti il duplice ruolo dei registri S e U, usati sia come puntatori di stack, sia come indici di consultazione dei parametri conservati nello stack.

ADDIZIONE IN PRECISIONE MULTIPLA

Ecco due frammenti di codice macchina che illustrano i due principali metodi per passare parametri ad una subroutine. Nel primo esempio, i parametri (usati per calcolare la somma di due valori a quattro byte) sono situati subito dopo la chiamata alla subroutine. I parametri sono nelle locazioni $100 e $104, il risultato è posto in $108:

```
        BSR     MPADD
        FCB     4
        FDB     $100
        FDB     $104
        FDB     $108
        ORG     $1000
MPADD   PSHS    X,Y,U,A,B,CC
        LDU     9,S
        PULU    X,Y,B
        LDU     ,U
        ANDCC   #%11111110
LOOP    LDA     ,X+
        ADCA    ,Y+
        STA     ,U+
        DECB
        BNE     LOOP
        LDU     9,S
        LEAU    7,U
        STU     9,S
        PULS    PC,U,X,Y,A,B,CC
```

- Lunghezza di ogni numero (in byte)
- Indirizzo del primo numero
- Indirizzo del secondo numero
- Indirizzo del risultato
- Salva tutti i registri (occupando 9 byte di stack)
- Questa istruzione carica in U il valore di PC (al nono posto nello stack)
- Usando la tabella dei parametri come se fosse uno stack, si recuperano gli indirizzi dei numeri in X e Y e la lunghezza in B
- L'indirizzo del risultato va in U
- Azzera il flag di Carry
- Somma un byte alla volta, assieme al Carry
- Deposita il risultato
- Controlla se somma conclusa
- In caso negativo continua a sommare
- Altrimenti recupera l'indirizzo di ritorno e vi somma 7, per saltare oltre i parametri
- Ripristina tutti i registri, compreso il contatore di programma PC, anziché eseguire una RTS

Nel secondo esempio è usato il passaggio di parametri tramite lo stack:

```
        LDU     #$108
        LDX     #$100
        LDY     #$104
        LDA     #4
        PSHS    U,X,Y,A
        BSR     MPADD
        ORG     $1000
MPADD   PSHS    X,Y,U,A,B,CC
        LEAU    11,U
        PULU    X,Y,B
        ANDCC   #%111111110
LOOP    LDA     ,X+
        ADCA    ,Y+
        STA     ,U+
        DECB
        BNE     LOOP
        PULS    PC,U,X,Y,A,B,CC
```

- Indirizzo del risultato
- Indirizzo dell'operando
- Lunghezza
- Deposita i parametri sullo stack
- Salva tutti i registri
- I registri occupano 9 byte di stack, l'indirizzo di ritorno 2, per un totale di 11: U adesso punta ai parametri
- Il resto è identico all'esempio precedente

Missile Command

Difendiamo l'umanità da ondate di attacchi con testate nucleari

Il successo di un gioco stile "arcade" dipende moltissimo dal modo in cui viene presentato lo scenario, che deve essere accattivante, durante lo svolgimento del gioco. Inoltre, quest'ultimo deve avvincere fin dalle prime mosse. I programmi che non soddisfano questi due requisiti fondamentali scompaiono presto di scena. Benché *Missile Command* non sia più una novità, rimane pur sempre un gioco ancora piacevole: ne è prova la versione recentemente introdotta per i computer Atari.

Lo scenario è al tempo stesso semplice e subdolo: il giocatore impersona il comandante di una base di difesa antimissile, durante un attacco nucleare. Il suo compito è proteggere sei grandi città, che rappresentano il bersaglio delle testate atomiche in arrivo, lanciando i propri missili. Indubbiamente, il gioco punta sulla prontezza di riflessi e sulla velocità dell'azione.

Sullo schermo sono riprodotte le sagome delle sei città, oltre a quella di una piramide (la base difensiva) armata di missili pronti al lancio. Le scie dei missili aggressori iniziano a comparire sulla parte superiore dello schermo. A questo punto il giocatore deve posizionare una crocetta sul loro percorso, tramite il joystick. Alla pressione del tasto di "fuoco", viene lanciato un missile di difesa, che distrugge tutti gli aggressori che si trovano entro il raggio dell'esplosione.

A complicare le cose, alcuni dei missili aggressori sono dotati di più testate nucleari: se coinvolti marginalmente dall'esplosione, essi si suddividono in numerose testate più piccole. Per di più ogni tanto appaiono velivoli nemici, in grado di proteggere parzialmente i missili in arrivo. Abbattere uno di questi fa aumentare notevolmente il proprio punteggio di gioco. Al termine di ogni ondata di attacco, viene presentato un quadro riassuntivo della situazione: quante città si sono salvate e quanti missili rimangono per la difesa.

Col procedere del gioco, l'azione diviene sempre più rapida e pericolosa ed occorre applicare una strategia globale, anziché limitarsi ad abbattere missili qua e là. Ad esempio, il giocatore può costruire una barriera missilistica, da far esplodere al momento opportuno, magari annullando in un sol botto un'intera ondata di testate nemiche.

Ai livelli di gioco più alti, sorge un altro inconveniente: i missili attaccanti sono dotati di testate fornite di paracadute: ciò rende la loro eliminazione estremamente difficile, specialmente se i missili del giocatore esplodono vicini, ma senza colpire direttamente il bersaglio.

Durante una partita, occorre ricordarsi che i missili da difesa sono in numero limitato (30 per il primo livello): esaurite le difese, non resta che assistere impotenti alla distruzione delle città e della stessa base di difesa.

Ciascun livello (contraddistinto da una diversa colorazione dello scenario) consiste in due ondate di attacco, dopodiché viene calcolato il punteggio. Come in altri giochi della Atari, è possibile "saltare" ad un livello di maggior difficoltà. Il gioco termina quando le sei città vengono distrutte e il fatto è sottolineato da una conclusione oltremodo pessimistica: appare, a caratteri cubitali, la scritta THE END.

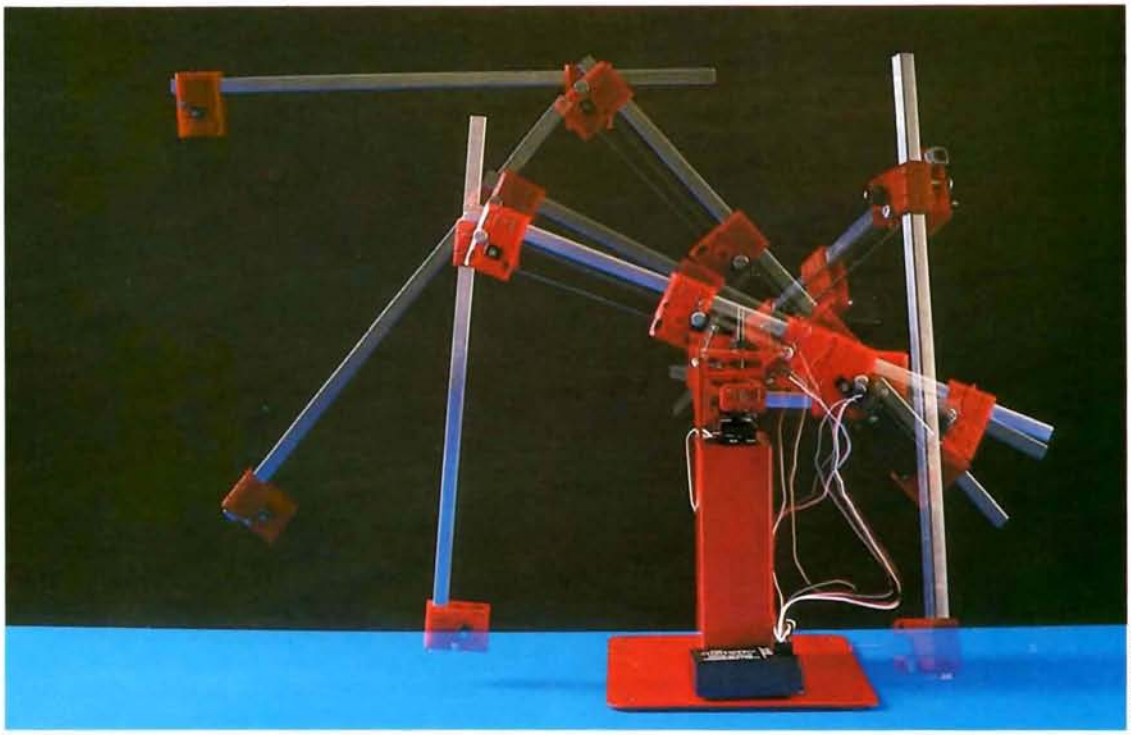
Il gioco, distribuito su cartuccia per i computer Atari, è corredato da un coloratissimo manuale, che surclassa la documentazione della maggior parte degli altri giochi per home computer: viene descritto come predisporre il gioco e come migliorare la propria strategia. Il tutto accompagnato da utili suggerimenti e da pregevoli illustrazioni.

Missile Command: per tutti i computer Atari
Distribuzione: Atari Corporation
Autori: Atari
Joystick: necessari
Formato: cartuccia

LIZ DIXON

Mani e gomiti

Come funzionano le articolazioni di un meccanismo robotizzato



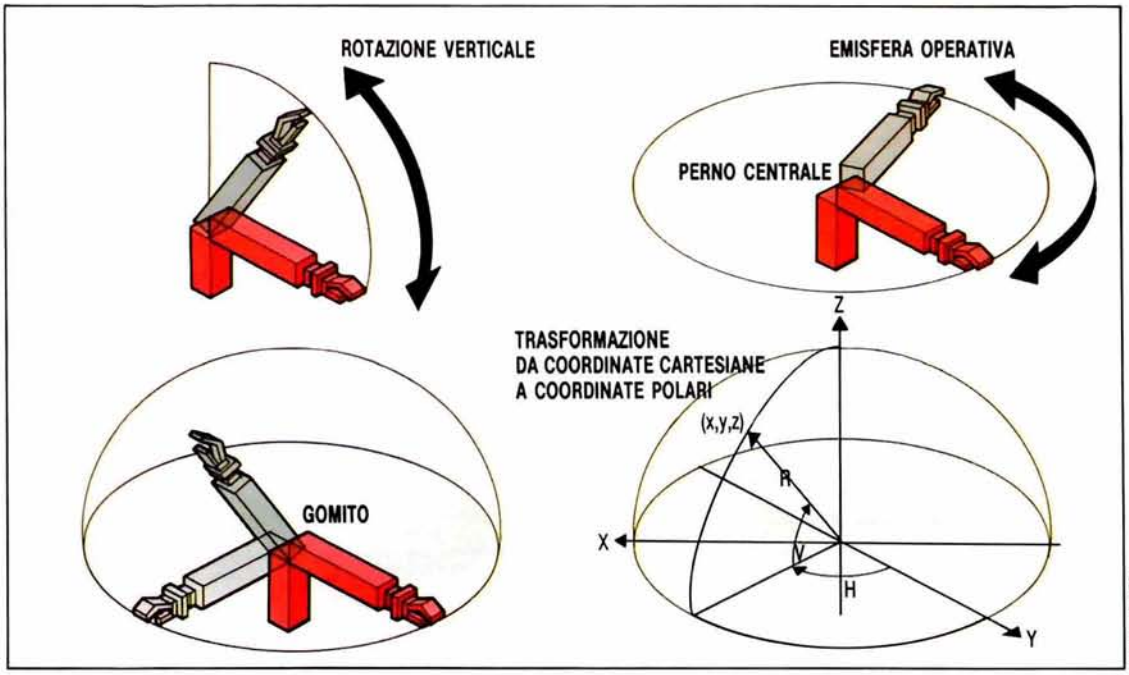
IAN McKINNELL

Doppia articolazione
Ancora per diverso tempo il robot più usato sarà un braccio dotato di mani diversificate per applicazioni industriali, domestiche ed hobbistiche. In effetti, solo poche attività richiedono veramente l'impiego di macchine "pensanti", simili a quelle resemitiche dalla fantascienza. I gripper programmabili, semi-intelligenti, stanno assumendo la stessa importanza di un aratro o di un telescopio

L'efficienza di un robot dipende soprattutto dalla precisione con cui esso riesce a manipolare gli oggetti. Molti robot vengono usati per operazioni del tipo "prendi e sposta" (in fabbrica, ad esempio, per trasferire alcune parti da un nastro trasportatore ad un altro), quindi la progettazione del braccio robot ha un'importanza primaria.
Questa deve tenere conto di tre aspetti fondamentali: il sistema che governa il robot deve essere costantemente in grado di definire la posizione del braccio, il quale deve possedere un proprio telaio o "scheletro" e, infine, deve avere un "sistema muscolare" che lo aziona e permette di controllarlo con precisione. I diversi modi in cui questi elementi interagiscono tra loro determinano la struttura complessiva del braccio robotizzato. I vari tipi di braccio vengono classificati in base al sistema di coordinate spaziali usate per controllarli.
Quando abbiamo descritto il movimento dei robot (vedere a pag. 1001) si è visto come impiegare il sistema di coordinate cartesiane per individuarne la posizione rispetto al piano d'appoggio, mediante i due assi x ed y tra loro perpendicolari. Lo stesso principio può essere applicato ad un braccio robot, ma, poiché il braccio si muove liberamente nello spazio, è necessario introdurre una terza coordinata z per definirne la posizione verticale. Il sistema a tre coordinate dà quindi una precisa descrizione tridimensionale della posizione del braccio robot.
In linea di principio, un braccio che usa questo sistema assomiglia ad una gru che può muoversi verso l'alto e il basso, a destra e a sinistra, avanti e indietro, oppure in una qualsiasi direzione intermedia. Simili bracci vengono impiegati per lavori che si svolgono in un'area circoscritta (su di un banco di lavoro, ad esempio). Tuttavia, la struttura stessa di tale braccio limita fortemente le possibili applicazioni: occorre infatti un'intelaiatura semifissa, che costringe ad operare soltanto in vicinanza dell'apposito banco di lavoro.
Un secondo sistema per definire la posizione del braccio è quello delle "coordinate cilindriche". Si immagini di avere un recipiente cilindrico vuoto: ogni suo punto interno può essere definito indicandone la distanza dall'asse centrale (con una variabile r), la distanza in senso rotatorio da un punto prefissato (usando una variabile angolare θ) e quella in senso verticale (con una terza variabile z). Le coordinate cilindriche permettono di sviluppare un sistema che può raggiungere un oggetto partendo da una posizione specificata (l'origine) interna al cilindro.

Bracci... in movimento

Anche il più semplice tipo di braccio, composto da un gripper e da uno snodo a gomito, può essere posizionato con notevole precisione all'interno di uno spazio tridimensionale.
Il gomito è "a cerniera" e ruota su un perno centrale: può quindi compiere movimenti semicircolari verticali e rotazioni complete orizzontali, raggiungendo qualsiasi punto della sua area operativa emisferica. Ricorrendo alla trigonometria, le coordinate cartesiane del sistema (x, y e z) permettono di calcolare la posizione di ogni punto interno all'emisfera: H, la rotazione del perno centrale, è data da ARCTAN (x/y); V, la rotazione del gomito, da ARCSIN (z/R). Il braccio viene programmato in base alla collocazione cartesiana degli oggetti e trasforma ciascun gruppo di coordinate in due rotazioni, che vengono poi trasmesse ai due servomotori responsabili del movimento



I bracci che adottano le coordinate sferiche (o polari), invece, identificano una data posizione servendosi di due angoli e di una "distanza": in questo caso la distanza corrisponde alla lunghezza del braccio, mentre gli angoli indicano la rotazione della base e l'elevazione del braccio. Il sistema può esser paragonato alla torretta di un carro armato, munita di un cannone a lunghezza variabile. Le coordinate polari si esprimono, in genere, mediante le variabili r, θ e Φ. Per il progettista del robot è abbastanza semplice creare un braccio "telescopico", in grado di estendersi o accorciarsi tramite un sistema idraulico.
Nonostante tutto, il sistema più comune per rappresentare la posizione del braccio è quello delle coordinate "di rivoluzione", appositamente sviluppato per far compiere al braccio robotizzato movimenti simili a quelli di un braccio umano. Come in precedenza, tre variabili definiscono la posizione del braccio, ma in questo caso il sistema usa esclusivamente coordinate angolari, esprimibili come θ, γ e Φ. θ indica l'angolo di rotazione della base, γ l'angolo di elevazione del braccio e Φ l'angolo dello snodo terminale o "polso".

I "muscoli" del braccio robot

Il tipo di "scheletro" del braccio dipende dunque dal particolare sistema di coordinate usato. Vediamo adesso che cosa sono e come funzionano i "muscoli" del braccio robot. In generale, per questa funzione vengono impiegati tre tipi di dispositivi: elettrici, idraulici o pneumatici. Nel primo caso si impiegano motori elettrici passo passo, di cui si è già parlato in precedenza. Essi possono trovarsi all'interno delle articolazioni del braccio ed azionarlo direttamente, facendogli eseguire di volta in volta piccoli movimenti rotatori, oppure possono muoverlo indirettamente mediante ingranaggi, pulegge o leve.

I sistemi migliori, comunque, devono consentire movimenti di contrazione ed estensione del tutto simili a quelli umani e richiedono un'azione da parte del muscolo sullo scheletro. Questo risultato si ottiene inserendo in ogni articolazione del braccio una serie di pistoni idraulici o pneumatici. Sui massicci robot industriali si preferisce usare pistoni idraulici, che garantiscono una maggiore pressione e conferiscono più forza al braccio: il fluido in essi contenuto, infatti, è meno deformabile rispetto all'aria e consente una maggiore precisione nei movimenti (vengono attenuati eventuali effetti di rimbalzo).
Per il movimento dei bracci robot, indipendentemente dal sistema adottato, si possono usare pistoni ad azione singola o doppia. Questo tipo di strumenti è chiamato "azionatore lineare".
Tuttavia, è possibile un ulteriore perfezionamento: anziché impiegare pistoni dai movimenti lineari, trasformati poi in rotazioni dell'articolazione, si può ricorrere ad un "azionatore rotazionale". Questo meccanismo produce direttamente una rotazione dello snodo, esercitando una pressione su una paletta fissata su un giunto circolare. Il procedimento è simile a quello usato nei motori elettrici passo passo, ma la pressione idraulica consente di fornire una forza molto più intensa. La pressione pneumatica, invece, è inadeguata a questo tipo di applicazione.
Dopo aver descritto il meccanismo del braccio robotizzato, passiamo al suo "azionatore terminale", ossia alla "mano", che permette al robot di svolgere il proprio lavoro. Per capire come funziona, conviene far riferimento alla mano dell'uomo. Con un polso ingessato, si hanno notevoli difficoltà a compiere anche le azioni più semplici: se scriviamo a macchina, ad esempio, i polsi alzano ed abbassano le mani sulla tastiera per permettere di agire sui tasti. Questo movimento è detto "innalzamento". Con i polsi bloccati

siamo invece costretti a muovere l'intero avambraccio.
Inoltre, i polsi spostano le mani da una parte all'altra della tastiera: questa è la "traslazione". Quando ciò non è possibile si è costretti a muovere i gomiti. Alla fine della battitura possiamo ruotare i polsi per poggiare le mani ai lati della tastiera con il palmo rivolto verso l'alto: questa è chiamata "rotazione". Anche questo movimento naturale diviene quasi impossibile se non possiamo articolare i polsi.
Teoricamente, il polso di un braccio robot deve essere in grado di compiere questi tre movimenti. Innalzamento, traslazione e rotazione possono avere due direzioni (alto/basso, destra/sinistra, oraria/antioraria), ciascuna delle quali è detta "grado di libertà". Un robot che le possiede tutte e tre ha dunque sei gradi di libertà.
Solitamente i robot dispongono solo di quattro o cinque gradi di libertà, ma ad ogni riduzione di capacità di movimento del polso corrisponde una maggiore mobilità di altri segmenti del braccio.

La "mano" del robot

Esaminiamo adesso l'architettura della mano del robot.
La configurazione ideale è quella di un braccio di tipo umano, che termina in una mano, anch'essa di aspetto umano. Alcune mani robot si avvicinano molto a questo risultato. Tuttavia, il tipo più comune di mano da robot è il "gripper" a tre dita, composto da due dita e da un pollice opponibile, che consente la presa di oggetti di varie forme e dimensioni. Per azionare la mano vengono impiegati i tre dispositivi già menzionati, scelti in base al tipo di lavoro che il robot deve eseguire. Se la mano deve spostare pesanti oggetti di considerevoli dimensioni, vengono abitualmente usati pistoni idraulici. Per molte altre applicazioni, invece, sono sufficienti pistoni pneumatici o motori elettrici, perché la mano deve semplicemente sollevare un oggetto e lasciarlo andare dopo un certo tempo. Se il braccio ed il polso sono posizionati in modo corretto, non è necessario che la mano possieda una grande precisione: basta che sia in grado di aprirsi e chiudersi!
In molti casi, l'azionatore terminale del braccio robotizzato non è una mano, bensì uno strumento in grado di compiere lavori specializzati.
Ad esempio, i robot usati per fare saldature hanno un saldatore fissato al posto del gripper. Spesso i robot stessi sono programmati per scegliere l'azionatore terminale adatto al lavoro da eseguire: automaticamente, l'attrezzo occorrente (una diversa punta per il trapano oppure una pistola a spruzzo) viene scelto da una rastrelliera e montato sul braccio, dopo aver riposto quello usato in precedenza. Questo processo ha poco di "umano", ma conferisce ai robot una grande adattabilità!

Il polso del robot

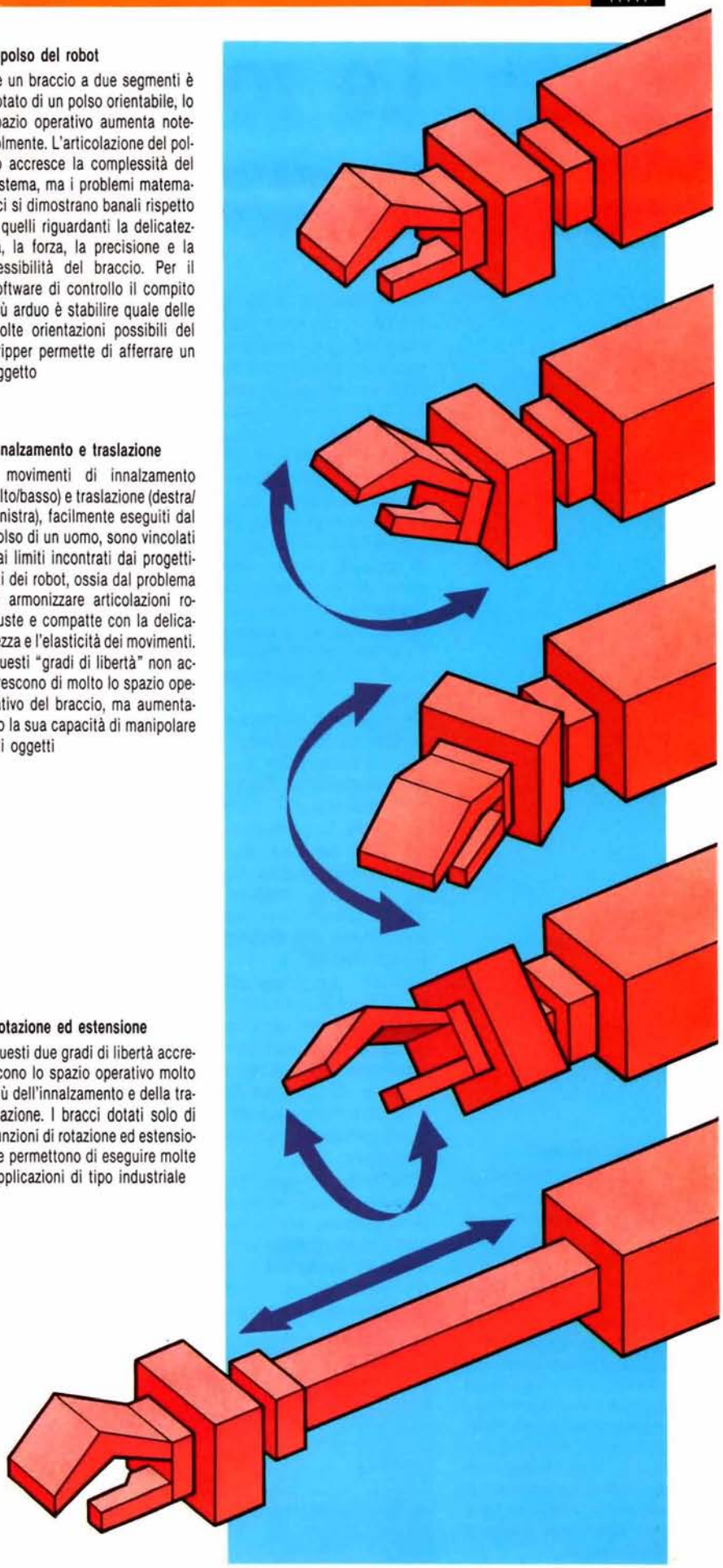
Se un braccio a due segmenti è dotato di un polso orientabile, lo spazio operativo aumenta notevolmente. L'articolazione del polso accresce la complessità del sistema, ma i problemi matematici si dimostrano banali rispetto a quelli riguardanti la delicatezza, la forza, la precisione e la flessibilità del braccio. Per il software di controllo il compito più arduo è stabilire quale delle molte orientazioni possibili del gripper permette di afferrare un oggetto

Innalzamento e traslazione

I movimenti di innalzamento (alto/basso) e traslazione (destra/sinistra), facilmente eseguiti dal polso di un uomo, sono vincolati dai limiti incontrati dai progettisti dei robot, ossia dal problema di armonizzare articolazioni robuste e compatte con la delicatezza e l'elasticità dei movimenti. Questi "gradi di libertà" non accrescono di molto lo spazio operativo del braccio, ma aumentano la sua capacità di manipolare gli oggetti

Rotazione ed estensione

Questi due gradi di libertà accrescono lo spazio operativo molto più dell'innalzamento e della traslazione. I bracci dotati solo di funzioni di rotazione ed estensione permettono di eseguire molte applicazioni di tipo industriale



Le routine di utilità

Alcuni programmi "utility" sono già presenti nelle ROM, ma se ne possono aggiungere altri

La presenza e la qualità delle routine di "utilità" variano notevolmente da apparecchio ad apparecchio: alcuni home computer, quali lo Spectrum della Sinclair, possiedono solo un semplice programma di Editor, mentre altri hanno funzioni molto più estese. Il Micro BBC, ad esempio, è dotato dei comandi TRACE e RENUMber; il primo visualizza il numero di linea delle istruzioni BASIC che vengono eseguite, il secondo riordina automaticamente la numerazione delle linee di un programma BASIC. Entrambe queste istruzioni sono di grande aiuto nello sviluppo e nel debugging di un programma.

Tuttavia, qualunque sia la dotazione di un apparecchio, è sempre comodo poter disporre di routine di utilità addizionali ed il mercato offre una vasta gamma di prodotti tra cui scegliere.

Di solito i programmi di utilità vengono scritti in Assembly, sia per la velocità del codice macchina sia perché non è facile per un programma BASIC automodificarsi senza compromettere il proprio funzionamento. Tuttavia, la presente analisi inizia con alcune semplici routine di utilità scritte in BASIC: concentreremo la nostra attenzione sul lavoro che una "utility" deve eseguire, senza approfondire problemi secondari come il ruolo giocato dal sistema operativo e dall'interprete BASIC.

Sebbene un programma BASIC incontri molte difficoltà a modificarsi, è abbastanza semplice crearne uno capace di esaminarne un altro. Il programma di utilità qui proposto in versione Spectrum e Micro BBC, individua tutte le ricorrenze del nome di una variabile o di una funzione in un programmma BASIC e visualizza i numeri delle linee in cui essa compare.

Entrambi i programmi cominciano individuando l'indirizzo iniziale di memoria del programma sotto analisi, quindi scorrono il suo listato linea dopo linea, estraendo tutti i nomi di variabili e saltando le parti che non ne contengono. Nella fase finale vengono confrontati i nomi estratti con quello cercato dal programma. Per prima cosa, quando il programma inizia l'esame di una nuova linea del listato BASIC, annota il numero di linea (memorizzato, su entrambi i micro, in due byte), e la lunghezza della linea (ossia il numero di byte da essa occupati). Sul Micro BBC, la lunghezza della linea è data dal numero dei byte compresi tra il numero di linea ed il marcatore di "fine linea" (codice ASCII 13): questo valore occupa un solo byte. Sullo Spectrum, invece, la lunghezza della linea è memorizzata in due byte e rappresenta il numero dei byte compresi tra il carattere che segue i byte della lunghezza ed il marcatore di fine-linea (risultano quindi esclusi dal totale i byte del numero e della lunghezza della linea).

In entrambe le versioni, il programma di ricerca ignora le istruzioni REM e tutto ciò che è racchiuso tra virgolette, in quanto le stringhe di caratteri in esse contenute non corrispondono a variabili di programma. Il Micro BBC consente di usare numeri esadecimali, preceduti dal carattere &: occorre assicurarsi che il programma non li scambi per nomi di variabili e che salti tutte le stringhe precedute dal carattere &.

Il programma non deve confondere, ad esempio, il numero esadecimale A0 di &A0 con il nome di variabile "A0".

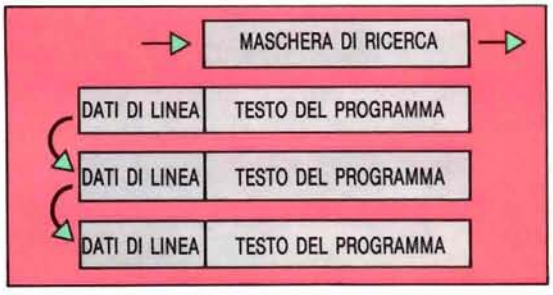
Sullo Spectrum, i numeri vengono memorizzati sotto forma di caratteri ASCII (uno per ogni cifra del numero) seguiti dal codice ASCII 14: quindi l'equivalente binario del numero è contenuto in cinque byte.

Il nostro programma deve essere in grado di saltare il codice per il numero e l'equivalente binario di cinque byte.

Solo dopo che il programma ha verificato queste condizioni esso può passare all'esame dei nomi contenuti nella linea. Entrambi i programmi partono dal presupposto che i nomi devono iniziare con una lettera seguita poi da un'altra lettera o da un numero. La versione BBC usa variabili intere (indicate dal carattere % dopo il nome) ed il carattere di sottolineatura, mentre entrambe le versioni prevedono l'impiego del simbolo $, per indicare le variabili stringa.

I nomi di una matrice, di una funzione o di una procedura, sul Micro BBC, sono seguiti da una parentesi tonda aperta: questo carattere, in realtà, non fa parte del nome, ma viene usato dai programmi per distinguere queste variabili da quelle semplici (senza indice).

Nella versione Spectrum c'è qualche complicazione. Il suo dialetto BASIC non fa distin-



Procedura di ricerca

Sul Micro BBC e sullo Spectrum una linea di programma inizia con tre o quattro byte contenenti il numero e la lunghezza della linea, seguiti dal testo BASIC. Quest'ultimo è intercalato dai "token" relativi alle parole chiave. Quando il programma esamina la linea, ne registra il numero e calcola l'indirizzo iniziale della linea successiva in base alla lunghezza in esame. Quindi esegue una scansione del testo BASIC, confrontandolo con la maschera di ricerca, finché non incontra un marcatore di fine linea o di fine programma, un token per l'istruzione REM oppure il nome di variabile cercato

zione tra maiuscole e minuscole all'interno di una variabile. Quindi i nomi TEMPO, Tempo oppure TEmpo vengono considerati come uno stesso nome di variabile. Pertanto, il programma trasforma tutte le lettere in maiuscole prima di iniziare il confronto. Lo Spectrum accetta nomi di variabili contenenti caratteri "spazio", ma è preferibile evitare un simile formato, che può confondere.

Diversamente da molti altri dialetti BASIC, quello dello Spectrum non fa molta distinzione tra le variabili stringa semplici e quelle a matrice.

Al contrario, i due formati sono piuttosto simili. Poiché in un programma possono comparire due variabili T$ e T$(i), riferite rispettivamente ad una stringa e ad un suo elemento interno, il programma di ricerca non opera alcuna distinzione tra variabili stringa semplici e quelle a matrice. In pratica, però, questa non è una limitazione, dato che il BASIC dello Spectrum non accetta, in uno stesso programma, variabili stringa e matrici con lo stesso nome.

Ciascuno dei due programmi di utilità va prima trascritto, poi salvato e quindi "fuso" con il programma da esaminare. Per lo Spectrum si adoperi il comando MERGE e per il BBC la procedura descritta nel manuale. La routine di ricerca si avvia con RUN 9000 (Spectrum) o con GOTO 30000 (Micro BBC): il programma invita subito ad immettere il nome di variabile da cercare. Se la ricerca riguarda un nome di matrice, questo deve essere seguito da una parentesi tonda aperta. Il listato di questo programma per il Commodore 64 verrà fornito in seguito. Si provi a scrivere un programma che, esaminandone un altro, riesca ad individuare le linee che chiamano una particolare subroutine.

SPECTRUM

```
9000 INPUT "Nome da cercare ? ";LINE T$
9010 FOR i=1 TO LEN (T$)
9020 IF T$(i)>="a" AND T$(i)<="z" THEN
     LET T$(i)=CHR$ (CODE (T$(i))-32)
9030 NEXT i
9040 LET SimboloREM=234
9050 LET Virgolette=34
9060 LET Nuovalinea=13
9070 LET Sottolin=95
9080 LET Numero=14
9090 LET PROG=23635
9100 LET Puntatxt=PEEK (PROG)+256*PEEK (PROG+1)
9110 LET Numlinea=256*PEEK (Puntatxt)+PEEK (Puntatxt+1)
9120 IF Numlinea>9000 THEN STOP
9130 LET Puntatxt=Puntatxt+2
9140 LET Ltesto=PEEK (Puntatxt)+256*PEEK (Puntatxt+1)
9150 LET Puntatxt=Puntatxt+2
9160 LET Lineasucc=Puntatxt+Ltesto
9170 IF PEEK (Puntatxt)=Nuovalinea
     THEN LET Puntatxt=Puntatxt+1: GO TO 9110
9180 IF PEEK (Puntatxt)<>TokendiREM THEN GO TO 9220
9190 REM Salta le linee REM
9200 LET Puntatxt=Lineasucc
9210 GO TO 9110
9220 IF PEEK (Puntatxt)<>Virgolette THEN GO TO 9280
9230 REM Salta i caratteri tra virgolette ma si ferma a
     fine linea se mancano le virgolette di chiusura
9240 LET Puntatxt=Puntatxt+1
9250 IF PEEK (Puntatxt)=Nuovalinea
     THEN LET Puntatxt=Puntatxt+1: GO TO 9110
9260 IF PEEK (Puntatxt)<>Virgolette THEN GO TO 9240
9270 LET Puntatxt=Puntatxt+1
9275 GO TO 9170
9280 IF PEEK (Puntatxt)<>Numero THEN GO TO 9320
9290 REM Salta i numeri binari a 5 byte
9300 LET Puntatxt=Puntatxt+6
9305 GO TO 9170
9310 REM Il primo carattere del nome deve essere
     una lettera maiuscola o minuscola
9320 IF PEEK (Puntatxt)<=CODE ("A") AND PEEK
     (Puntatxt)<=CODE ("Z") THEN LET c$=CHR$
     (PEEK (Puntatxt)): GO TO 9370
9330 REM Usare le maiuscole al posto delle minuscole
9340 IF PEEK (Puntatxt)>=CODE ("a") AND PEEK
     (Puntatxt)<=CODE ("z") THEN LET c$=CHR$
     (PEEK (Puntatxt)-32): GO TO 9370
9350 LET Puntatxt=textpointer+1
9360 GO TO 9170
9370 LET n$=""
9380 LET n$=n$+c$
9390 LET Puntatxt=Puntatxt+1
9400 REM Lettere, numeri o caratteri di sottolineatura
     dopo il primo carattere del nome
9410 IF PEEK (Puntatxt)>=CODE ("A") AND PEEK
     (Puntatxt)<=CODE ("Z") THEN LET c$=CHR$
     (PEEK (Puntatxt)): GO TO 9380
9420 REM Usare le maiuscole al posto delle minuscole
9430 IF PEEK (Puntatxt)>=CODE ("a") AND PEEK
     (Puntatxt)<=CODE ("z") THEN LET c$=CHR$
     (PEEK (Puntatxt)-32): GO TO 9380
9440 IF PEEDK (Puntatxt)>=CODE ("0") AND PEEK
     (Puntatxt)<=CODE("9") THEN LET c$=CHR$
     (PEEK (Puntatxt)): GO TO 9380
9450 IF PEEK (Puntatxt)=Virgolette THEN GO TO 9380
9460 REM Le variabili stringa terminano col carattere $
9470 IF PEEK (Puntatxt)=CODE ("$") THEN LET n$=n$+
     "$": LET Puntatxt=Puntatxt+1: GO TO 9500
9480 REM Le matrici e le funzioni terminano con (
9490 IF PEEK (Puntatxt)=CODE ("(") THEN LET n$=n$+CHR$
     (PEEK (Puntatxt)): LET Puntatxt=Puntatxt+1
9500 IF n$=t$ THEN PRINT n$;" IN LINE ";Numlinea
9520 GO TO 9170
```

MICRO BBC

```
30000 INPUT "Nome da cercare ";TARGET$
30010 SimboloREM=244
30020 Virgolette=34
30030 Esadec=38
30040 Nuovalinea=13
30050 Sottolin=95
30060 Puntatxt=PAGE
30070 Puntatxt=Puntatxt+1
30080 Numlinea=256*?Puntatxt+?(Puntatxt+1)
30090 IF Numlinea>=30000 THEN END
30110 Llinea=?Puntatxt
30120 Finelinea=Puntatxt+Llinea-3
30130 Puntatxt=Puntatxt+1
30140 IF ?Puntatxt=Nuovalinea THEN GOTO 30070
30150 IF ?Puntatxt<>TokendiREM THEN GOTO 30180
30160    REM Salta le linee REM
30170    Puntatxt=Endline:GOTO 30070
30180 IF ?Puntatxt<>Virgolette THEN GOTO 30240
30190     REM Salta i caratteri tra virgolette ma si
          ferma alla fine della linea se mancano le
          virgolette di chiusura
30200    Puntatxt=Puntatxt+1
30210    IF ?Puntatxt=Nuovalinea THEN GOTO 30070
30220    IF ?Puntatxt<>Virgolette THEN GOTO 30200
30230    Puntatxt=Puntatxt+1
30235 GOTO 30140
30240 IF ?Puntatxt=<>Hex THEN GOTO 30300
30250     REM Salta gli esadecimali per non
          confonderli con i nomi di variabile
30260    Puntatxt=Puntatxt+1
30270    IF ?Puntatxt>=ASC("0") AND
         ?Puntatxt<=ASC("9") THEN GOTO 30260
30280    IF ?Puntatxt>=ASC("A") AND
         ?Puntatxt<=ASC("F") THEN GOTO 30260
30285    GOTO 30140
30290  REM Il primo carattere del nome deve essere
       una lettera maiuscola o minuscola
30300 IF  ?Puntatxt>=ASC("A") AND ?Puntatxt<=ASC("Z")
      THEN GOTO 30330
30310 IF  ?Puntatxt>=ASC("a") AND ?Puntatxt<=ASC("z")
      THEN GOTO 30330
30320 GOTO 30130
30330 Nome$=""
30340 Nome$=Nome$+CHR$(?Puntatxt)
30350 Puntatxt=Puntatxt+1
30360 REM  Lettere, numeri o caratteri di
      sottolineatura dopo il primo carattere
30370 IF ?Puntatxt>=ASC("A") AND ?Puntatxt<=ASC("Z")
      THEN GOTO 30340
30380 IF ?Puntatxt>=ASC("a") AND ?Puntatxt<=ASC("z")
      THEN GOTO 30340
30390 IF ?Puntatxt>=ASC("0") AND ?Puntatxt<=ASC("9")
      THEN GOTO 30340
30400 IF ?Puntatxt=Sottolin THEN GOTO 30340
30410 REM Le variabili stringa terminano con $ quelle
      intere con %
30420 IF ?Puntatxt=ASC("$") THEN Nome$=Nome$+CHR$
      (?Puntatxt):Puntatxt=Puntatxt+1:GOTO 30450
30430 IF ?Puntatxt=ASC("%") THEN Nome$=Nome$+CHR$
      (?Puntatxt):Puntatxt=Puntatxt+1
30440 REM Le matrici, le procedure e le funzioni
      terminano con (
30450 IF?Puntatxt=ASC("(") THEN Nome$=Name$+CHR$
      (?Puntatxt):Puntatxt=Puntatxt+1
30460 IF Name$=TARGET$ THEN PRINT Name$;
      " IN LINE " Numlinea
30470 GOTO 30140
30480 END
```

Sprite col LOGO

Usiamo il LOGO sul Commodore, per immagini animate con gli sprite

In LOGO gli sprite si comportano come le "tartarughe" ed obbediscono agli stessi comandi; la loro forma può essere definita dall'utente, ma, diversamente dalla tartaruga, gli sprite non ruotano sullo schermo quando cambia la direzione del loro movimento. Il LOGO del Commodore prevede l'uso di sette sprite, numerati da 1 a 7, mentre alla tartaruga è assegnato il numero di sprite 0. All'inizio, è selezionato automaticamente lo sprite 0, che obbedisce a tutti i comandi per gli sprite che vengono battuti. Inserendo il comando TELL 1, si abilita lo sprite 1 che diviene lo sprite "corrente", finché non ne viene selezionato uno nuovo.
Tuttavia, dopo l'esecuzione del comando TELL 1, sullo schermo non compare nulla: tutti gli sprite, diversamente dalla tartaruga, inizialmente sono oggetti "invisibili" e le loro penne di tracciamento sono in posizione "sollevata". Per visualizzare lo sprite 1 ed i suoi movimenti, occorre usare il comando ST, che riproduce sullo schermo una griglia quadrata vuota. Ci si può esercitare con questa griglia per sprite, usando i comandi per la tartaruga FD, BK, RT, LT, PD, PU, ST, HT, ecc. Collocando lo sprite 1 nella stessa posizione dello sprite 0 (la tartaruga), si nota che lo sprite compare "al di dietro" della tartaruga. Infatti, gli sprite con numero di codice inferiore vengono visualizzati "davanti" a quelli con numero di codice più alto. Ciò conferisce una certa tridimensionalità alle immagini sullo schermo.
Il dischetto delle utility del LOGO Commodore contiene un programma di "sprite editor" che viene caricato in memoria con il comando READ "SPRED. Per disegnare la sagoma dello sprite 1, si deve innanzi tutto renderlo "corrente" con TELL 1, quindi inserire il comando EDSH. Sullo schermo appare un'immagine ingrandita della griglia dello sprite sulla quale è possibile muovere il cursore. Ciascun pixel viene poi acceso premendo il tasto "asterisco" (*) o spento con la barra spaziatrice.
Dopo aver creato uno sprite, occorre battere un Control-C, per fissarne la definizione, ed usare il comando ST per renderlo visibile. Per trasferire sullo sprite corrente la figura definita per lo sprite 1 si usa l'istruzione SETSHAPE 1.
Dopo aver creato una serie di sprite, si possono salvare su un file con l'istruzione SAVESHAPES "NOMEFILE e richiamare in memoria con READSHAPES "NOMEFILE.
In un problema matematico, quattro insetti vengono disposti agli angoli di un quadrato: ciascun insetto insegue e deve raggiungere quello che si trova alla sua destra nonostante che tutti abbiano la stessa velocità. Lo scopo del programma a pagina 1148 è quello di tracciare il percorso compiuto dai quattro insetti; grazie al LOGO il problema può essere risolto servendosi degli sprite.
La procedura usata posiziona quattro copie dello stesso sprite sugli angoli del quadrato, quindi le muove l'una all'inseguimento dell'altra. L'immagine dell'insetto è stata definita come sprite 3, e tale definizione viene "copiata" per gli altri sprite mediante il comando SETSHAPE 3 nella procedura POSIZIONE.
Il "cuore" della soluzione si trova nella procedura SEGUI: prima ad X e Y vengono assegnate le coordinate x ed y dello sprite inseguito (:B), poi, tramite il comando TOWARDS, viene assegnata, allo sprite inseguitore (:A), la direzione del movimento verso quel punto. TOWARDS prevede due parametri in input, rappresentanti le coordinate del punto verso cui lo sprite deve muoversi e calcola la direzione del suo movimento verso quel punto.

Animazione

Spesso la grafica a sprite viene usata per creare animazioni: in un programma d'animazione viene definita una serie di sprite, relativi a una stessa figura, ma ciascuno si differenzia dal precedente solo per un piccolo particolare. Quando gli sprite vengono visualizzati in successione, si ottiene una figura che sembra muoversi sullo schermo. Il LOGO del Commodore possiede tre sprite, che creano una rappresentazione piuttosto grossolana di un uomo in movimento. Le seguenti procédure ripuliscono lo schermo e visualizzano l'animazione delle tre immagini.

```
TO ANIMAZIONE
   TELL 0
   DRAW
   PU
   BIGX BIGY
   SETH 90
   MOVIMENTO 2
END

TO MOVIMENTO :FIGURA
   FD 5
   SETSHAPE :FIGURA
   IF :FIGURA = 4 THEN MAKE "FIGURA1
   MOVIMENTO :FIGURA + 1
END
```



Prima di eseguire queste 2 procedure, è necessario "leggere" il file SPRITES dal dischetto delle utility; quest'ultimo contiene alcune procedure di grande utilità, tra le quali BIGX e BIGY, che raddoppiano le dimensioni di uno sprite e SMALLX e SMALLY invece lo riportano alle dimensioni originali. Dopo aver caricato in memoria i tre sprite con l'istruzione READSHA-

PES "RUNNER, si possono inviare in esecuzione le 2 procedure. I quattro sprite definiti nella pagina seguente verranno in seguito usati nel gioco *Cavaliere contro Drago*.

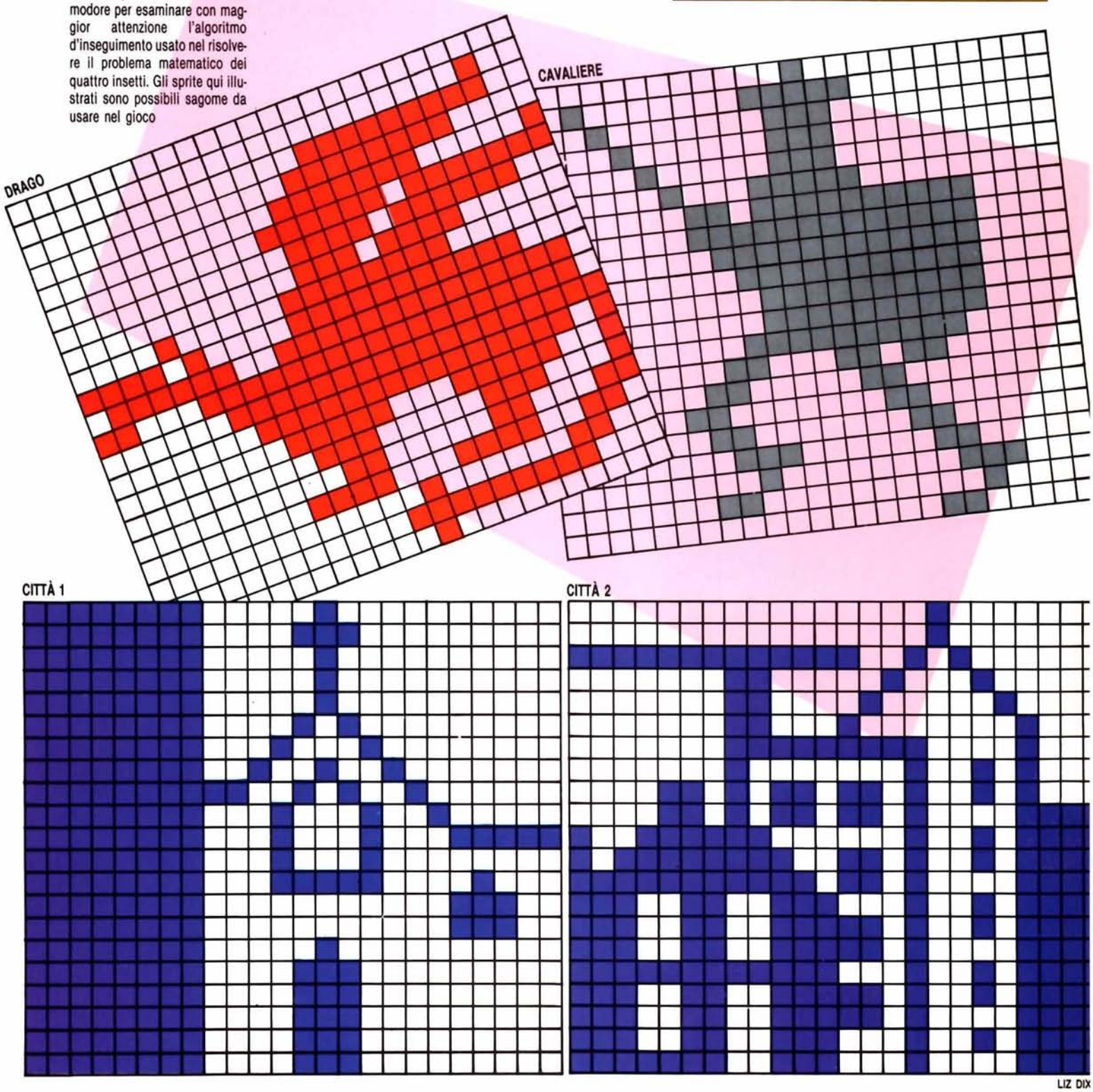
Logo & Logo

I dialetti LOGO dello Spectrum e dell'Apple sono privi di grafica a sprite. Il LOGO per Atari presenta le seguenti differenze:

1) possiede solo 4 sprite;
2) usa il comando SETSH al posto di SETSHAPE;
3) lo "sprite editor" fa parte delle primitive. Battendo la barra spaziatrice si accende un pixel spento oppure se ne spenge uno acceso.

IL DRAGO RAMPANTE

Il gioco "Cavaliere contro Drago", presentato più avanti, ricorre alle capacità del LOGO Commodore per esaminare con maggior attenzione l'algoritmo d'inseguimento usato nel risolvere il problema matematico dei quattro insetti. Gli sprite qui illustrati sono possibili sagome da usare nel gioco



LIZ DIX

I QUATTRO INSETTI

In questa dimostrazione geometrica, diventata classica, i quattro insetti si muovono in successione, ognuno all'inseguimento di quello alla propria destra. L'algoritmo crea una spirale che si sviluppa verso l'interno, i cui "bracci" descrivono una "curva d'inseguimento", ben nota ai piloti di caccia ed agli appassionati dei giochi di tipo arcade

```
TO INSETTI
   PREPARAZIONE
   MOVIMENTO
END

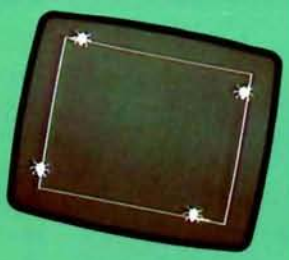
TO PREPARAZIONE
   DRAW
   FULLSCREEN
   TELL 0
   HT
   PU
   SETXY (—100) (—100)
   QUADRATO 200
   POSIZIONE 1 (—100) (—100)
   POSIZIONE 2 (—100) 100
   POSIZIONE 3 100 100
   POSIZIONE 4 100 (—100)
END

TO QUADRATO :LATO
   PD
   REPEAT 4[FD :LATO RT 90]
   PU
END

TO POSIZIONE :NUM :X :Y
   TELL :NUM
   SETSHAPE 3
   PU
   SETXY :X :Y
   PD
   ST
END

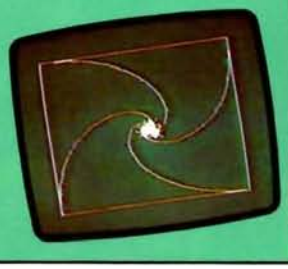
TO MOVIMENTO
   SEGUI 1 2
   SEGUI 2 3
   SEGUI 3 4
   SEGUI 4 1
   MOVIMENTO
END

TO SEGUI :A :B
   TELL :B
   MAKE "X XCOR
   MAKE "Y YCOR
   TELL :A
   SETH TOWARDS :X :Y
   FD 10
END
```









IAN McKINNELL

ESERCIZIO CON TRE INSETTI

Scrivere un programma LOGO per risolvere un problema matematico simile al precedente, ma con tre insetti disposti agli angoli di un triangolo

PROGRAMMA DEL MODULO LUNARE

Il seguente programma è una delle possibili soluzioni per l'esercizio a pag. 1135. Digitare ALLUNAGGIO, per dare il via a questa semplice versione del gioco:

```
TO ALLUNAGGIO
   PREPARAZIONE AZIONE
END
TO PREPARAZIONE
   DRAW TRACCIA.PIATTAFORMA MESSA.A PUNTO.RAZZO
END
TO TRACCIA.PIATTAFORMA
   PU SETXY (—20) (—60) PD SETXY 20 (—60)
   PU
END
TO MESSA.A.PUNTO.RAZZO
   SETXY 0 120 MAKE "VEL 0 MAKE "CARB 50
END
TO AZIONE
   COMANDO SPOSTAMENTO
   IF YCOR<—53 THEN ESPLOSIONE STOP
   GRAVITÀ CARBURANTE.RESIDUO AZIONE
END
TO COMANDO
   IF TASTO="F THEN ACCENSIONE
END
TO TASTO
   IF RC? THEN OUTPUT RC
   OUTPUT"
END
TO ACCENSIONE
   IF :CARB0>0 THEN MAKE "VEL :VEL+0.5 MAKE
   "CARB :CARB—1
END
TO SPOSTAMENTO
   SETY YCOR+:VEL
END
TO ESPLOSIONE
   IF :VEL>(—1) THEN PRINT [ALLUNAGGIO EFFETTUATO.
   COMPLIMENTI!] STOP
   PRINT [L'IMPATTO HA UCCISO L'INTERO EQUIPAG-
   GIO!]
END
TO GRAVITÀ
   MAKE "VEL :VEL—0.2
END
TO CARBURANTE.RESIDUO
   (PRINT "CARB :CARB)
END
```

QUALCOSA IN PIÙ...

Standard MSX

CPU	**Z80A, 3,85 MHz**
RAM	Minimo 8 Kbyte
ROM	32 Kbyte, compreso il BASIC
SCHERMO	16 colori, video grafico a 256x192 pixel, 32 sprite, video per testi 40 x 24 (oppure 32 x 24) (chip video TI 9918 o equivalente)
SUONO	Tre canali accessibili dal BASIC (chip generatore di suoni AY38910)
INTERFACCE	Porta MSX per cartuccia, output modulato per TV, stampante parallela Centronics, interfaccia per registratore a cassette
TASTIERA	Tastiera QWERTY, più tasti di funzione, 4 tasti per il cursore, 10 tasti di funzione programmabili

MSX & MSX

HIT-BIT SONY	Database incorporato, output RGB, "Ram Packs" opzionali da 4 Kbyte
TOSHIBA HX-10	Bus d'espansione, 2 porte per joystick
YAMAHA CX-5	Mini-tastiera musicale e software con MIDI
PIONEER	Interfaccia di controllo per unità a video-dischi
SANYO MPC100	Penna luminosa opzionale e software
JVC HC7GB	Output RGB
SPECTRAVIDEO SVI 728	Tastierina numerica completa

Anche se la normativa MSX richiede un minimo di 8 Kbyte di memoria, tutti i fabbricanti hanno dotato i loro apparecchi di 64K di RAM per l'utente e di 16K di RAM per il video

del gioco come ad esempio, muovere gli alieni sullo schermo e verificare se è stato premuto il pulsante di fuoco. Deve quindi eseguire due operazioni contemporaneamente e passare rapidamente da una all'altra.

Per far ciò il BASIC MSX definisce alcune situazioni come "eventi" e, usando alcune istruzioni, informa il computer; quando tali eventi si verificano, questo passa automaticamente ad una subroutine per la gestione di quella particolare situazione.

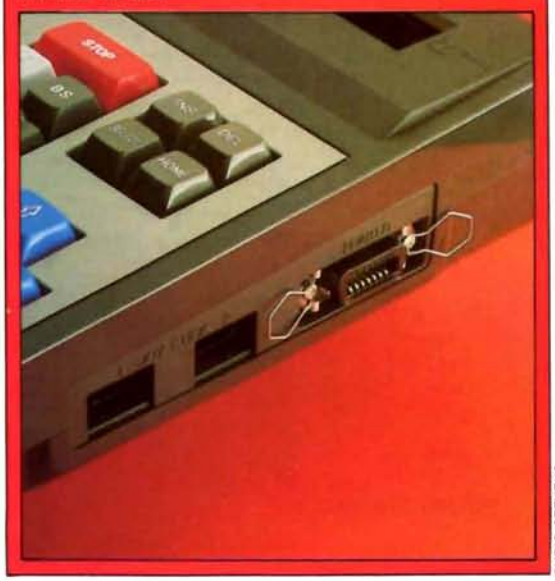
Il video grafico degli apparecchi MSX consente di visualizzare 16 colori con una risoluzione di 256 x 192 pixel. Inoltre, è possibile creare fino a 32 sprite di 8 x 8 pixel (o 16 sprite di 16 x 16 pixel oppure 8 sprite di 32 x 32 pixel). Per il loro controllo, il BASIC MSX offre una gamma completa di appositi comandi. Tra questi, SPRITE e PUT SPRITE usati, rispettivamente, per creare e posizionare uno sprite sullo schermo.

Gli apparecchi MSX già oggi possono contare su un discreto numero di cartucce software. La compatibilità tra i diversi apparecchi, propagandata dai fabbricanti MSX, è una realtà e permette di utilizzare il software prodotto per il Toshiba HX-10 sull'Hit-Bit e viceversa. Questa compatibilità vale tanto per il software su cartuccia quanto per i programmi su cassetta. Grazie a ciò, le compagnie che producono apparecchi MSX sperano di poter rendere disponibile, entro breve tempo, una vasta scelta di applicazioni software, adatte a tutti gli apparecchi.

Resta tuttavia da vedere se l'MSX avrà il successo di mercato che i giapponesi si augurano: già si profila un'accanita concorrenza da parte di case come la Sinclair, la Commodore e la Amstrad. Comunque, il livello qualitativo degli apparecchi MSX è quello dichiarato dai loro fabbricanti: sono ben corredati, facili da usare e di prezzo ragionevole.



Interfacce standard



CHRIS STEVENS

TOSHIBA HX-10 MSX

DIMENSIONI

mm. 365 x 245 x 60

CPU

Z80A, frequenza di lavoro a 3.58 MHz

MEMORIA

64 Kbyte di RAM (di cui 28 Kbyte a disposizione del BASIC) 16 Kbyte di RAM per il video, 32 Kbyte di ROM (BASIC incluso)

VIDEO

24 righe di 40 caratteri. Risoluzione grafica: 256 x 192 pixel, con 16 colori e fino a 32 sprite

INTERFACCE

Per stampante Centronics, per TV, per monitor, uscita audio, due porte per joystick, porta per registratore, connettore per cartucce, bus d'espansione

LINGUAGGI FORNITI

Extended BASIC della Microsoft

TASTIERA

Tipo QWERTY, con 68 tasti, di cui 4 per il cursore, più 5 di funzione (programmabili)

DOCUMENTAZIONE

I due manuali (uno per l'installazione e l'altro per il BASIC) sono abbastanza buoni, anche se piuttosto superficiali

PRO

Il BASIC MSX possiede molte interessanti doti, specialmente per il controllo della grafica e del suono. Trattandosi di un linguaggio standard, è logico attendersi una discreta disponibilità di software interscambiabile

CONTRO

Il BASIC MSX è carente per quanto concerne la programmazione strutturata. La disponibilità di periferiche MSX è per il momento molto limitata

DIFFERENZE

L'Hit-Bit della Sony possiede 3 programma applicativi nella ROM un'uscita monitor RGB. La configurazione della tastiera è leggermente diversa

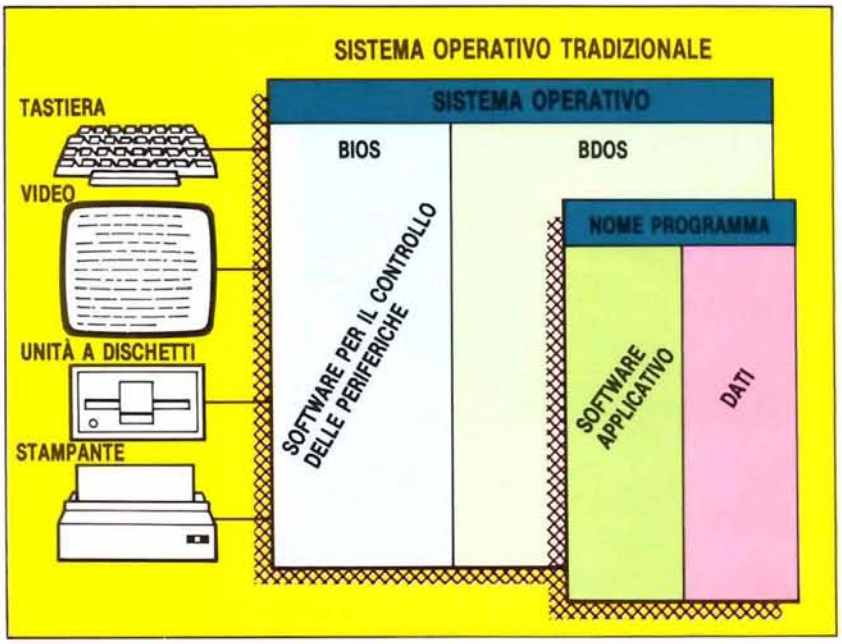
Un maggior controllo

Un metodo più versatile per realizzare package integrati

Un modo per sviluppare software integrato, in modo diverso da quello visto precedentemente, si affida al sistema operativo del computer per fissare i meccanismi di base dell'integrazione ed usa programmi indipendenti appositamente sviluppati che vengono combinati e fatti lavorare assieme sotto il controllo del sistema.
La progettazione di un simile sistema operativo è piuttosto complessa ed il computer che ne è destinatario deve possedere un hardware ed un software molto sofisticati. La Apple è all'avanguardia in questo campo con i modelli Lisa e Macintosh, ma altre compagnie (la Microsoft ad esempio) stanno sviluppando sistemi integrati per apparecchi molto diffusi, quali il PC della IBM. I programmi per questi nuovi sistemi operativi sono molto diversi da quelli "tradizionali". Gran parte dei programmi, ossia le routine che ricevono comandi e informazioni dall'utente e visualizzano i risultati, sono dedicate all'interfaccia utente. Ci sono idee contrastanti sul modo in cui i programmi devono funzionare, perciò il software prodotto adotta procedure operative notevolmente diverse tra loro.
Un sistema operativo integrato possiede al proprio interno una serie di routine d'interfaccia utente dedicate a ciascuno dei suoi programmi operativi. Grazie a questa impostazione, tutti i programmi scritti per lavorare con quel sistema operativo hanno più o meno le stesse procedure; quindi basta imparare il funzionamento di un programma per essere in grado di usare tutti gli altri!

Funzionamento gerarchico

Con un sistema operativo tradizionale il programma in uso è interamente responsabile del sistema: esso controlla le immagini che compaiono sullo schermo, le unità a dischetti e le comunicazioni con la tastiera. Le sue istruzioni vengono trasferite al sistema operativo che, a sua volta, gestisce la componente hardware del sistema. L'esecuzione del programma si trova al vertice di una simbolica piramide la cui base poggia sul sistema operativo



Un'interfaccia per l'utente caratteristica di questi sistemi è il "mouse", uno strumento di controllo del cursore che serve per scegliere un'applicazione da un menu sullo schermo. In alternativa al mouse viene usato uno "schermo tattile", al cui interno si trova una matrice di fasci luminosi che risponde al tocco di un dito. Lo schermo è suddiviso in "finestre" indipendenti, corrispondenti ad applicazioni di vario tipo. Dal punto di vista tecnico, per un'interfaccia di questo tipo, occorre un processore veloce, una discreta quantità di memoria e grafica ad alta risoluzione. Ovviamente, queste caratteristiche comportano costi maggiori, ma permettono al sistema di lavorare con quasi tutti i programmi oggi sul mercato, lo rendono facile da usare e consentono all'utente di avere sullo schermo ed adoperare in modo combinato più applicazioni contemporaneamente.

Procedure di controllo

È importante capire come il sistema ed i programmi applicativi interagiscono tra loro. Programma ed utente non sono mai in contatto diretto, ma si scambiano informazioni solo tramite il sistema operativo, che svolge una costante funzione di controllo. In pratica, il programma applicativo diviene una vera e propria estensione del sistema operativo: il computer nel suo complesso si trasforma in un unico "ambiente integrato".
Nei sistemi tradizionali, le comunicazioni tra il programma ed il sistema operativo avvengono attraverso un solo canale; il programma esegue un particolare lavoro ed il sistema operativo ne consente l'esecuzione.
In un sistema integrato, il sistema operativo controlla il programma e gli invia, contemporaneamente, messaggi di vario genere. Ad esempio, il sistema operativo può inviare al programma informazioni di questo tipo: «Ridisegna l'immagine sullo schermo perché l'utente si è spostato sull'altro lato del video» o «Conserva tutte le informazioni: l'utente ha selezionato con il mouse una nuova applicazione» oppure «Esamina questi dati provenienti dallo spreadsheet». Il programma deve cioè essere in grado di rispondere sia alle domande del sistema operativo, sia a quelle provenienti da altre direzioni.
Quando tutte le componenti software dell'apparecchio raggiungono un simile livello di operazione è abbastanza facile creare un ambiente integrato. Ogni programma ha una propria finestra sullo schermo e, quando l'utente sposta il cursore controllato dal mou-

Il Toshiba HX-10

Anatomia di un microcomputer a diffusione europea: tecnologie a confronto

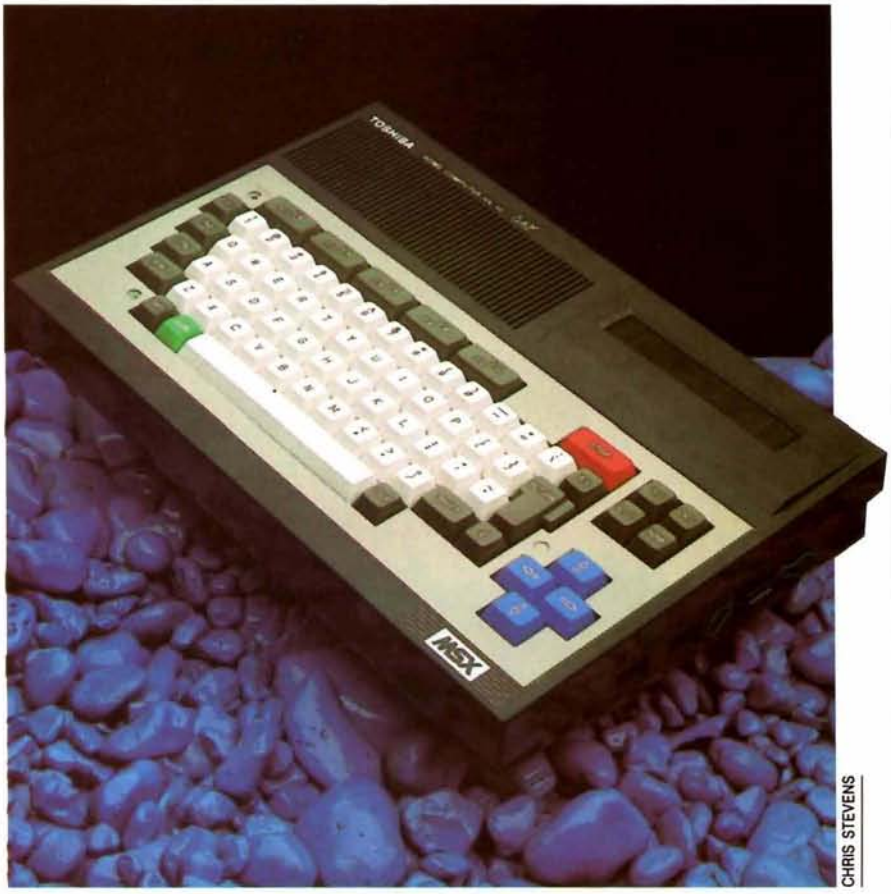
Lo standard MSX (vedere a pag. 621) richiede una serie di caratteristiche piuttosto precise: il tipo di CPU da usare (Z80), le dimensioni minime della ROM (32 Kbyte) e della RAM (8 Kbyte), il tipo di chip per la grafica e per il suono, il numero e la funzione dei tasti (anche se la loro disposizione può variare), il numero minimo di interfacce e le loro caratteristiche, il formato del video per la grafica e per i testi e, infine, il dialetto BASIC residente nelle ROM. È quindi logico aspettarsi che tutti gli apparecchi aderenti allo standard MSX siano tra loro simili. Tuttavia, i fabbricanti hanno la facoltà di variare la capacità della memoria (purché non scendano al di sotto del minimo fissato), il tipo di tastiera usata ed il numero di interfacce aggiuntive. In pratica, la Sony e la Toshiba, come molti altri produttori di apparecchi MSX, hanno ampiamente superato gli standard minimi imposti dalla normativa MSX.

L'Hit-Bit della Sony ed il Toshiba HX-10 hanno tastiere di buona qualità, anche se alcuni utenti le giudicano troppo "sensibili". Entrambi i micro possiedono una memoria centrale di 64 Kbyte ed una RAM addizionale di 16 Kbyte, dedicata al video. La memoria complessiva è quindi di 80 Kbyte, superiore a quella degli altri home computer. I modelli Sony e Toshiba hanno un'interfaccia standard per stampante Centronics ed una coppia di porte per joystick, che spesso, su altri home computer, sono accessori opzionali.

Originariamente i computer MSX dovevano essere apparecchi di basso costo, ma le fluttuazioni monetarie e la crescita dei costi di produzione hanno fatto lievitare i loro prezzi. Un altro fattore che ha inciso sui prezzi è stato l'impeto con cui le case giapponesi hanno cercato di lanciare i loro prodotti sul mercato europeo: le esportazioni della Toshiba, ad esempio, avvengono tutte per via aerea, con costi nettamente superiori alle spedizioni per mare. Inoltre, la compagnia si è affrettata ad adattare tutte le linee di produzione per realizzare una versione europea di apparecchi MSX nella speranza di essere tra le prime a commercializzarli in Europa.

All'accensione di un micro MSX, sull'ultima riga dello schermo compare una serie di parole: sono alcune delle parole chiave più usate nel linguaggio BASIC (quali RUN, CLOAD", LIST), che svolgono la funzione di "etichette" per i cinque tasti di funzione posti sopra la tastiera. Le funzioni di questi tasti vengono definite automaticamente al momento dell'accensione, ma si possono modificare col comando KEY. Ricorrendo all'uso simultaneo dello Shift e dei tasti funzione, vengono abilitate cinque nuove funzioni, per un totale di 10 parole chiave BASIC richiamabili in questo modo. Inoltre, il tasto Shift cambia l'ultima riga dello schermo, visualizzando la seconda serie di etichette associate ai tasti di funzione. Ogni etichetta può contenere un massimo di 15 caratteri, ma solo i primi sette compaiono sullo schermo.

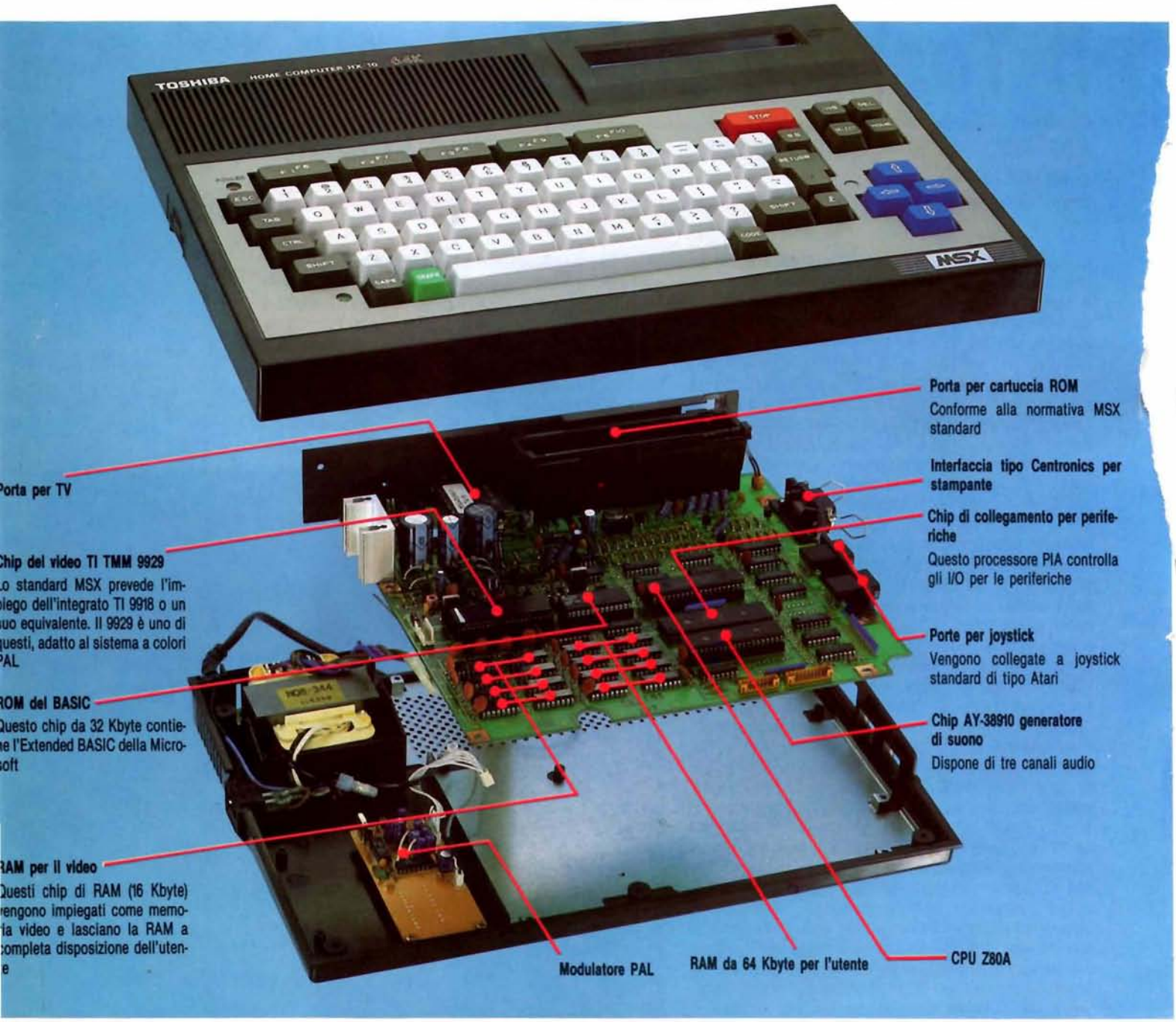
L'impiego combinato della tastiera e del programma di "screen editor" facilita notevol-



CHRIS STEVENS



Sistema Standard
Il Toshiba HX-10 ha due porte per joystick, un'interfaccia parallela per stampante di tipo Centronics ed una tastierina separata per il cursore. Il BASIC MSX usa gli stessi comandi per controllare tanto il joystick quanto il cursore, quindi i giochi scritti per un tipo di controllo possono essere tranquillamente usati anche con l'altro

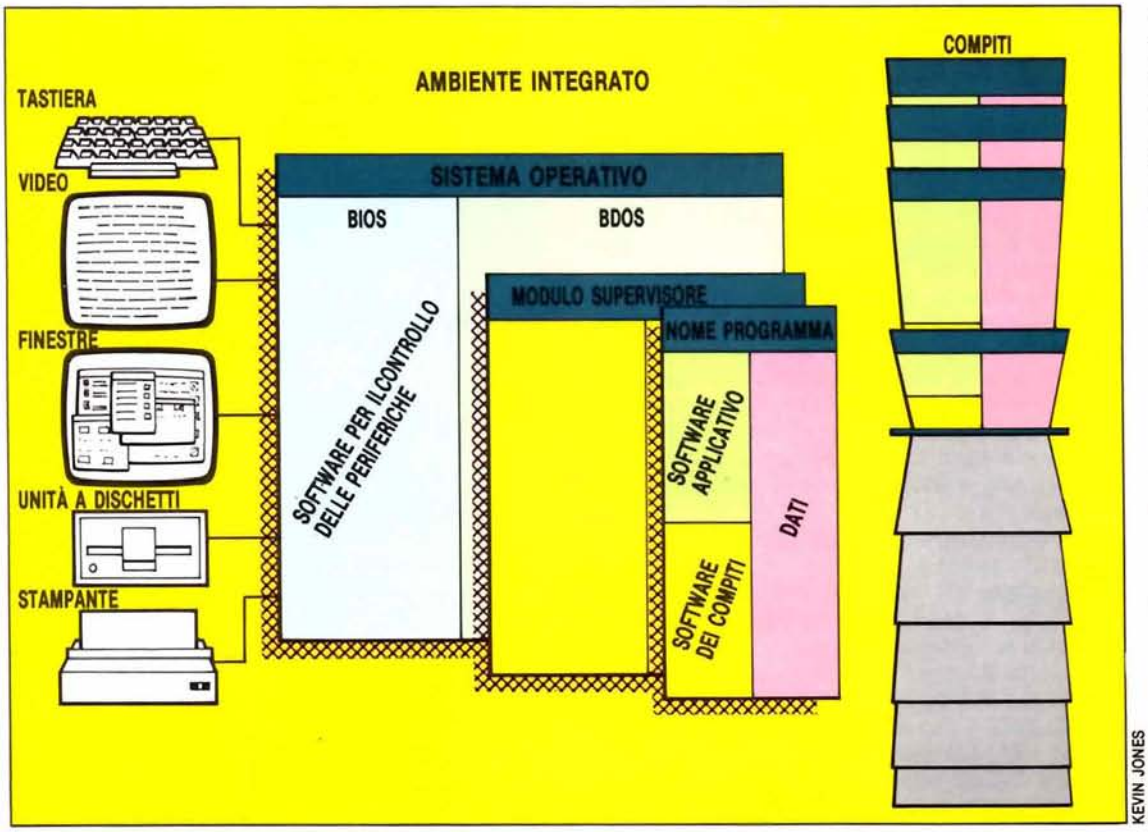


mente la redazione dei testi. Quattro tasti controllano i movimenti del cursore, consentendo di appotare correzioni in qualsiasi zona dello schermo, semplicemente sovrapponendo i caratteri nuovi a quelli che si vogliono cambiare. Le procedure di inserimento e di cancellazione richiedono quindi la pressione di un solo tasto. Sul Toshiba HX-10, i tasti per il cursore hanno le stesse dimensioni degli altri tasti mentre l'Hit-Bit della Sony presenta una tastierina quadrata composta da 4 grossi tasti.

Come l'hardware, anche il software MSX possiede alcune caratteristiche speciali. Il BASIC MSX possiede comandi come AUTO e RENUMber oltre ad altre istruzioni per la generazione dei suoni, la grafica e la gestione delle "interrupt". La grafica si basa su tre comandi principali: LINE traccia una linea tra due punti, ma se le sue coordinate sono seguite dalla lettera B (per box) traccia un quadrato. Se invece sono seguite dalle lettere FB, viene tracciato un quadrato, colorato all'interno. Il comando CIRCLE può essere usato per tracciare cerchi, ma anche ellissi ed archi. Il comando PAINT permette di colorare l'interno di figure chiuse anche molto complesse.

IL BASIC MSX possiede altre utili funzioni, ma il gruppo di comandi più interessanti è certamente quello per la gestione delle "interrupt". Queste istruzioni si dimostrano fondamentali nella programmazione della grafica ad alta velocità di visualizzazione. In particolari situazioni, ad esempio nei giochi del tipo *Space Invaders*, un programma deve svolgere un compito e, allo stesso tempo, controllare continuamente ciò che accade nello sviluppo

se su una finestra e seleziona l'applicazione, il sistema operativo richiama quel particolare programma e lo invia in esecuzione.
Se, ad esempio, l'utente muove il cursore su una finestra, seleziona l'opzione "trasferimento" e la sposta in una nuova posizione, le routine del sistema operativo eseguono il lavoro e, se necessario, informano il programma del cambiamento consentendogli così di modificare l'immagine sullo schermo. Se l'utente dirige il mouse su una nuova finestra, il programma precedente viene temporaneamente abbandonato ed il sistema operativo inizia a lavorare con la nuova applicazione. Dunque, basta muovere il mouse per passare da un'applicazione all'altra.
Come accade per i grandi programmi polivalenti (tutto-in-uno), nei sistemi integrati tutti i programmi e le informazioni presenti sullo schermo devono sempre essere caricati in memoria per poter essere utilizzati. I computer devono quindi possedere una notevole quantità di memoria: il Lisa, ad esempio, ha una memoria da 1 Megabyte, mentre il Macintosh ne ha una da 512 Kbyte. Nonostante ciò, il loro sistema operativo spesso è costretto a trascrivere e caricare dai dischetti programmi ed informazioni. Per rendere questi apparecchi più veloci, è necessario dotarli di un disco rigido (hard disk).
Per agevolare lo scambio di informazioni tra programmi, il sistema operativo usa una serie di routine interne. Quando si desidera trasferire un gruppo di dati da un programma ad un altro, il sistema operativo interrompe il primo programma ed avvia l'esecuzione del secondo. Quindi chiede all'applicazione che controlla il trasferimento di "leggere" ed elaborare l'informazione in arrivo. I canali di comunicazione possono essere attivati automaticamente, perciò quando si modificano i dati contenuti, ad esempio, in uno spreadsheet, viene automaticamente corretto anche il suo diagramma grafico.
Dato che i due programmi non possono essere eseguiti contemporaneamente, il sistema operativo porta a termine l'operazione "saltando" dall'uno all'altro.
Il Lisa della Apple applica una tecnica più sofisticata, in base alla quale i dati provenienti da un programma prima vengono "parcheggiati" (evento segnalato nella finestra simboleggiata da un fermacarte) e successivamente trasferiti al nuovo programma. L'informazione che ne controlla la redazione (formattazione) viene trasmessa assieme ai dati: in questo modo un grafico creato con il software per la grafica gestionale può essere trasferito in un altro programma pur mantenendo inalterato il suo formato.
Quello della Apple è il sistema più funzionale per la creazione di software integrato: permette di fondere e far lavorare assieme qualsiasi programma appartenente al sistema, di passare da un'applicazione all'altra e di scambiare tra loro informazioni. Purtroppo l'hardware occorrente è sofisticato e molto costoso, mentre assai limitata è la disponibilità del software necessario per questo tipo di integrazione.
Simili innovazioni tecnologiche impiegano sempre un certo tempo prima di divenire di uso comune. Il mouse e le finestre, ad esempio, sono stati sviluppati dai ricercatori della Xerox intorno al 1970, ma solo da poco sono comparsi sul mercato.



Operazioni combinate

In un sistema integrato, il sistema operativo viene potenziato dalla presenza di un modulo "supervisore", che considera il programma in uso ed i dati come "compiti" da pianificare ed elaborare e tratta il sistema operativo ad esso sottoposto come un semplice programma di supporto al sistema. Il modulo supervisore trasferisce i compiti da e nella memoria principale oppure li copia o li richiama dal dischetto a seconda delle esigenze dell'utente e delle necessità del lavoro al momento in esecuzione. Inoltre, esso trasferisce le informazioni da un'applicazione all'altra, mantenendo il loro formato originale. In realtà, il modulo supervisore non è altro che un "compito ad alta priorità", che opera in simbiosi con gli altri compiti

Semplici applicazioni

Dopo aver realizzato il dispositivo a relais, vediamo come utilizzarlo in pratica

La scatola del relais permette di controllare l'alimentazione di rete di tutti gli apparecchi ad essa collegati. In risposta ad un segnale di controllo, a basso voltaggio, il relais lascia passare o interrompe la corrente di rete inviata alla presa collegata alla scatola. La presa viene alimentata solo quando al relais viene applicata una tensione a basso voltaggio. Il dispositivo a relais può essere quindi collegato direttamente al box di uscita a basso voltaggio, realizzato in precedenza (vedere a pag. 1054). Esiste una corrispondenza diretta tra la corrente trasmessa dalla scatola del relais e quella di basso voltaggio inviata al relais dal box di uscita. Per controllare comuni apparecchi domestici (a tensione di rete), si possono quindi impiegare le stesse tecniche software usate per guidare strumenti a basso voltaggio.

Se, ad esempio, i terminali a basso voltaggio del relais vengono collegati ai due poli della linea 0 del box di uscita e i terminali di commutazione ad una presa di rete, allora la corrente di rete raggiunge la presa sopra la scatola del relais quando il bit 0 del registro dati, nella porta utente, è posto ad 1. Viceversa, quando il bit 0 è a zero, la corrente di rete non raggiunge la presa collegata al relais. Il box di uscita a basso voltaggio può accogliere ed azionare nel modo descritto fino a 4 dispositivi a relais.

Questo semplice circuito di commutazione può essere impiegato per creare vari sistemi di controllo per comuni elettrodomestici. Come primo esercizio, proviamo a programmare il nostro computer in modo che risponda alla pressione di un pulsante con frasi registrate su un registratore a nastro.

Innanzi tutto si devono registrare una serie di frasi del tipo: «Stai schiacciando il mio pulsante!», seguita da «L'hai fatto di nuovo!» e «Attento, ti ho già avvisato!». Quindi si devono collegare il pulsante ed il registratore alla porta utente e scrivere un programma che faccia pronunziare al registratore le frasi, una alla volta, quando il pulsante viene premuto.

Si devono creare i seguenti collegamenti per la porta utente:

1) innestare gli spinotti del relais nei poli positivo e negativo della linea 0 del box di uscita a basso voltaggio;
2) inserire la spina d'alimentazione della scatola del relais nella presa di rete;
3) collegare i due cavetti provenienti dal pulsante ai poli positivo e negativo della linea 7 del buffer box;

Il problema principale che dobbiamo affrontare, nello sviluppo del software di controllo è quello di assicurarsi che il registratore venga acceso e spento con la massima precisione quando deve trasmettere un messaggio. Quindi, prima di scrivere il programma, dobbiamo cronometrare la durata di ciascuna frase ed inserire questi dati nel programma di controllo. Se le frasi registrate sono tre e durano rispettivamente T(1), T(2) e T(3) secondi, scriveremo un programma che, dopo la pressione del pulsante, accenda il registratore per il tempo necessario ad emettere una particolare frase. Se la durata delle frasi è stata misurata con precisione, ciascuna verrà pronunciata appena il registratore viene acceso.

I seguenti programmi (per il Commodore 64 e il Micro BBC) accendono il registratore per tre intervalli di tempo successivi in risposta al segnale proveniente dal pulsante. Le tre variabili T(1), T(2) e T(3) devono contenere i valori di durata delle rispettive frasi registrate.

MICRO BBC

```
10 REM CONTROLLO REGISTRATORE - BBC
20 DIM T(3)
30 DDR=&FE62:REGDAT=&FE60
40 ?DDR=127:REM LINEA 7 IN INPUT
50 ?REGDAT=0:REM TUTTO SPENTO
60 CLS
70 FOR I=1 TO 3
80 INPUT "INTERVALLO DI TEMPO (SEC.)";T(I)
90 NEXT I
100 :
110 FOR L=1 TO 3
120 CLS
130 REPEAT
140 UNTIL (?REGDAT AND 128)=0: REM L7 A 0
150 ?REGDAT=1: REM ACCENDE IL REGISTRATORE
160 TIME=0: REM AVVIA IL TEMPORIZZATORE
170 REPEAT
180 UNTIL TIME > T(L)*100
190 ?REGDAT=0: REM SPEGNE IL REGISTRATORE
200 NEXT L
210 END
```

COMMODORE 64

```
10 REM CONTROLLO REGISTRATORE - CBM 64
20 DDR=56579:REGDAT=56577
30 POKE DDR,127: REM LINEA 7 IN INPUT
40 POKE REGDAT,0: REM TUTTO SPENTO
50 PRINT CHR$(147): REM CANCELLA LO SCHERMO
60 FOR I=1 TO 3
70 INPUT "INTERVALLO DI TEMPO (SEC.)";T(I)
80 NEXT I
90 :
100 FOR L=1 TO 3
110 IF(PEEK(REGDAT)AND128)<>0 THEN 110
115 POKE REGDAT,1: REM ACCENDE IL
    REGISTRATORE
120 T=TI: REM INIZIALIZZA IL TEMPORIZZATORE
130 IF T(L)>(TI-T)/60 THEN 130
140 POKE REGDAT,0: SPEGNE IL REGISTRATORE
150 NEXT L
160 END
```

Sveglia programmabile

In questo secondo esercizio cerchiamo di trasformare il nostro micro in una sveglia programmabile. Entrambe le versioni del programma di controllo (per il Commodore 64 ed il Micro BBC), permettono all'utente di inserire:

1) l'ora del giorno;
2) il numero degli intervalli di "snooze" (o "pisolino") durante i quali viene emessa musica o un "cicalino";
3) se durante un intervallo si vuole silenzio oppure l'emissione di musica o di un cicalino;
4) se durante l'intervallo deve essere accesa una luce;
5) l'orario di sveglia desiderato.

I collegamenti tra gli apparecchi usati ed il box d'uscita di basso voltaggio sono i seguenti:

1) un registratore viene collegato alla linea 0 attraverso un relais di rete;
2) una lampada da tavolo viene collegata alla linea 1 mediante un relais per la linea 1;
3) un campanello (o un cicalino) elettrico, a 9 Volt, viene collegata, direttamente alla linea 3.

Il programma calcola, in base all'orario di sveglia e agli intervalli di snooze programmati, il momento iniziale di ogni intervallo. Tutte le informazioni riguardanti il funzionamento dei vari apparecchi durante un intervallo vengono memorizzate in matrici del programma. Si noti che gli elementi di matrice ricevono valori che corrispondono al valore richiesto affinché il bit all'interno del registro dati accenda uno degli apparecchi del sistema. Usando una semplice OR logica, possiamo calcolare la configurazione complessiva da inserire nel registro dati per azionare una particolare combinazione di apparecchi.
Gran parte della programmazione è dedicata alla gestione delle variabili stringa per l'esecuzione dei calcoli numerici. Ciò vale soprattutto per la versione del Commodore 64, perché il suo dialetto BASIC è privo dei comandi MOD e DIV presenti invece sul Micro BBC.
Si è così realizzato un flessibile sistema di input/output per microcomputer, che consente di controllare una serie di LED e di apparecchi a basso ed alto voltaggio. Inoltre, questo sistema permette al micro di ricevere ed interpretare le informazioni provenienti da sensori di diverso tipo. Nell'esempio illustrato, il computer svolge le funzioni di un sofisticato timer programmabile. Applicazioni di diverso genere possono accendere e spegnere segnalatori d'incendio in risposta ad una coppia di sensori per il calore oppure accendere una luce quando fa buio. Le applicazioni possibili sono praticamente infinite!



IAN McKINNELL

COMMODORE 64

```
100 REM ** SVEGLIA PROGRAMMABILE CBM 64 **
110 DDR=56579:REGDAT=56577
120 POKE DDR,255:POKE REGDAT,0
130 PRINT CHR$(147):REM CANCELLA LO SCHERMO
140 INPUT "N. INTERVALLI DI 'SNOOZE'";N
150 M=N+1
160 DIM A(M),M(M),L(M),TS$(M),T(M)
170 :
180 REM ** PARAMETRI D'INTERVALLO **
190 FOR C=1 TO N
200 PRINT: PRINT "N. D'INTERVALLO";C
210 INPUT "MUSICA, ALLARME O SILENZIO
    (M/A/S)";AN$
215 AN$=LEFT$(AN$,1)
220 IF AN$<>"M"ANDAN$<>"A"ANDAN$<>"S"
    THEN 210
230 IF AN$="M" THEN M(C)=1:A(C)=0
240 IF AN$="A" THEN A(C)=8:M(C)=0
250 IF AN$="S" THEN A(C)=0:M(C)=0
260 INPUT "LUCE ACCESA (S/N)";AL$
270 L$=LEFT$(AL$,1)
280 IF L$<>"S" AND L$<>"N" THEN 260
290 IF L$="S" THEN L(C)=2: GOTO 310
300 L(C)=0
310 INPUT "DURATA INTERVALLO (MIN.)";T(C)
320 NEXT C
330 :
340 INPUT "ORARIO DI SVEGLIA (HHMM)";LT$
350 LT$=LT$+"00": REM AGGIUNGE I SECONDI
360 TS$(N+1)=LT$: REM ULTIMO ORARIO
370 REM TRASFORMA L'ULTIMO ORARIO IN MINUTI
380 LM=60*VAL(LEFT$(LT$,2))+VAL(MID$(LT$,3,2))
390 :
400 INPUT "REGOLAZIONE DELL'ORA (HHMM)";TN$
410 TI$=TN$+"00": REM AVVIA IL TEMPORIZZATORE
420 :
430 REM ** ANALISI E CALCOLO **
440 REM ** CALCOLA ORARIO INIZIO INTERVALLO **
450 FOR C=N TO 1 STEP -1
460 LM=LM-T(C):REM ORARIO D'INIZIO IN MINUTI
470 HR=INT(LM/60)
480 MN=INT(60*(LM/60-HR+.000001))
490 HR$=STR$(HR): REM ORE
500 MN$=STR$(MN): REM MINUTI
510 MN$=MID$(MN$,2,LEN(MN$))
520 HR$=MID$(HR$,2,LEN(HR$))
530 REM ** AGGIUNGE ZERO NON SIGNIFICATIVI **
540 SP$="00"
550 HR$=LEFT$(SP$,2-LEN(HR$))+HR$
560 MN$=LEFT$(SP$,2-LEN(MN$))+MN$
570 TS$(C)=HR$+MN$+"00"
580 NEXT C
590 :
600 REM ** ACCENSIONE DEL SISTEMA **
610 PRINTCHR$(147)
620 FOR C=1 TO N+1
630 IF TI$<TS$(C)THENGOSUB710:GOTO630
640 DN=M(C) OR A(C) OR L(C):REM DATI PER
    REGDAT
650 POKE REGDAT,DN
670 NEXT C
680 POKE REGDAT,0
690 END
700 :
710 REM ** S/R VISUALIZZAZ. TEMPORIZZATORE **
720 PRINTCHR$(145);:REM CURSORE IN SU
730 PRINTLEFT$(TI$,2);":";MID$(TI$,3,2);
740 PRINT" ";RIGHT$(TI$,2)
750 RETURN
```

MICRO BBC

```
 10 REM **** SVEGLIA PROGRAMMABILE BBC ****
 15 MODE7
 20 DDR=&FE62:REGDAT=&FE60
 30 CLS
 40 INPUT "NUMERO INTERVALLI DI 'SNOOZE'";N
 45 M=N+1
 50 DIM A(M),M(M),L(M),T(M),TS(M)
 70 REM **** PARAMETRI D'INTERVALLO ****
 80 FORC=1 TO N
 90   PRINT "N. D'INTERVALLO";C
 96   REPEAT
100      PRINT "MUSICA, ALLARME O SILENZIO";
103      INPUT " (M/A/S)";AN$
105    AN$=LEFT$(AN$,1)
110    UNTIL AN$="M" ORAN$="A"ORAN$="S"
120  IF AN$="M"THEN M(C)=1:A(C)=0
130  IF AN$="A"THEN M(C)=0:A(C)=8
140  IF AN$="S"THEN M(C)=0:A(C)=0
150  REPEAT
160    INPUT "LUCE ACCESA (S/N)";AL$
170    AL$=LEFT$(AL$,1)
180    UNTIL AL$="S" OR AL$= "N"
190  IF AL$="S" THEN L(C)=2 ELSE L(C)=0
200  INPUT "INTERVALLO DI TEMPO (MIN)";T(C)
210  NEXT C
220  :
230 INPUT "ORARIO DI SVEGLIA (HHMM)";LT$
232  TS(N+1)=6000*(60*VAL(LEFT$(LT$,2)))
234    TS(N+1)=TS(N+1)+VAL(RIGHT$(LT$,2))
236 REM TRASFORMA L'ULTIMO ORARIO IN MINUTI
237 LM=60*VAL(LEFT$(LT$,2))
239  LM=LM+VAL(RIGHT$(LT$,2))
240 INPUT "CHE ORE SONO ADESSO (HHMM)";TN$
250  TIME=6000*(60*VAL(LEFT$(TN$,2))
255    TIME=TIME+VAL(RIGHT$(TN$,2))
260 :
270 REM ANALISI E CALCOLO
280 FORC=N TO 1 STEP -1
290   LM=LM-T(C): REM INIZIO INTERVALLO
300   TS(C)=6000*LM
310    NEXT C
320 :
330  REM **** ACCENSIONE DEL SISTEMA ****
340 CLS
350 FOR C=1 TO N+1
360   REPEAT
370     PROCtemporizzatore
380     UNTIL TIME>=TS(C)
390   REGDATA=M(C) OR A(C) OR L(C)
400   ?REGDAT=REGDATA
420   NEXT C
430 ?REGDAT=0
440 END
999 :
1000 DEF PROCtemporizzatore
1020 MIN=(TIME DIV 6000) MOD 60
1030  HR=(TIME DIV 6000) MOD 60
1040 MIN$=STR$(MIN):HR$=STR$(HR)
1042 REM AGGIUNGE ZERO NON SIGNIFICATIVI
1043 SP$="00"
1044 HR$=LEFT$(SP$,2-LEN(HR$))+HR$
1045 MIN$=LEFT$(SP$,2-LEN(MIN$))+MIN$
1050 PRINTTAB(18,12)HR$;":";MIN$
1060 ENDPROC
```

DATABASE

Quinta scheda delle istruzioni per lo Z80 della Zilog. Istruzioni aritmetiche e di controllo per la CPU

Istruzioni aritmetiche

Aggiustamento decimale dell'Accumulatore, 'DAA'	27
Collegamento dell'Accumulatore, 'CPL'	2F
Negazione dell'Accumulatore, 'NEG' (Complemento a 2)	ED 44
Complemento del flag di Carry, 'CCF'	3F
Attivazione del flag di Carry, 'SCF'	37

Istruzioni di controllo sulla CPU

Nessuna operazione, 'NOP'	00	
Blocco della CPU, 'HALT'	76	
Disabilitazione interruzioni, 'DI'	F3	
Abilitazione interruzioni, 'EI'	FB	
Selezione Modo d'interruzione 0, 'IM 0'	ED 46	Varianti 8080A
Selezione Modo d'interruzione 1, 'IM 1'	ED 56	Chiamata alla locazione 38_H
Selezione Modo d'interruzione 2, 'IM 2'	ED 5E	Chiamata indiretta usando come puntatore il registro 1 e 8 bit forniti dal dispositivo di interruzione

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
DAA	Converte il contenuto dell'Acc. nel formato BCD dopo un'operazione aritmetica tra due operandi BCD	↕	↕	X	↕	X	P	•	↕	00 100 111	27	1	1	4	Aggiustamento decimale dell'Accumulatore, 'DAA'
CPL	$A \leftarrow \overline{A}$	•	•	X	1	X	•	1	•	00 101 111	2F	1	1	4	Complemento dell'Accumulatore (complemento a 1)
NEG	A←0−A	↕	↕	X	↕	X	V	1	↕	11 101 101 01 000 100	ED 44	2	2	8	Negazione dell'Accumulatore (Complemento a 2)
CCF	$CY \leftarrow \overline{CY}$	•	•	X	X	X	•	0	↕	00 111 111	3F	1	1	4	Complemento del flag di Carry
SCF	CY←1	•	•	X	0	X	•	0	1	00 110 111	37	1	1	4	Attiva il flag di Carry (Flag di Carry = 1)
NOP	Nessuna operazione	•	•	X	•	X	•	•	•	00 000 000	00	1	1	4	
HALT	Blocco della CPU	•	•	X	•	X	•	•	•	01 110 110	76	1	1	4	
DI ★	IFF←0	•	•	X	•	X	•	•	•	11 110 011	F3	1	1	4	
EI ★	IFF←1	•	•	X	•	X	•	•	•	11 111 011	FB	1	1	4	
IM 0	Selezione modo d'interr. 0	•	•	X	•	X	•	•	•	11 101 101 01 000 110	ED 46	2	2	8	
IM 1	Selezione modo d'interr. 1	•	•	X	•	X	•	•	•	11 101 101 01 010 110	ED 56	2	2	8	
IM 2	Selezione modo d'interr. 2	•	•	X	•	X	•	•	•	11 101 101 01 011 110	ED 5E	2	2	8	

NOTE:

IFF indica il flip-flop di abilitazione delle interruzioni
CY indica il flip-flop di Carry (riporto)
★ indica che le interruzioni non vengono campionate alla fine di EI e DI

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

per concessione della Zilog Inc.

Ingresso e uscita dal 6809

Il funzionamento di due chip di I/O per il microprocessore 6809

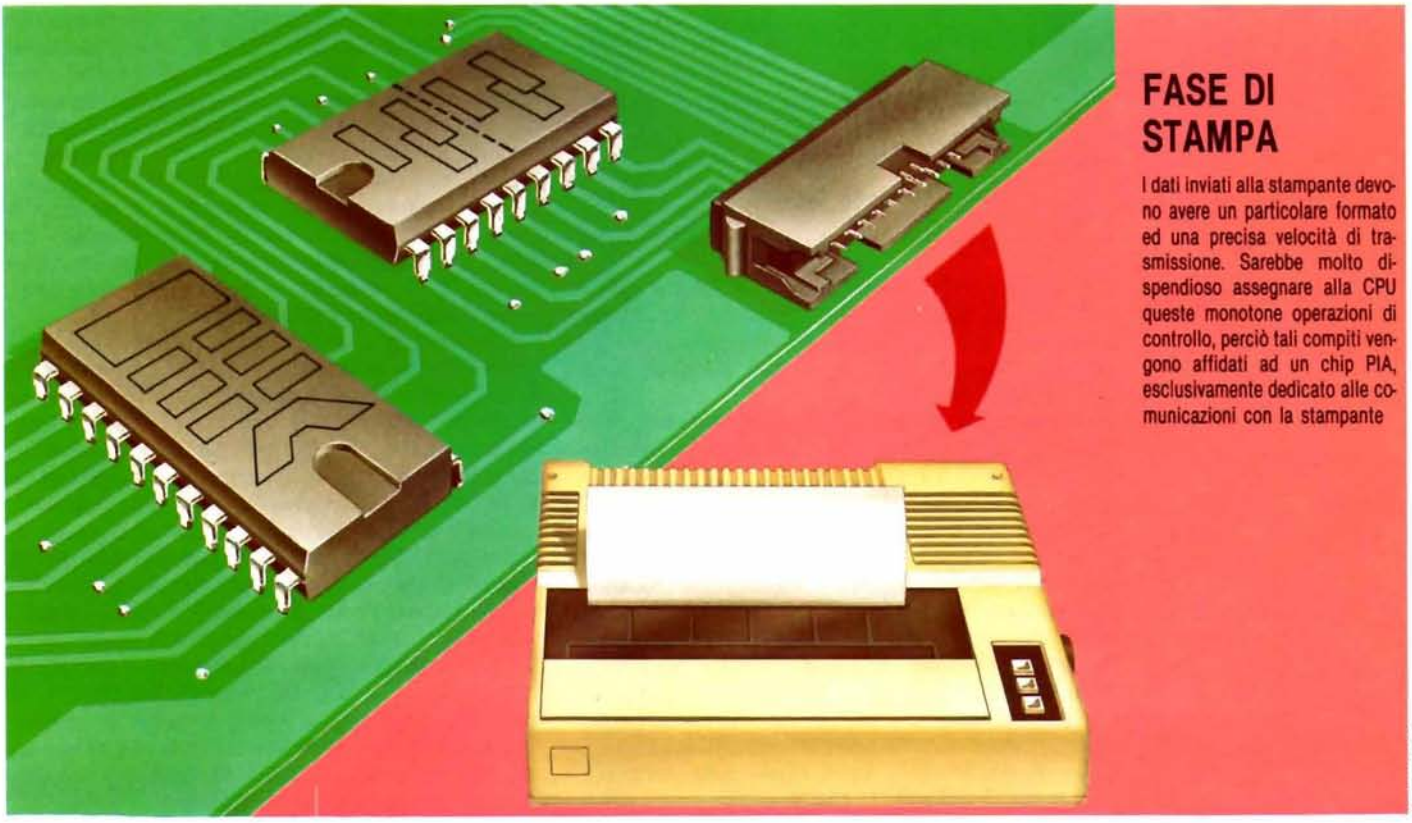
Il processore 6809, come il 6502 (ma diversamente dallo Z80), non possiede uno spazio d'indirizzo di I/O indipendente, né speciali istruzioni di I/O. Al contrario, i chip di I/O d'interfaccia occupano un comune indirizzo di memoria e l'accesso avviene tramite le normali istruzioni destinate alla memoria. Ciò facilita le operazioni di input/output, ma occupa blocchi di memoria inutilizzabili come normale RAM. Per cui il 6809, pur possedendo un bus d'indirizzo a 16 bit in grado di indirizzare direttamente 64 Kbyte di memoria, può al massimo disporre di circa 56 Kbyte (essendo i restanti 8K occupati dai chip di I/O).

Alcune periferiche di input/output possono essere collegate direttamente al bus dati, ma abitualmente vengono interfacciate con sofisticati chip di I/O, di complessità pari a quella dello stesso microprocessore. Di solito, si adoperano chip della stessa "famiglia" del microprocessore, perché è più facile collegarli e controllarli. I due chip più usati con il 6809 sono il 6802 (o 6821) PIA (Peripheral Interface Adaptor), che controlla ingressi/uscite paralleli ed il 6850 ACIA (Asynchronous Communications Interface Adaptor) che si occupa degli I/O seriali. Entrambi possiedono un certo numero di registri e le loro funzioni di controllo si basano sulla lettura e scrittura dei contenuti di questi registri, trattati come normali locazioni di memoria. I registri sono di tre tipi:

- Registri di controllo (di sola scrittura): i valori in essi immagazzinati permettono di programmare il chip per svolgere particolari funzioni. Ad esempio: per la definizione della velocità di trasmissione.
- Registri di stato (di sola lettura): i loro valori forniscono informazioni sullo stato del chip. Per esempio, possono segnalare se il processore ha ricevuto un input o se ha trasmesso l'ultimo valore in output, oppure se si è verificato un errore.
- Registri dati (di lettura-scrittura): possono venire letti e scritti separatamente e contengono i dati in ingresso o in uscita.

Per ottenere un risparmio di memoria, spesso più registri occupano uno stesso indirizzo. Ad esempio, un registro di stato ed uno di controllo possono avere lo stesso indirizzo e la comparsa dell'uno e dell'altro dipende dal tipo di chiamata (un'operazione di lettura o di scrittura) fatta sull'indirizzo. Uno stesso indirizzo può anche essere comune ad un registro per dati in input ed uno per dati in output.

Il 6820 PIA contiene 6 registri ed occupa in memoria 4 byte consecutivi. Il chip, in realtà, possiede due porte indipendenti, ciascuna delle quali usa tre registri. Il lato del chip



FASE DI STAMPA

I dati inviati alla stampante devono avere un particolare formato ed una precisa velocità di trasmissione. Sarebbe molto dispendioso assegnare alla CPU queste monotone operazioni di controllo, perciò tali compiti vengono affidati ad un chip PIA, esclusivamente dedicato alle comunicazioni con la stampante

KEVIN JONES

rivolto verso la periferica possiede 8 linee (o canali) per i dati e due linee di controllo per ciascuna porta. Le due linee di controllo collegate a quelle della periferica saranno usate per determinarne lo stato. Sulla linea di controllo 1, vengono trasmessi i segnali di controllo in ingresso, mentre la linea di controllo 2 può essere programmata per ricevere o trasmettere segnali di controllo.
I tre registri sono:

- Un registro dati: può lavorare tanto in input quanto in output, perché ogni bit può essere selezionato individualmente.
- Un registro di direzione dei dati: i suoi bit vengono usati per fissare la direzione dei dati, ponendo a 0 (input) o 1 (output) i bit corrispondenti a quelli del registro dati.
- Un registro con funzioni combinate di controllo/stato.

Il registro di direzione dei dati ed il registro dati hanno lo stesso indirizzo: l'accesso all'uno o all'altro è determinato dallo stato di uno dei bit del registro di controllo. (Si veda la tabella che mostra l'offset, rispetto all'indirizzo di base del chip, relativo ai vari registri). I bit del registro di controllo/stato hanno le seguenti funzioni:

Registro	Offset
Reg. Dati A	0
Reg. Direzione Dati A	0
Reg. Controllo/Stato	A 1
Reg. Dati B	2
Reg. Direzione Dati B	2
Reg. Controllo/Stato B	3

Bit	Funzione
7	Bit di stato per la linea di controllo 1. È posto ad 1 quando arriva un segnale di controllo ed automaticamente azzerato quando viene letto il registro dati
6	Bit di stato per la linea di controllo 2. Come il bit 7
5	Stabilisce se la linea di controllo 2 deve esser usata per l'input (0) o per l'output (1)
4	Definisce la natura del segnale di controllo sulla linea 2
3	Se la linea di controllo 2 è selezionata per l'input, un 1 in questo bit abilita le "interruzioni" (segnalate dal bit 6). Se invece è selezionata per l'output, partecipa alla definizione della natura del segnale
2	Seleziona tra il registro dati (1) ed il registro di direzione dei dati (0)
1	Definisce la natura del segnale di controllo sulla linea 1
0	Un 1 in questo bit abilita le interruzioni (segnalate dal bit 7)

Per il momento tralasciamo l'uso delle interruzioni e gli effetti prodotti dai bit 1 e 4. Si noti però che, quando interveniamo sul registro per selezionare i bit di controllo, è impossibile modificare i bit 6 e 7.
Il primo dei programmi d'esempio usa il chip 6820 per controllare una stampante mediante un'interfaccia Centronics standard. Quest'ultima comprende un certo numero di linee di controllo ed 8 linee dati. Una delle linee di controllo (chiamata strobe) viene usata per segnalare alla stampante l'invio di un carattere. La linea "strobe" viene collegata alla linea di controllo 2, che deve essere abilitata per l'output. Un'altra linea di controllo (detta "acknowledge") viene usata dalla stampante per segnalare che è pronta a ricevere un nuovo carattere: questa linea viene collegata alla linea di controllo 1. Le 8 linee dati vengono ovviamente collegate alle 8 uscite per i dati, previste dalla porta del chip PIA. Per mettere a punto la porta, prima occorre selezionare il registro di direzione dei dati e predisporre gli 8 bit in uscita, poi attivare il registro dati e selezionare per l'output la linea di controllo 2. Va poi eseguita una continua lettura del registro di controllo/stato, fino alla comparsa di un 1 nel bit 7: ciò indica che la stampante è pronta a ricevere un carattere. A questo punto si può depositare un carattere nel registro dati, che automaticamente trasmette un segnale sulla linea di controllo 2. Dopo l'invio del carattere, il bit 6 viene posto ad 1. Infine si deve leggere il registro dati per azzerare i bit 6 e 7 e ripetere l'intero processo per tutti i caratteri. La trasmissione e la ricezione dei segnali di controllo fra il processore e la periferica è chiamata "handshaking" (a "stretta di mano") o procedura di sincronizzazione.
Poniamo che l'indirizzo di base del PIA sia contenuto in una tabella di indirizzi situata alla locazione $3000. All'ingresso della subroutine di stampa, il registro A del processore contiene l'indice per la tabella, mentre Y contiene l'indirizzo della stringa da stampare, memorizzata secondo il consueto formato (il primo byte indica la lunghezza della stringa). Il programma prevede due subroutine: la prima mette a punto la porta, la seconda stampa la stringa.

Registro	Offset
Registro di Controllo	0
Registro di Stato	0
Trasmissione Dati	A 1
Ricezione Dati	2

Il 6850 ACIA

Il 6850 ACIA è un chip UART (Universal Asynchronous Receiver/Transmitter) impiegato per trasmissioni seriali. Di solito è usato il protocollo RS232 e un modem. Questo chip possiede 4 registri, occupa due indirizzi e, sul lato rivolto verso la periferica, prevede cinque collegamenti: due linee per la trasmissione e la ricezione dei dati e tre linee di controllo per le procedure di sincronizzazione, se necessarie. Due di queste linee, DCD (Data Carrier Detect, ossia rilevazione della portante) e CTS (Clear To Send, ossia trasmissione consentita), trasportano i segnali di controllo in ingresso, mentre la terza, RTS (Request To Send, ossia richiesta d'invio) è in uscita. Queste tre linee vengono collegate alle loro omonime della porta RS232 standard.
I nomi dei quattro registri ACIA sono dati nella tabella a margine pagina. Nel registro di controllo, il bit più significativo (7) viene usato per abilitare le interruzioni nella ricezione dei dati. I bit 5 e 6 abilitano o disabilitano le interruzioni di trasmissione e definiscono la natura del segnale di controllo trasmesso sulla linea RTS. I bit 2, 3 e 4 fissano le dimensioni del "blocco" di dati trasmesso. La trasmissione seriale di un byte di solito coinvolge almeno 10 bit: il primo è il bit di "start", che comunica all'apparecchio ricevente l'arrivo di un'informazione. Il dato vero e proprio è composto da 7 o 8 bit, seguiti da un bit di "parità", aggiuntivo, che permette di individuare gli errori di trasmissione. Ci possono poi essere uno o due bit di stop. La tabella illustra le combinazioni possibili:

Registro di controllo			Numero dei Bit Dati	Bit di Parità	Numero dei Bit di Stop
Bit 4	Bit 3	Bit 2			
0	0	0	7	Pari	2
0	0	1	7	Dispari	2
0	1	0	7	Pari	1
0	1	1	7	Dispari	1
1	0	0	8	(Non Usato)	2
1	0	1	8	(Non Usato)	1
1	1	0	8	Pari	1
1	1	1	8	Dispari	1

I due bit meno significativi (0 ed 1) vengono usati per fissare la velocità di trasmissione e di ricezione, calcolata predisponendo un divisore per la frequenza di clock. Il 6850 non ha un proprio orologio interno, quindi deve usarne uno esterno, in genere a 1,760 Hz.

Registro di controllo		Frequenza di clock Divisore
Bit 1	Bit 0	
0	0	1
0	1	16
1	0	64

Quando entrambi i bit sono ad 1, le combinazioni mostrate nella precedente tabella provocano il reset del chip (con azzeramento di tutte le selezioni). Nel registro di stato, i bit svolgono le seguenti funzioni:

Bit	Funzione
7	Richiesta d'interruzione
6	Ad 1 se c'è un errore di parità in ricezione
5	Ad 1 per un superamento di capacità (overflow): la periferica ha ricevuto troppi bit e gli ultimi caratteri hanno ricoperto quelli trasmessi in precedenza
4	Ad 1 se in ricezione si verifica un errore di inquadramento (numero errato dei bit di start e di stop)
3	Ad 1 quando viene ricevuto un segnale sulla linea CTS
2	Ad 1 quando viene ricevuto un segnale sulla linea DCD
1	Ad 1 quando il registro di trasmissione dati è vuoto
0	Ad 1 quando il registro di ricezione dati è pieno

Il secondo programma usa un chip 6850 per ricevere da un terminale a distanza una stringa che si conclude con un carattere di ritorno-carrello. Prima occorre programmare il chip in modo adeguato, quindi viene eseguito il loop per controllare se il registro dati ricevente è pieno. Se si preleva il byte ricevuto, si azzera il bit 0 del registro di stato. Questo procedimento viene ripetuto finché il carattere ricevuto non è un ritorno-carrello (cod. ASCII 13). Nel programma viene ignorato qualsiasi errore di trasmissione, anche se è abbastanza facile verificarne la presenza "mascherando" il contenuto del registro di stato. In tal modo possiamo controllare se uno dei bit che indicano la presenza di un errore si trova a livello alto. Il protocollo adottato è di uso piuttosto comune: 8 bit dati, nessun bit di parità, due bit di stop, divisione per 16 della velocità di clock. La prima subroutine programma il chip, la seconda riceve i dati.

PROGRAMMA PER CHIP PIA

```
TABEL   EQU   $3000          S/R messa a punto Porta A
        ORG   $1000
        ASLA                 Muove A a sin., ossia lo moltiplica per 2 (la tabella è composta da indirizzi a due byte)
        LDX   #TABEL         Legge l'indirizzo di base di PIA
        LDX   A,X            Accede al registro di direzione dei dati
        CLR   1,X            Seleziona tutti i bit per l'output
        LDB   #%11111111     Disabilita le interruzioni, seleziona la linea di controllo 2 per l'output ed il registro dati
        STB   ,X
        LDB   #%00101100
        STB   1,X
        RTS                  S/R per stampare la stringa il cui indirizzo si trova nel registro Y
        ASLA                 Muove A a sin. per moltip. per 2
        LDX   #TABEL         Legge l'indirizzo di base di PIA
        LDX   A,X            Lunghezza della stringa in A
        LDA   ,Y+            Controlla se la lunghezza è 0
CICL1   BEQ   FINITO         Verifica per bit successivo
CICL2   LDB   1,X            Maschera tutti i bit salvo il bit 7
        ANDB  #%10000000     Se non è pronto
        BEQ   CICL2          Carica il carattere successivo
        LDB   ,Y+            Lo stampa
        STB   ,X             Controlla se è stato trasmesso
CICL3   LDB   1,X            Esamina il bit 6
        ANDB  #%01000000     Gira in ciclo se non pronto
        BEQ   CICL3          Legge il registro dati per azzerare i bit di stato
        LDB   ,X
        DECA                 Sottrae uno dalla lunghezza
        BRA   CICL1          Passa al carattere successivo
FINITO  RTS
```

PROGRAMMA PER IL CHIP ACIA

```
TABEL   EQU   $3000          S/R per programmare il 6850
        ORG   $1000          S/R per inizializzare il chip ACIA
ACIAST  ASLA                 Muove A a sin., ossia lo moltiplica per 2
        LDX   #TABEL         Legge l'indirizzo di base di ACIA
LDX     A,X                  Reset generale del chip ACIA
        LDA   #%00000011     All'interno registro di controllo
        STA   ,X             Programma ACIA (8 bit dati, nessun bit di parità, 2 bit di stop)
        LDA   #%00010001
        STA   ,X             S/R per ricevere la stringa di caratteri
        RTS
BUFFER  EQU   $4000          Indirizzo per la stringa
CR      EQU   13             Cod. ASCII per ritorno-carrello
        BSR   ACIAST         Prepara l'ACIA. Il registro X contiene l'indirizzo dell'ACIA
        LDY   #BUFFER        Destinazione in Y
CICLO   LDB   ,X             Legge lo stato
        ASLB                 Muove il bit 7 fuori dal registro B nel flag di riporto di CCR
        BCC   CICLO          Torna indietro se il bit non è stato selezionato (cioè non ci sono interruzioni)
        LDA   1,X            Carica il byte dati
        STA   ,Y+            Lo immagazzina
        CMPA  #CR            A contiene un ritorno-carrello?
        BNE   CICLO          Carattere successivo
        RTS
```

"Star Raiders"

Derivato dal famoso "Star Trek" garantisce ore di intenso divertimento

Un "contatto" con la Zylons



Carta galattica



Nello *Star Raiders* il giocatore è al comando della nave spaziale Star Raider e attraversa la galassia alla caccia di un'astronave nemica, la Zylons. Per controllare il gioco sono necessari tanto il joystick quanto la tastiera del computer. Quando il giocatore preme il tasto F, sullo schermo compare un'immagine del settore della galassia occupato dall'astronave, vista dalla cabina di pilotaggio.

La posizione della nave nemica è indicata in una griglia di riferimento, situata nella parte inferiore dello schermo. Con il tasto L si attiva un "analizzatore a lungo raggio", che produce una visione ingrandita di una zona contenuta nella griglia di riferimento. L'astronave Zylons compare al centro di questa griglia. Adesso il comandante deve raggiungere il nemico: per far ciò può usare i normali motori dell'astronave (ma in questo caso la Zylons può sfuggire) oppure saltare nell'"iperspazio" (tasto H): questa manovra consente alla Star Raider di coprire in pochi secondi la distanza che la separa dalla nave avversaria.

Prima del salto, il giocatore deve ricordarsi di attivare il computer che calcola la rotta, battendo il tasto T. In caso contrario, il passaggio attraverso l'iperspazio può condurre in una zona della galassia totalmente sconosciuta. Altri elementi da tener presenti sono il computer d'attacco (abilitato dal tasto C) e gli "scudi protettivi" della Star Raider, azionati tramite il tasto S. Quando l'astronave esce dall'iperspazio, il computer fa lampeggiare un segnale di Allarme Rosso e la battaglia ha inizio. La Zylons può attaccare la Star Raider e le sue dimensioni aumentano via via che s'avvicina.

Il joystick permette al giocatore di ruotare l'astronave in tutte le direzioni, mentre i tasti numerici ne controllano la velocità. Questi controlli consentono di manovrare l'astronave con grande precisione, picchiando e cabrando attorno al vascello nemico.

Purtroppo, queste evoluzioni consumano molto carburante e la Zylons conquista un notevole vantaggio se la Star Raider è costretta a disimpegnarsi ed a dirigersi verso una base stellare per fare rifornimento e per riparare i danni subiti. Per compiere questa operazione, il giocatore deve visualizzare una griglia che mostra la base stellare: all'avvicinarsi dell'astronave, la base si ingrandisce trasformandosi in una "salsiccia" volante gialla. La manovra di "ingresso" nell'orbita è la più difficile del gioco: prima il giocatore deve inquadrare la base nella sua griglia di avvicinamento, poi rallentare la velocità finché l'altimetro non riporta il valore 0.

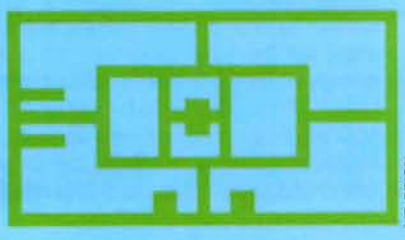
A questo punto la Star Raider può fermarsi e, se la manovra ha avuto successo, nella parte alta dello schermo compare il messaggio lampeggiante ORBIT ESTABLISHED. L'intera operazione è molto delicata ed è facile commettere errori fatali. Una volta in orbita, dalla base stellare emerge un ponte di rifornimento, che entra in contatto con la l'astronave. Completato il rifornimento, la Star Raider può tornare a combattere.

Quando sullo schermo compare il messaggio STARBASE SURROUNDED (Base Stellare Circondata), il giocatore deve correre in soccorso della base per evitarne la distruzione, ma contemporaneamente, deve fare molta attenzione a non colpirla con i propri cannoni.

Lo *Star Raiders* offre quattro livelli di gioco, da "principiante" a "comandante". Al primo livello, il giocatore non deve preoccuparsi eccessivamente dei pericoli che insidiano la Star Raider, perché ci sono poche astronavi Zylons e la loro mira è piuttosto approssimativa. Al livello massimo, è già un successo risuscire a sopravvivere per pochi minuti!

Negli Stati Uniti lo *Star Raiders* è stato giustamente insignito del titolo "Gioco dell'Anno" per il 1980 ed è un vero peccato che l'Atari non abbia sviluppato versioni per un numero maggiore di computer.

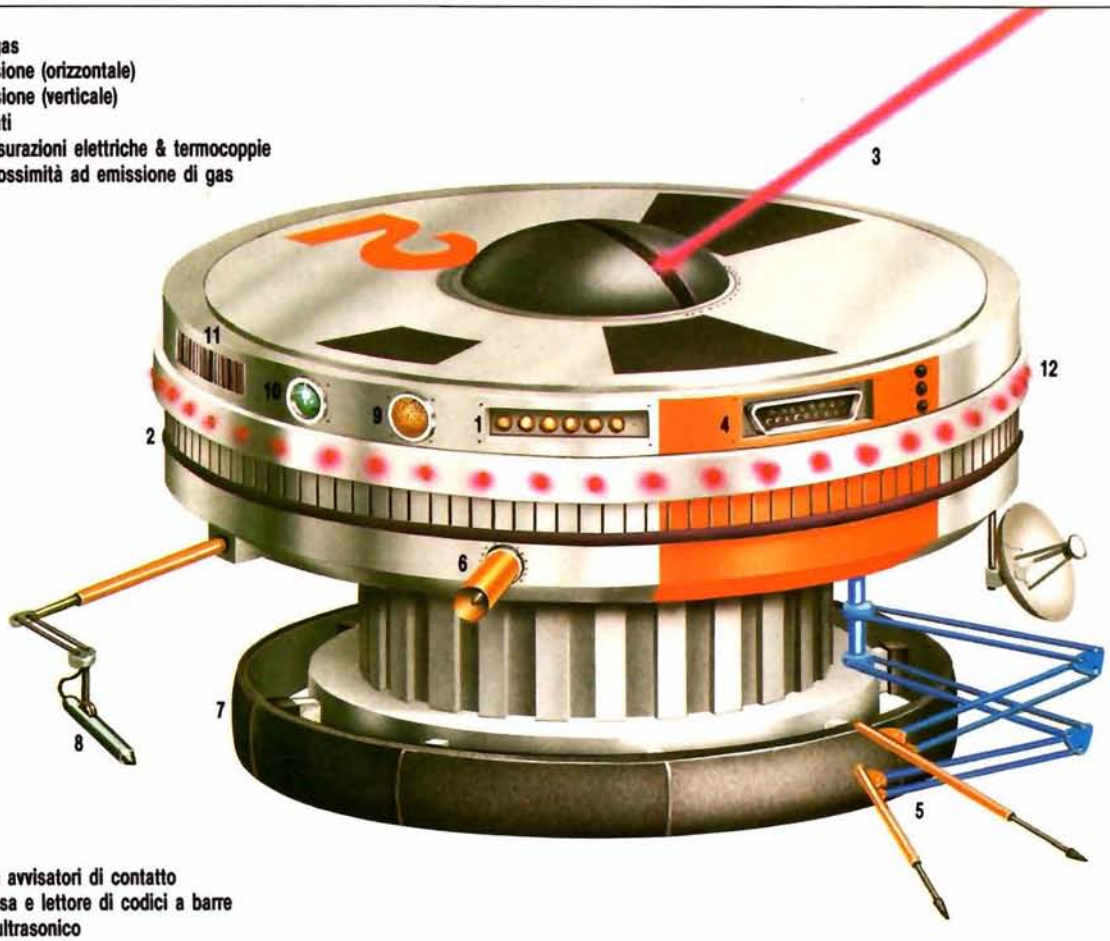
Star Raiders: Per tutti i computer Atari
Editori: Atari Corporation
Autori: Atari
Joystick: Richiesto
Formato: Su cartuccia



LIZ DIXON

I "sensi" dei robot

Speciali sensori permettono ai robot di ricavare precise informazioni sull'ambiente



STEVE CROSS

L'uomo interagisce col mondo circostante tramite i cinque sensi: un individuo che disgraziatamente ne fosse privo sarebbe totalmente indifeso; gli sarebbe impossibile persino camminare, perché il cervello non riceverebbe informazioni sui movimenti del suo corpo.

Abbiamo già visto come si muovono i robot ma, per poter godere di una certa indipendenza, devono essere dotati di un sistema di sensori. Il sogno di ogni progettista di robot è quello di creare una macchina che possieda sensi del tutto simili a quelli umani, ma questa meta è ancora molto lontana. I problemi connessi alla progettazione di una "vista" e di un "udito" per un robot sono talmente complessi da richiedere un'approfondita descrizione, che verrà fatta in seguito: queste pagine contengono un'introduzione generale all'argomento. Va detto, comunque, che il livello di complessità raggiunto nel realizzare sensori per robot è ancora di gran lunga inferiore rispetto a quello dei sensi dell'uomo. Un robot può "vedere" gli oggetti che lo circondano usando un dispositivo fotosensibile, che, di solito, produce una variazione di corrente proporzionale alla luce che lo colpisce. Tuttavia, simili sensori sono piuttosto primitivi e vengono impiegati in applicazioni poco sofisticate, in cui, ad esempio, si chiede al robot di dirigersi verso una sorgente luminosa nello stesso modo in cui segue una linea sul pavimento (vedere a pag. 1121). Questa funzione risulta utile per individuare una presa di corrente dalla quale ricaricare le proprie batterie, quando queste siano quasi

I "sensi" dei robot

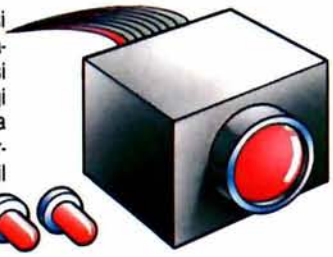
Le caratteristiche sensorie di un robot dipendono in larga parte dai suoi compiti specifici, quindi i robot meno specializzati, ossia di applicazione più generica, hanno bisogno di una maggior varietà di sensori. L'apparecchio in figura, puramente teorico, possiede quasi tutti i sensori oggi disponibili, ma è improbabile che un unico robot possieda in pratica una simile gamma di dispositivi

scariche (ovviamente, il robot deve possedere un circuito capace di controllare lo stato delle batterie).
Una semplice cellula fotoelettrica permette di eseguire diverse operazioni: i robot che lavorano sulle linee di montaggio possono "vedere" il componente da assemblare, controllando le variazioni di luminosità prodotte dalla sua presenza. Spesso, per accentuare il contrasto, dietro al componente c'è una sorgente luminosa, momentaneamente offuscata dal passaggio dell'oggetto.

I SENSORI



Il sensore ottico è una telecamera monocromatica a bassa scansione e bassa risoluzione. Genera immagini in varie sfumature di grigio, che contengono informazioni sufficienti per semplici applicazioni, quali seguire una linea o individuare uno spigolo

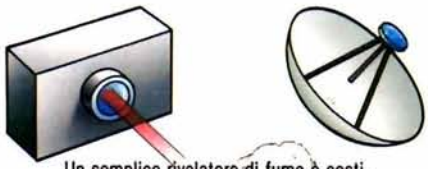


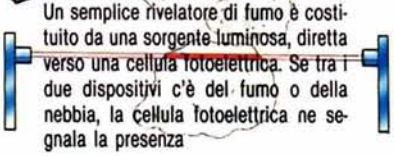
La telecamera a raggi infrarossi funziona come una normale telecamera, ma è sensibile agli infrarossi e non alla luce visibile. I raggi infrarossi penetrano il fumo e la nebbia molto più della luce e permettono di misurare, a distanza, il calore irradiato da un oggetto

Gli ultrasuoni sono suoni ad alta frequenza e vengono usati sui robot come telemetri direzionali. Un analizzatore ultrasonico è composto da un generatore di ultrasuoni e da un ricevitore direzionale. Quando gli ultrasuoni incontrano un oggetto, la conformazione della superficie riflettente distorce l'onda di ritorno generando una configurazione unica ed inconfondibile



Per misurazioni direzionali di alta precisione viene usato un laser a scansione, di bassa potenza. Il suo fascio luminoso può essere messo a fuoco con grande accuratezza e consente analisi dettagliate di oggetti vicini





Un semplice rivelatore di fumo è costituito da una sorgente luminosa, diretta verso una cellula fotoelettrica. Se tra i due dispositivi c'è del fumo o della nebbia, la cellula fotoelettrica ne segnala la presenza

I sensori di vicinanza ad emissione di gas sono composti da un ugello di emissione e da un misuratore di pressione. L'ugello emette il gas in una camera concava, creando una variazione di pressione standard. Quando un oggetto è molto vicino all'imboccatura della camera, viene captata la variazione di pressione



La penna ottica viene usata per numerose applicazioni: la lettura di codici a barre, ad esempio



Le sonde per misurazioni elettriche permettono di rilevare la resistenza, la capacità, il voltaggio e la corrente. Usate come termocoppie, misurano la temperatura



KEVIN JONES

Un robot può anche distinguere i colori, purché venga dotato di tre cellule fotoelettriche, ciascuna sensibile ad un diverso colore: il rosso, il verde ed il blu coprono tutto lo spettro visibile. Un simile robot può essere programmato, ad esempio, per estrarre da una pila di mattonelle colorate soltanto quelle rosse.
Un robot dotato di sensori acustici (microfoni e dispositivi simili) è capace di "udire" i suoni, anche se non è in grado di comprenderne il significato. Tuttavia, è possibile un repertorio di comandi, sotto forma di "modelli" o sequenze di suoni, che il robot "impara" a riconoscere. I comandi ai quali il robot può rispondere, benché in numero limitato, sono del tipo «vai avanti», «fermati», «gira a sinistra» e permettono di pilotarlo con una certa affidabilità.
I sensori preposti al "tatto" possono essere semplici microinterruttori, che trasmettono segnali elettrici quando subiscono una pressione. Ovviamente, la sensibilità tattile risultante è piuttosto grossolana, ma spesso sufficiente per le esigenze di un robot. Ad esempio, i sensori tattili applicati alla base di un robot mobile consentono alla macchina di reagire al contatto con un ostacolo, scegliendo una nuova direzione. Grazie ai sensori inseriti sul suo azionatore terminale, un braccio robot può sapere se ha "afferrato" o no un oggetto e comportarsi di conseguenza.
I rivelatori di gas o di fumo sono paragonabili all'olfatto dell'uomo. I primi si basano su un elemento sensorio, spesso un filo di platino che, in presenza di particolari gas, modifica la propria resistenza interna alla corrente che lo attraversa. I secondi sono formati da due "camere", entrambe contenenti elio ionizzato: una è chiusa, con funzione di parametro di controllo, e l'altra aperta. La presenza di fumo fa variare il numero delle particelle ionizzate contenute nella camera aperta. Il dispositivo, contando le particelle cariche presenti in ciascuna camera, rileva eventuali differenze ed emette un segnale se viene superato un valore "di soglia" prefissato.
A tutt'oggi, non è stato inventato alcun sensore per simulare il senso del gusto. Il massimo che un robot può fare è "vedere, sentire, toccare" e "annusare". Ciò gli consente, ad esempio, di individuare un incendio, di allontanarsi dal fuoco evitando gli ostacoli sul proprio cammino e (se dotato di un estintore) di spargere schiuma sulle fiamme.

Potenziali vantaggi

Tutto sommato, però, se si realizza un robot dotato di "sensi" simili a quelli dell'uomo, si riduce notevolmente il suo potenziale. È molto più ragionevole partire dai dispositivi di percezione offerti dalla tecnologia, per poi scegliere quelli che hanno qualche utilità pratica. Prendiamo come esempio un braccio robot, utilizzato per prendere un oggetto e spostarlo altrove. Un metodo per raggiungere questo risultato è fissare dei limiti o dei

"fermi" al movimento in ogni direzione, all'interno dei quali il robot svolge il suo lavoro: il braccio ruota fino ad incontrare uno dei fermi e, se ogni cosa è disposta in modo corretto, la sua "mano" si trova esattamente sopra l'oggetto da prendere. Effettuata la presa, il braccio ruota nella direzione opposta raggiungendo un secondo fermo: a questo punto può appoggiare l'oggetto nel punto previsto.
Questa semplice tecnica, molto usata in passato, è oggi superata, ma dimostra ugualmente che i robot possono usare sistemi non derivati da quelli umani.
Un esempio ancor più calzante riguarda un particolare impiego della "vista" del robot. Mentre la vista umana è limitata (gran parte dello spettro elettromagnetico è infatti invisibile all'occhio umano), il robot può essere dotato di sensori adatti ai raggi infrarossi. Questi sofisticati strumenti consentono di misurare la quantità di calore irradiata da un oggetto e vengono spesso montati su quei robot industriali che lavorano in ambienti dove vengono manipolati oggetti ad alte temperature.
I sensori infrarossi permettono anche di rilevare il calore emesso da un corpo umano, quindi un robot domestico potebbe essere programmato per andare incontro al proprio padrone. Altri sensori permettono ai robot di individuare e seguire i campi magnetici generati dal passaggio di una corrente elettrica, ma questa applicazione può essere usata anche per distinguere i materiali magnetizzati da quelli che non lo sono.
I sensori di vicinanza, che non hanno un equivalente umano, permettono al robot di "sentire" la prossimità di un oggetto: l'uomo ottiene lo stesso risultato combinando percezioni visive e tattili. Alcuni tipi di robot usano un getto d'aria, emesso da un ugello, che viene riflesso dagli oggetti vicini. Si crea quindi una pressione di ritorno, rilevata da un apposito trasduttore di pressione. Altri tipi si servono di un circuito elettrico la cui capacità varia all'approssimarsi di un oggetto. In tal caso viene misurata la dispersione elettrica tra un condensatore e l'oggetto, comunicando al robot che si sta avvicinando ad un ostacolo.

Trasduttori

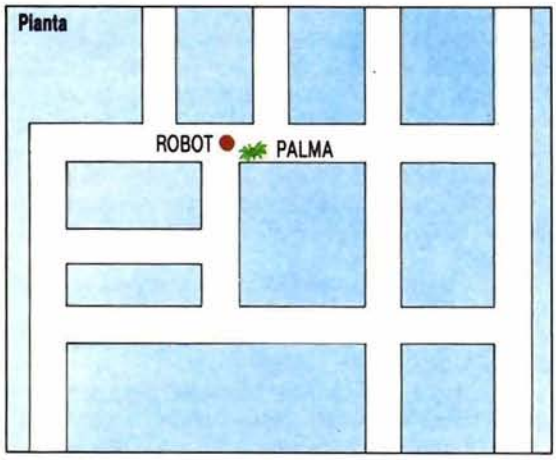
Esistono anche sensori ultrasonici: il suono emesso (inavvertibile all'uomo), riflesso dagli oggetti circostanti, viene captato dal sensore. Il lasso di tempo tra l'emissione del segnale, altamente direzionale, e la ricezione dell'eco consente di misurare con precisione la distanza dell'oggetto.
Un metodo simile viene usato dai pipistrelli in volo e sullo stesso principio si basano alcune macchine fotografiche con messa a fuoco automatica.
Ancor più sofisticati sono i sensori a laser. Questi inviano su un oggetto un raggio laser, che viene poi riflesso verso il sensore. Confrontando i due raggi è possibile calcolare la distanza dell'oggetto con una precisione assoluta. Questa tecnica viene usata anche per grandi distanze. Durante il primo allunaggio con equipaggio umano, gli astronauti posero sulla superficie lunare un riflettore ed un sensore laser, che permise di misurare l'esatta distanza tra la Terra e la Luna. La precisione di questa misurazione, su una distanza di 384.400 Km., è dell'ordine di 15 centimetri!
I sensori elastici permettono di ottenere informazioni tattili molto più sofisticate rispetto ai microinterruttori. Il loro funzionamento è basato sulle proprietà elettriche di un cristallo piezo-elettrico sottoposto a pressione oppure sulle variazioni di conduttività di granuli di grafite, anch'essi soggetti ad una pressione (la tecnica è simile a quella impiegata nei microfoni a carbone). Per forze di maggior entità possono essere usati degli estensimetri, che rilevano le variazioni di resistenza elettrica in un filo metallico sottoposto a stiramento.
Tutti questi sensori vengono detti "trasduttori" perché eseguono misurazioni di un dato tipo (ad esempio, luminoso, acustico o di pressione) e le convertono o "trasducono", appunto, in segnali di diverso tipo, ma proporzionali alle misurazioni originali. Nei robot controllati da computer, i trasduttori possono generare segnali elettrici binari (un dato segnale è presente oppure assente) o analogici (il segnale varia al variare della misurazione). Nel secondo caso, il segnale elettrico deve essere trasformato in digitale tramite un apposito convertitore A/D, prima di applicarlo al computer.
In definitiva, i sensi dell'uomo sono molto più completi ed efficienti di quelli dei robot, ma questi hanno una gamma di percezioni molto più vasta e divengono ogni giorno più sofisticati.

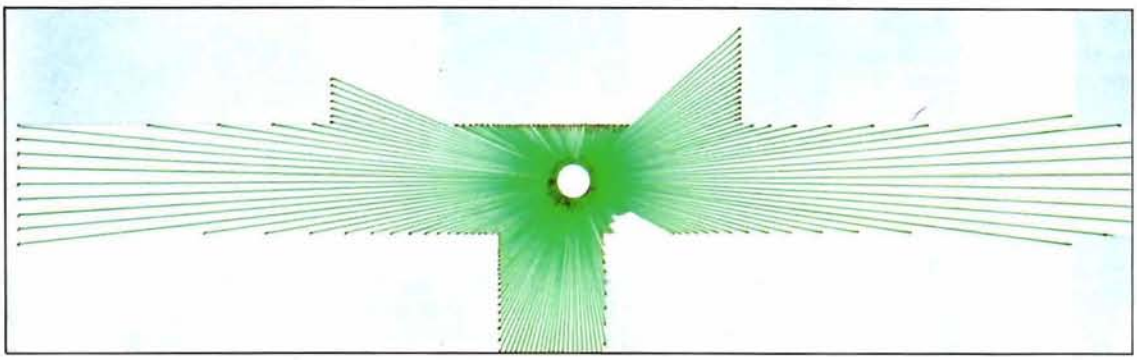


Un braccio
Questo braccio robot ripulisce fusioni prelevandole dagli stampi, quando sono ancora ad altissima temperatura. Naturalmente, il robot è insensibile al calore e può lavorare i pezzi molto più velocemente di un operatore umano

TRARRE SENSO DAI "SENSI"

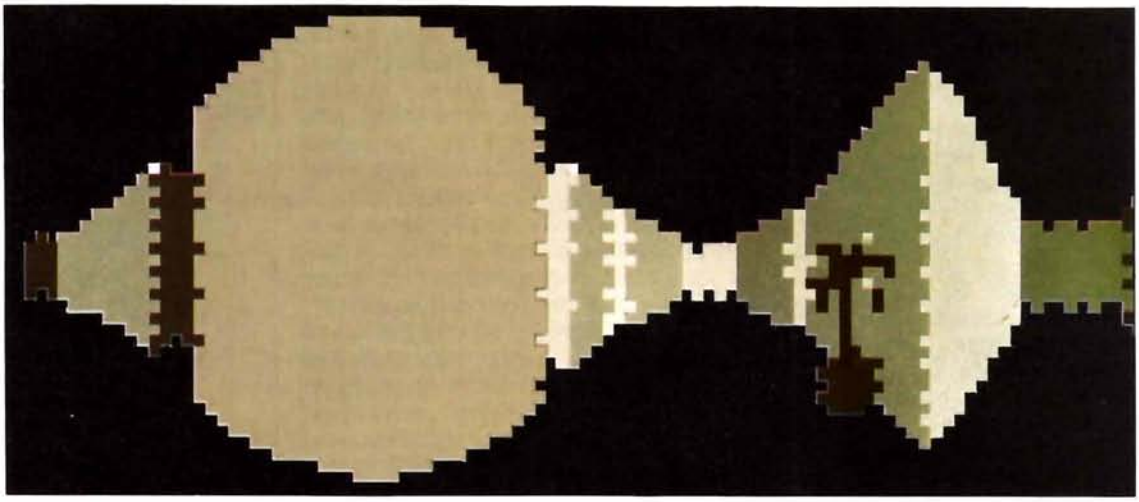
La percezione del mondo circostante è il principale problema dei robot ed i progettisti hanno tentato di superarlo dotandoli di numerosi sensori altamente sofisticati. Un solo sensore non può fornire un'informazione completa, ma talvolta troppi sensori sembrano contraddirsi tra loro. La "conoscenza" del mondo esterno, dunque, dipende dalla capacità del robot di integrare e confrontare i segnali che riceve dai vari dispositivi.
La pianta in figura mostra un robot all'interno di un corridoio i cui muri sono interamente verniciati di bianco. C'è una sola sorgente luminosa quindi l'illuminazione delle pareti dipende dalla loro orientazione. Il robot ha di fronte una piccola palma





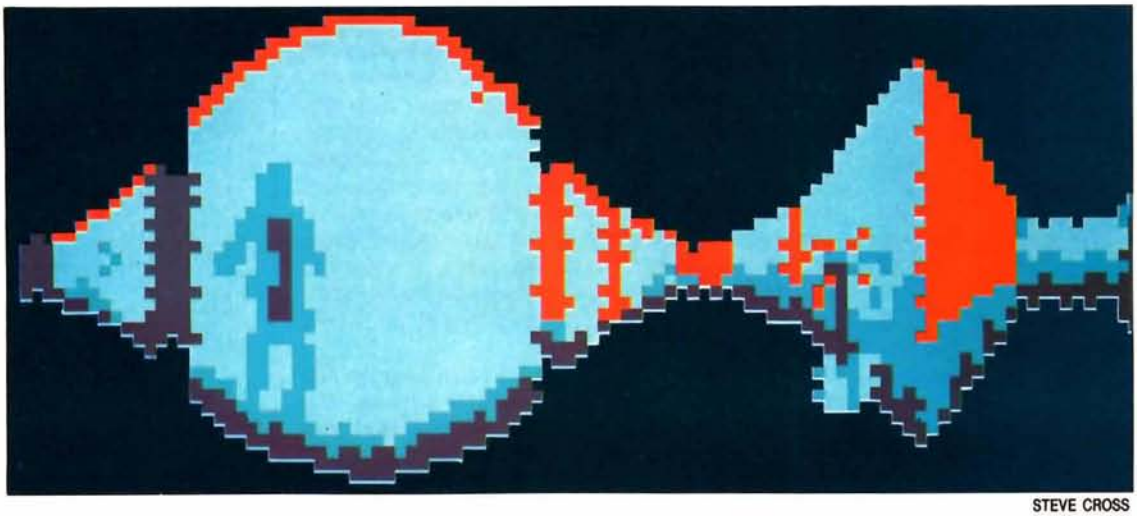
Esame laser

Il laser direzionale consente al robot di tracciare una precisa pianta dell'ambiente e di disegnare la sagoma della palma. Un piccolo spostamento permette al robot di osservare la palma sotto una diversa angolazione, per poterla distinguere dalle pareti del corridoio



Scansione luminosa

La telecamera genera un'immagine monocromatica a bassa risoluzione dell'ambiente. Questa può essere migliorata sviluppando tecniche per l'individuazione degli spigoli dei muri e dei movimenti. In pratica, la scansione luminosa non consente di ottenere una buona immagine della palma ma, integrata ad altre osservazioni, può essere un valido aiuto



STEVE CROSS

Scansione a raggi infrarossi

L'immagine infrarossa è simile a quella precedente, ma rivela che la temperatura della palma è leggermente diversa da quella dei muri circostanti, evidenziandola quindi come "organismo vivente". Contemporaneamente, a sinistra, compare l'"ombra" termica lasciata da un uomo, appoggiatosi in precedenza ad una delle pareti. Confrontando questa immagine con le precedenti, il robot individua e scarta la sagoma umana ed identifica la palma come unico organismo vivente nel corridoio

Un comodo visualizzatore

Colleghiamo alla porta del computer un display a sette segmenti

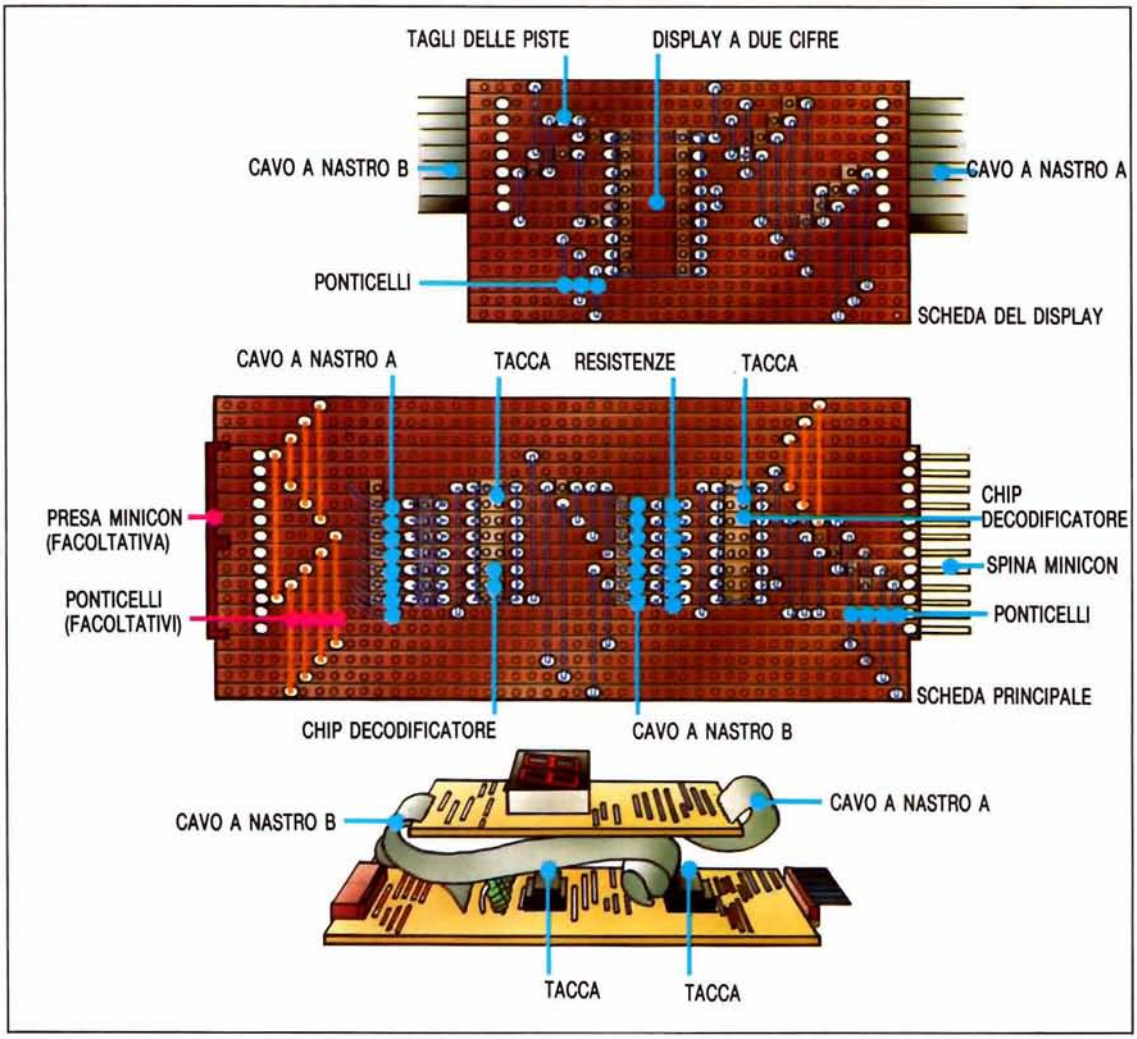
Per visualizzare cifre esadecimali occorrono quattro bit (che consentono 16 combinazioni di zero e uno). Qualsiasi numero ad otto bit può, quindi, essere rappresentato con due cifre esadecimali: la prima per i quattro bit "bassi", la seconda per i quattro "alti". Benché i display a LED contengano 7 segmenti, bastano quattro linee per pilotare il circuito di codifica/decodifica ad essi collegato.
I decodificatori sono speciali circuiti che trasformano le informazioni, trasmesse da un computer alle periferiche, in segnali elettrici e viceversa. La realizzazione di un circuito decodificatore è stata descritta in precedenza (vedere a pag. 626), ma per lo strumento qui presentato esiste un circuito integrato già pronto, ossia il chip 7447A.
Il decodificatore collegato a ciascun display prevede quattro ingressi (nel nostro caso provenienti dalla porta utente) e produce, tramite una serie di porte logiche, 7 linee di uscita. Se, ad esempio, ai quattro ingressi è applicata la combinazione 0111, il circuito logico attiva i segmenti del display che compongono il numero 7 (0111 in binario equivale a 7 in esadecimale). La tavola della verità per il display è a pag. 1167.
I numeri esadecimali superiori a 9 vengono di solito rappresentati con le prime 6 lettere dell'alfabeto (da A a F). Il decodificatore qui impiegato è di tipo decimale, quindi le combinazioni di segmenti per valori superiori al 9 non corrispondono alle lettere da A-F, bensì ad altri simboli. Ciò permette di ridurre il numero di porte logiche del decodificatore e la complessità globale del circuito.
Il circuito per il display qui descritto è in grado di visualizzare continuamente, in notazione esadecimale, il contenuto del registro dati della porta utente. Le otto linee dati previste consentono di controllare contemporaneamente due visualizzatori LED, ma in alcune applicazioni diversi display a 7 segmenti condividono lo stesso gruppo di linee. La tecnica che permette a più display di visualizzare contemporaneamente informazioni diverse è detta "trasmissione in multiplex". Le linee dati provenienti dal decodificatore vengono applicate in successione ai vari display mentre, in perfetto sincronismo, variano le informazioni trasmesse. Questa operazione è talmente rapida che i display sembrano essere tutti accesi allo stesso tempo, mentre invece visualizzano l'informazione ricevuta per una frazione di secondo dalle linee dati in comune.
Proviamo ad applicare il principio della trasmissione in multiplex ai due visualizzatori a 7 segmenti che stiamo costruendo. Poiché il numero decimale 15, decodificato, rappresenta uno spazio vuoto (blank), possiamo servircene per cancellare un display mentre accendiamo l'altro. Il seguente programma chiede la cifra da visualizzare quindi la fa apparire contemporaneamente su entrambi i display. Il programma contiene una routine che, tramite la pressione della barra spaziatrice, rallenta l'alternanza tra i due display. Premendo la barra si può infatti vedere che la cifra "salta" dall'uno all'altro, mentre lasciandola andare, il rallentamento viene eliminato, la velocità dell'alternanza torna a crescere e la cifra sembra comparire simultaneamente sui due display.

ELENCO DEI COMPONENTI

Quantità	Descrizione
14	resistenze da 330 ohm da 1/4 Watt
2	decodificatori 7447A per display a 7 segmenti
1	display ad anodo comune, a due cifre
2	zoccoli per chip DIL a 16 piedini
1	presa minicon a 12 canali "ad angolo retto"
1	spina minicon a 10 canali "ad angolo retto"
	cavo a nastro a 8 canali*
	cavo a nastro a 7 canali*
	filo stagnato*
1	veroboard da 36 piste x 50 fori
1	scatola di plastica 116x61x36 mm.

* Questi componenti dovrebbero essere avanzati da lavori precedenti. La presa a 12 canali è necessaria solo se si intende estendere il bus: questa, ed alcuni collegamenti sulla scheda, sono dunque facoltativi

```
10 REM TRASMISSIONE IN MULTIPLEX — BBC
20 DDR = &FE62:REGDAT = &FE60
30 ?DDR = 255
40 blank_sinistro = 15*16
50 blank_destro = 15
55 :
60 REPEAT
70 INPUT "NUMERO DA VISUALIZZARE";dato
80 ?REGDAT = dato + blank_sinistro
90 PROCrallentamento
100 ?REGDAT = dato*16 + blank_sinistro
110 PROCrallentamento
120 GOTO80
130 END
140 :
150 DEF PROCrallentamento
155 REM LA BARRA SPAZIATRICE È ABBASSATA ?
160 IF INKEY(—99) = —1 THEN PROCrallentamento
```




```
170 ENDPROC
180 :
190 DEF PROCrallentamento
200 FOR I = 1 TO 500:NEXT
210 ENDPROC
```

```
 10 REM TRASMISSIONE IN MULTIPLEX — CBM 64
 30 DDR=56579:REGDAT=56577
 40 POKE DDR,255
 50 LB=15*16:RB=15
 55 :
 60 INPUT "NUMERO DA VISUALIZZARE";DT
 70 POKE REGDAT,DT+LB
 80 GOSUB 1000: REM RALLENTAMENTO
 90 POKE REGDAT,DT*16+RB
100 GOSUB 1000: REM RALLENTAMENTO
110 GOTO 70
120 :
1000 S/R PER IL RALLENTAMENTO
1010 GET A$
1020 IF A$=" " THEN GOSUB 2000: REM RITARDO
1030 RETURN
1999 :
2000 REM S/R PER IL RITARDO
2010 FOR I=1 TO 250:NEXT
2020 RETURN
```

La coppia dei due display a 7 elementi può essere usata come contatore esadecimale per i segnali inviati alla porta del computer da un interruttore collegato alla sua linea 1. A prima vista questa operazione sembra piuttosto semplice, ma in realtà le otto linee della porta occorrono tutte per la visualizzazione: non ne rimane alcuna libera per l'input. Se selezioniamo per l'input la linea 0, il sistema di I/O del computer la mantiene costantemente a livello alto (1), indipendentemente dal numero contenuto nel registro per i dati. Quando carichiamo nel registro il numero 128 (10000000 in binario), il sistema di I/O lo trasforma immediatamente in 129 (10000001) perché la linea 0 deve essere mantenuta a livello alto per l'input. Ovviamente, la correzione produce un errore di conteggio sul display. Questo problema viene risolto con una tecnica simile alla trasmissione in multiplex: se la linea 0 si trova in input solo per un breve periodo mentre le altre linee restano in output per un tempo più lungo, il display riporterà costantemente il conteggio esatto. Il valore errato (che compare quando la linea 0 viene selezionata per l'input) lampeggerà solo per un breve istante.

```
10 REM CONTATORE — BBC
20 DDR=&FE62:REGDAT=&FE60
55 :
50 conteggio=0
60 REPEAT
```

```
  70 PROCimmissione
  72 PROCsomma
  73 FOR I=1TO40
  75 PROCdisplay
  77 NEXT I
  80 UNTIL conteggio255
  90 END
 999:
1000 DEFPROCsomma
1010 IF flag=1 THEN conteggio=conteggio+1
1050 ENDPROC
1499:
1500 DEF PROCimmissione
1510 ?DDR=254
1515 flag=0
1520 IF (?REGDAT AND 1)=0 THEN flag=1
1525 REPEAT UNTIL (?REGDAT AND 1)=1
1530 ENDPROC
1999:
2000 DEF PROCdisplay
2010 ?DDR=255
2030 ?REGDAT=conteggio
2040 ENDPROC
```

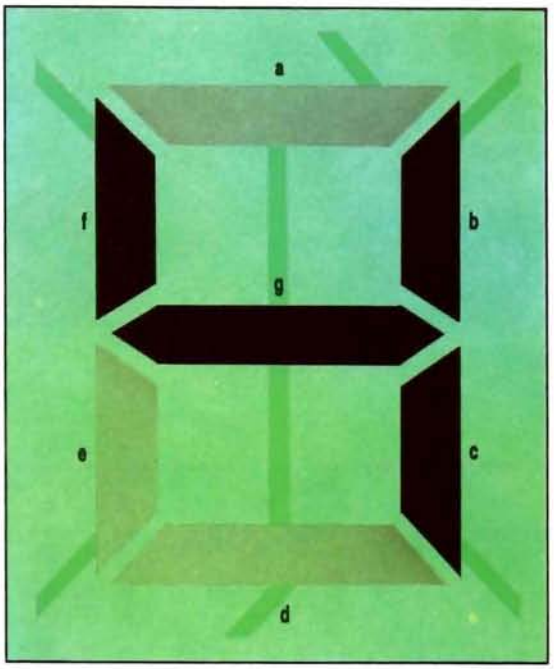
```
  10 REM CONTATORE — CBM 64
  30 DDR=56579:REGDAT=56577
  40 CC=0: REM INIZIALIZZA IL CONTATORE
  50:
  60 GOSUB 1000: REM IMMISSIONE
  70 GOSUB 2000: REM SOMMA
  80 FOR I=1 TO 20
  90 GOSUB 30J0: REM DISPLAY
 100 NEXT I
 110 IF CC
 120 END
 999:
1000 REM S/R PER L'INPUT
1010 POKE DDR,254
1020 FL = 0
1030 IF(PEEK(REGDAT)AND1)=0 THEN FL=1
1040 IF(PEEK(REGDAT)AND1)<>1 THEN 1040
1050 RETURN
1999:
2000 REM S/R PER LA SOMMA
2010 IF FL=1 THEN CC=CC+1
2020 RETURN
2999:
3000 REM S/R PER IL DISPLAY
3010 POKE DDR,255
3020 POKE REGDAT,CC
3030 RETURN
```

Per ogni passaggio del programma, il ciclo FOR.. NEXT esegue una sola volta la routine che pone in input la linea 0 e molte volte la routine che seleziona le altre linee in uscita. Per il Commodore 64, l'esecuzione della routine di input avviene ogni 20 esecuzioni di quella per la visualizzazione. Questo rapporto è di 40 a 1 nella versione BBC, grazie alla maggior velocità di elaborazione. In queste condizioni è possibile vedere il lampeggiamento ma, se il rapporto cresce ancora, può accadere che il tempo impiegato per leggere l'input divenga tanto breve da causare una totale perdita dei dati in arrivo. Nella elaborazione in "tempo reale" si devono ricercare compromessi di questo genere tra le esigenze delle varie parti di un sistema.

Decimale	Binario				Uscita	Display
	D3	D2	D1	D0	a b c d e f g	
0	0	0	0	0	0 0 0 0 0 0 1	0
1	0	0	0	1	1 0 0 1 1 1 1	1
2	0	0	1	0	0 0 1 0 0 1 0	2
3	0	0	1	0	0 0 0 0 1 1 0	3
4	0	1	0	0	1 0 0 1 1 0 0	4
5	0	1	0	1	0 1 0 0 1 0 0	5
6	0	1	1	0	1 1 0 0 0 0 0	6
7	0	1	1	1	0 0 0 1 1 1 1	7
8	1	0	0	0	0 0 0 0 0 0 1	8
9	1	0	0	1	0 0 0 1 1 0 0	9
10	1	0	1	0	1 1 1 0 0 1 0	c
11	1	0	1	1	1 1 0 0 1 1 0	J
12	1	1	0	0	1 0 1 1 1 0 0	u
13	1	1	0	1	0 1 1 0 1 0 0	c
14	1	1	1	0	1 1 1 0 0 0 0	E
15	1	1	1	1	1 1 1 1 1 1 1	

Tavola della verità

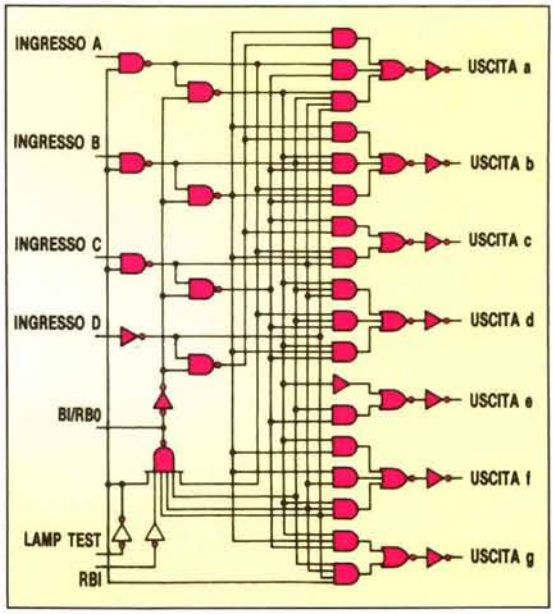
Il dato trasmesso dalla porta utente al display digitale è un numero binario di 4 bit. A ciascun valore binario compreso tra 0000 e 1111 viene associato un unico numero a 7 bit: ciascun bit segnala lo stato di uno dei sette segmenti del display. In questo circuito, i valori 0 e 1 indicano, rispettivamente, che il segmento corrispondente è acceso o spento



La scatola del display

Un singolo display digitale ha un codice di input di 4 bit, quindi la porta per l'utente può controllarne contemporaneamente due. Questi compongono un'unità compatibile con gli apparecchi realizzati in precedenza





Il decodificatore BCD 7447A

La circuiteria interna del chip evidenzia la sostanziale semplicità della sua logica: le porte logiche decodificano il dato in ingresso a 4 bit e lo trasformano negli output per i sette segmenti del display. Un segnale di controllo (LAMP TEST) verifica il funzionamento del chip accendendo contemporaneamente tutti i segmenti. Altre linee (BI/RBO e RBI) servono per lo spengimento del display nel caso degli zero non significativi e per collegare più decodificatori "in cascata"

DATABASE

Sesta scheda delle istruzioni per lo Z80 della Zilog. Istruzioni aritmetiche a 16 bit

DESTINAZIONE		SORGENTE: BC	DE	HL	SP	IX	IY
Somma, 'ADD'	HL	09	19	29	39		
	IX	DD 09	DD 19		DD 39	DD 29	
	IY	FD 09	FD 19		FD 39	FD 29	
Somma con riporto (Carry), 'ADC'	HL	ED 4A	ED 5A	ED 6A	ED 7A		
Sottrazione con riporto (Carry), 'SBC'	HL	ED 42	ED 52	ED 62	ED 72		
Incremento, 'INC'		03	13	23	33	DD 23	FD 23
Decremento, 'DEC'		08	1B	2B	3B	DD 2B	FD 2B

Codice mnemonico	Operazione simbolica	Flag S	Z		H		P/V	N	C	Op-code 76 543 210	Prefisso Hex	N. di byte	N. di Cicli M.	N. di Stati T.	Commenti
ADD HL, ss	HL←HL+ss	•	•	X	X	X	•	0	↕	00 ss1 001		1	3	11	ss Reg.: 00 BC, 01 DE, 10 HL, 11 SP
ADC HL, ss	HL←HL+ss+CY	↕	↕	X	X	X	V	0	↕	11 101 101 01 ss1 010	ED	2	4	15	
SBC HL, ss	HL←HL−ss−CY	↕	↕	X	X	X	V	1	↕	11 101 101 01 ss0 010	ED	2	4	15	
ADD IX, pp	IX←IX+pp	•	•	X	X	X	•	0	↕	11 011 101 01 pp1 001	DD	2	4	15	pp Reg.: 00 BC, 01 DE, 10 IX, 11 SP
ADD IY, rr	IY←IY+rr	•	•	X	X	X	•	0	↕	11 111 101 00 rr1 001	FD	2	4	15	rr Reg.: 00 BC, 01 DE, 10 IY, 11 SP
INC ss	ss←ss+1	•	•	X	•	X	•	•	•	00 ss0 011		1	1	6	
INC IX	IX←IX+1	•	•	X	•	X	•	•	•	11 011 101 00 100 011	DD 23	2	2	10	
INC IY	IY←IY+1	•	•	X	•	X	•	•	•	11 111 101 00 100 011	FD 23	2	2	10	
DEC ss	ss←ss−1	•	•	X	•	X	•	•	•	00 ss1 011		1	1	6	
DEC IX	IX←IX−1	•	•	X	•	X	•	•	•	11 011 101 00 101 011	DD 2B	2	2	10	
DEC IY	IY←IY−1	•	•	X	•	X	•	•	•	11 111 101 00 101 011	FD 2B	2	2	10	

NOTE:

ss è una qualsiasi delle coppie di registri BC, DE, HL, SP
pp è una qualsiasi delle coppie di registri BC, DE, IX, SP
rr è una qualsiasi delle coppie di registri BC, DE, IY, SP

Notazione per i flag:

• nessun effetto
0 flag azzerato
1 flag posto a 1
X effetto non definito
↕ flag rispecchia il risultato dell'operazione

per concessione della Zilog Inc.

Un giro tra le vetrine

Analisi critica dei più diffusi home computer: rispondono alle aspettative?

Come accade nel settore automobilistico, anche in quello degli home computer gli acquirenti si dividono in due gruppi: ci sono quelli che acquistano un apparecchio una volta per tutte, dotandolo di un'infinita serie di accessori, e quelli che partono già con l'intenzione di rivenderlo dopo poco, per comperarne uno nuovo, appena il mercato offre qualcosa di meglio.

Le caratteristiche dei più recenti home computer sono più soddisfacenti: memorie più ampie, dialetti BASIC più sofisticati, maggior risoluzione grafica e software residente. Tuttavia, gli apparecchi meno recenti, soprattutto quelli che hanno riscosso un buon successo di mercato, possono vantare una maggior disponibilità di software, mentre, per un computer da poco immesso sul mercato, occorre spesso attendere mesi prima che sia disponibile un soddisfacente corredo di pacchetti software.

In alcuni casi, come ad esempio per l'Atmos della Oric, il software non viene neppure sviluppato. Come logica conseguenza, le vendite dell'Atmos sono crollate.

I tre micro più seguiti dalle software house sono lo Spectrum della Sinclair, il Commodore 64 ed il Micro BBC. In particolare, lo Spectrum ha dimostrato come un buon corredo di software possa compensare i limiti dovuti all'hardware di un computer: alcuni programmi per lo Spectrum sono addirittura migliori di quelli sviluppati per apparecchi molto più sofisticati. Tuttavia, difficilmente i tre modelli menzionati avrebbero riscosso lo stesso successo, se lanciati sul mercato a distanza di qualche anno dalla loro effettiva apparizione: lo Spectrum ha una tastiera decisamente insufficiente, il BASIC del Commodore 64 non consente di esaltarne le potenzialità e il Micro BBC ha una memoria limitata ed è molto costoso rispetto ad altri apparecchi della stessa categoria.

Il mercato degli home computer offre apparecchi sempre più sofisticati a prezzi molto contenuti. Tutti i più recenti modelli possiedono tastiere di ottima fattura, ampie memorie RAM (64 Kbyte o più) ed una buona grafica. In apparecchi come il Commodore Plus/4, il Commodore 16, il QL Sinclair ed i vari micro MSX, la qualità dell'interprete BASIC è notevolmente migliorata. L'Amstrad include addirittura nel prezzo un monitor monocromo o a colori.

Un'altra interessante tendenza è quella di fornire, incluso nel prezzo, un intero package di software. Il QL Sinclair è venduto con quattro programmi di corredo: un word processor, uno spreadsheet, un database ed un programma grafico gestionale. Il Commodore Plus/4 offre programmi simili, ma meno sofisticati e che richiedono l'impiego di un'unità a dischetti. Altri micro puntano sui giochi: al Commodore 16 vengono abbinati quattro programmi di gioco, mentre la Sinclair ha abbinato allo Spectrum una confezione di 6 giochi.

Nell'acquisto di un nuovo micro occorre tener conto anche di altri fattori: alcune macchine sono facilmente espandibili e possono collegarsi ad unità a dischetti, stampanti, modem ed altre periferiche; alcuni computer accettano accessori standard, altri solo periferiche fabbricate appositamente dallo stesso fabbricante, con ovvie limitazioni di scelta per l'utente; fondamentale è la presenza di un buon manuale, ma ne esistono alcuni che confondono le idee; infine, occorre valutare il tipo, la qualità e la quantità di software disponibile: il Micro BBC, ad esempio, possiede soprattutto software didattico, mentre lo Spectrum punta molto sui giochi.



IAN McKINNELL

Fascino ... a otto bit

L'acquisto di un computer dovrebbe assomigliare a quello di un abito, ma spesso l'influenza dell'informazione tecnica e l'immensa gamma di prodotti disponibili lo rendono simile ad una "lotteria". Prima di entrare in un negozio di computer è bene considerare con calma le proprie esigenze e confrontarle con le capacità dell'apparecchio, senza lasciarsi influenzare da fattori emotivi







AMSTRAD CPC 464

La notevole esperienza della Amstrad nel campo dell'alta fedeltà si nota chiaramente dal design compatto del suo primo micro (vedere a pag. 909). L'Amstrad è dotato di monitor (monocromo oppure a colori) e di un registratore a cassette incorporato. Possiede un'interfaccia Centronics ed una per il joystick. Utile accessorio è l'unità a dischetti, venduta assieme al linguaggio LOGO ed al sistema operativo CP/M. L'Amstrad è una macchina completa, con una risoluzione grafica massima (bicromatica) di 640x200 pixel, una tavolozza con 27 sfumature di colore e suono stereofonico

QL SINCLAIR

IL QL viene venduto con una memoria di 128 Kbyte, una coppia di Microdrive incorporati e quattro programmi gestionali di corredo (vedere a pag. 981). Purtroppo, però, l'apparecchio non risponde alle aspettative: la tastiera è semplicemente una versione migliorata della tastiera "a membrana" dello Spectrum; il BASIC è strutturato, ma presenta alcuni errori ed appare piuttosto lento; l'affidabilità dei Microdrive, a lungo andare, suscita qualche perplessità ed il suo programma di Editor è deludente.
Il QL ha una risoluzione grafica di 512x256 pixel in quattro colori o di 256x256 in otto. Ciascun pixel può essere acceso individualmente: in questo caso i 32 Kbyte di memoria sono appena sufficienti per la gestione dello schermo. La macchina non dispone di unità a cassette, né a dischetti, ma possiede due interfacce RS232 e porte per monitor, joystick e per collegamenti in rete. Il suo punto di forza è il software di corredo sviluppato dalla Psion. I quattro programmi (word processor, spreadsheet, database, grafica gestionale) sono molto sofisticati, ma subiscono le limitazioni imposte dall'hardware

SPECTRUM SINCLAIR

Nonostante i suoi limiti, lo Spectrum ha avuto un grande successo. La tastiera è di modesta qualità ed usa un sistema "a parole chiave" che facilita la programmazione ai principianti, ma può creare qualche difficoltà ai più esperti. La risoluzione del video è di 256x176 pixel con otto colori, visualizzabili a coppie in un'unica cella di carattere. Le funzioni di suono sono quasi inesistenti: c'è un solo canale di suono ed il volume è molto basso. Il BASIC è soddisfacente, anche se un po' lento, mentre il manuale è paragonabile ad un buon testo didattico. Lo Spectrum manca di interfacce standard, ma numerose case indipendenti hanno prodotto periferiche da collegare alla porta utente. La Sinclair ha realizzato due interfacce: l'Interface 1, che consente il collegamento ai Microdrive, ad una rete ed alla porta RS232, seguita poco dopo dall'Interface 2, che permette all'apparecchio di usare software su cartuccia. Il software dello Spectrum è un po' "datato", ma contribuisce ancora alle vendite dell'apparecchio: il modello a 48 Kbyte è corredato da un package composto da 6 programmi di gioco

BBC MODELLO B

Questo apparecchio, ufficialmente adottato dal sistema scolastico britannico, ha molti ammiratori tra gli utenti più giovani e possiede una buona gamma di programmi didattici (vedere a pag. 929). Il Micro BBC viene prodotto dalla Acorn e possiede caratteristiche molto interessanti: un BASIC strutturato molto veloce, una discreta risoluzione grafica e buone funzioni di suono. Presenta inoltre una notevole gamma di interfacce: porte Centronics, RS423, RGB e video composito, 4 canali analogico/digitali, una porta per l'utente, un bus ad 1 MHz ed un "Tube", che permette di collegare un secondo processore. L'apparecchio è predisposto per collegamenti in rete e può usare unità a dischetti. Purtroppo, la memoria RAM è piccola e non basta se si adopera molta grafica. A seconda del MODE selezionato, restano a disposizione dell'utente da 9 a 28 Kbyte. L'impiego di uno Z80, come secondo processore, riduce i problemi di memoria, consente di utilizzare il software CP/M ed attribuisce al Modello B caratteristiche gestionali



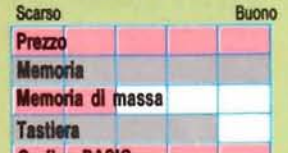


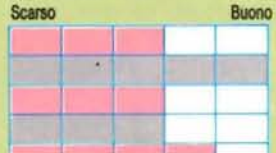
ELECTRON DELLA ACORN

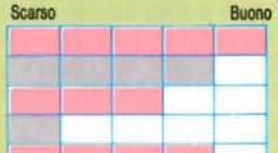
L'Electron è una versione "ridotta" del Micro BBC, possiede lo stesso BASIC strutturato (anche se più lento) ma non la gamma d'interfacce né la grafica del MODE 7 (teletext) del BBC (vedere a pag.929). Solo parte del software BBC è compatibile con l'Electron, che può tuttavia contare su una serie di programmi sviluppati appositamente. La massima risoluzione grafica è di 640x256 pixel, quella testuale è di 32 righe x 80 caratteri. Sfortunatamente, l'alta risoluzione riduce la memoria a disposizione del programmatore da 32 a circa 9 Kbyte. La periferica PLUS 1 fornisce all'apparecchio le porte per la stampante, il joystick e le cartucce, assenti nella versione base, ma non l'interfaccia per l'unità a dischetti

Come orientarsi nell'acquisto

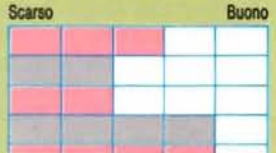
Comprare un nuovo micro non è facile! Queste dettagliate schede analizzano le principali caratteristiche dei micro ed offrono una serie di profili comparativi delle macchine descritte in queste pagine

















COMMODORE PLUS/4

Questa macchina è destinata a sostituire il Commodore 64. Entrambi gli apparecchi hanno una risoluzione grafica di 320x200 pixel e 64 Kbyte di RAM, ma il Plus/4 può visualizzare fino a 121 colori. Inoltre, possiede un BASIC più sofisticato che consente un controllo maggiore del video. Nel modo di massima risoluzione si possono adoperare solo due colori in un'unica cella di carattere, mentre il modo a 160x200 pixel ne accetta 4. Le funzioni di suono del Plus/4 sono a due "voci", ma non si avvicinano alla qualità di quelle offerte dal Commodore 64. Tuttavia, grazie ad una migliore versione del BASIC, la loro programmazione è molto più semplice. Purtroppo il Plus/4 manca della grafica a sprite, punto di forza del suo predecessore. Il software residente è composto da un programma "monitor" per codice macchina e da 4 programmi gestionali: word processor, spreadsheet, database e programma grafico. Il loro uso richiede la presenza di un'unità a dischetti. In sintonia con la politica commerciale della Commodore, il Plus/4 accetta solo i joystick ed il registratore a cassetta fabbricati dalla casa. Le stampanti e le lente unità a dischetti sono le stesse del Commodore 64, ma in futuro è previsto il lancio di un drive più veloce



COMPUTER CON STANDARD MSX

La normativa MSX definisce alcune caratteristiche di base, che possono essere ampliate e migliorate dai fabbricanti, ma senza influire sulla compatibilità tra gli apparecchi. La macchina qui raffigurata è il Toshiba HX-10 (vedere a pag. 1149). I computer MSX adottano un buon dialetto BASIC, con discrete funzioni grafiche, numerosi comandi per la generazione dei suoni ed un valido programma Editor. Possiedono inoltre vari tasti di funzione programmabili, per 10 differenti funzioni o comandi. La risoluzione grafica dello schermo è di 256x192 pixel in 16 colori, mentre gli sprite disponibili sono 32. Per la gestione del video vengono riservati 16 degli 80 Kbyte di RAM. Dei rimanenti 64 Kbyte, 28 sono a disposizione del programmatore BASIC, gli altri sono riservati al codice macchina o all'impiego di un'unità a dischetti

COMMODORE 64

Come lo Spectrum, il Commodore 64 è un apparecchio di grande successo, ma il suo BASIC manca di comandi adeguati alle eccellenti capacità grafiche e di suono (vedere a pag. 490). È dotato di grafica a sprite e la risoluzione massima è di 320x200 pixel, con 16 colori. Anche in questo caso i colori visualizzabili in una cella di carattere sono solo due. Sebbene l'apparecchio possieda una RAM da 64 Kbyte, restano soltanto 39 Kbyte per i programmi. Il Commodore 64 usa esclusivamente accessori Commodore: si tratta di periferiche abbastanza economiche, ma lente (specialmente l'unità a dischetti) e non esenti da inconvenienti



ATARI 600/800 XL

Questi computer sono versioni aggiornate della serie Atari 400/800 (vedere a pag. 669) il cui software è compatibile coi nuovi modelli. L'Atari 600XL è ormai fuori produzione, ma resta un valido acquisto: infatti la sua RAM di 16 Kbyte può essere espansa fino a 64 Kbyte trasformandolo così nell'Atari 800XL. La massima risoluzione grafica è di 320x192 pixel a due colori. Una risoluzione più bassa consente l'uso di 16 colori in 16 diverse tonalità. Il dialetto BASIC è un po' "sorpassato", ma possiede buoni comandi di suono e grafica a sprite. Gli apparecchi Atari usano un particolare registratore a cassette che va acquistato a parte. La gamma delle periferiche (non adattabili ad altri computer) è piuttosto ampia e di prezzo contenuto

COMMODORE 16

Sviluppato per sostituire il "vecchio" Vic-20, il Commodore 16 viene venduto in "kit", assieme ad un registratore a cassette, ad un manuale e una cassetta didattica per il BASIC ed a 4 programmi di gioco. Esteticamente ricorda il Commodore 64 ed il Vic-20, ma all'interno somiglia molto al Plus/4: infatti usa lo stesso BASIC e un identico programma "monitor" per codice macchina. In risoluzione "normale", il programmatore BASIC può disporre di 12 Kbyte, ma con la grafica ad alta risoluzione la memoria è di appena 2 Kbyte. Per questo motivo la maggior parte del software viene prodotto su cartucce. Tra i nuovi programmi sviluppati per il Commodore 16 ci sono due giochi di tipo arcade, un programma didattico di scacchi e uno di disegno grafico. La macchina è particolarmente adatta ai principianti. La poca quantità di memoria è una forte limitazione

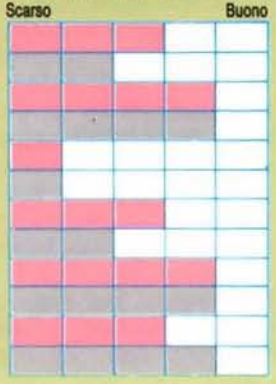
Variazioni di prezzo

I prezzi dei micro e degli home computer possono subire variazioni dovute alle fluttuazioni delle varie monete europee e del dollaro. Tuttavia, grazie ai continui sviluppi tecnologici, negli ultimi tempi i prezzi di molti apparecchi sono notevolmente diminuiti

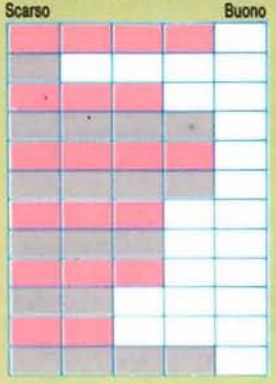
COMMODORE PLUS/4



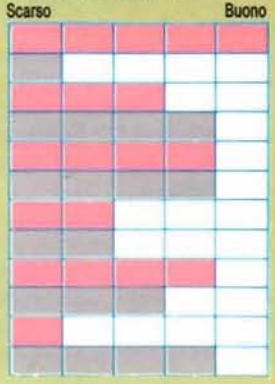
COMMODORE 64



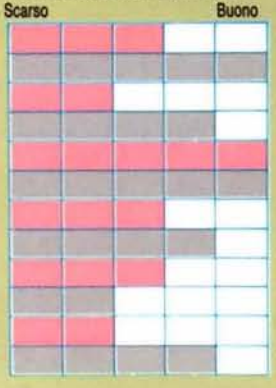
COMMODORE 16



ATARI 600/800 XL



MSX STANDARD COMPUTERS



Una macchina da calcolo

L'impiego di spreadsheet non è limitato ai soli ragionieri o esperti finanziari

Oggi il mercato offre numerosi ed economici pacchetti di gestione finanziaria, anche su cassetta, i quali sono adatti ai più diffusi home computer. La seguente descrizione vuol essere una guida pratica e articolata all'uso dei programmi di spreadsheet per applicazioni di diverso tipo, con particolare riferimento al bilancio familiare.
In questo viene calcolato il rapporto esistente tra i tassi d'interesse ed il pagamento dei mutui ipotecari e vengono messi a confronto i vantaggi e gli svantaggi dell'affitto rispetto all'acquisto nelle principali voci del bilancio domestico.
Il "cuore" di uno spreadsheet è la cosiddetta "lavagna elettronica", ossia una particolare tabella di calcolo, composta da alcune righe e colonne.
Lo schermo viene usato come una finestra mobile di quattro o cinque linee per 10 o 15 colonne, nella quale visualizzare le diverse parti della tabella.
Le intersezioni tra righe e colonne sono chiamate "celle" e possono contenere lettere, numeri o formule di calcolo. Il cursore dello spreadsheet (di solito un rettangolino luminoso) viene spostato tramite i normali tasti di controllo.
Ciascun valore immesso da tastiera viene inserito dal programma nella cella occupata in quel momento dal cursore.
Dopo questa schematica descrizione della struttura di uno spreadsheet, analizziamo più approfonditamente il pacchetto *Vu-Calc*, sviluppato dalla Psion per lo Spectrum e per il BBC Modello B.
Esaminiamo, in particolare, la versione BBC, quasi identica all'altra, salvo per i nomi di alcuni comandi.
Il *Vu-Calc* offre un ottimo esempio della flessibilità e dei vantaggi tipici degli spreadsheet.
Pur non possedendo le sofisticate funzioni del *Lotus 1-2-3* (il costo e le caratteristiche del quale sono spiccatamente commerciali), il *Vu-Calc* permette di costruire "modelli" gestionali molto efficienti, da poterli memorizzare su cassetta assieme ai dati per essere riutilizzati al momento opportuno.
La versione per il Micro BBC prevede un massimo di 28 colonne, numerate a partire da 1 e di 52 righe etichettate alfabeticamente. Le righe dopo la "Z" sono contrassegnate da lettere doppie ("AA", "BB", ecc.).
Le ridotte dimensioni di questa versione sono dovute alla limitata quantità di memoria disponibile sul BBC (32 Kbyte).
Una delle applicazioni più interessanti di uno spreadsheet su di un home computer è il bilancio familiare annuale. Questo modello, che non presenta particolari difficoltà di definizione, permette di avere un'idea generale di come alcune spese non previste (ad esempio, una bolletta telefonica particolarmente elevata o un imprevisto viaggio all'estero) influiscono sui risparmi domestici (sempre che ve ne siano!). In altre parole, permette di "giocare" con uno o più dati in bilancio e



vedere come questi possano influenzare l'intero spreadsheet.

La costruzione di un "modello"

Innanzi tutto, conviene annotarsi su un foglio tutte le voci di spesa previste, riportando sotto ogni voce l'ammontare delle uscite mensili. È in questo calcolo e nella registrazione all'interno del modello delle varie uscite mensili che il *Vu-Calc* mostra tutte le sue potenzialità.
Le voci di spesa vengono inserite nella prima colonna, mentre l'ammontare delle diverse spese viene registrato nelle colonne successive indicate dalle rispettive etichette "Gen", "Feb", "Mar", ecc. Nel *Vu-Calc* tutte le voci devono essere precedute da virgolette, anche se queste non compaiono poi sullo schermo.
Le prime righe e colonne dello spreadsheet potrebbero essere le seguenti:

	1	2	3	4
A		GEN	FEB	MAR
B	Automobile			
C	Mutui ipotecari	150000		
D	Tasse			

Questa è una configurazione di base del modello nel quale vengono inseriti i relativi valori usando l'istruzione REPLICATE.
Il *Vu-Calc* possiede un numero limitato di comandi, preceduti dal segno #. Battendo il carattere # in una cella vuota, il programma passa nel "modo di comando" ed attende l'inserimento di un comando. La fase più noiosa, nella creazione del bilancio annuale familiare, è forse la registrazione dei dati, elaborati in seguito dal programma. L'istruzione REPLICATE trascrive automaticamente uno stesso dato in più celle, riducendo così i tempi di lavoro.
Le voci del bilancio familiare che si ripetono sono le spese mensili "fisse" come le tasse, i mutui ipotecari e l'affitto. Se, ad esempio, la rata di un mutuo è di 150.000 lire al mese, REPLICATE permette di inserirla in tutte e 12 le celle mensili.
Quando si chiama questo comando battendo #R, sulla prima riga dello schermo compare il messaggio:

Replicate — Enter the cell to replicate, RETURN for the current cell

che chiede all'utente di indicare le coordinate (prima la lettera della riga, poi il numero della colonna: ad esempio (C2) della cella da copiare (premendo RETURN vengono prese le coordinate della cella in cui si trova il cursore).
È possibile copiare soltanto una cella alla volta: questo è uno dei limiti più evidenti del *Vu-Calc*.
Ricevute le coordinate, il programma visualizza un secondo messaggio:

Enter the range over which the data is to be replicated

che chiede di indicare la sequenza di celle nelle quali si vuol copiare un'informazione: si devono cioè specificare le coordinate della prima cella a sinistra e dell'ultima a destra della serie.
Ad esempio, l'istruzione per trasferire il mutuo mensile di 150.000 lire in una serie di celle della riga C, comprese tra C3 e C13 (ossia le colonne da "Feb" a "Dic") ha il seguente formato: #R,C2,C3:C13.
Il comando REPLICATE consente di copiare da una cella ad un'altra anche formule di calcolo, che possono essere "relative" a una data cella o "assolute": questa è un'area più specializzata del programma, che procederemo ad esaminare in seguito.
Per cambiare l'ammontare di una particolare spesa mensile, basta spostare il cursore sulla cella da correggere e inserire il nuovo valore, che automaticamente sostituisce il precedente.
Nel nostro modello, sulla colonna 14 vengono riportati i totali delle spese mensili. Talvolta può essere necessario aggiungere una certa somma alle celle di una voce di spesa mensile: in questi casi il *Vu-Calc* può essere usato come una calcolatrice, per eseguire l'operazione su qualsiasi riga o colonna. Il comando @ indica che si intendono sommare i valori contenuti in una serie di celle, specificate nel modo già visto.
Pertanto, volendo calcolare i versamenti annuali di un mutuo ed inserire il risultato nella cella C14, occorre posizionare il cursore in C14 e scrivere l'istruzione @C2:C13. Il risultato (in questo caso 1.800.000 lire) compare immediatamente sullo schermo.
Con la stessa facilità (ed il medesimo risultato) si potrebbe adoperare una formula di calcolo: C2–12. Il *Vu-Calc* riconosce (C non è preceduto da virgolette) ed esegue la formula.
Esiste quindi più di un modo per ottenere uno stesso risultato.
In seguito verrà mostrato come incrementare un valore di un tasso percentuale "fisso" e come duplicare le formule di calcolo che si riferiscono a celle relative e assolute.

PACCHETTI SPREADSHEET

Commodore 64

Busicalc della Supersoft (GB) su dischetto o cassetta.
Insta-Calc Graphic, della Dataview Wordcraft Ltd. (GB), su cartuccia e dischetto.

Micro BBC

Vu-Calc: su cassetta della Psion Ltd. (GB)

Spectrum

Vu-Calc: su cassetta della Psion Ltd. (GB)

Caccia al drago

Alcuni interessanti sviluppi, nell'uso di sprite con il LOGO

Le procedure di questo gioco usano gli sprite del LOGO: il giocatore controlla, tramite joystick, i movimenti di un drago che deve raggiungere e distruggere una città, difesa da un cavaliere volante (guidato dal computer) che cerca a sua volta di uccidere il drago. Se quest'ultimo riesce a sfuggire al cavaliere, può planare sulla città ed incendiarla con il suo soffio infuocato.

Per giocare è necessario caricare in memoria il file SPRITES, creare le diverse figure, inserire le procedure e battere il nome di procedura GIOCO. Effettuate le operazioni di messa a punto, GIOCO chiama la procedura AZIONE, che è la routine principale del programma. AZIONE muove il drago e il cavaliere sullo schermo e verifica se il drago riesce a raggiungere la città o se viene colpito dal cavaliere. Le altre procedure svolgono operazioni di contorno.

I comandi di colore sono molto semplici. BACKGROUND, seguito dal numero di codice del colore, definisce il colore dello sfondo mentre PENCOLOR seleziona il colore degli sprite (e della linea che essi tracciano se la "penna" è abbassata). Nella procedura INIZ.VARIABILI, ai numeri di codice vengono assegnati i nomi dei colori: è possibile selezionare i colori usando il loro nome (ad esempio, PENCOLOR :ROSSO). Nella procedura AZIONE, la linea:

```
IF COLPITO? THEN DRAGO.DISTRUTTO STOP
```

verifica se il cavaliere ha colpito il drago. La procedura COLPITO? illustra come vengono programmate le condizioni di verifica nel LOGO. La procedura restituisce la risposta logica "TRUE (vero) o "FALSE (falso), usata poi come dato in ingresso dall'istruzione IF. Se la condizione è vera ("TRUE), l'azione condizionale viene eseguita. COLPITO? utilizza la procedura TS?, contenuta nel file SPRITES, che restituisce la risposta "TRUE se c'è stata una collisione tra

Cavalieri e Draghi



Il gioco prosegue...



Città in fiamme



Morte del drago



IAN McKINNELL

CAVALIERE CONTRO DRAGO

```
TO GIOCO
   INIZ.VARIABILI
   PREP.SCHERMO
   AZIONE
END

TO INIZ.VARIABILI
   MAKE "FIAMMA 1
   MAKE "FIAMMA1 2
   MAKE "DRAGO 3
   MAKE "CAVALIERE 4
   MAKE "CITTÀ5
   MAKE "CITTÀ1 6
   MAKE "ROSSO 2
   MAKE "NERO 0
   MAKE "BLU 6
   MAKE "ARANCIONE 8
   MAKE "GIALLO 7
END

TO PREP.SCHERMO
   DRAW
   FULLSCREEN
   BACKGROUND :BLU
   TELL 0
   PU
   HT
   TELL :FIAMMA1
   HT
   HT
   TELL :FIAMMA
   HT
   POSIZIONE :DRAGO 100 100 :RED
   BIGX BIGY
   POSIZIONE :CITTÀ (—52) (—80) :NERO
   BIGX BIGY
   POSIZIONE :CITTÀ1 (—100) (—80) :NERO
   BIGX BIGY
   TELL 4
   PU
   POSIZIONE :DRAGO (100) (—100) :GIALLO
   SMALLX SMALLY
END

TO AZIONE
   MOV.DRAGO
   IF DISTANZA :DRAGO :CITTÀ<50 THEN DISTRUZ. CITTÀ STOP
   IF DISTANZA :DRAGO :CITTÀ1<50 THEN
      DISTRUZ.CITTÀ STOP
   MOV.CAVALIERE
   IF COLPITO? THEN DISTRUZ.DRAGO STOP
   AZIONE
END

TO MOV.DRAGO
   TELL :DRAGO
   MOV.JOY JOYSTICK 1
   FD 10
END

TO MOV.JOY :DIR
   IF: DIR SETH :DIR * 45
END

TO DISTANZA :A :B
   TELL :A
   MAKE "X1 XCOR
   MAKE "Y1 YCOR
   TELL :B
   MAKE "X2 XCOR
   MAKE "Y2 YCOR
   OUTPUT SQRT ((:X1—:X2) * (:X1—:X2) +
   (:Y1—:Y2) * (:Y1—:Y2))
END
```

uno sprite e lo sprite in esame. COLPITO? prima seleziona lo sprite del drago, poi verifica se è entrato in contatto con qualcosa. I parametri del comando JOYSTICK sono 0 ed 1 (corrispondenti alle porte 1 e 2): il joystick emette il valore −1 quando si trova in posizione centrale, 0 quando viene spostato in avanti, 1 a 45 gradi, 2 a 90 gradi e così via fino a 7. Il programma seleziona la direzione del movimento del drago a 45 gradi e la moltiplica per il numero emesso dal joystick.
L'esplosione e gli altri effetti visivi sono formati da sprite. L'immagine dell'esplosione lampeggia sopra lo sprite che viene distrutto. Allo sprite FIAMMA è stato assegnato un numero di sprite basso, per conferirgli un'alta priorità: può così comparire al di sopra degli altri sprite. Il computer controlla il cavaliere, ma con una strategia difensiva piuttosto semplice: il cavaliere punta contro il drago, il quale riesce a schivarlo abbastanza facilmente e a distruggere la città.
Per migliorare la strategia difensiva del cavaliere si può aumentarne la velocità: basta passare da 10 ad 11 per mettere il drago in serie difficoltà. (È vietato muoversi sullo schermo per guadagnare tempo!) Comunque, la migliore strategia, per il cavaliere, è quella di frapporsi tra il drago e la città. In seguito verranno esposte alcune caratteristiche degli sprite relative al LOGO per gli Atari.

PROGRAMMA "TRE INSETTI"

Gli insetti partono dagli angoli del triangolo:

```
TO INSETTI
   PREPARAZIONE MOV.INSETTI
END

TO PREPARAZIONE
   DRAW FULLSCREEN TELL 0 HT PU SETXY
   (−100) (−100) TRIANGOLO 200 POSITION 1 (−100)
   (−100) POSITION 2 0 73 POSITION 3 100 (−100)
END

TO TRIANGOLO :LATO
   PD REPEAT 3 [FD :LATO RT 120] PU
END

TO POSIZIONE :NO :X :Y
   TELL :NO SETSHAPE 3 PU SETXY :X :Y PD ST
END

TO MOV.INSETTI
   INSEGUE 1 2 INSEGUE 2 3 INSEGUE 3 1
   MOV.INSETTI
END

TO INSEGUE :A :B
   TELL :B MAKE "X XCOR "Y YCOR TELL :A
   SETH TOWARDS :X :Y FD 10
END
```



Lo Sprite Editor (Creazione di uno sprite)

```
TO DISTRUZ.CITTÀ
   TELL :CITTÀ
   MAKE "X XCOR
   MAKE "Y YCOR
   LAMPEGGIO :X :Y :ARANCIONE
   TELL CITTÀ1
   MAKE "X2 XCOR
   MAKE "Y2 YCOR
   LAMPEGGIO1 :X2 :Y2 :ARANCIONE
   SOVRAPP.SPRITE
   SPLITSCREEN
   REPEAT 3 [PRINT"]
   PRINT [CITTÀ DISTRUTTA!]
END

TO LAMPEGGIO :X :Y :COLOR
   TELL :FIAMMA
   PENCOLOR :COLOR
   SETXY :X :Y
   ST
   REPEAT 6 [SMALLX SMALLY ATTESA BIGX BIGY
   ATTESA]
END

TO LAMPEGGIO1 :X2 :Y2 :COLOR
   TELL :FIAMMA1 PENCOLOR :COLOR
   SETXY :X2 :Y2
   ST REPEAT 6 [SMALLX SMALLY ATTESA BIGX
   BIGY ATTESA]
END

TO MOV.CAVALIERE
   TELL :DRAGO
   MAKE "X XCOR
   MAKE "Y YCOR
   TELL :CAVALIERE
   SETH TOWARDS :X :Y
   FD 10
END

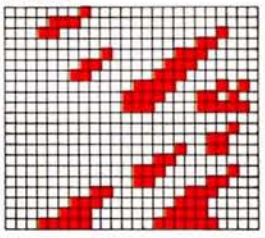
TO COLPITO?
   TELL :DRAGO
   IF TS? THEN OUTPUT "TRUE
   OUTPUT "FALSE
END

TO DISTRUZ.DRAGO
   TELL :DRAGO
   MAKE "X XCOR
   MAKE "Y YCOR
   LAMPEGGIO :X :Y :NERO
   SOVRAPP.SPRITE
   SPLITSCREEN
   REPEAT 3 [PRINT"]
   (PRINT [DRAGO UCCISO AD UNA DISTANZA DI]
   DISTANZA :DRAGO :CITTÀ
END

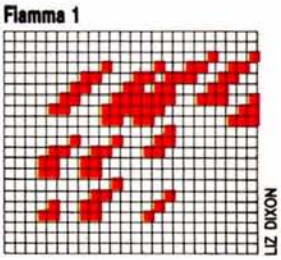
TO SOVRAPP.SPRITE
   TELL :FIAMMA HT
   TELL :FIAMMA1 HT
   TELL :CITTÀ HT
   TELL :CITTÀ1 HT
END

TO ATTESA
   REPEAT 100 [ ]
END
```

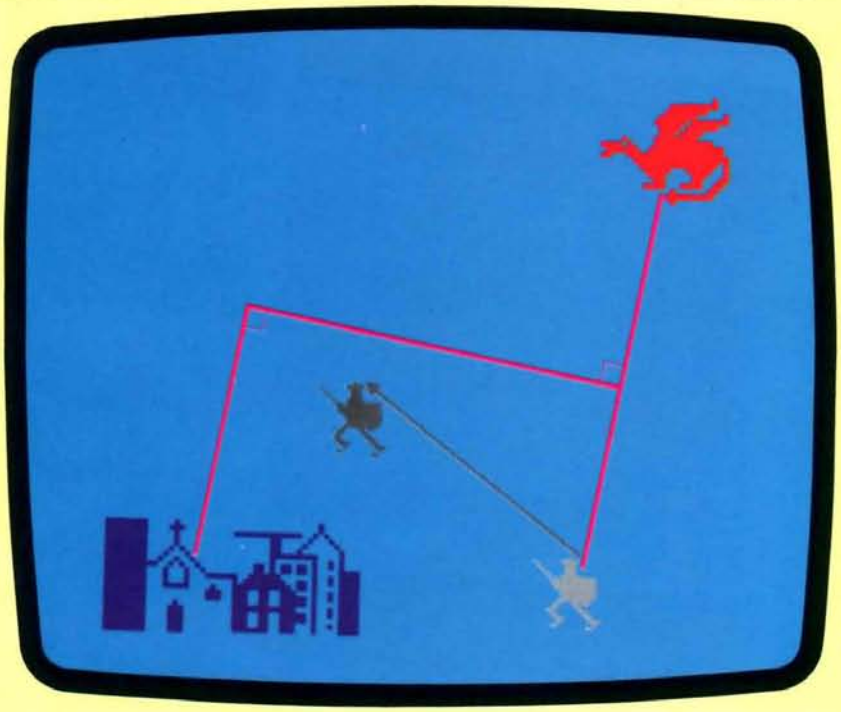
Fiamma



Fiamma 1



©1984 ZX



LIZ DIXON

STRATEGIA VINCENTE

La strategia migliore (per entrambi i contendenti) è quella di portarsi nella posizione più vicina alla città, sulla bisettrice perpendicolare alla linea che congiunge il cavaliere al drago (vedere figura):

```
TO MOV.CAVALIERE
   TELL :DRAGO MAKE "DX XCOR MAKE "DY
   YCOR TELL :CAVALIERE MAKE "KX XCOR
   MAKE "KY YCOR TELL :CITTÀ MAKE "CX
   XCOR MAKE "CY YCOR
   MAKE "SX (:DX+:KX)/2
   MAKE "SY (:DY+:KY)/2
   MAKE "VX (:DY—:KY)
   MAKE "VY (:KX—:DX)
   MAKE "FACT (:VX * (:CX—:SX) + :VY *
   (:CY—:SY)) / ((:VX*:VX)*(:VX*:VX)+(:VY*:VY)*(:VY*:VY))
   MAKE "X :SX + :FACT * :VX
   MAKE "Y :SY + :FACT * :VY
TELL :CAVALIERE SETH TOWARDS :X :Y
FD 10
END
```

UN MIGLIOR MOVIMENTO

Ecco come migliorare il movimento del cavaliere, facilitando l'abbattimento del drago:

```
TO MOV.CAVALIERE
   TELL :DRAGO MAKE "X XCOR MAKE "Y YCOR
   TELL :CITTÀ
   MAKE "BEARING TOWARDS :X :Y
   TELL :CAVALIERE SETH 270 + :BEARING
   IF XCOR < :X THEN LEFT 180
   FD 10
END
```

CONTROLLO DA TASTIERA

Controllo da tastiera del drago:

```
TO MOV.DRAGO
   TELL :DRAGO MOVIMENTO LEGGI.TASTO FD 10
END

TO MOVIMENTO :DIR
   IF :DIR = "W THEN SETH 0
   IF :DIR = "S THEN SETH 90
   IF :DIR = "Z THEN SETH 180
   IF :DIR = "A THEN SETH 270
END

TO LEGGI.TASTO
   IF RC? THEN OUTPUT READCHARACTER
   OUTPUT"
END
```



ABBREVIAZIONI

BACKGROUND	BG
PENCOLOR	PC

Logo & Logo

Le versioni LOGO per lo Spectrum e per l'Apple non prevedono la grafica a sprite.
I possessori di Atari devono modificare il programma nel seguente modo:

1) Poiché nel LOGO Atari TS? non esiste, si deve eliminare la linea IF COLPITO? ecc. della procedura AZIONE ed inserire la seguente alla fine della procedura PREP.SCHERMO:

```
WHEN TOUCHING :DRAGO :CAVALIERE
[DISTRUZ.DRAGO STOP]
```

2) Mancano le istruzioni BIGX, BIGY, SMALLX e SMALLY quindi vanno eliminate.

3) Usare SETBG e SETPC al posto di BACKGROUND e PENCOLOR. Naturalmente sono differenti anche i codici di colore.

4) Stranamente, nel LOGO Atari, manca il comando TOWARDS (presente invece nelle versioni LCSI dell'Apple e dello Spectrum). Quindi si devono sostituire le linee:

```
SETH TOWARDS :X :Y
FD 10
```

nelle procedure MOV.CAVALIERE e INSEGUE con:

```
MAKE "FRAC 10 / (SQRT((XCOR—:X) * (XCOR—:X) +
(YCOR—:Y) * (YCOR—:Y)))
SETPOS LIST (XCOR+(:X—XCOR) * :FRAC)
(YCOR + (:Y—YCOR) * :FRAC)
```

Sospendere l'esecuzione

Un microprocessore si occupa, ma solo apparentemente, di più operazioni simultanee

Per la gestione delle immissioni da tastiera, un computer ricorre comunemente alle "interruzioni" (interrupt). Infatti, se è il programma applicativo ad occuparsi della tastiera, può accadere che, passando attraverso il sistema operativo, vadano perdute alcune battute, effettuate mentre il programma sta eseguendo un'altra operazione. Inoltre, la perdita di un carattere, soprattutto di quelli che hanno un particolare significato (ad esempio, un ritorno-carrello), può anche verificarsi quando il processore è esclusivamente impegnato nella elaborazione di input da tastiera.

Questo problema viene solitamente risolto facendo in modo che la pressione di un qualsiasi tasto "richiami l'attenzione" del processore, che interrompe il lavoro in corso per dedicarsi all'esecuzione di una "routine di servizio dell'interruzione". Quest'ultima accetta il carattere digitato e lo deposita in un'area di memoria riservata, detta "buffer della tastiera". Quindi il processore riprende il lavoro interrotto come se nulla fosse accaduto.

Di conseguenza, quando viene chiamata la routine di immissione da tastiera del sistema operativo, essa non "legge" direttamente la tastiera, bensì estrae il primo carattere del buffer (oppure, se questo è vuoto, attende la comparsa di un carattere). Questa tecnica permette all'utente di procedere nella battitura, indipendentemente da ciò che compare sullo schermo, evitando la perdita di caratteri.

Tuttavia, possono verificarsi due inconvenienti. Può accadere che il ritmo di battitura sia talmente veloce, che la routine di immissione non riesca a "smaltire" in tempo i caratteri depositati nel buffer: in tal caso si verifica un "overflow" (superamento della capacità) del buffer. Ovviamente, basta dimensionare adeguatamente il buffer, ma bisogna far attenzione a non sprecare memoria preziosa. Il secondo problema, riguarda l'immediata comparsa sullo schermo del carattere battuto, a conferma visiva dell'accettazione: se ciò non avviene, l'utente è portato a premere più volte un tasto, facendo così confluire nel buffer (rischiando tra l'altro un overflow) decine di caratteri.

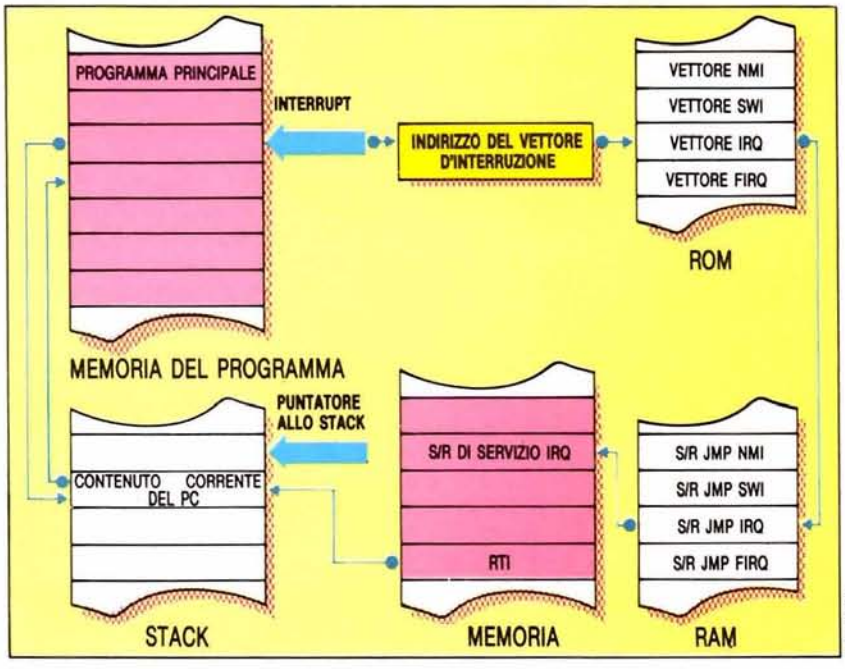
Le interruzioni sono particolarmente utili anche durante la fase di stampa, una procedura che talvolta rallenta moltissimo l'esecuzione di altre funzioni. Durante l'invio di caratteri alla stampante, può accadere che il processore impieghi 100 microsecondi per trasmettere un singolo carattere e debba poi fermarsi per alcuni millisecondi in attesa che la stampante lo elabori. Per eliminare questo ritardo viene usato un sistema detto "spooling" (ossia, esecuzione simultanea di operazioni periferiche e di un programma principale): i file da stampare vengono letteralmente messi "in coda", e trasferiti, un blocco alla volta, ad un secondo buffer. La stampante, inviando un segnale tramite la propria porta, interrompe il processore quando è pronta a ricevere un nuovo carattere. A questo punto la routine di servizio per l'interrupt invia un carattere estratto dal buffer, previo trasferimento in esso di un nuovo blocco, se il buffer dovesse risultare vuoto. Ciò consente al processore di effettuare una stampa, pur dedicandosi ad un altro lavoro, apparentemente simultaneo al primo.

Tipi di interruzioni

Durante alcune operazioni svolte dal processore (quelle di lettura o scrittura sui dischetti, ad esempio), la comparsa di una interrupt può causare la perdita di informazioni o effetti ancor più catastrofici. Occorre quindi un meccanismo che "mascheri" le interrupt, permettendo al processore di ignorarle durante l'esecuzione di operazioni particolarmente delicate. In tal caso, è utile segnalare

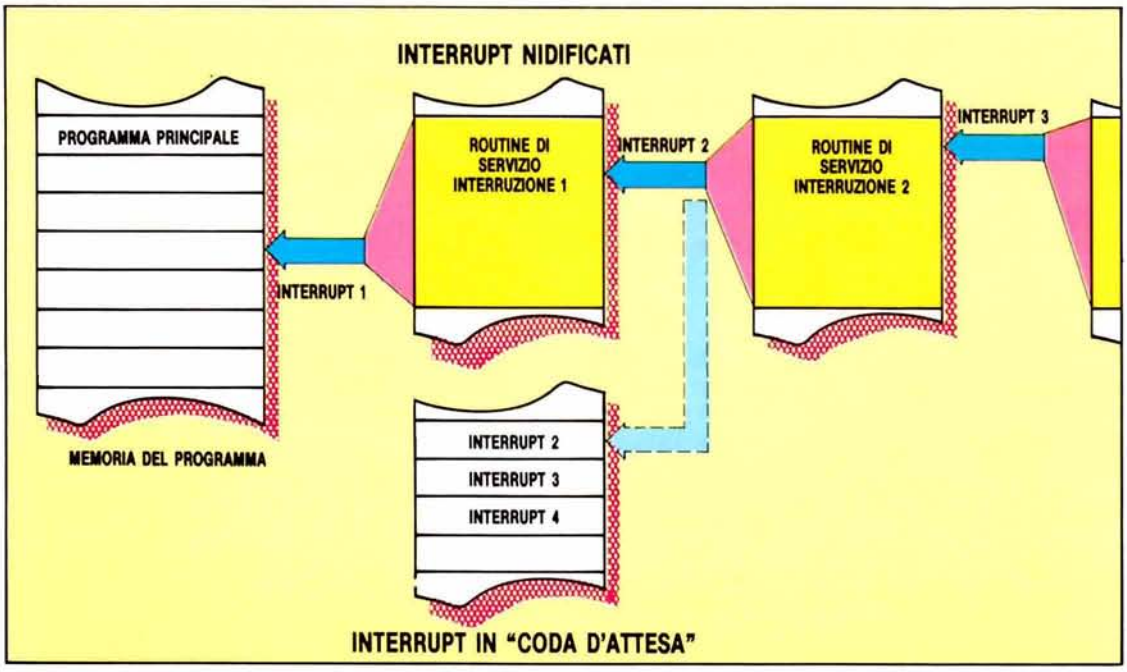
Gestione di un interrupt

Quando riceve una interrupt, il processore termina l'esecuzione dell'istruzione corrente ed appoggia sullo stack il contenuto del contatore di programma. Nel PC (Program Counter) viene caricato l'indirizzo del vettore associato all'interrupt, al quale poi viene trasferito il controllo. Tale indirizzo punta a sua volta ad un altro indirizzo, di solito in RAM, dove un'istruzione di salto (JMP) invia il controllo ad una particolare routine di servizio dell'interruzione. Questa deve terminare con l'istruzione RTI, che riporta il controllo al programma principale, usando l'indirizzo di ritorno precedentemente appoggiato sullo stack. Poiché l'istruzione JMP si trova in RAM, il programmatore può modificarla in modo che, ad ogni interrupt, il controllo passi ad una particolare routine (aggiunta dal programmatore), prima di eseguire la normale routine di servizio dell'interrupt



Interrupt... Interrotte

Se durante l'esecuzione di una interrupt, il processore ne riceve una seconda, una soluzione consiste nella nidificazione: all'arrivo della seconda interrupt, il PC viene appoggiato sullo stack, l'interrupt viene immediatamente eseguita ed il controllo rinviato all'indirizzo contenuto nello stack. I limiti di questa nidificazione dipendono dalle dimensioni dello stack e dalla possibilità, da parte degli strumenti che generano gli interrupt, di "sopportare" eventuali interruzioni nelle routine ad essi associate.
In alternativa, il processore può inserire le informazioni riguardanti successive interrupt in una "coda d'attesa". Completata la routine di servizio della prima interrupt, il processore estrae ed elabora quelle successive dalla "coda", finché questa non si svuota. Infine, il controllo torna al programma principale



(ad esempio tramite un flag) l'avvenuta richiesta d'interruzione, per poterla trattare in un momento successivo.
Alcune interfacce per dischetti, però, sono esse stesse pilotate da interrupt: è ovvio che queste devono avere la massima priorità, senza mai essere mascherate. Altri tipi di interruzioni "non mascherabili" possono provenire, ad esempio, da un circuito per l'individuazione di una caduta di tensione: le routine di servizio ad esse abbinate iniziano immediatamente a salvare il lavoro, finché c'è corrente.
Quando le interruzioni provengono da più periferiche, si parla di interruzioni "nidificate". Se il processore riceve un'interruzione mentre ne sta controllando un'altra, le alternative sono due: ignorare la seconda finché non ha terminato l'esecuzione della prima oppure classificare la nuova interruzione con un parametro d'urgenza: in questo modo le interruzioni con alta priorità vengono trattate prima di quelle con priorità inferiore.
L'istruzione SWI (vedere a pag. 1057) serve per generare, in un programma, particolari interruzioni che restituiscono il controllo al sistema operativo: sono dette "interruzioni da software" per distinguerle da quelle prodotte dall'hardware, viste fin qui. Inoltre, le interruzioni da software vengono spesso utilizzate come "breakpoint" (punti d'interruzione) per sospendere l'esecuzione di un programma in codice macchina e favorirne così la messa a punto (debugging). Questa tecnica è spesso applicata nei programmi "monitor" di controllo di codice macchina residenti in ROM.
Il 6809 dispone di tre diversi meccanismi d'interruzione: IRQ (Interrupt ReQuest), FIRQ (First Interrupt ReQuest) e NMI (Non-Maskable Interrupt). La selezione di questi diversi criteri avviene applicando appropriati segnali su tre piedini del microprocessore. Se la funzione è attivata da un segnale logico 0 (anziché da un 1) sulla sigla corrispondente viene posta una lineetta (come in $\overline{\text{IRQ}}$). I tre piedini sono connessi al bus principale, cosicché divengono accessibili ai chip periferici (come il 6820 e il 6850). Opportunamente programmati, questi chip possono trasmettere gli appropriati segnali d'interruzione al processore 6809 del computer.
Le interruzioni da software, tre anch'esse, vengono generate dalle istruzioni SWI, SW2 ed SW3.
Quando ha luogo un'interruzione, il controllo passa al "vettore" d'indirizzi, solitamente contenuto in specifiche locazioni poste all'estremità superiore della mappa di memoria. Questi vettori si trovano in ROM e trasferiscono il controllo sempre ad uno stesso indirizzo RAM, che contiene un'istruzione di salto (JMP). Il programmatore può quindi modificare l'istruzione di salto, cambiando la destinazione finale dell'interruzione, per eseguire una particolare utility.
Le locazioni di memoria sono:

Tipo d'interruzione	Vettore
$\overline{\text{NMI}}$	$FFFC
SWI	$FFFA
$\overline{\text{IRQ}}$	$FFF8
$\overline{\text{FIRQ}}$	$FFF6
SW2	$FFF4
SW3	$FFF2

È importante rilevare che i due byte superiori della memoria ($FFFE e $FFFF) contengono il "vettore di reset", ossia l'indirizzo al quale viene trasferito il controllo quando il computer viene acceso o azzerato. Di solito, questo è l'indirizzo iniziale del programma "monitor" ROM. Inoltre, la Motorola ha riservato i due byte con indirizzo $FFF0 e $FFF1 per eventuali applicazioni future.
I dati riguardanti le interruzioni sono conte-

nuti in tre bit del registro di stato (CC): i bit 4 (I), 6 (F) e 7 (E). Il bit I posto a 1 maschera l'interruzione IRQ, il bit F l'interruzione FIRQ. Il processore usa il bit E per distinguere IRQ o NMI da FIRQ: E viene posto ad 1 in presenza di un'interruzione IRQ o MNI a 0 per FIRQ.

Il processore tratta le interruzioni in modo analogo alle chiamate di subroutine e il contenuto di alcuni o di tutti i registri viene appoggiato sullo stack. Così facendo, l'esecuzione del programma può tranquillamente riprendere dal punto in cui era stata interrotta. La routine di servizio dell'interruzione termina con l'istruzione RTI (simile a RTS), che recupera i valori appoggiati sullo stack e ritrasferisce il controllo al programma originario.

FIRQ si differenzia dai due comandi d'interruzione, perché appoggia sullo stack solo il contenuto del contatore di programma (PC) e del registro di stato (CC); è quindi più veloce di IRQ ed NMI. Poiché la routine di servizio dell'interruzione deve riportare allo stato originario tutti i registri che usa, FIRQ non può essere impiegato con routine che adoperano più di uno o due registri. Vediamo adesso il funzionamento del bit E. Dato che viene usata la stessa istruzione RTI per chiudere le utility d'interruzione IRQ e FIRQ, il processore ricorre al bit E per determinare quali registri debbano essere riportati allo stato iniziale. La procedura è la seguente:

1) Viene eseguita l'istruzione corrente.
2) Il bit I, che maschera le interrupt IRQ, viene posto a 1. Se il processore ha ricevuto una interrupt FIRQ o NMI, viene posto a 1 il bit F, che disabilita FIRQ. SWI, contrariamente a SW2 ed SW3 non maschera le altre interrupt.
3) Se si tratta di una FIRQ, il bit E viene azzerato, altrimenti posto ad 1.
4) Il vettore contenuto nelle appropriate locazioni viene caricato nel contatore di programma e l'esecuzione prosegue da tale indirizzo.

Il primo programma presentato riguarda una particolare applicazione delle interruzioni: il controllo di un clock in tempo reale. Supponiamo di aver collegato ad un PIA (alla locazione $5000) uno circuito di temporizzazione (che può essere un chip specializzato, come il 6840, oppure un semplice divisore della frequenza di clock del sistema o di quella di rete). La prima subroutine abilita le interrupt e predispone un contatore a 16 bit all'indirizzo $50. La routine di servizio dell'interruzione si limita a incrementare il contatore. Esaminando, in qualsiasi momento, il contenuto di $50 si legge il numero dei segnali di temporizzazione ricevuti: da questi si può ricavare il tempo trascorso, conoscendo il suo valore iniziale e la frequenza dei segnali di temporizzazione.

Per il secondo programma occorre collegare una stampante al PIA (alla locazione $E000). Il buffer all'indirizzo $100, di lunghezza non precisata, viene usato per immagazzinare una sequenza di caratteri che viene in seguito trasmessa alla stampante dalla routine di servizio. Il flag all'indirizzo $50 viene azzerato durante la stampa della linea e posto ad uno al termine della stampa. Ciò consente di abilitare o meno una serie di altre routine (qui non specificate) per il successivo riempimento del buffer. Le locazioni $51 e $52 contengono un puntatore, che individua il "prossimo" carattere da stampare. La prima subroutine predispone il PIA, il flag e il buffer per la stampa di una nuova linea.

PRIMO PROGRAMMA

```
PIACR   EQU   $E001     Registro di controllo PIA
PIADR   EQU   $E000     Registro dati PIA
INTRP   EQU   $2000
CLK1    EQU   $50
CLK2    EQU   $51
        ORG   $1000     S/R di inizializzazione del clock
INITCK  CLR   CLK1      Azzera le locazioni del clock
        CLR   CLK2
        LDA   #%00000101  Abilita le interrupt PIA
        STA   PIACR
        ANDCC #%11101111  Abilita IRQ
WAITCK  TST   CLK2      Attesa per il primo incremento
        BEQ   WAITCK
        RTS
        ORG   INTRP     Routine di servizio dell'interruzione
        LDA   PIADA     Cancella l'interrupt
        LDD   CLK1      Legge il contatore
        ADDD  #1        Incrementa il contatore
        STD   CLK1
        RT1
```

SECONDO PROGRAMMA

```
PIACR   EQU   $E001     Registro di controllo PIA
PIADR   EQU   $E000     Registro dati PIA
INTRP   EQU   $2000
CR      EQU   13        Ritorno-carrello
BUFFER  EQU   $100      Indirizzo del buffer
BUFPTR  EQU   $51       Puntatore al buffer
FLAG    EQU   $50       Flag di "fine linea"
        ORG   $1000     S/R di preparazione del sistema
        CLR   FLAG      Azzera il flag di fine linea
        LDX   #BUFFER   Inizializza il puntatore al buffer
        STX   BUFPTR
        CLR   PIACR     Indirizzo registro direzione dati
        LDA   $FF       Pone tutte le linee in uscita
        STA   PIADR     Abilita le interrupt PIA
        LDA   #%00000101
        STA   PIACR     Abilita IRQ
        ANDCC #%11101111
        RTS             Routine di servizio dell'interruzione
        ORG   INTRP     Puntatore al buffer
        LDX   BUFPTR    Legge carattere success. dal buffer
        LDA   ,X+       Lo stampa
        STA   PIADR     Azzera le interrupt
        LDB   PIADR     Incrementa il puntatore al buffer
        STX   BUFPTR    Fine della linea ?
        CMPA  #CR       Prosegue se la linea non è finita
        BNE   FINITO    Altrimenti pone a 1 il flag
        INC   FLAG
FINITO  RTI
```

Una utility per il Commodore 64

La routine di ricerca dei nomi di variabili

Come già accennato in precedenza, i nomi delle variabili vanno scelti con molta cura, evitando quelli che contengono parole riservate del BASIC. Ad esempio, non si possono usare NEWLINEA o INTERO, perché iniziano rispettivamente con NEW e INT, che sono due "parole chiave" del BASIC.
Le modifiche all'inizio del programma, rispetto alle versioni precedenti, sono dovute al diverso sistema usato per conservare in memoria le linee di BASIC. Sul Micro BBC e sullo Spectrum, la linea, espressa nel suo "formato interno", inizia con un numero di linea a due byte (il byte alto precede quello basso), seguiti dalla lunghezza della linea, anch'essa a due byte. Sul Commodore 64, invece, una linea BASIC inizia con un puntatore di due byte (che contiene l'indirizzo iniziale della linea successiva) e con un numero di linea a due byte (il byte basso precede quello alto).
Il programma Commodore si limita a saltare le linee REM e le stringhe tra virgolette, senza prendere in esame gli altri casi, ossia i numeri esadecimali come sul Micro BBC o l'equivalente binario dei numeri come sullo Spectrum.

PROGRAMMA DI RICERCA PER COMMODORE 64

```
30000 INPUT "NOME DA CERCARE";T$
30010 RE=143
30020 VIRGOL=34
30030 NL=0
30040 PUNTXT=2049
30050 NUOVALIN=PEEK(PUNTXT) + 256 * PEEK(PUNTXT+1)
30070 PUNTXT = PUNTXT +2
30080 NUMLIN=PEEK(PUNTXT) + 256 * PEEK(PUNTXT+1)
30085 IF NUMLIN >=30000 THEN END
30090 PUNTXT =PUNTXT + 2
30140 IF PEEK(PUNTXT)=NL THEN PUNTXT=PUNTXT+1:GOTO30050
30150 IF PEEK(PUNTXT)<>RE THEN GOTO 30180
30160 REM SALTA LE LINEE REM
30170 PUNTXT =NUOVALIN:GOTO 30050
30180 IF PEEK(PUNTXT)<>VIRGOL THEN GOTO 30300
30190 REM SALTA I CARATTERI TRA VIRGOLETTE MA SI FERMA ALLA
      FINE DELLA LINEA SE MANCANO LE VIRGOLETTE DI CHIUSURA
30200 PUNTXT = PUNTXT + 1
30210 IF PEEK(PUNTXT)=NL THEN PUNTXT=PUNTXT+1: GOTO 30050
30220 IF PEEK(PUNTXT)<>VIRGOL THEN GOTO 30200
30230 PUNTXT = PUNTXT + 1
30235 GOTO 30140
30290 REM IL PRIMO CARATTERE DEL NOME DEVE ESSERE UNA LETTERA
30300 IF PEEK(PUNTXT)>=ASC("A") AND PEEK(PUNTXT)<=ASC("Z") THEN GOTO 30330
30310 PUNTXT = PUNTXT + 1
30320 GOTO 30140
30330 NOME$= ""
30340 NOME$=NOME$+CHR$(PEEK(PUNTXT))
30350 PUNTXT = PUNTXT + 1
30360 REM LETTERE O NUMERI DOPO IL PRIMO CARATTERE
30370 IF PEEK(PUNTXT)>=ASC("A") AND PEEK(PUNTXT)<=ASC("Z") THEN GOTO 30340
30390 IF PEEK(PUNTXT)>=ASC("0") AND PEEK(PUNTXT)<=ASC("9") THEN GOTO 30340
30410 REM LE VARIABILI STRINGA TERMINANO CON $, QUELLE INTERE CON %
30420 IF PEEK(PUNTXT)=ASC("$")THEN NAME$=NAME$+"$":PUNTXT=PUNTXT+1: GOTO 30450
30430 IF PEEK(PUNTXT)=ASC("%")THEN NAME$=NAME$+"%":PUNTXT=PUNTXT+1
30440 REM LE VARIABILI DI MATRICE TERMINANO CON (
30450 IF PEEK(PUNTXT)=ASC("(")THEN NAME$=NAME$+"(":PUNTXT=PUNTXT+1
30460 IF NOME$=T$ THEN PRINT NOME$;" ALLA LINEA ";NUMLIN
30470 GOTO 30140
30480 END
```

La sezione del programma che estrae i nomi delle variabili cerca quelli che iniziano con una lettera alfabetica, quindi esamina le lettere o i numeri successivi ed infine i simboli $, % e (che indicano, rispettivamente, le variabili stringa semplici, le variabili intere e quelle che indicano una funzione o una matrice. Diversamente dal Micro BBC e dallo Spectrum, il Commodore 64 non accetta caratteri di sottolineatura all'interno dei nomi di variabili. Il Commodore può visualizzare tanto lettere maiuscole quanto minuscole, ma i loro codici di carattere sono quelli delle maiuscole: la procedura di ricerca si limita quindi ad una ricerca delle lettere maiuscole.
La versione Commodore funziona in modo simile a quelle già viste. Prima si trascrive e si registra il programma di ricerca, poi si carica in memoria quello da esaminare, eseguendo un "merge" col precedente. Infine, digitando "RUN 30000", si avvia il programma di ricerca. Esso chiede di immettere il nome della variabile da cercare, seguito dal simbolo "(" se si tratta di un nome di matrice.
Sul Commodore 64, prima di unire i due programmi in memoria, occorre controllare che i numeri di linea del primo siano più piccoli di quelli del secondo. La procedura di collegamento usa due puntatori in pagina zero: TXTTAB (indirizzi 43 e 44), che contiene l'indirizzo iniziale del programma BASIC e VARTAB (indirizzi 45 e 46). Un programma BASIC termina sempre con tre byte che contengono degli 0: il primo indica la fine dell'ultima linea di programma, gli altri due la fine del programma. Per collegare due programmi va caricato per primo il programma che ha i numeri di linea più bassi, quindi battere l'istruzione:

```
PRINT PEEK(45), PEEK(46)
```

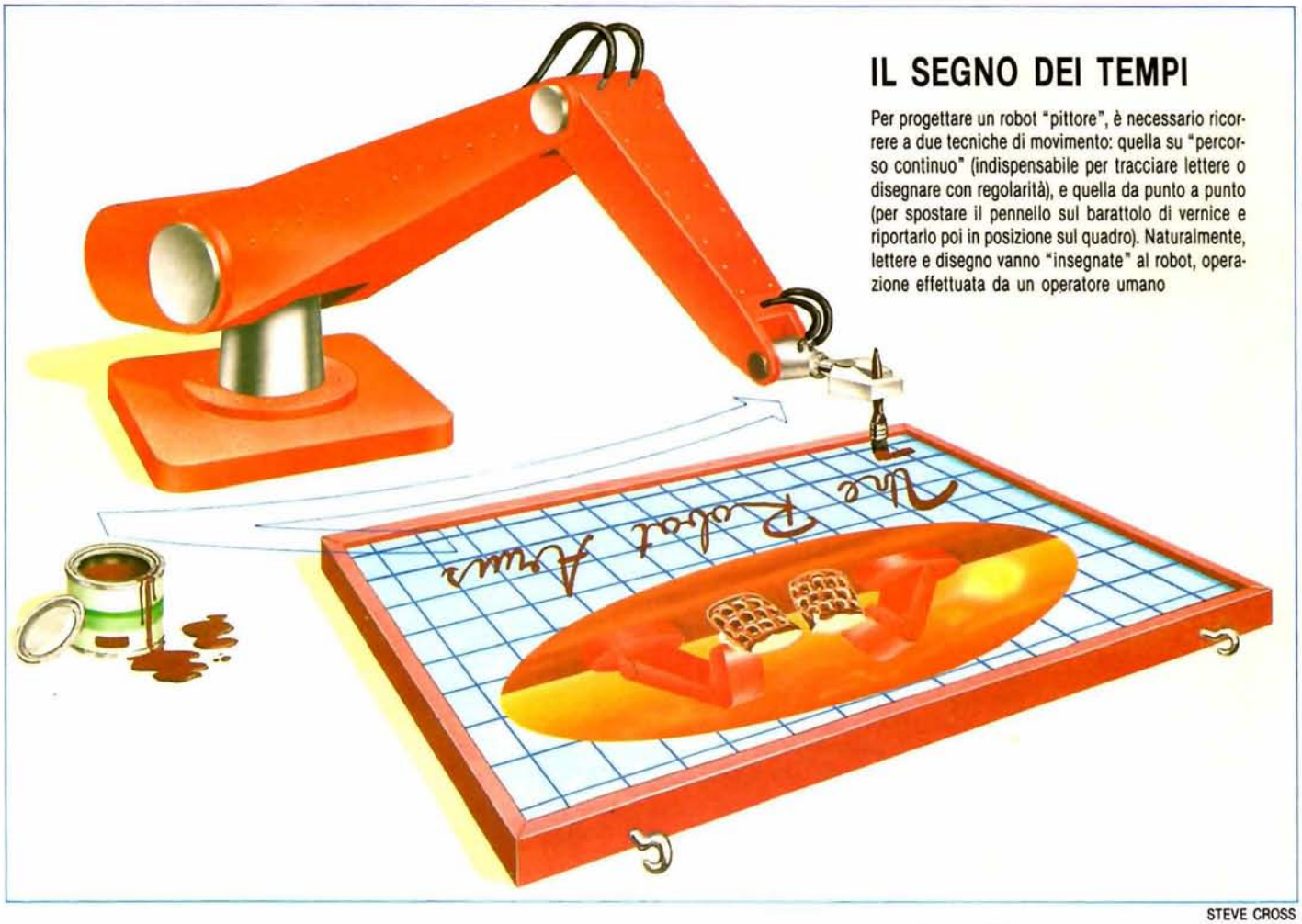
Se il primo numero è compreso tra 2 e 255, si sottrae 2 e si deposita il risultato all'indirizzo 43. Se invece il primo numero è uno zero o un uno, all'indirizzo 43 vanno depositati, rispettivamente, i valori 254 e 255. In tal caso, nella locazione 44 va depositato il risultato di PEEK(46), diminuito di uno. Infine, si carica il secondo programma e si digita:

```
POKE 43,1: POKE 44,8
```

Ciò riporta il puntatore all'indirizzo iniziale del programma BASIC al suo valore normale. Ora i programmi sono collegati.

Movimenti di precisione

Non bastano sofisticati meccanismi: occorre anche una certa dose di "intelligenza"



IL SEGNO DEI TEMPI

Per progettare un robot "pittore", è necessario ricorrere a due tecniche di movimento: quella su "percorso continuo" (indispensabile per tracciare lettere o disegnare con regolarità), e quella da punto a punto (per spostare il pennello sul barattolo di vernice e riportarlo poi in posizione sul quadro). Naturalmente, lettere e disegno vanno "insegnate" al robot, operazione effettuata da un operatore umano

STEVE CROSS

Abbiamo visto come la realizzazione di bracci robotizzati si ispiri alla conformazione umana: uno "scheletro" per sostenere la struttura e "muscoli" per fornire potenza motoria. Tuttavia, manca ancora un elemento per completare il tutto: l'"intelligenza".
Ovviamente, in questo caso non si tratta di intelligenza ad alto livello, quale può essere quella umana, bensì di qualcosa molto più elementare. Consideriamo un semplice processo motorio umano: lo spostamento di un oggetto posto su di un ripiano. Sono implicate due forme d'intelligenza: la prima concerne la percezione del piano e dell'oggetto e la seconda la decisione di muovere l'oggetto da un luogo all'altro. Ciò implica un "pensiero cosciente", strettamente connesso a concetti quali "intenzione" e "comportamento finalizzato". Per un braccio robotizzato è sufficiente un'intelligenza di livello molto inferiore: basta solo che esegua con precisione l'azione di spostamento, senza esser coinvolto nell'aspetto decisionale.

Apprendistato umano

Un simile processo sembra semplice e banale ma, se si dubita che sia in effetti un atto intelligente, si pensi ai tentativi fatti da un bambino molto piccolo nel compiere questa semplice azione: probabilmente, non riuscirà ad afferrare l'oggetto al primo colpo, e poi non riuscirà a collocarlo nella posizione richiesta. Il bambino, non ancora in grado di coordinare alla perfezione i propri movimenti, sta ancora imparando a muovere le proprie braccia e le proprie mani in un ambiente tridimensionale. Tuttavia, una volta acquisito il processo mentale, i movimenti divengono automatici e naturali, senza richiedere alcun pensiero "cosciente", tanto da farci

credere che non richiedano alcuna forma d'intelligenza.
La situazione di un braccio robotizzato è la stessa: deve "imparare" a muoversi con precisione, prima di poter compiere qualsiasi azione richiesta.
Il più semplice metodo per insegnare movimenti ad un robot è quello di pilotarlo attraverso tutta la sequenza di spostamenti, facendo in modo che esso "ricordi" tale sequenza. Questo metodo è largamente applicato ai robot industriali, soprattutto perché non occorre che l'operatore, facente le funzioni di "insegnante", conosca a fondo il funzionamento del braccio: basta che lo guidi nella corretta sequenza di movimenti. Né, del resto, è necessario che il robot "sappia" cosa sta facendo: basta che si ricordi la particolare sequenza di azioni.

Metodi di addestramento

Esistono due metodi per "insegnare" ai robot: quello cosiddetto "da punto a punto" e quello su "percorso continuo". Nel primo l'operatore sposta il braccio in una posizione e preme un apposito pulsante, per segnalare al robot che la posizione va memorizzata; il braccio viene poi spostato in un nuovo punto e il processo prosegue finché tutta un'intera sequenza è depositata nella memoria del robot. Dopodiché, ogni volta che la sequenza viene richiamata, il braccio compie tutte le azioni "imparate". Nell'addestramento su "percorso continuo", invece, il robot impara tutta la serie dei movimenti pilotati dall'operatore, senza riferirsi soltanto a particolari punti del percorso. La scelta del metodo dipende dalla funzione del braccio. Il primo è adatto ad impieghi del tipo «raccogli e sposta», simile a quello dell'esempio del bambino, in cui esistono delle posizioni terminali ben precise. Se, invece, il braccio viene impiegato nella verniciatura a spruzzo di una piastra, per esempio, allora il movimento deve essere continuo, senza scatti: ecco dunque che il secondo metodo si rivela indispensabile.
Esaminiamo adesso il criterio di memorizzazione dei movimenti. In genere, durante la fase di apprendimento, vengono rilevati i dati forniti dai sensori collocati in ciascuna giunzione o articolazione del robot. Normalmente, si tratta di codificatori assiali, la cui lettura viene tradotta in valori numerici, depositati poi in memoria oppure, per un apprendimento più durevole, su nastro o dischetto. In seguito, la sequenza viene richiamata ed ogni valore corrispondente a ciascuna articolazione viene convertito ed inviato all'appropriato motore.
Sorprendentemente, il robot apprende più facilmente un percorso continuo, sebbene ciò implichi l'uso di una maggiore quantità di memoria (spesso migliaia di informazioni per un semplice movimento), rispetto ai pochi dati occorrenti per uno spostamento da punto a punto. Tuttavia, per ottenere continuità nei movimenti bisogna risolvere un problema non indifferente: più articolazioni vanno attivate contemporaneamente, affinché non vi siano indesiderati scatti negli spostamenti. Un robot "verniciatore" deve sapersi muovere lungo tre assi, e nello stesso tempo il suo "polso" deve adattarsi per dirigere lo spruzzo in modo corretto. I casi sono due: adoperare un computer estremamente veloce, oppure impiegare un processore per ogni articolazione. Solo la seconda soluzione produce movimenti realmente simultanei.

Movimenti calcolati

I robot che operano col criterio da punto a punto hanno una vita più complicata: pur

GEOMETRIA SPAZIALE

Nei movimenti "da punto a punto", un braccio a due giunti deve ruotare attorno alla propria base (angolo R), oltre a variare l'inclinazione della "spalla" (S) e del gomito (G). Se le coordinate cartesiane del punto di partenza e di arrivo sono (X1,Y1,Z1) e (X2,Y2,Z2), allora le variazioni si calcolano con:

A1 = SQR(X1 + Y1 + Z1)
A2 = SQR(X2 + Y2 + Z2)

Base:

R1 = ARCTAN(Y1/X1)
R2 = ARCTAN(Y2/X2)
Variazione = (R2 − R1)

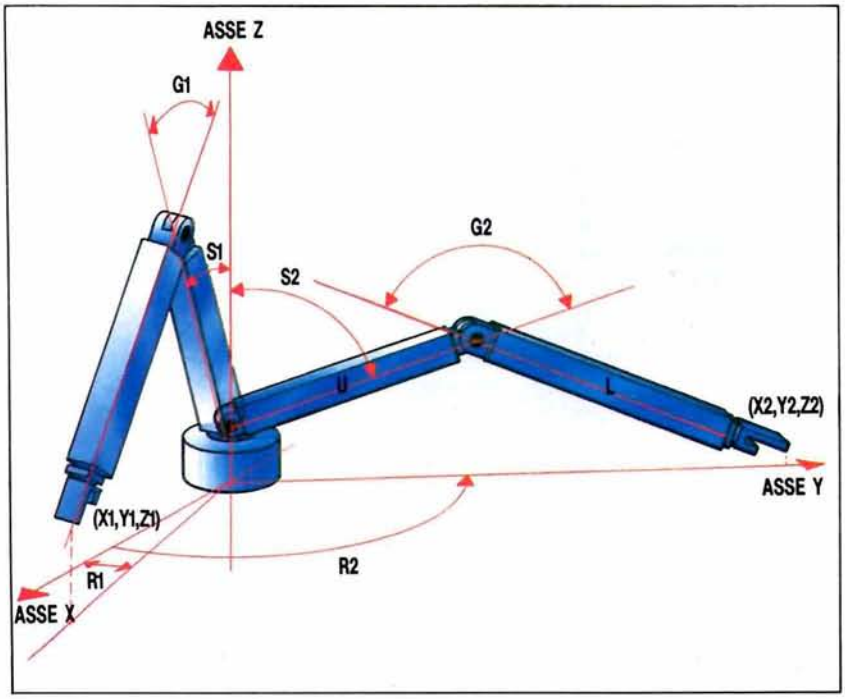
Spalla:

S1 = ARCTAN(Z1/A1) + ARCCOS((A1 + U − L)/(2 * A1 * U))
S2 = ARCTAN(Z2/A2) + ARCCOS((A2 + U − L)/(2 * A2 * U))
Variazione = (S2 − S1)

Gomito:

G1 = ARCCOS((U + L − A1)/(2 * U * L))
G2 = ARCCOS((U + L − A2)/(2 * U * L))
Variazione = (G2 − G1)

dove U e L sono, rispettivamente, la lunghezza della sezione superiore e di quella inferiore del braccio



conoscendo il punto di partenza e quello di arrivo, non viene indicato loro il percorso da seguire per passare dall'uno all'altro. Una soluzione banale, ma non certo applicabile in pratica, sarebbe quella di muovere tutte le articolazioni finché non viene rilevata la posizione finale. Ha molta più efficacia un movimento diretto lungo il percorso più breve, che simula un comportamento più economico e quasi umano. Ciò implica calcoli complessi per convertire le coordinate cartesiane dei punti appartenenti alla retta che congiunge le due posizioni del braccio (per le quali è impiegato un sistema di coordinate totalmente diverso). Pertanto, il robot deve risolvere problemi geometrici non indifferenti, se deve lavorare con precisione. Inoltre, nelle applicazioni industriali, si tratta spesso di trasportare per distanze considerevoli oggetti di notevole peso: la scelta di un percorso ottimale può evitare enormi sprechi di tempo e di denaro.

Un ulteriore aspetto da considerare, nei robot che si muovono da punto a punto, è quello dinamico. In un movimento umano, infatti, il braccio aumenta la propria velocità, allontanandosi dal punto di partenza, fino a raggiungere quella massima. Poi decelera fino a posizionarsi con precisione sul punto di arrivo. In un braccio robotizzato, invece, la velocità massima è raggiunta quasi istantaneamente e, con altrettanta immediatezza, il braccio si arresta di colpo nel punto di arrivo. Ciò sottopone le articolazioni a notevoli sforzi e richiede una potenza maggiore, rispetto a bracci che si muovono con continuità. Come conseguenza non trascurabile, specialmente se l'oggetto da afferrare è delicato, la minima imprecisione del movimento potrebbe risultare in un poderoso colpo sull'oggetto, magari frantumandolo. Oltre ad un percorso ottimale, quindi, va calcolata la velocità più adatta allo spostamento.

Coreografia nei robot

Nonostante il braccio robotizzato venga programmato con cura, può darsi che i movimenti risultanti, in seguito, non corrispondano con esattezza a quanto voluto. Ciò può dipendere da un errore dell'operatore, o da leggeri mutamenti del compito o, infine, perché due robot entrano in collisione, muovendosi sullo stesso percorso. Per evitare quest'ultimo inconveniente, noto come "coreografia per robot", occorre dunque un metodo che consenta di correggere le sequenze impostate. Un metodo di memorizzazione dei movimenti, che facilita la correzione, consiste nel conservare la sequenza sotto forma di lista concatenata: ogni posizione memorizzata è seguita dall'indirizzo al quale si trova la posizione successiva. Se occorre modificare la sequenza, basta posizionare il braccio nel punto immediatamente precedente la modifica ed inserire la variazione.

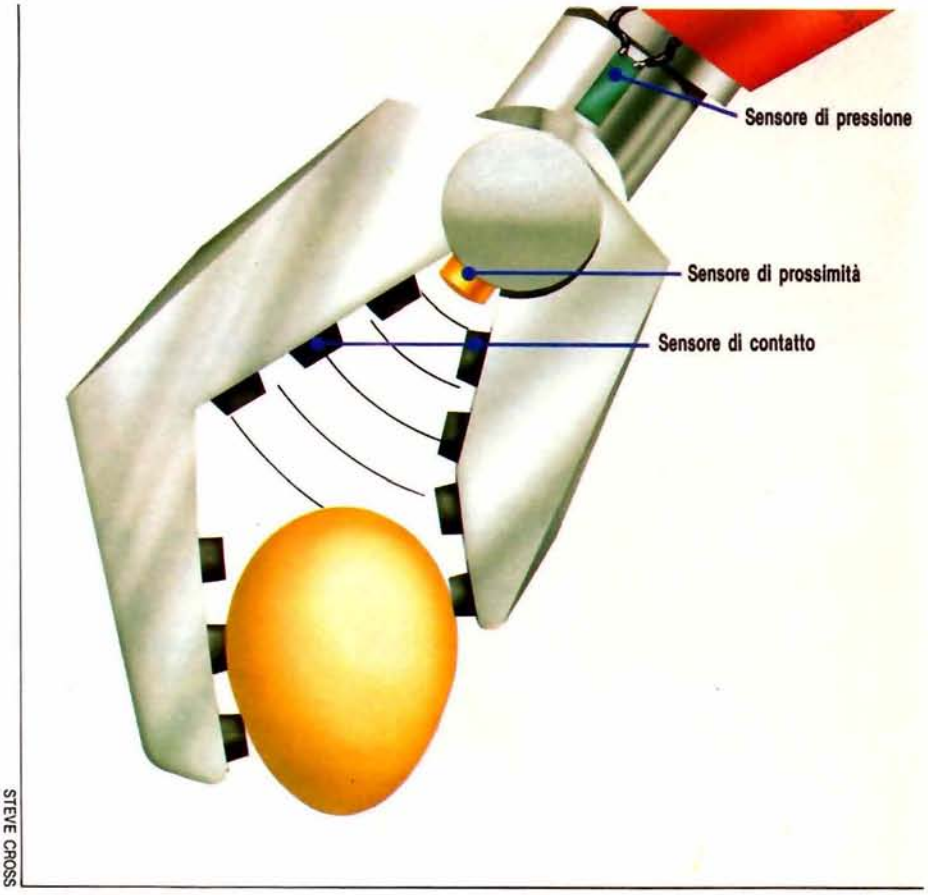
Un altro diffuso metodo di controllo consiste in una serie di istruzioni (un vero e proprio programma) redatto in un particolare linguaggio. I linguaggi variano in base al modello e alla marca del braccio, ma assomigliano tutti, più o meno, al LOGO: esistono comandi di movimento per ciascuna articolazione, oltre ad istruzioni del tipo «afferra» o «lascia la presa».

I problemi di programmazione sono affini al metodo di apprendimento da punto a punto: ogni movimento si può ottenere in più modi diversi, agendo sulle varie articolazioni, quindi va scelta la sequenza ottimale.

Infine, va considerato l'aspetto concernente la "presa" della mano. Nonostante l'accuratezza nel posizionamento, non è detto che afferrare l'oggetto sia semplice, specialmente se questo ha una forma asimmetrica: è necessaria una mano "intelligente", capace di rilevare la presenza o l'assenza dell'oggetto e la sua distanza, e di valutare infine la forza da usare nella presa. Per far ciò, la mano viene equipaggiata di appositi sensori di prossimità, di pressione e di contatto che informano il computer di controllo (tramite un feedback), in modo da agire di conseguenza.

Abbiamo così esaminato i problemi connessi alla realizzazione di bracci robotizzati dotati di una forma di "intelligenza" molto elementare. Per ottenere robot capaci di azioni complete, dettate da criteri complessi, occorre progettare una forma di controllo più sofisticata.

Un meccanismo in grado di evitare ostacoli, ad esempio, costituisce già un notevole passo avanti.



STEVE CROSS

Delicatezza meccanica

Raccogliere un uovo è una delle prove decisive sulla precisione dei sensori e del sistema di feedback di un braccio robotizzato. Il sensore di prossimità deve verificare che la distanza sia adatta alla presa, prima che le dita inizino a stringere. I sensori di contatto rilevano quando l'uovo è stato afferrato. A questo punto, il controllo del feedback diventa molto critico: una diminuzione nella prossimità indica che l'uovo sta sfuggendo alla presa, cosicché le dita devono applicare una maggiore forza, fino ad un limite massimo prestabilito oppure fino a che il sensore di pressione indichi una leggera deformazione della superficie dell'oggetto

Utility ibride

Utility scritte in codice macchina possono intervenire nei programmi BASIC

Per utilizzare il programma di ricerca delle variabili, occorre "fonderlo" assieme al programma sotto verifica. Con questo metodo, l'unica informazione da estrarre dal sistema operativo è l'indirizzo iniziale del programma BASIC (il termine è trovato dallo stesso programma di ricerca).
Il programma che intendiamo realizzare adesso, invece, provvede a cercare ed a sostituire nomi di variabili. Ciò può tornare molto utile se, ad esempio, si è scoperto che il nome adoperato per una variabile non è consentito oppure se, volendo consegnare un programma a terzi, si vuol rendere meno comprensibile il significato delle variabili impiegate. In entrambi i casi, un programma che esegua automaticamente le sostituzioni è utilissimo. Tale utility, però, va scritta in codice macchina: iniziamo così lo sviluppo teorico del programma (il listato verrà presentato in seguito).

Spazio in memoria

In questo caso, l'utility va collocata in una zona di memoria ben distinta da quella del programma sotto esame. Inoltre, occorre un nuovo sistema per individuare la fine del programma BASIC, ed un meccanismo che consenta di avere in memoria, allo stesso tempo, due programmi BASIC.
Sui tre computer qui esaminati (lo Spectrum, il Commodore 64 ed il BBC) la posizione del programma BASIC viene comunicata al sistema operativo e all'interprete BASIC tramite puntatori. Sfortunatamente, però, i dettagli dei tre apparecchi differiscono.
Sul Micro BBC, ci sono quattro importanti puntatori. Due sono PAGE e TOP, che contengono l'indirizzo iniziale e quello finale del programma BASIC. Gli altri due sono LOMEM, che contiene l'indirizzo iniziale dell'area riservata alle variabili, e HIMEM, che contiene l'indirizzo finale dell'area BASIC. Tali puntatori sono accessibili tramite le omonime variabili riservate del BASIC. Se si desidera la presenza di due programmi BASIC in memoria, basta variare PAGE, affinché sia maggiore di TOP (usando il comando OLD per aggiustare TOP e LOMEM), per poi caricare il secondo programma senza danneggiare il primo. Per passare dall'uno all'altro, occorre assegnare nuovi valori a PAGE e HIMEM, usando nuovamente il comando OLD.
Una volta realizzata l'utility, questa deve poter intervenire sui puntatori per trovare l'inizio e la fine di ciascun programma, cosicché occorre copiarne il contenuto in locazioni protette da eventuali sovrascritture. Un metodo alternativo, per trovare la fine del programma BASIC, consiste nella ricerca del marcatore di "fine linea" (il cui valore è di 128 o più), inserito dall'interprete BASIC in fondo a ciascuna linea di programma dopo il "ritorno carrello". Il byte cercato, e quello successivo normalmente indicano il numero della linea seguente successiva (nel formato HI-LO): se il valore "alto" della coppia è uguale o maggiore di 128, ciò significa che il numero di linea è superiore al massimo consentito, ossia 32767, quindi si è trovata la fine del programma.
Il Commodore 64 adopera sette puntatori, collocati nella pagina zero della memoria. TXTTAB (indirizzo 43 e 44), punta all'inizio del programma BASIC; VARTAB, ARYTAB, STREND, FRETOP e FRESPC (indirizzi da 45 a 54), puntano alle varie sezioni dell'area per le variabili; MEMSIZ (indirizzo 55 e 56), infine, punta alla fine dell'area BASIC. Tramite opportune POKE, è possibile variare tali puntatori per riservare una zona di memoria a un secondo programma. Tuttavia, è molto più sicuro affidare simili cambiamenti a un programma in codice macchina, evitando così le conseguenze di eventuali errori di battitura nelle POKE, che farebbero bloccare il computer.
Nel Commodore 64, il termine di un programma BASIC è indicato da due byte di valore 0, immediatamente successivi al byte 0, che segna la fine di ciascuna linea di programma. Pertanto, basta seguire la catena di puntatori situati all'inizio di ciascuna linea di programma, finché non si trova un puntatore di valore 0.

Per lo Spectrum

Nel caso dello Spectrum, la realizzazione dell'utility è leggermente più complessa, poiché la memoria non viene suddivisa in precise sezioni raggiungibili tramite variabili, e ciò impedisce la possibilità di caricare contemporaneamente due programmi BASIC. Procederemo copiando il programma in una zona di memoria superiore a RAMTOP e lavorando su tale copia. Ciò lascia insoluto il problema di come riportare la versione modificata nella zona originaria: occorre un breve programma in codice macchina che provveda a questo.
Il manuale dello Spectrum fornisce ampie

informazioni su come vengono memorizzati i programmi BASIC e la destinazione delle varie aree di memoria. Tuttavia, è difficile redigere utility senza ricorrere alle subroutine in codice macchina presenti nelle ROM, su cui il manuale, invece, è poco esauriente. Fortunatamente, esiste una vasta letteratura sull'argomento, facilmente ottenibile presso rivenditori specializzati. Due importanti subroutine di sistema, impiegate nella nostra utility, sono quella che riserva e quella che recupera aree di lavoro in memoria.

ESPERIMENTI IN BASIC

Modificare il contenuto di un programma BASIC, durante l'esecuzione, è possibile, ma prima conviene salvare il programma su nastro o disco, poiché è facile far "inchiodare" il sistema a seguito dei cambiamenti.
Si adoperi il programma Monitor (a pagina 598), che permette di ispezionare e modificare le locazioni di memoria: anche quelle dello stesso programma Monitor.
Si inseriscano alcune linee REM all'inizio del programma, poi si eseguano le seguenti prove.

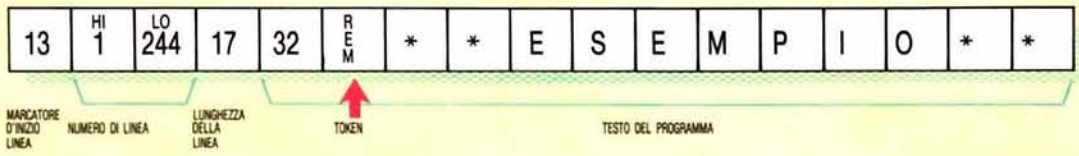
- Si trovi l'inizio dell'area testuale del BASIC (vedere a pag. 538) e si ispezioni la memoria fino a riconoscere le varie linee del programma Monitor.
- Si modifichi il valore successivo al token di una REM. Poi si esca dal programma e si LISTi la linea in questione.
- Si esegua una prova come quella precedente, ma inserendo un valore superiore a 127, listando poi la linea.
- Si provi ad alterare il numero di linea: i risultati sono imprevedibili, specialmente se il nuovo valore è "fuori sequenza" rispetto agli altri.
- Si provi a modificare il valore che indica la lunghezza di una linea (ricordandosi però di inserire un marcatore di "fine linea" nella nuova posizione specificata.
- Sul Commodore 64 si provi a modificare il valore del puntatore "alla prossima linea", sostituendolo con quello contenuto nella linea successiva, poi si dia un LIST.
- Consultando il manuale del computer, si provi ad individuare, tra le varie zone di memoria riservate, quella per le variabili. Esistono fino a sei tipi diversi di variabili: numeriche semplici, a matrice, intere, intere a matrici, stringhe e matrici di stringhe. Le variabili intere e le stringhe sono le più facili da individuare.
- Si provi a modificare il valore di uno o più token all'interno di una linea: nel listato le parole chiave risulteranno diverse dalle originali. Se ciò viene fatto sullo stesso programma Monitor, in corso d'esecuzione, i risultati possono essere paradossali.

I PROGRAMMI BASIC IN MEMORIA

Quasi tutti i micro adottano lo stesso criterio per codificare le linee di BASIC in memoria. Ogni linea di programma inizia con informazioni concernenti il numero di linea (nei primi due byte) ed eventualmente sulla sua lunghezza. Il testo dei commenti e delle variabili stringa costanti rimane generalmente inalterato, mentre i comandi e le istruzioni vengono sostituite da speciali codici di un solo byte (i "token")

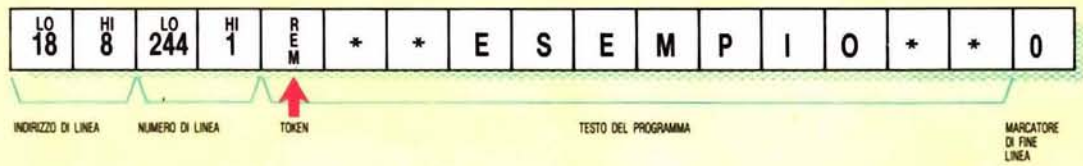
500 REM ** ESEMPIO **

Versione BBC



In questo formato, il codice 13 (ASCII per il "ritorno carrello") è usato come marcatore per l'inizio della linea (mentre in genere si trova al termine della linea). La lunghezza è espressa da un solo byte, quindi è limitata a 255 caratteri. Si noti che viene conservato anche lo spazio subito seguente il numero di linea

Versione COMMODORE 64



I due byte iniziali contengono l'indirizzo di Link, ossia il puntatore all'inizio della successiva linea di BASIC: siccome questa linea è lunga 17 byte e l'area BASIC inizia da 2049 tale indirizzo corrisponde a 2066 (0812 in hex)

Versione SPECTRUM



I criteri dello Spectrum non finiscono mai di stupire: poiché la lunghezza della linea è conservata in due byte, una singola linea potrebbe in teoria esser lunga ben 65.535 caratteri! Nel nostro caso la lunghezza è 13, in quanto i byte occupati dal numero di linea e dalla lunghezza non vengono computati

LIZ DIXON

Il LOGO per gli Atari

Esaminiamo la potente gestione degli sprite offerta da questa versione di LOGO

Il LOGO dei computer Atari prevede l'uso di 4 sprite, numerati da 0 a 3, che sul manuale vengono chiamati tartarughe. In assenza di istruzioni specifiche, è selezionata automaticamente la tartaruga di codice 0, per certi versi "privilegiata".
Il comando TELL 1 abilita la tartaruga 1. Si provi la seguente procedura:

```
TELL 1
FD 40
RT 90
TELL 2
BK 40
RT 90
TELL 3
RT 90
```

È anche possibile selezionare più tartarughe contemporaneamente:

```
TELL [1 2 3]
FD 50
```

Adesso abbiamo tre tartarughe identiche a quella 0, capaci di ruotare e di reagire ai comandi. Il programma Editor permette di disegnare ed assegnare alle tartarughe fino a 15 diverse sagome che però non possono ruotare come la tartaruga privilegiata. Il comando EDSH 1 predispone il programma Editor per la creazione della sagoma 1. Il movimento del cursore sullo schermo è controllato dai normali tasti per il cursore, mentre la barra spaziatrice viene usata per accendere o spegnere le caselle della griglia per la creazione degli sprite. Il tasto <ESC> fissa l'immagine finale. Il comando SETSH 1 assegna il nuovo sprite alle tartarughe 1, 2 e 3, che sono state selezionate in precedenza.
ASK, invece, invia un comando ad una particolare tartaruga senza modificare le altre. Ad esempio, con:

```
ASK 1[FD 20]
```

si sposta solo la tartaruga 1, mentre una successiva istruzione FD 20 muove tutte e tre le tartarughe selezionate.
Ad ogni tartaruga può essere assegnata una propria velocità, direzione e posizione. SETSP 30 conferisce alla tartaruga in uso una velocità di 30 unità, che viene mantenuta finché non la si modifica nuovamente. SETSP 0 riporta la velocità a 0, fermando così il movimento. Richiamando il programma Editor, tutte le velocità vengono azzerate; ciò dipende dal fatto che esso occupa la stessa area di memoria impiegata dalla grafica.
L'Atari possiede ben 128 diverse tonalità di colori: per lo sfondo, per la "penna" che traccia le linee e per la tartaruga. SETBG 92, ad esempio, seleziona il verde per lo sfondo e SETPC 0 23 il colore arancione per la penna. Nel LOGO Atari il 23 è il codice di colore per l'arancione, mentre lo 0 seleziona una delle tre penne disponibili. La tabella dei colori e dei corrispondenti codici è a pag. 26 del manuale originale.

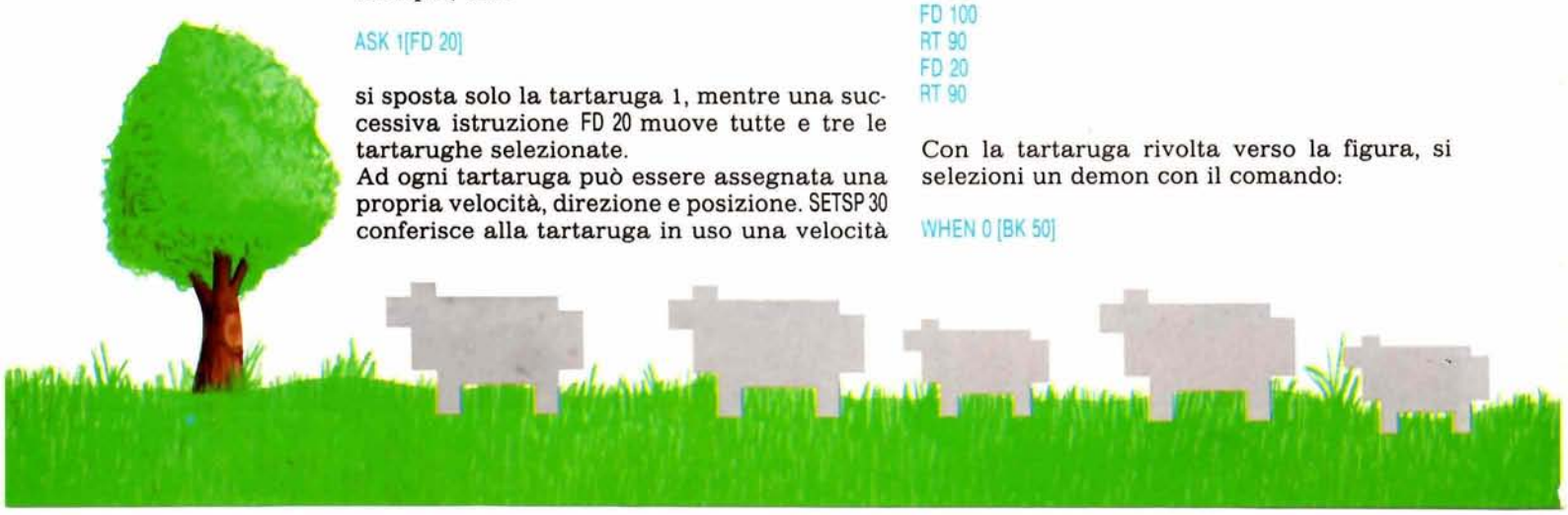
I demon

Il LOGO Atari è in grado di individuare fino a 21 "collisioni" o "eventi speciali" (vedere a pag. 145 del manuale originale), la maggior parte delle quali riguarda collisioni tra tartarughe, o "incroci" tra tartarughe e linee. Una delle caratteristiche più curiose del LOGO Atari, a tal riguardo, è la presenza dei cosiddetti "demon": un demon comunica al LOGO come deve comportarsi quando si verifica una collisione.
Se, per ipotesi, la tartaruga 0 incrocia una linea tracciata con la penna 0, si ha la collisione numero 0. Per selezionare un demon si usa il comando WHEN. Trascriviamo la seguente procedura:

```
CS
TELL 0
PD
FD 50
PU
RT 90
FD 100
RT 90
FD 20
RT 90
```

Con la tartaruga rivolta verso la figura, si selezioni un demon con il comando:

```
WHEN 0 [BK 50]
```





Inizialmente non accade nulla: il demon attende semplicemente, in memoria, che si verifichi l'evento 0. Inseriamo adesso il comando SETSP 30: la tartaruga si muove in avanti ma, quando incrocia la linea, il demon entra in azione e la respinge! La tartaruga ritenta il passaggio, sempre a una velocità di 30, ma ogni volta che raggiunge la linea viene nuovamente respinta.
Questo demon resta in funzione finché non viene disattivato dal comando WHEN 0 [ ]. Il comando CS, l'uso del programma Editor oppure la comparsa di un messaggio d'errore cancellano tutti i demon.
Per evitare di imparare a memoria i vari codici di collisione, si ricorre a due speciali comandi. OVER <numerotartaruga> <numeropenna> genera il numero della collisione tra una tartaruga e una linea tracciata con una data penna e TOUCHING <numerotartaruga 1> <numerotartaruga 2> quello per la collisione tra le due tartarughe specificate.

Una tartaruga in gabbia

Le seguenti procedure formano un recinto attorno a una tartaruga. Quando la tartaruga incontra uno dei margini (WHEN OVER 0 0), un demon chiama la procedura TURN che la respinge indietro di 10 unità e le imprime una rotazione casuale (il comando RANDOM abbinato ad un numero N genera un numero casuale compreso tra 0 e N−1).

```
TO GABBIA
   TRACC.GABBIA
   HOME
   WHEN OVER 0 0 [ROTAZIONE]
   SETSP 50
END

TO TRACC.GABBIA
   CS
   PU
   SETPOS [−50 −50]
   PD
   QUADRATO
   PU
END

TO QUADRATO
   REPEAT 4 [FD 100 RT 90]
END

TO ROTAZIONE
   BK 10
   RT RANDOM 45
END
```

I demon si possono usare anche per il controllo dei joystick. Dei 21 eventi speciali prima citati, il caso 3 si verifica quando si preme il pulsante del joystick ed il 15 quando si muove la sua cloche. Il comando JOY 1 produce un numero compreso tra −1 e 7 corrispondente alla posizione del joystick (sulla porta 2). Definiamo la procedura MOV.JOY nel seguente modo:

```
TO MOV.JOY
   IF (JOY 1)
   ASK 0 [SETH 45 * JOY 1]
END
```

selezioniamo poi la velocità della tartaruga con SETSP 50 ed infine abilitiamo un demon con l'istruzione:

```
WHEN 15 [MOV.JOY]
```

A questo punto il joystick è in grado di controllare la direzione della tartaruga 0.
Si possono impartire più comandi WHEN, ma essi non vengono eseguiti simultaneamente: quando uno è attivo (per il verificarsi dell'evento ad esso abbinato), gli altri sono passivi. Ciò può comportare la mancata individuazione di alcune collisioni.
Per superare questo problema, ciascun demon deve azzerare la velocità ed eseguire ripetutamente una procedura di controllo. Serviamoci di questa tecnica per modificare la procedura precedente:

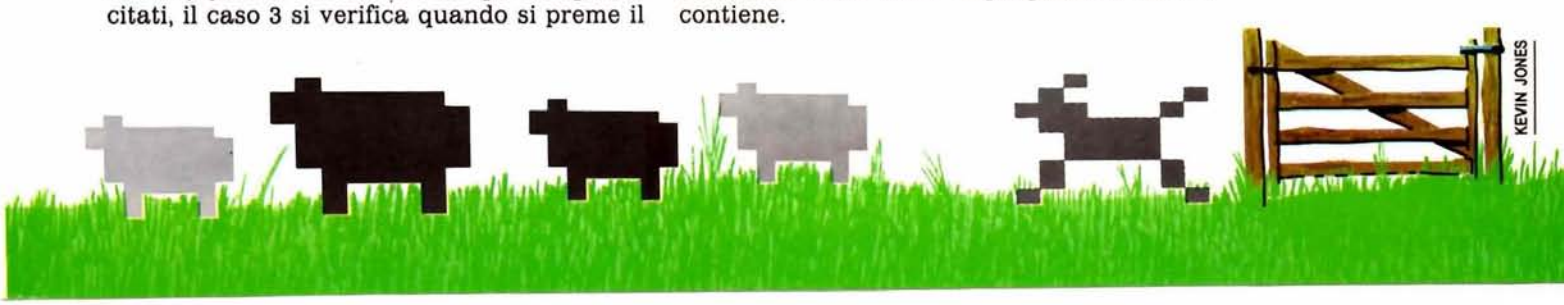
```
TO GABBIA
   TRACC.GABBIA
   HOME
   WHEN OVER 0 0 [SETSP 0]
   SETSP 50
   CONTROLLO
END

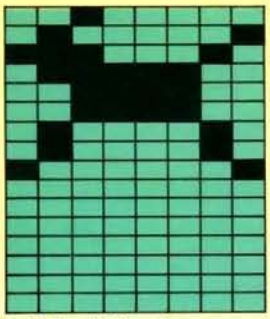
TO CONTROLLO
   IF :SPEED = 0 [VERIFICA]
   CONTROLLO
END
```

In questa procedura, SPEED fornisce il valore per la velocità della tartaruga in uso. La procedura VERIFICA deve invece individuare il tipo di collisione che si è verificata, eseguire le operazioni necessarie al caso e riportare la velocità al valore originario.

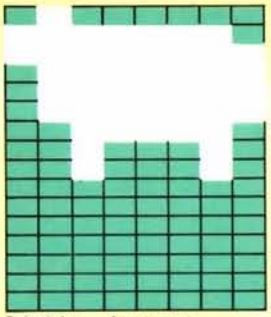
```
TO VERIFICA
   IF COND OVER 0 0 THEN [TURN]
   SETSP 50
END
```

L'istruzione COND, seguita da un numero, risulta "vera" quando si verifica una collisione che corrisponde al numero indicato. COND può verificare soltanto un caso alla volta, durante l'esecuzione della linea di programma che la contiene.





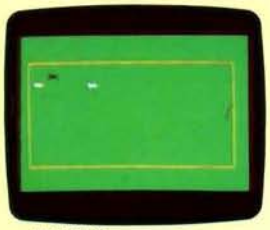
Canis Familiaris — Il cane



Ovis Aries — La pecora



Il gioco inizia ...



... prosegue ...



... si conclude !

IAN McKINNELL

Un cane da pastore

Il seguente gioco contiene molte delle funzioni appena descritte. Il giocatore usa il joystick per controllare un cane da pastore che insegue due pecore in un campo. Se le pecore urtano il recinto girano su se stesse e tornano indietro; se si urtano tra loro cambiano direzione in modo casuale. Se il cane le tocca ruotano di 90 gradi a destra. Premendo il pulsante del joystick, il giocatore traccia un piccolo recinto nell'angolo in basso a sinistra del campo; una seconda pressione del pulsante lo cancella. Il compito del cane è spingere le pecore dentro il recinto.

```
TO INSEGUIMENTO
   DEF.VAR
   ASK :TARTARUGA [DEF.VIDEO]
   DEF.DEMON
INIZIO
CONTROLLO
END

TO DEF.VAR
   MAKE "RECINTO 0
   MAKE "TARTARUGA 0
   MAKE "PECORA1 3
   MAKE "PECORA2 2
   MAKE "CANE 1
   MAKE "VERDE 92
   MAKE "ARANCIONE 23
   MAKE "NERO 0
   MAKE "BIANCO 7
END

TO DEF.VIDEO
   CS
   FS
   SETBG :VERDE
   HT
   PU
   SETPOS [-150 -80]
   PD
   SETPC 0 :MARRONE
   RETT 160 300
   PU
END

TO RETT :LATO1 :LATO2
   REPEAT 2 [FD :LATO1 RT 90 FD :LATO2 RT 90]
END

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]
   WHEN 3 [SETSP 0]
   WHEN 15 [MOV.JOY]
END

TO MOV.JOY
   IF (JOY 1)
   ASK :CANE [SETH 45 * JOY1]
END

TO INIZIO
   DEF :PECORA1 1 [-150 20] 45 :BIANCO
   DEF :PECORA2 1 [150 20] 315 :BIANCO
   DEF :CANE 2 [0 0] 0 :NERO
   DEF.VELOCITA
END

TO DEF :NO :FORMA :POSIZ :DIR :COLORE
   TELL :NO
   PU
   SETSH :FORMA
   SETC :COLORE
   ST
   SETPOS :POSIZ
   SETH :DIR
END

TO
 DEF.VELOCITA
 ASK :PECORA1 [SETSP 10]
 ASK :PECORA2 [SETSP 10]
 ASK :CANE [SETSP 60]
END

TO CONTROLLO
   IF SPEED = 0 [VERIFICA]
   CONTROLLO
END

TO VERIFICA
   IF COND OVER :PECORA1 :RECINTO [ASK :PECORA1 [BK 20
   RT 90]]
   IF COND OVER :PECORA2 :RECINTO [ASK :PECORA2 [BK 20
   RT 90]]
   IF COND TOUCHING :PECORA1 :PECORA2 [COLLISIONE]
   IF COND TOUCHING :CANE :PECORA1 [ASK :PECORA1 [RT
   90]]
   IF COND TOUCHING :CANE :PECORA2 [ASK :PECORA2 [RT
   90]]
   IF COND 3 [ASK :TARTARUGA [TRACCIA.GABBIA]]
   DEF.VELOCITA
END

TO COLLISIONE
   ASK :PECORA1 [SETH RANDOM 360]
   ASK :PECORA2 [SETH RANDOM 360]
   END

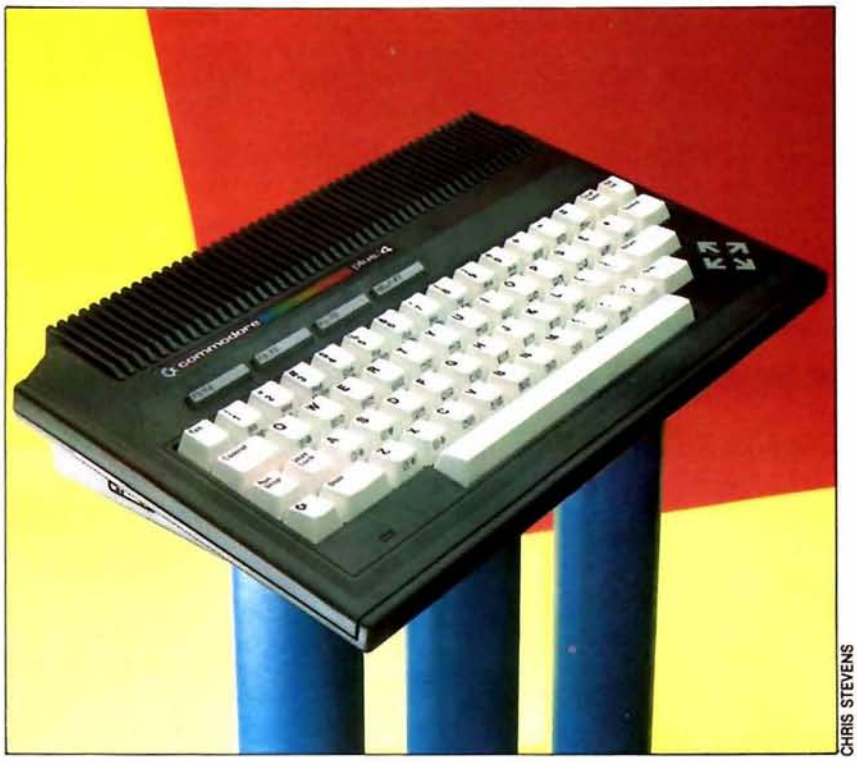
TO TRACCIA.GABBIA
   PU
   SETPOS [-150 -30]
   PX
   SETH 90
   REPEAT 2 [FD 50 RT 90]
   PU
END
```

ESERCIZI LOGO

1) Modificare il gioco in modo da poter controllare il cane tramite la tastiera.
2) Scrivere un programma nel quale il giocatore, alla guida di un'astronave, cerca di evitare le meteoriti che gli vengono incontro. Usare uno sprite per l'astronave e gli altri per le meteoriti. Adoperare i demon, tramite il comando WHEN, per individuare le collisioni. Le meteoriti si muovono a velocità costante, ma con direzione casuale. L'astronave può essere controllata tramite il joystick.

Il Plus/4

Anatomia di un microcomputer a diffusione europea: tecnologie a confronto



CHRIS STEVENS

Nelle intenzioni della Commodore, il Commodore Plus/4 dovrebbe affiancare (e non sostituire) il Commodore 64, ma il nuovo modello offre prestazioni talmente superiori, che probabilmente è destinato a spodestare il buon vecchio 64.

Il Plus/4, usa il microprocessore 7501, una versione perfezionata del 6502, capace di indirizzare più di 64 Kbyte di memoria. L'apparecchio, quindi, possiede spazio sufficiente per un buon BASIC e mantiene la RAM a completa disposizione dell'utente. I programmi BASIC possono occupare fino a 64 Kbyte (ridotti a 50 se viene adoperato anche il pacchetto grafico).

Tra tutti i micro, solo il QL della Sinclair (vedere a pag. 981) e l'Advance 86a (vedere a pag. 829) offrono di meglio.

Il Plus/4 dispone di un'eccellente versione del BASIC Microsoft. Inoltre, i comandi grafici e quelli audio sono particolarmente versatili. Nel modo grafico, l'istruzione DRAW traccia punti o linee, mentre PAINT permette di colorare qualsiasi figura chiusa.

Il comando BOX traccia quadrati e rettangoli vuoti o colorati, mentre con CIRCLE si disegnano circonferenze o ellissi, specificandone le dimensioni; per tracciare un arco è sufficiente indicare nel comando le coordinate iniziali e finali.

Di solito, la risoluzione grafica dei comandi è di 320 x 200 pixel (come sul Commodore 64), ma il Plus/4 possiede una maggiore gamma di colori: può infatti visualizzarne contemporaneamente 120, oltre al "nero", partendo da 15 colori base, ciascuno con 8 diversi livelli di luminosità. Purtroppo però, il Plus/4 è privo di sprite.

I comandi di suono sono standard: l'istruzione SOUND genera una nota di durata e tono specificati, il comando VOL seleziona uno degli 8 livelli di volume di ciascun canale di suono. Tutti i suoni vengono emessi dall'altoparlante dell'apparecchio televisivo collegato. Sebbene il BASIC consenta di indirizzare tre canali di suono, il Plus/4 ne possiede solamente due. Il terzo è in realtà un generatore di rumore, molto utile nei programmi di gioco, che trasforma in "effetti speciali" le note specificate.

Al corpo principale del linguaggio BASIC sono state aggiunte alcune istruzioni che ne migliorano le prestazioni. Il comando AUTO numera automaticamente le linee del programma in fase di creazione; RENUMBER assegna nuovi numeri di linea ai programmi e VERIFY controlla che i programmi vengano salvati in modo corretto su cassetta o su disco. La presenza di nuovi comandi per la gestione dei dischetti fa pensare che la Commodore preveda di venderne un buon numero.

Il Plus/4 possiede, nel modo testuale, un formato di 40 x 25 caratteri. L'utente può selezionare due punti dello schermo, che definiscono gli angoli di una "finestra": al suo interno possono essere visualizzati listati e comandi, lasciando inalterato il resto dello schermo.

La tastiera del Plus/4 segue lo standard QWERTY e possiede una discreta sensibilità: presenta tasti individuali per i caratteri @, =, +, – e £, permettendo inoltre di generare direttamente una completa serie di caratteri grafici. Sopra la tastiera ci sono quattro tasti di funzione che, al momento dell'accensione, vengono automaticamente selezionati sui comandi di uso più comune; l'utente, tuttavia, può abbinare ad essi funzioni diverse, con un massimo di 128 caratteri per ciascun tasto. L'uso combinato dei quattro tasti e del tasto Shift consente di generare complessivamente 8 diverse funzioni.

La discreta quantità di memoria disponibile consente al Plus/4 di tenere in ROM quattro programmi applicativi: un word processor, uno spreadsheet, un database ed un package di grafica, tutti progettati per lavorare assieme. Disgraziatamente, il software non sembra avere lo stesso livello qualitativo dell'apparecchio.

Il word processor, infatti, risulta piuttosto difficile da usare. Il Plus/4 può visualizzare solo linee di 40 caratteri mentre molte stampanti producono 80 caratteri per linea; per armonizzare le due misure, lo schermo "carrella" fino

Un passo nel futuro?

Il Plus/4, successore del Commodore 64, possiede tutte le caratteristiche presenti sui micro più moderni: conformazione esterna di tipo MSX, tastierina separata per il cursore, una discreta memoria e software residente. Tuttavia, la concorrenza in questo settore è molto forte, anche se la Commodore rappresenta un nome ormai ben affermato

alla colonna 37, poi prosegue fino alla colonna 77, infine torna alla prima colonna. Il programma possiede alcune istruzioni per la formattazione e l'allineamento, ma che intervengono sul testo solo in fase di stampa. I comandi SEARCH e REPLACE permettono di individuare (ed eventualmente sostituire) parole o frasi all'interno di un documento. Il testo inserito può contenere al massimo 99 linee. Poiché ogni linea è di 77 caratteri, si possono introdurre un massimo di circa 1.500 parole: non molto per applicazioni più "serie".

Il programma di spreadsheet è più facile da usare, ma è anch'esso limitato dallo schermo a 40 colonne: la griglia visualizzata è composta da 3 x 12 celle, sebbene sia in grado di gestire modelli con griglie di 17x50 celle.

Anche il programma grafico è piuttosto deludente. La sua unica funzione è quella di trasformare una serie di valori numerici, provenienti dallo spreadsheet, in un grossolano grafico a barre e di trasferire quest'ultimo al word processor per la visualizzazione o la stampa.

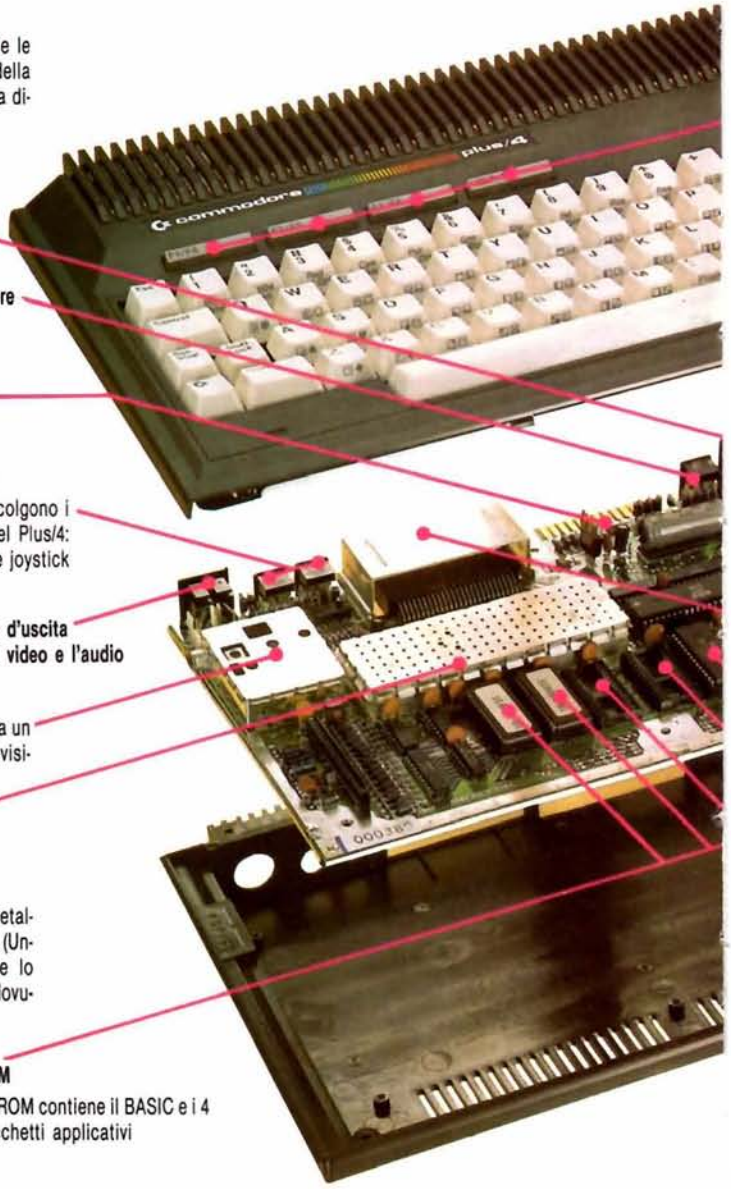
Il word processor e lo spreadsheet possono essere adoperati sul modello base del Plus/4, ma per conservare il lavoro fatto non basta il registratore a cassette: occorre un'unità a dischetti. Il programma di database, che funziona anch'esso solo in presenza di un'unità a dischetti, crea una serie di record vuoti di lunghezza standard che vengono registrati sul dischetto e, successivamente, ricevono tutte le informazioni. Quindi un dischetto è abbinato ad un solo archivio e il formato non può essere modificato dopo l'inserimento dei dati. Ogni database può contenere fino a 999 record, ciascuno composto da un massimo di 17 campi di 38 caratteri. Riassumendo, il software del Plus/4 è piuttosto insoddisfacente, inadatto ad applicazioni commerciali ed obbliga all'acquisto di un'unità a dischetti.

Molti utenti forse apprezzeranno più il programma "monitor" per il codice macchina, chiamato *Tedmon*, che non la dotazione software. Il *Tedmon* è di grande aiuto al programmatore in codice macchina e viene abilitato con il comando MONITOR.

Come gli altri micro prodotti dalla Casa, il Plus/4 impiega un registratore a cassette fabbricato dalla Commodore, che si differenzia dai precedenti per il particolare spinotto di collegamento. Il Plus/4 possiede anche una diversa presa per i joystick.

Per questo modello viene offerta una versione leggermente aggiornata della lenta unità a dischetti per il Commodore 64 e pare che in futuro sarà disponibile un'unità a dischetti ancora più veloce, da collegare alla porta per cartucce dell'apparecchio. Il Plus/4 può operare con cinque diverse stampanti Commodore: una stampante a margherita, due comuni stampanti ad aghi, una stampante ad aghi a colori ed una stampante/plotter a 4 penne.

Il costo del Plus/4, superiore a quello del Commodore 64, è giustificato dalla maggior memoria disponibile e dai miglioramenti apportati al BASIC (questi ultimi verranno esaminati più approfonditamente in seguito).



Bus seriale
Ad esso vengono collegate le periferiche standard della Commodore, quali l'unità a dischetti e la stampante

Presa per registratore a cassetta

Porta per utente

Prese per il joystick
Queste due prese accolgono i joystick "dedicati" del Plus/4: non è possibile usare joystick standard

Porta d'uscita per il video e l'audio

Modulatore TV
Genera un segnale adatto a un normale apparecchio televisivo

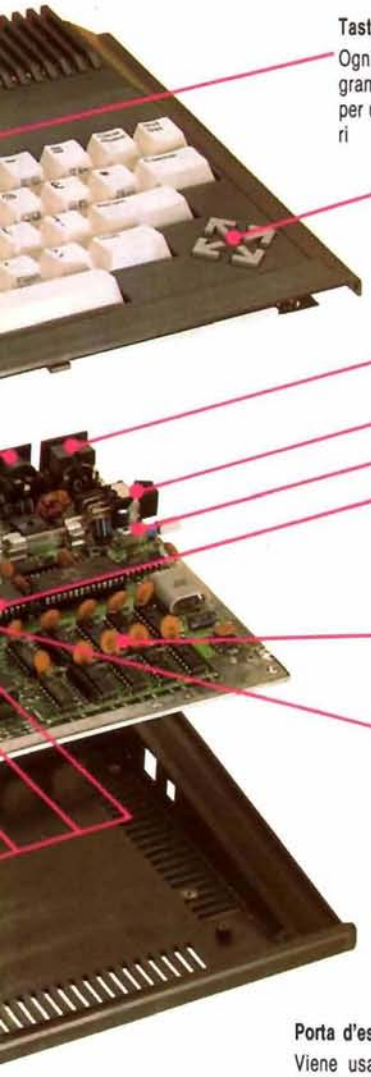
Custodia ULA
Questa piccola scatolà metallica contiene un chip ULA (Uncommitted Logic Array) e lo protegge da interferenze dovute ad onde radio

ROM
La ROM contiene il BASIC e i 4 pacchetti applicativi



IAN McKINNELL

QL contro Plus/4
Il confronto è obiettivamente improponibile: il QL possiede Microdrive interni, maggiore memoria, il SuperBASIC ed un software apprezzabile; il Plus/4, abbinato a un registratore a cassette, può contare solo sull'eccellente qualità della sua tastiera. Una fredda valutazione porterebbe a optare per il QL, ma, nonostante tutto, molti hanno deciso di restare fedeli al marchio Commodore



Tasti di funzione
Ogni tasto può essere programmato con due comandi per un massimo di 128 caratteri

Tasti per il cursore

Presa d'alimentazione

Interruttore d'accensione

Pulsante di reset

CPU 7501

RAM da 64K

Porta d'espansione
Viene usata dal software su cartuccia e dall'unità a dischetti "veloce"

CHRIS STEVENS

Tasti di funzione



Tastierina per il cursore



Pulsante di Reset



Porta d'espansione/Porta per utente



Nuove caratteristiche
Il Plus/4 possiede il tasto Escape ed il pulsante di Reset, oltre a una tastierina separata per il cursore, 4 tasti di funzione (programmabili in ambiente BASIC) ed il tasto Help. Le prese per l'alimentazione, per il joystick e per il registratore a cassette, come del resto quelle per l'alimentazione e per l'utente, sono diverse da quelle del Commodore 64 e del Vic-20

COMMODORE PLUS/4

DIMENSIONI

mm. 67 x 203 x 338

CPU

7501, frequenza di lavoro a 0,9 o 1,8 MHz

MEMORIA

64 Kbyte di RAM, 64 Kbyte di ROM

VIDEO

Testuale: 40 x 25 caratteri. Grafico: 320 x 200 pixel in 121 colori

INTERFACCE

2 prese per joystick, interfaccia seriale, interfaccia per registratore, porta per cartucce e porta parallela

LINGUAGGI FORNITI

BASIC

TASTIERA

Tipo QWERTY, con 67 tasti di cui 4 per funzioni programmabili

DOCUMENTAZIONE

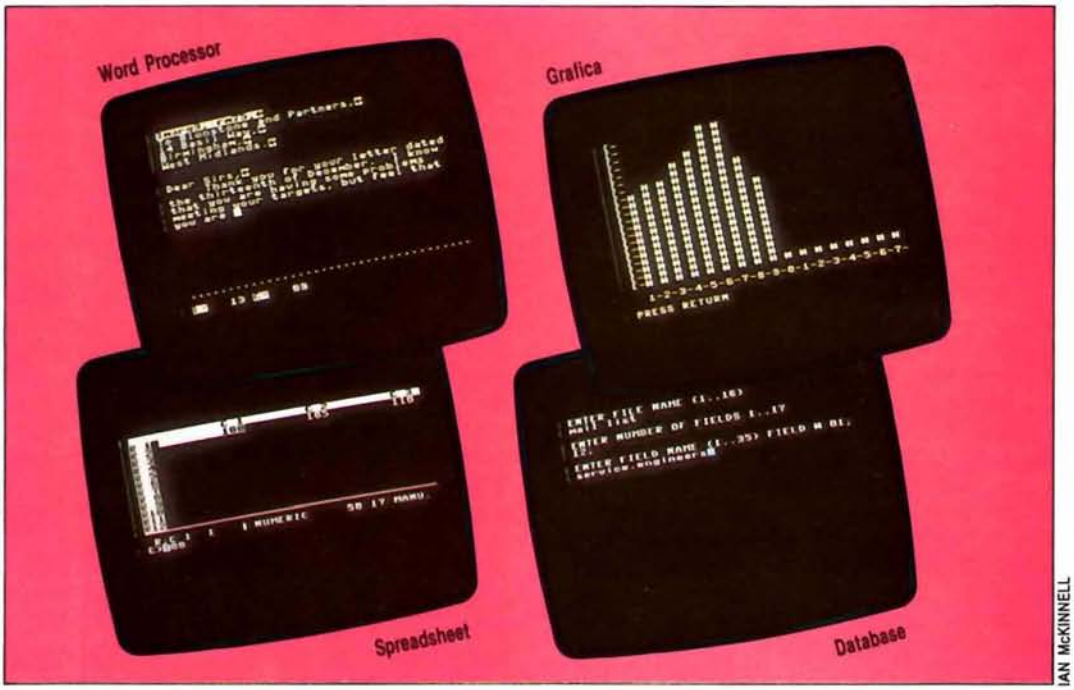
I manuali originali, rivolti alla programmazione in generale ed al software incluso, sono sufficientemente adeguati ma non privi di lacune

PRO

Ottima versione di BASIC, dotato di buoni comandi grafici e audio. Tutti i 64 Kbyte di RAM sono disponibili per i programmi

CONTRO

La qualità del software incorporato, per il quale è indispensabile l'impiego dei dischetti, è alquanto opinabile. Occorrono periferiche Commodore, diverse da quelle per il Commodore 64 e per il Vic 20



IAN McKINNELL

COSTI "NASCOSTI"

Il software residente nella ROM del Plus/4 comprende un word processor, uno spreadsheet, un database ed un programma grafico. Tutti questi programmi sono integrati tra loro, ma richiedono la presenza di un'unità a dischetti che accresce il costo complessivo del sistema

Mutui e prestiti

Vediamo come adoperare il Vu-Calc per calcolare gli interessi

La notevole forza dei programmi spreadsheet, anche di quelli mediamente sofisticati quali il Vu-Calc della Psion, consiste nella facilità con la quale si possono applicare ai dati complesse formule, che però non sempre sono incorporate nel programma ma vanno immesse nel modello creato. In effetti, la sola formula presente nel Vu-Calc è 'sum', che consente di sommare il contenuto di una serie di celle. Per applicarla, si digita il prefisso @ prima dell'indirizzo della cella.
Altri spreadsheet contengono tutta una gamma di formule già pronte, che l'utente può richiamare per nome ed applicare. Nel *Multiplan,* ad esempio, il calcolo dei versamenti per ripagare un prestito su differenti periodi (per ipotesi su 15, 20 e 25 anni) è molto semplice: basta richiamare la formula ed immettere i dati, dopodiché *Multiplan* esegue il calcolo.
Un simile calcolo con Vu-Calc, però, richiede più tempo ed impegno, perché la formula va introdotta dall'utente. Inoltre, il Vu-Calc pone altre restrizioni: il numero massimo di colonne è 28, quindi un prospetto suddiviso in mesi può coprire al massimo poco più di due anni. Un secondo problema deriva dal fatto che Vu-Calc prende in considerazione soltanto numeri interi, perciò 99,99 viene tranquillamente troncato a 99 (la parte intera). È consentito immettere nelle celle sia i valori, sia il risultato di un'espressione: se, per esempio, il cursore è posizionato in una cella vuota, poniamo H5, si può digitare 500—2 nella linea dei comandi e, appena premuto ENTER, nella cella H5 compare 1000.
Correggere le formule immesse nel Vu-Calc è abbastanza fastidioso poiché occorre riscriverle per intero, dopo aver invocato il comando di EDIT (#E). Nel *Lotus 1-2-3,* invece, le correzioni non implicano affatto la completa re-impostazione della formula. Attenzione, dunque, nel trascrivere formule lunghe e complesse!
Un comando molto comodo è REPLICATE (#R), col quale è possibile duplicare il contenuto (dato o formula) di una cella in un'altra. Supponiamo che nel modello impiegato a pag. 1172 si voglia introdurre, per esempio, una voce "spese alimentari" in cui un fattore di correzione tenga conto di un tasso d'inflazione mensile costante dello 0,5%. Eseguire i calcoli necessari a mano è un compito lungo e tedioso, ma se ricorre al REPLICATE del Vu-Calc i risultati si ottengono con estrema facilità. In alcuni spreadsheet esiste addirittura un comando GROW BY (accresci di) appositamente previsto per questi casi.
Nelle formule immesse, il Vu-Calc riconosce l'indirizzo delle celle se questo viene prefissato con $ o %. Tali simboli non hanno niente a che vedere con dollari o percentuali: indicano soltanto se l'indirizzo della cella in questione è assoluto ($) oppure relativo (%). Nel primo caso, viene preso in considerazione il contenuto della cella specificata, indipendentemente dalla sua posizione.
Durante l'inserimento di formule in un modello, però, risulta spesso più comodo riferirsi ad una cella che abbia una posizione relativa a quella in cui si trova il cursore, in quanto ciò evita di dover immettere una stessa formula in più celle, variando di volta in volta soltanto l'indirizzo delle celle interessate. In sostanza, si tratta di costruire una formula "campione" in una cella, e poi di replicarla su di un arco di celle: ci pensa il Vu-Calc ad aggiustare automaticamente tutti gli indirizzi. Per passare ad un caso pratico, torniamo all'esempio di una voce che aumenti mensilmente in modo progressivo. In questo frangente, il valore della cella B5, risulta dall'applicazione del tasso di aumento dello 0,5% sul valore contenuto nella cella B4. Nella cella B6 dobbiamo trovare un valore pari a quello della cella precedente, aumentato dello 0,5% e così via. Per calcolare il valore di una cella, dunque, basta applicare la percentuale alla cifra contenuto nella cella precedente e questo meccanismo viene innescato ricorrendo all'indirizzamento relativo. Nella formula (creata nella cella B4), l'indirizzo relativo è %B5, a indicare la cella immediatamente precedente a quella in cui si trova il cursore. Ovviamente, una complessa formula può contenere anche più indirizzi relativi.
La formula per accrescere progressivamente un valore dello 0,5% è %B3— 100.5/100, dove % indica l'indirizzo relativo di una cella e B3 è l'indirizzo della cella che contiene la somma destinata mensilmente all'alimentazione. Avendo digitato la formula nella cella B4, adesso è necessario ricopiarla in tutte le caselle della riga, per ottenere un quadro annuale. In B4 appare il risultato della formula (quest'ultima compare nella linea dei comandi, quando il cursore si trova in B4). Per copiare la formula, non occorre ritrascriverla: basta usare il comando REPLICATE: #R,B4,B5:B14. Questo comunica al programma che la formula contenuta in B4 va copiata nelle celle da B5 a B14. I risultati del calcolo appaiono pressoché istantaneamente:

	1	2	3	4	5
A			GEN	FEB	MAR
B	Spese alimentari		200	201	202

Si noti come le celle B1 e B2 siano occupate dalla descrizione: ecco perché il primo valore si trova in B3. Inoltre, tutti i valori (espressi in migliaia) sono interi: la cifra relativa a marzo dovrebbe essere 202,215 ma il Vu-Calc arrotonda a 202.
Questo semplice esempio dimostra l'effetto del comando REPLICATE, quando vengono usati indirizzi di celle relativi. La formula iniziale, contenuta in B4, è: %B3—100.5/100 e viene automaticamente riprodotta in B5 come se si digitasse %B4—100.5/100 e in B6 come %B5—100.5/100 e così via. In ciascun caso, l'indirizzo della colonna viene aumentato di uno. L'uso di REPLICATE su più righe di una stessa colonna ha un effetto analogo, stavolta aumentando l'indirizzo della riga. Se la formula impiegata non fosse espressa con indirizzi relativi, ma assoluti ($), il risultato finale sarebbe totalmente diverso: la formula applicherebbe il tasso d'inflazione sempre al valore della medesima cella (B4).

Analisi dei costi

Adesso proviamo ad applicare il modello realizzato ad un caso concreto. Si vuole ottenere una previsione dei costi di una materia grezza, partendo da un costo base di 100.000 lire, con un tasso di aumento mensile dello 0,5% su di un arco di due anni. Quanto costerà la materia grezza a metà del secondo anno? La risposta, usando il nostro modello, si ha in pochi secondi.
Si sposta il cursore nella cella B3, immettendo il valore 100.000. Adesso si estende l'arco dei mesi considerato, usando REPLICATE per copiare la solita formula sulle celle da B14 a B26. In molti spreadsheet, a questo punto, compaiono i risultati del calcolo. Con Vu-Calc, però, è necessario chiedere che le cifre vengano nuovamente ricalcolate e per far ciò si adopera il comando CALCULATE, ossia #C. Il risultato richiesto si trova alla cella B20: 109931, cioè quasi 10.000 lire di aumento. A causa di arrotondamenti, il risultato può non essere preciso alla lira, tuttavia ciò non invalida l'utilità di tutto il sistema.
Come ultimo esempio di un problema su singola riga, consideriamo una formula più complessa, adatta a calcolare il tempo occorrente per ripagare un prestito bancario di un milione di lire, soggetto ad un tasso d'interesse passivo del 27%, con versamenti mensili di 80.000 lire.
La formula deve tener conto del capitale iniziale, più l'interesse passivo mensile, meno la cifra versata. Dopo aver immesso 1000 (che sta per un milione) in B1, la formula da immettere è: %B1+%B1—.27/12-80. Si "replichi" la formula su tutto l'arco di 28 celle. Eseguito il calcolo, si vada a cercare la casella in cui il risultato diventa di segno positivo: essa corrisponde al momento in cui il prestito è totalmente ripagato. In base al nostro modello, ciò richiede 16 mesi. Come effetto secondario, si ottiene un tabulato delle cifre progressive del debito, a condizione che i versamenti vengano fatti regolarmente.
In seguito viene analizzato il funzionamento del programma *Abacus*, lo spreadsheet venduto in dotazione al QL della Sinclair.

	1	2	3	4	5	
A			ENTRATE ANNUALI = 5400			
B			GEN	FEB	MAR	
C	ENTRATE FAMILIARI		A5/12 450	A5/12 450	A5/12 450	
D	SPESE FAMILIARI		DATA 400	D3*1.01 404	D4*1.01 408	
E	BILANCIO MENSILE		C3-D3 50	C4-D4 46	C5-D5 42	

RELATIVAMENTE ASSOLUTO

In questo semplice modello è stato usato il comando REPLICATE, lungo le righe C, D ed E. Ai soli fini dell'esempio, in ciascuna cella è riportata la relativa formula, assieme al valore. Nella riga C la formula replicata contiene indirizzi assoluti, cosicché la stessa formula (A/12) appare in ogni cella della riga, producendo il medesimo risultato: 450. Le formule nelle righe D ed E, invece, contengono indirizzi relativi, quindi le rispettive formule variano da una posizione all'altra, producendo risultati non costanti (come nel caso precedente), ma dipendenti dal valore delle celle precedenti o superiori

Un convertitore D-A

Un dispositivo per controllare apparecchi analogici e produrre suoni

Per questo progetto si è preferito usare un chip convertitore digitale/analogico (D-A) già pronto: la realizzazione del circuito diventa così molto meno complicata.
Il segnale analogico in uscita da questo chip DAC viene "bufferizzato" da un amplificatore in un secondo circuito integrato, la cui uscita viene applicata a una delle prese di output e, attraverso un potenziometro ed un controllo di livello, all'altra presa di output.
Fase 1: Praticare sulla scatola del circuito le aperture necessarie per accogliere i connettori dei due bus e, eventualmente, per una presa da utilizzare per futuri progetti.
Fase 2: Tagliare una piastrina di veroboard delle giuste dimensioni (30 fori per 16 piste). Eseguire i tagli delle piste illustrati in figura. Saldare per primi gli zoccoli dei chip ed i ponticelli, poi i due condensatori. Il loro orientamento sulla scheda non è difficile. Se si desidera dotare il circuito di un connettore d'espansione del bus, è il momento di saldarlo e di collegarlo al cavo a nastro.
Fase 3: Inserire le prese (due rosse e due nere) ed il potenziometro sulla scatola e collegarle con filo stagnato seguendo lo schema in figura. Infine saldare i 3 cavetti "volanti" che collegano il circuito al potenziometro.
Fase 4: Innestare i due chip sugli zoccoli, facendo attenzione al loro orientamento: osservando dall'alto, con il connettore "maschio" del bus rivolto in avanti, la tacca del chip convertitore D-A deve trovarsi a sinistra, mentre quella del chip amplificatore deve essere a destra. La costruzione è terminata.
A questo punto, dopo aver ricontrollato tutti i collegamenti, si può collaudare il convertitore D-A. Lo strumento trasforma in una tensione qualsiasi numero binario ad 8 bit caricato nel registro dati della porta utente. La corrente prodotta dal convertitore è di due tipi. Sulla coppia di prese per la corrente continua (CC) si ottiene un voltaggio compreso tra 0 e 2,5 volt, corrispondente a valori digitali della porta utente che variano da 0 a 255. Sulla seconda coppia di prese viene invece simulata una corrente alternata (CA). Il livello globale del voltaggio viene controllato da un potenziometro e può essere regolato in modo da inviare il voltaggio richiesto ad un altro strumento del sistema. Per verificare il buon funzionamento del convertitore D-A, proviamo a modificare la luminosità di un LED, in base alla seguente procedura:
Fase 1: Prendere un LED del tipo usato per il buffer box (vedere a pag. 1003) e collegarlo "in serie" ad una resistenza da 50 ohm.
Fase 2: Connettere il convertitore D-A direttamente al buffer box, che a sua volta deve essere collegato alla porta utente dalla quale riceve corrente nel modo consueto.
Fase 3: Collegare il LED e la resistenza alle prese di corrente alternata (CA) del convertitore D-A. Trascrivere e lanciare il seguente programma:

```
10 REM VERIFICA DEL CONVERTITORE D-A - CBM 64
20 DDR=56579:REGDAT=56577
30 VL=127
40 POKE DDR,255: REM TUTTE LINEE IN USCITA
50 POKE REGDAT,VL
60 PRINT VL
70 GET A$
80 IFA$<>"Z" AND A$<>"X" THEN 70
90 IF A$="X" THEN DV=1
100 IF A$="Z" THEN DV=—1
110 VL=VL+DV
120 IF VL<256 AND VL>=0 THEN 50
```

```
10 REM VERIFICA DEL CONVERTITORE D-A - BBC
20 DDR=&FE62:REGDAT=&FE60
25 valore=127
30 ?DDR=255: REM TUTTE LINEE IN USCITA
40 REPEAT
55 ?REGDAT=valore
57 PRINT valore
60 A$=GET$
62 IF A$<>"Z" AND A$<>"X" THEN 60
65 IFA$="X" THEN dv=1 ELSE dv=—1
67 valore=valore+dv
70 UNTIL(valore>255 OR valore<0)
```

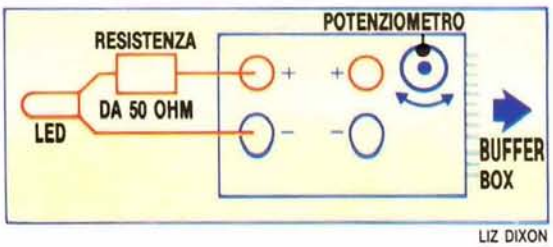
Il programma pone in uscita tutte le linee della porta utente, depositando il valore 255 (11111111 in binario) nel registro di direzione dei dati, quindi carica il valore iniziale 127 nel registro dati della porta utente. I tasti Z e X rispettivamente decrementano ed incrementano il valore contenuto nel registro dati: il programma termina quando questo diviene minore di 0 o maggiore di 255. Incrementando il valore "digitale" contenuto nel registro dati si ottiene un aumento del voltaggio "analogico" che alimenta il LED. La luminosità cresce con l'aumentare del voltaggio applicato ed è massima quando il valore contenuto nel registro dati della porta utente è pari a 255. Se il LED non si accende, si provi a invertire i collegamenti che raggiungono la scatola del convertitore D-A prima di cercare difetti di altro genere. Infatti, le normali lampade si accendono qualunque sia la direzione della corrente, mentre i LED funzionano solo quando il flusso di corrente ha una particolare direzione.

Il convertitore D-A

La levetta del potenziometro inserito in questo convertitore D-A è troppo lunga e, per un comodo impiego dell'apparecchio, va accorciata. Spesso bisogna adattare i componenti acquistati nei negozi di elettronica prima di poterli adoperare

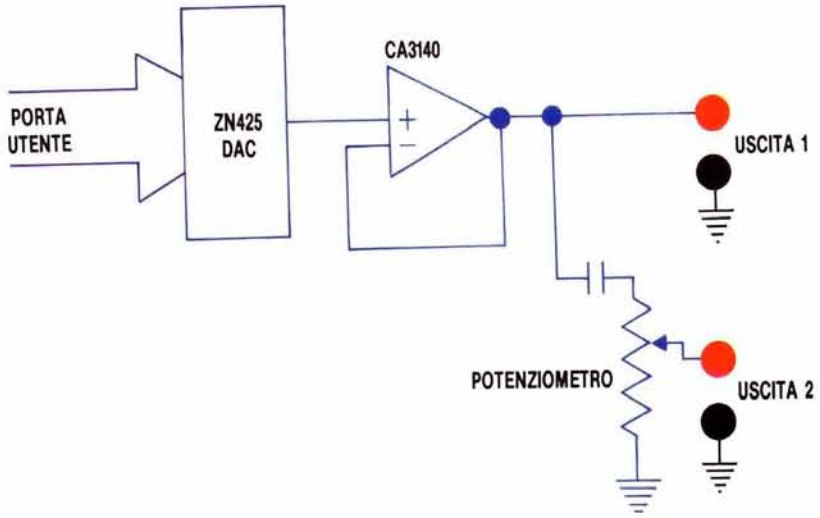


IAN McKINNELL



LIZ DIXON

IL CIRCUITO FINALE



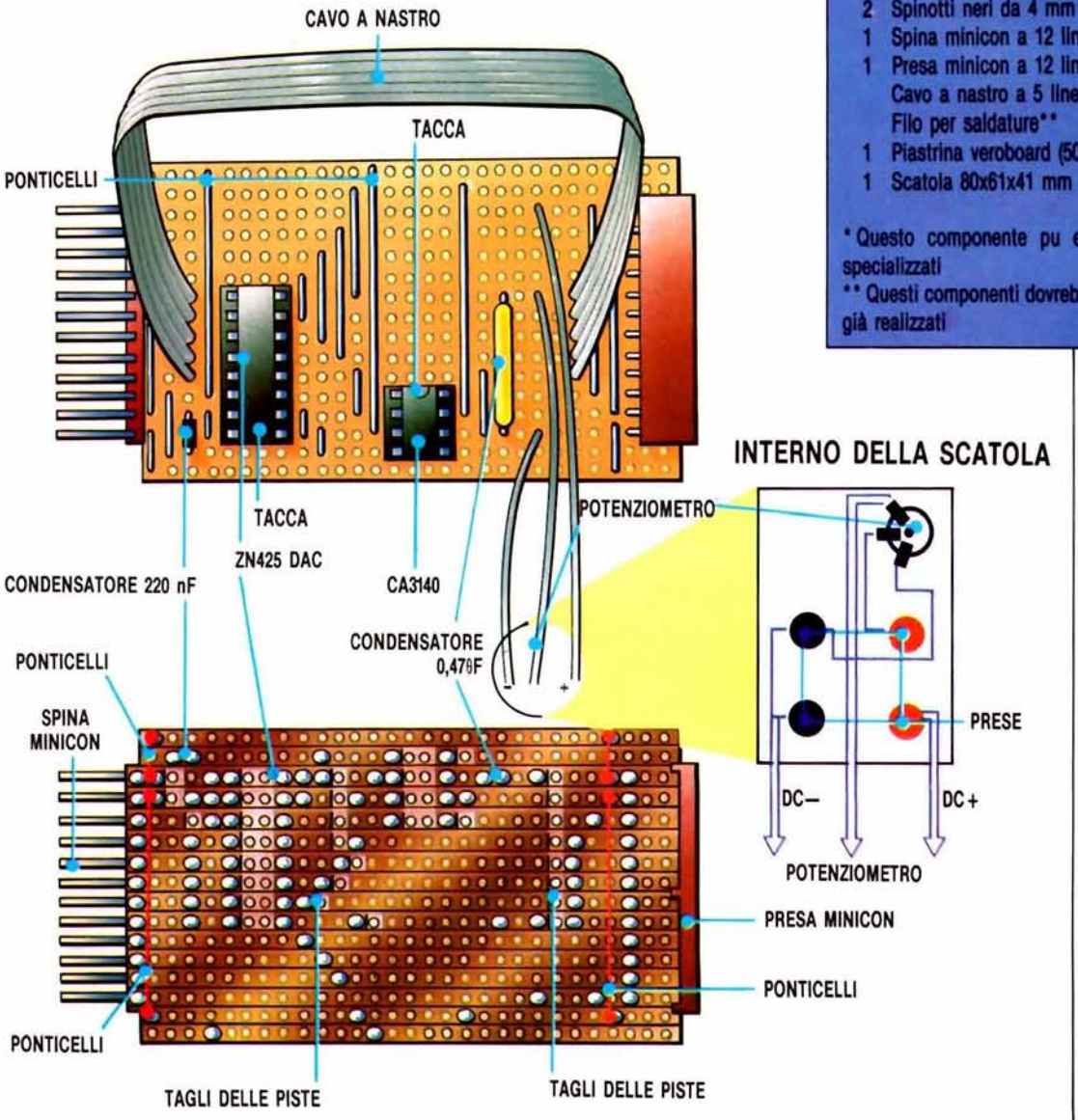
Quando si costruiscono circuiti di questo tipo, occorre molta attenzione nei particolari, soprattutto alla disposizione ed alla precisione dei tagli delle piste. Inoltre, ci si deve attenere alle seguenti regole: verificare ripetutamente i contatti con uno strumento di misurazione (multimetro), inserire per primi nel circuito i componenti passivi (resistenze e condensatori), non indugiare a lungo sui componenti col saldatore. Infine, fare molta attenzione alla orientazione dei chip

COMPONENTI

- 1 ZN425 DAC*
- 1 Amplificatore CA3140
- 1 Zoccolo DIL a 16 piedini
- 1 Zoccolo DIL a 8 piedini
- 1 Condensatore 220nF
- 1 Condensatore 0,47µF
- 1 Potenziometro da 10K
- 2 Prese rosse da 4 mm
- 2 Prese nere da 4 mm
- 2 Spinotti rossi da 4 mm
- 2 Spinotti neri da 4 mm
- 1 Spina minicon a 12 linee
- 1 Presa minicon a 12 linee
- Cavo a nastro a 5 linee**
- Filo per saldature**
- 1 Piastrina veroboard (50 fori x 24 piste)
- 1 Scatola 80x61x41 mm

* Questo componente pu essere acquistato nei negozi specializzati

** Questi componenti dovrebbero essere avanzati dai lavori già realizzati



KEVIN JONES

DATABASE

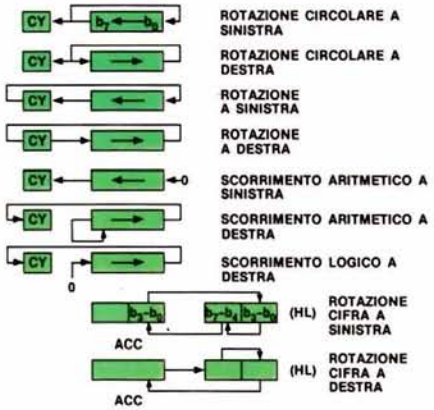
Settima scheda delle istruzioni per lo Z80 della Zilog. Istruzioni di scorrimento e rotazione

SORGENTE E DESTINAZIONE

TIPO DI SCORRIMENTO O ROTAZIONE

	A	B	C	D	E	H	L	(HL)	(IX + d)	(IY + d)
'RLC'	CB 07	CB 00	CB 01	CB 02	CB 03	CB 04	CB 05	CB 06	DD CB d 06	FD CB d 06
'RRC'	CB 0F	CB 08	CB 09	CB 0A	CB 0B	CB 0C	CB 0D	CB 0E	DD CB d 0E	FD CB d 0E
'RL'	CB 17	CB 10	CB 11	CB 12	CB 13	CB 14	CB 15	CB 16	DD CB d 16	FD CB d 16
'RR'	CB 1F	CB 18	CB 19	CB 1A	CB 1B	CB 1C	CB 1D	CB 1E	DD CB d 1E	FD CB d 1E
'SLA'	CB 27	CB 20	CB 21	CB 22	CB 23	CB 24	CB 25	CB 26	DD CB d 26	FD CB d 26
'SRA'	CB 2F	CB 28	CB 29	CB 2A	CB 2B	CB 2C	CB 2D	CB 2E	DD CB d 2E	FD CB d 2E
'SRL'	CB 3F	CB 38	CB 39	CB 3A	CB 3B	CB 3C	CB 3D	CB 3E	DD CB d 3E	FD CB d 3E
'RLD'								ED 6F		
'RRD'								ED 67		

	A
'RLCA'	07
'RRCA'	0F
'RLA'	17
'RRA'	1F



Codice mnemonico	Operazione simbolica	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
RLCA		•	•	X	0	X	•	0	:	00 000 111	07	1	1	4	Rotazione circolare sinistra dell'accumulatore
RLA		•	•	X	0	X	•	0	:	00 010 111	17	1	1	4	Rotazione a sinistra dell'accumulatore
RRCA		•	•	X	0	X	•	0	:	00 001 111	0F	1	1	4	Rotazione circolare a destra dell'accumulatore
RRA		•	•	X	0	X	•	0	:	00 011 111	1F	1	1	4	Rotazione a destra dell'accumulatore
RLC r		:	:	X	0	X	P	0	:	11 001 011 00 000 r	CB	2	2	8	Rotazione circolare a destra del registro r
RLC (HL)		:	:	X	0	X	P	0	:	11 001 011 00 000 110	CB	2	4	15	
RLC (IX + d)	r,(HL),(IX + d) (IY + d)	:	:	X	0	X	P	0	:	11 011 101 11 001 011 ←d→ 00 000 110	DD CB	4	6	23	
RLC(IY + d)		:	:	X	0	X	P	0	:	11 111 101 11 001 011 ←d→ 00 000 110	FD CB	4	6	23	
RL m	m = r,(HL),(IX + d),(IY + d)	:	:	X	0	X	P	0	:	010					Il formato delle istruzioni e degli stati è identico a quello per le istruzioni RLC. Per formare i relativi codici sostituire il gruppo 000 delle istruzioni RLC con quelli indicati
RRC m	m = r,(HL),(IX + d),(IY + d)	:	:	X	0	X	P	0	:	001					
RR m	m = r,(HL),(IX + d),(IY + d)	:	:	X	0	X	P	0	:	011					
SLA m	m = r,(HL),(IX + d),(IY + d)	:	:	X	0	X	P	0	:	100					
SRA m	m = r,(HL),(IX + d),(IY + d)	:	:	X	0	X	P	0	:	101					
SRL m	m = r,(HL),(IX + d),(IY + d)	:	:	X	0	X	P	0	:	111					
RLD		:	:	X	0	X	P	0	•	11 101 101 01 101 111	ED 6F	2	5	18	Rotazione di una cifra a sinistra e a destra tra l'accumulatore e la locazione (HL).
RRD		:	:	X	0	X	P	0	•	11 101 101 01 100 111	ED 67	2	5	18	Il contenuto del semi-byte superiore dell'accumulatore rimane inalterato

r	Reg.
000	B
001	C
010	D
011	E
100	H
101	L
111	A

NOTA:

d rappresenta un numero con segno nella gamma < —128/ + 127 >

Notazione per i flag:

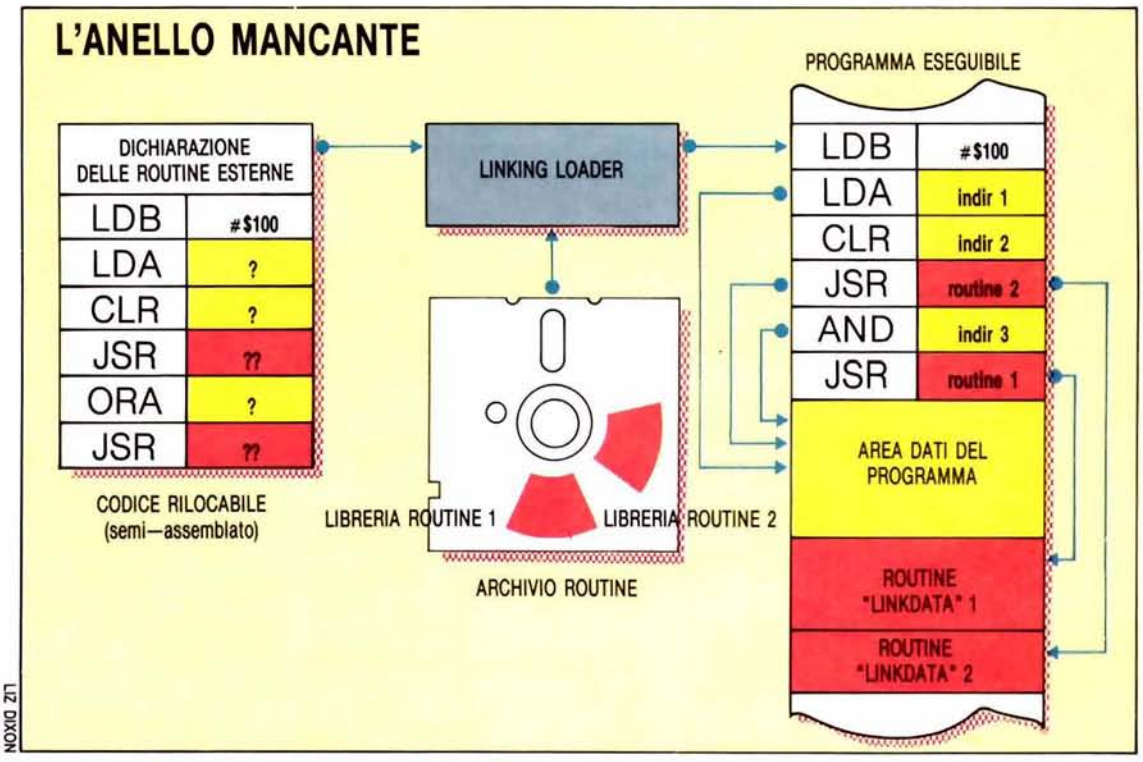
- • nessun effetto
- 0 flag azzerato
- 1 flag posto a 1
- X effetto non definito
- : il flag rispecchia il risultato dell'operazione
- p pari

per concessione della Zilog Inc.

Libero movimento

Codice rilocabile, lunghezza delle istruzioni e routine di temporizzazione

Un programma "rilocabile" si differenzia dagli altri perché il codice che lo compone può essere collocato ed eseguito ovunque in memoria, senza alcuna modifica. Questa caratteristica assume notevole importanza in sistemi multi-utente o multi-funzione, nei quali più programmi vengono simultaneamente caricati ed eseguiti in memoria. È il sistema operativo che assegna ad essi le zone di memoria libera.
Tuttavia, anche in applicazioni mono-utente, specialmente quando si tratta di raccogliere in una "libreria" routine di uso generico, l'impiego di programmi rilocabili è di enorme utilità.
Su molti computer questo aspetto è gestito dai "linking loader", che sono appositi programmi di caricamento del codice rilocabile, che provvede ad aggiustare opportunamente gli indirizzi dei salti e delle chiamate alle subroutine.
Grazie a questo sistema, più moduli (magari compilati in linguaggi diversi tra loro quali il BASIC e il FORTRAN) possono essere raccolti e utilizzati in uno stesso programma. Col chip 6809 tale compito è ancor più facilitato, poiché sono previste specifiche istruzioni che rendono i programmi rilocabili, eliminando così un passaggio in più. Il trucco consiste nello specificare gli indirizzi in termini di "offset" o scarto rispetto al Contatore di Programma (PC).
Il programma utilizza gli indirizzi in due modi ben distinti: come dati oppure come destinazione per un trasferimento di controllo. Le istruzioni di Branch (BRA, BSR, ecc.) calcolano la propria destinazione in base ad un valore di scarto rispetto al contenuto del PC e dovrebbero essere sempre preferite, nella redazione dei propri programmi, alle istruzioni di trasferimento assoluto (JMP e JSR). Queste ultime sono adatte soltanto quando l'indirizzo di destinazione è comunque fisso (si riferisce, ad esempio, a routine del sistema operativo o delle ROM).



Il "Linking loader"

In alcuni sistemi computerizzati, i programmi in codice macchina sono caricati in memoria da uno speciale programma chiamato "Linking Loader" (Caricatore di collegamento), che legge il programma semi-assemblato (privo di indirizzi assoluti) prodotto dall'assembler e che determina la migliore ORG da usare, in base alla corrente occupazione di memoria nel sistema. Usando tale indirizzo come base, il loader calcola tutti gli indirizzi assoluti occorrenti nel programma. Inoltre, aggiunge e collega ad esso le eventuali routine di libreria necessarie al funzionamento. Le routine vengono automaticamente estratte da un'apposita raccolta, detta "libreria", conservata su dischetto. Infine, il loader deposita in memoria (pronta per l'uso) o su dischetto (per essere riutilizzata più volte) la versione completa e definitiva del programma. In gergo, le routine estratte ed accodate dal loader vengono dette "linkdata"

Per quanto concerne gli indirizzi dei dati, esiste un semplice metodo per renderli indipendenti dalla posizione in memoria. L'istruzione:

```
LDA OFFSET,PC
```

serve a sommare un valore al PC, per ottenere l'"indirizzo effettivo" di un dato. L'unico problema, con tale metodo, è quello di calcolare correttamente l'offset, poiché implica il cal-

colo della differenza tra l'indirizzo dei dati e quello contenuto nel PC, che, tra l'altro, viene incrementato subito dopo il caricamento nella CPU di un'istruzione; mentre questa viene eseguita, durante l'esecuzione di un'istruzione, dunque, il PC punta già a quella successiva.
Tutto quanto è complicato ulteriormente dal fatto che la lunghezza delle istruzioni, nel 6809, non è affatto fissa: può variare da 1 a 5. Per esempio:

```
LDX OFFSET,Y
```

occupa un byte per l'op-code e un byte per il "post-byte G", che comunica al computer quale registro usare come indice e, inoltre, se si tratta di indirizzamento indiretto oppure no. Inoltre, l'offset può occupare da zero a due byte, a seconda delle sue dimensioni.
Gli offset 0 (e quelli il cui valore può essere espresso con soli 5 bit) vengono generalmente incorporati nel post-byte (benché gli assembler siano poco precisi in proposito). Offset di dimensioni maggiori occupano uno o due byte in più. In aggiunta, se il PC è usato nell'indicizzamento, non sono ammessi offset speciali di zero o cinque bit.
Inoltre, l'istruzione:

```
LDY OFFSET,X
```

impegna un byte in più, perché l'op-code di LDY è lungo due byte.
Per chi si diverte a programmare in Assembly, il calcolo degli indirizzi relativi ai dati e quello per ricavare l'offset da usare diventa quasi un fatto automatico. Tuttavia, esiste una più comoda alternativa: acquistare un buon assembler che provveda ad eseguire tutti i calcoli necessari.
Alcuni assembler, addirittura, prevedono una speciale notazione, PCR (Program Counter Relative), che costringe l'assembler ad usare il contatore di programma per calcolare l'offset necessario.
Ad esempio:

```
DATO    FCB   0
        LDA   DATO,PCR
```

Emulazione di un terminale

Qui di seguito viene presentata una subroutine che, ricorrendo alla tecnica appena descritta, consente di "emulare" tutta una serie di terminali. Ciò permette di eseguire sul proprio sistema programmi destinati a particolari tipi di terminali. La maggior differenza consiste nei codici adoperati dai terminali per funzioni quali la pulizia dello schermo e il posizionamento del cursore. Possono esser previsti codici di controllo (il cui valore ASCII è inferiore a 32) oppure "sequenze di escape" (composte dal codice ASCII 27, seguito da uno o più caratteri).
La nostra routine permette soltanto la sostituzione di un codice di controllo con un altro oppure del carattere di una sequenza di escape con un altro. Il listato mostra come avvenga la conversione.
Vengono utilizzate due tabelle: una contiene i codici di controllo e l'altra i caratteri usati nelle sequenze di escape. Se il programma adopera un codice di controllo, esso viene adoperato come indice nella rispettiva tabella, per ricavare il codice da usare al suo posto.
Essendo completamente rilocabile, la subroutine può essere aggiunta a qualsiasi programma ed in qualsiasi posizione. Viene data per scontata l'esistenza di una routine del sistema operativo (OUTCHR), che emette sullo schermo il carattere contenuto nell'accumulatore. Inoltre, l'indirizzo di tale routine è considerato fisso, tanto che viene usata un'istruzione JMP.
Si noti che la direttiva ORG va egualmente specificata, benché non abbia alcun effetto pratico.

Lunghezza delle istruzioni

Il problema che concerne il calcolo della lunghezza delle istruzioni non riguarda soltanto l'indirizzamento "relativo al contatore di programma".
Talvolta occorre conoscere con precisione la lunghezza di una routine in codice macchina per poterla collocare in uno spazio ristretto di memoria. Qualsiasi buon manuale sul codice macchina del 6809 riporta le tabelle dei codici mnemonici, assieme alle informazioni ad essi relative: l'op-code equivalente, la lunghezza dell'istruzione (benché ciò non sia sempre determinabile a priori, nel qual caso è indicata la lunghezza minima, seguita da un +), il numero di cicli di clock necessari all'esecuzione dell'istruzione e l'effetto della stessa istruzione sui vari flag del registro di condizione.
Le regole generali, per calcolare la lunghezza di un'istruzione, sono:

1) La maggior parte delle istruzioni sono composte da un solo byte, tranne quelle che coinvolgono S o Y (eccetto LEA) ed alcune che coinvolgono U (quali LDY e STS) le quali sono lunghe due byte.

2) Se viene impiegato l'indirizzamento indiretto, occorre un post-byte e, possibilmente uno o due byte per l'offset.

3) Nel modo immediato, i dati che seguono l'op-code occupano uno o due byte, a seconda della lunghezza del registro usato.

4) Gli indirizzi in pagina diretta (locazioni da $00 a $FF) occupano un byte, altrimenti due. Alcuni assembler, però, non sembrano apprezzare la differenza ed usano talvolta due byte quando ne basta uno.

Calcolare il tempo impiegato nell'eseguire un'istruzione è altrettanto complicato, non foss'altro perché questo dipende dallo stesso numero di byte che compongono l'istruzione,

e diviene critico quando il programma riguarda applicazioni in tempo reale o il controllo di alcune periferiche. Il tempo occorrente per ciascuna istruzione (o perlomeno il tempo minimo) è dato in cicli di clock. Quindi il tempo effettivo dipende dalla frequenza del clock di sistema. Una frequenza piuttosto standard, per il 6809, è di 1 MHz, ossia un milione di cicli al secondo, cosicché un ciclo di clock equivale a un milionesimo di secondo. La semplice istruzione:

```
LDA DATO
```

usa un indirizzo a 16 bit ed impiega 5 cicli (5 milionesimi di secondo), mentre l'istruzione:

```
PSHS PC, B, CC
```

impiega 5 cicli, più un ciclo per ogni byte appoggiato sullo stack: in questo caso un totale di 9 cicli (il registro PC è di due byte).
Invece, in sistemi privi di un clock in tempo reale, l'unico modo per misurare il tempo trascorso è ricorrere ad una routine di ritardo o di temporizzazione, che esegua una sequenza di istruzioni tale da far trascorrere l'intervallo di tempo desiderato. Di solito questi intervalli durano qualche millisecondo, per cui non importa una precisione assoluta (qualche milionesimo di secondo in più o in meno conta poco).
Supponendo che la frequenza di clock sia di 1 MHz, la routine di ritardo qui presentata produce intervalli compresi tra 1 e 255 millisecondi: l'esatto numero di millisecondi (ms) viene passato alla routine tramite l'accumulatore A. Il calcolo per definire la costante CONTO può essere così espresso:

Istruzioni	Numero di Clock Cicli	Numero di tempo Esecuzioni	Tempo occorrente (Clock Cicli
PSHS B,CC	7	1	7
LDB#COUNT	2	(A)	(A)*2
DECB	2	(A)*COUNT	(A)*COUNT*2
BNE LOOP2	3	(A)*COUNT	(A)*COUNT*3
DECA	2	(A)	(A)*2
BNE LOOP1	3	(A)	(A)*3
PULS PC,B,CC	9	1	9

Il totale è dato da (A)*(7+5*CONTO)+16 cicli di clock. Per facilitare il calcolo, ignoriamo il 16. Ad una frequenza di 1 MHz, ci sono 1000 cicli di clock in un millisecondo, cosicché il tempo totale dovrebbe essere (A)*1000 cicli di clock.

```
(A)*(7+5*CONTO) = (A)*1000
    (7+5*CONTO) = 1000
          5*CONTO = 973
            x CONTO = 195 (arrotondato all'intero)
```

Il principio appena esposto vale anche per routine di ritardo più accurate, in cui vengono usati registri a 16 bit per eseguire più volte una stessa sequenza di istruzioni.

ROUTINE PER EMULAZIONE DI TERMINALE VIDEO

```
ESCAPE  EQU   27
SPAZIO  EQU   32        (32 = spazio in ASCII)
OUTCHR  EQU             Inserire l'indirizzo della routine di
                        visualizzazione
        ORG   $1000
CTABEL  RMB   32        Tabella dei codici di controllo
ETABEL  RMB   128       Tabella dei caratteri di 'Escape'
EFLAG   FCB   0         Questo flag indica se l'ultimo ca-
                        rattere è un Escape
DISPCH  PSHS  X         Salva X
        TST   EFLAG,PCR Controlla se ultimo carattere era
                        Escape
        BEQ   DISP1     Se non lo è, salta a DISP1
        LEAX  ETABLE,PCR Altrimenti carica ETABEL in X
        LDA   A,X       Estrae il carattere dalla tabella,
                        usando A come 'offset'
        CLR   EFLAG,PCR Azzera il flag
        BRA   FINITO
DISP1   CMPA  SPAZIO    Verifica se è un carattere di con-
                        trollo
        BGE   FINITO    Se non lo è, salta a FINITO
        CMPA  ESCAPE    Altrimenti, verifica se è Escape
        BEQ   ESCCH     Se lo è, salta a ESCCH
        LEAX  CTABEL,PCR Indirizzo di CTABEL in X
        LDA   A,X       Estrae il carattere dalla tabella,
                        usando A come 'offset'
        BRA   FINITO
ESCCH   INC   EFLAG,PCR EFLAG=1 per indicare Escape
FINITO  PULS  X         Recupera X
        JMP   OUTCHR    Visualizza il carattere in A
        END             Si noti che la RTS posta alla fine
                        di OUTCHR fa tornare il controllo
                        al programma chiamante
```

ROUTINE DI RITARDO

```
                        Subroutine per creare ritardi di (A)
                        millisecondi
CONTO   EQU   195       (vedere testo)
        ORG   $1000
RITAR   PSHS  B,CC      Salva gli altri due registri usati
CICL1   LDB   #CONTO    Conto per 1 ms
CICL2   DECB            Continua a decrementare
        BNE   CICL2     finché B diventa zero
        DECA            Decrementa A dopo ogni ms
        BNE   CICL1     finché A diventa zero
        PULS  PC,B,CC   Ritorna al programma chiamante
```

Psytron

Col passare del tempo, i giochi diventano sempre più sofisticati

Lo *Psytron* è un gioco emozionante, dotato di ottima grafica a rapido movimento. Le regole sono piuttosto complicate e richiedono un lungo periodo di apprendimento.
Il giocatore interpreta il ruolo di Psytron, un essere metà uomo e metà computer, che opera in una colonia spaziale del pianeta Betula 5. La colonia è composta da vari edifici, ciascuno dei quali ha una propria funzione. I sistemi di sopravvivenza sono indispensabili per i coloni umani, mentre la piena efficienza dei generatori elettrici è fondamentale per mantenere operativo il computer. L'installazione più importante è il generatore principale, senza il quale l'intera colonia si estinguerebbe. La tastiera o il joystick permettono al giocatore di controllare costantemente il generatore, mentre lo schermo fornisce un'immagine realistica e dettagliata delle varie sezioni della colonia.
Il gioco si sviluppa in fasi o livelli: nelle fasi iniziali, il Psytron somiglia ad un qualsiasi altro gioco stile arcade; il giocatore controlla una serie di armi con le quali si difende dagli attacchi di invasori alieni. I veri problemi strategici si manifestano solo a partire dal quarto livello di gioco!
Al primo livello, Psytron si serve di un "droide", posto sotto il suo controllo, per distruggere i sabotatori a tre gambe che, "teletrasferiti" all'interno della base, cercano di distruggere i passaggi che collegano gli edifici della colonia. Al secondo ed al terzo livello il giocatore deve contrastare un'invasione di "salsicce" aliene abbattendole una alla volta: solo l'arma chiamata "Disruptor" consente di distruggere contemporaneamente tutte le salsicce visibili sullo schermo. Purtroppo il Disruptor è uno strumento poco affidabile e ci sono 10 probabilità su 100 che esploda durante l'uso.
Solo al quarto livello il giocatore può prendere decisioni di carattere strategico, in base ai danni riportati dalle istallazioni della colonia. Anche durante questa fase le astronavi aliene continuano ad attaccare bombardando le aree vitali della colonia e lanciando sabotatori con missioni suicide. Tuttavia, in questa fase, il giocatore può difendersi battendo il tasto Return e attivando la funzione "Freezetime" (Tempo Congelato), che blocca lo sviluppo dell'azione e gli consente di ricevere ed elaborare le comunicazioni sui danni subiti. Le informazioni da analizzare sono: il numero dei membri della colonia morti o feriti, il livello delle scorte ancora disponibili ed i danni subiti dal generatore principale. Durante il Freezetime, il giocatore può riparare i danni e trasferire i membri della colonia nelle aree in cui la presenza umana è di importanza vitale. Questa fase di gioco richiede molta fantasia ed abilità: Psytron deve tener conto del fatto che curare i membri della colonia comporta un consumo maggiore di scorte alimentari e di ossigeno che non riparare i droidi e che può essere necessario abbandonare alcuni edifici danneggiati per risparmiare carburante, aria e scorte.
Al quinto livello il giocatore può contattare un' astronave di soccorso e richiedere il lancio di rifornimenti indispensabili alla sopravvivenza. Si deve però controllare che le piste d'atterraggio non siano state danneggiate dagli attacchi precedenti.
Al sesto ed ultimo livello, le scelte strategiche assumono un'importanza primaria. In questa fase finale, l'unico obiettivo è sopravvivere ancora per un'ora, difendendo la base con tutte le armi ed i sistemi disponibili. Crescono sia il numero degli attaccanti che la velocità dell'azione e proteggere la colonia dal volume di fuoco scatenato dagli assalitori alieni sembra un'impresa disperata: il giocatore deve decidere in pochi istanti quali edifici sacrificare, poiché è essenziale difendere le piste d'atterraggio che consentono di ricevere provvidenziali rifornimenti.
I giochi per computer hanno fatto molta strada dall'epoca dello *Space Invaders*, e lo *Psytron* rappresenta un'alternativa realistica ed appassionante ai giochi di tipo arcade che fino ad oggi hanno dominato il mercato.

L'"occhio" del droide

Queste immagini sono state tratte dalle prime fasi di gioco dello "Psytron". Quando il droide caccia i sabotatori, che si muovono all'interno dei corridoi, il suo occhio esegue una "panoramica" su tutta la base. Grazie alla finestra posta nell'angolo in basso a destra dello schermo, il giocatore può vedere attraverso gli occhi del droide. Quando il sabotatore compare all'interno della finestra, il giocatore può annientarlo premendo il pulsante di fuoco



Per concessione della Spectrum

Psytron: per lo Spectrum a 48K e per il Commodore 64
Editori: Beyond Sofware (GB)
Autori: Tayo Olowu e Paul Voysey
Joystick: opzionale
Formato: su cassetta (Spectrum), su cassetta o disco (Commodore 64)

21495