SPECIALE

MSX COMPUTER MAGAZINE

N. 1/1985

Sped. in abb. post. Gr. III L. 9.000

CON UNA CASSETTA DI PROGRAMMI MSX

PER CHI COMINCIA

CORSO DI MSX BASIC

1ª PUNTATA

Spectravideo SVI 728

MSX SPRITE STORY

PHILIPS UNA MACCHINA TUTTA EUROPEA

IL PRIMO ARCHIVIO ELETTRONICO

ISTRUZIONI: TUTTE LE EQUIVALENZE

OLTRE LE BARRIERE DELL'HARD E DEL SOFT.

MSX Computer Magazine è edita da Arcadia srl, C.so Vitt. Emanuele 15, Milano. Pubblicità: Printer Pubblicità, Via Palmanova 131, Milano. Tel. 02/25.91.957. Una copia L. 9.000. Fotocomposizione: Composit. Fotolito: Eurofotolit. Stampa: Garzanti, Milano. Distribuzione: SO.DI.P. Angelo Patuzzi srl, Via Zuretti 25, Milano. Registrato Trib. Milano (in attesa assegnazione numero). Resp. Sira Rocchi. Sped. in abb. post. Gr. III/70.

IN QUESTO NUMERO

SOMMARIO

MSX TAPE SOFT

Avete tutti trovato, allegata a questo fascicolo, una cassetta nella quale si è cercato di darvi un assaggio di quello che si può fare con gli MSX. Ci è sembrato doveroso risparmiarvi lunghi listati passibili di errori e ladri di tempo.
Naturalmente abbiamo in serbo cose sempre migliori ed interessanti per i mesi futuri. Basterà seguirci per approfittarne!
Eccovi intanto sei programmi per cominciare: è ovvio che vanno bene per qualsiasi computer MSX abbiate.
Ricordiamo di collegare al registratore anche il cavetto del controllo motore affinché possa essere completamente automatizzato il caricamento dei successivi programmi. Se il vostro registratore non possiede la presa del controllo motore, allora fate particolare attenzione a quando un programma è stato caricato o deve essere caricato, affinché il nastro scorra per il giusto tempo. Ricordiamo, infatti, che tra un programma e l'altro vi sono circa 10 secondi di pausa. I caratteri racchiusi tra virgolette, nelle parentesi, rappresentano il nome con cui è stato salvato il programma.

Quindi se si vuole caricare un determinato programma digitate:

LOAD "CAS: nome programma ", R

altrimenti

LOAD "CAS:",R

e vedrete il primo programma che il computer leggerà da nastro.
Sul lato A, dopo il programma introduzione (titolo MSXCM1) che è la «copertina» della rivista, il gioco dello Slalom, l'utility Testo in Screen ed ancora un gioco, Ultra Code.
Sul lato B un altro gioco, Black Car, e due interessanti utility, Sprite Design e Sound Explorer.
Svelti, caricate la cassetta e buon divertimento!!!

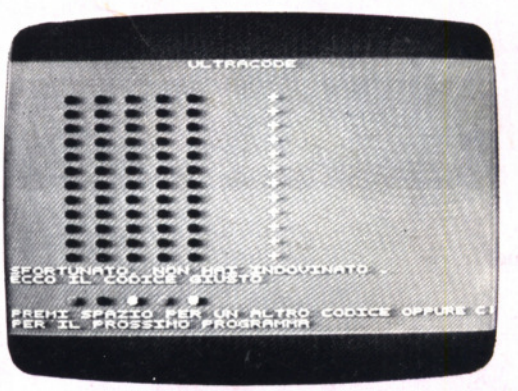
Siete su una favolosa pista da sci contornata da conifere e dovete portare a termine una gara di slalom gigante. Se ce la farete passerete ad un'altra gara di difficoltà superiore. Attenzione, però, perché potete essere squalificati solo tre volte. Il programma è un buon esempio di utilizzo degli sprites e anche del potente linguaggio macchina Z80.

Comandi:
- tasto cursore sinistra: sposta lo sciatore a sinistra
- tasto cursore destra: sposta lo sciatore a destra
- per caricare il programma successivo spegnere il computer, riaccenderlo e digitare LOAD"CAS:",R.

Dal gioco passiamo ad una utility. Purtroppo il BASIC-MSX non può gestire testo e grafica contemporaneamente (a meno di procedure di apertura file); ecco allora per voi un programma che vi risolve tale problema. La sua lunghezza è tale da poter essere utilizzato, senza problemi di memoria, in altri programmi. Nella sua esecuzione, oltre ad un esempio, vi è la descrizione dell'intero listato; questo è assai importante per poter sfruttare al meglio l'utility. Vi anticipiamo la sua conoscenza dicendo che sfrutta le stringhe assegnate all'istruzione DRAW.
Il caricamento del programma successivo è chiesto esplicitamente.

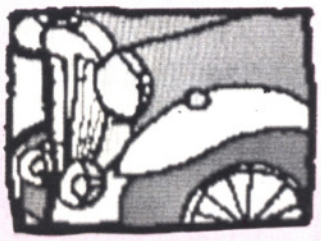
Siamo passati dal gioco all'utility ed eccoci nuovamente ad un gioco. Ultracode non è nient'altro che Master Mind rappresentato con delle palline colorate anziché con i numeri. Tu dovrai indovinare la combinazione formulata dal computer avendo a disposizione 12 tentativi. Il livello di difficoltà va da 4 a 6 palline. Tale gioco sfrutta l'utility Testo in screen 2.

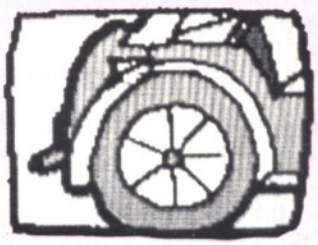
Comandi:
- Tasto cursore alto e basso: cambia il colore della pallina
- Tasto cursore sinistra e destra: specifica la pallina che deve cambiare colore.



4

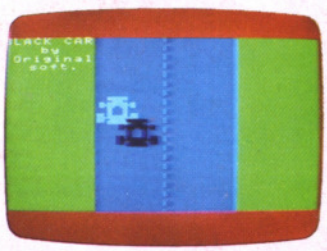
BLACK CAR
("BCAR")

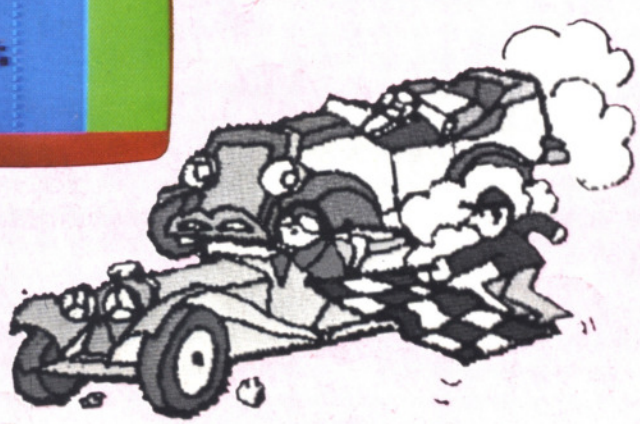




Ancora giochi con questa corsa di formula 1. Pilotate l'auto nera a velocità sempre più elevata ma state attenti alle auto che sorpassate. Se vi scontrate perdete una delle 5 vetture a vostra disposizione. Riuscirete a battere il record? Correte subito a provarci.
Il programma, di limitata lunghezza, utilizza in modo semplice le possibilità di controllo degli sprites del BASIC MSX. Vi consigliamo di dare anche un'occhiata al listato; chissà che non vi sia qualcosa di utile anche per i vostri lavori di programmazione.

Comandi:
- Tasto cursore alto: accelera
- Tasto cursore basso: frena
- Tasto cursore destra e sinistra: spostano l'auto rispettivamente a destra e a sinistra.





HITACHI

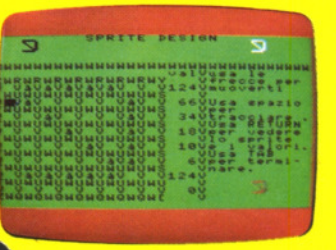
5

SPRITE DESIGN
("SPRITE")

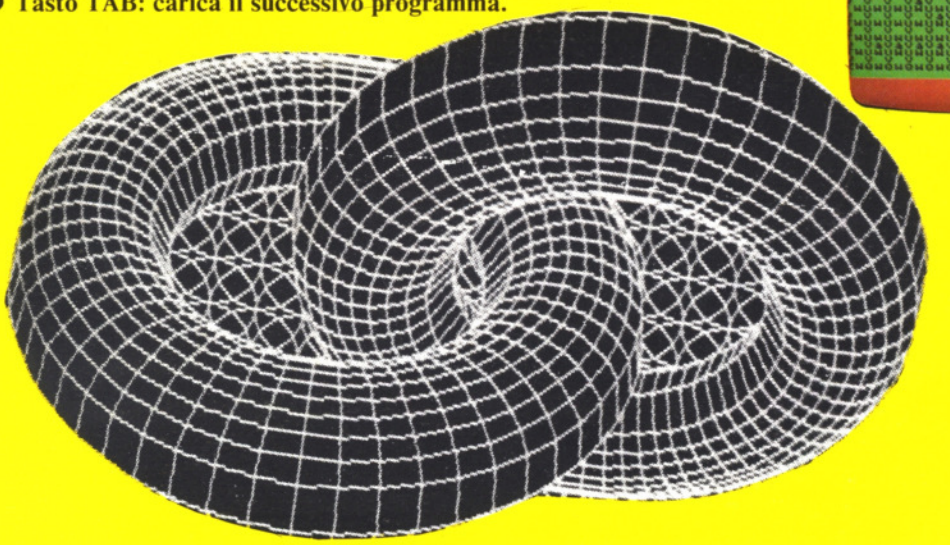
6

SOUND EXPLORER
("SOUND")

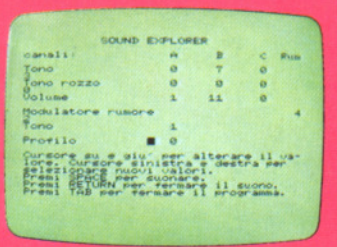
Lo sapevate che il vostro computer MSX è in grado di tenere fino a 256 sprites di formato piccolo? Tanti! Però vi sarete anche accorti che è noioso doverli definire. Non preoccupatevi, Sprite Design vi aiuterà in questo con estrema facilità. Infatti basterà muovere un cursore su di una matrice di 64 caselle (rappresentante lo sprite) e, con appositi tasti, disegnare la figura che dovrà contenere lo sprite. Quando avrete finito il vostro disegno, potrete vedere lo sprite in differenti colori sullo schermo e gli 8 valori necessari per generarlo.



Comandi:
- Tasto di SPAZIO: traccia (disegna)
- Tasto RETURN: visualizza lo sprite creato e i suoi valori
- Tasto TAB: carica il successivo programma.



Chiudiamo la rassegna software con un programma che vi aiuterà ad esplorare le capacità sonore del vostro computer MSX. L'istruzione PLAY consente, con estrema facilità, di far suonare al computer ben 8 ottave su tre canali differenti. Però se si volessero realizzare effetti speciali, come per il programma TITOLO, bisognerebbe ricorrere all'istruzione SOUND che agisce direttamente sul processore sonoro. L'uso di questa istruzione, però, comporta lo studio dei registri dell'AY-3-8910.
Non volete studiare e siete curiosi di adoperare SOUND? Ecco per voi il programma che vi aiuterà a conoscere, giocando, il processore sonoro e gli effetti speciali che esso può generare con l'istruzione appena citata.



Comandi:
- Tasti cursore alto e basso: alterano i valori dei registri
- Tasti cursore destra sinistra: selezionano il registro da alterare
- Tasto di SPAZIO: esegue il suono programmato
- Tasto RETURN: ferma il suono
- Tasto TAB: ferma l'esecuzione del programma.

macchine

MSX

L'SVI-728 MICROCOMPUTER STANDARD MSX SUPER COMPLETO

SPECTRAVIDEO

Probabilmente alcuni di voi vorranno comprare prima o dopo un microcomputer standard MSX. Proviamo a vederne qualcuno, per esempio lo Spectrovideo (USA) che subito appare come una macchina dalle caratteristiche professionali. Oltre ad una comoda tastiera tradizionale di tipo QWERTY, risiede sulla parte destra dell'SVI-728 anche un tastierino numerico unito ai tasti delle quattro operazioni, del controllo del cursore ed altri. Questo computer incorpora tre processori che gli permettono di realizzare una fantastica grafica (VDP), un suono da sintetizzatore musicale (PSG) ed un potente linguaggio di programmazione (Z80A).

L'interprete BASIC risiede in 32K di ROM, mentre al programma utente sono dedicati 64K di RAM. Rimangono altri 16K di RAM che sono utilizzati per le informazioni da inviare al video (non dimentichiamo che sono disponibili 16 colori e la possibilità di gestire fino a 32 sprites).

Dando un'occhiata ai lati della macchina è possibile identificare ben otto prese. Ciò dimostra l'alto grado di espandibilità che questa macchina può supportare.

Sulla destra della scritta SVI 728 è alloggiato uno slot per l'inserzione, in senso verticale, di cartridge quali l'SVI-727 per espandere il video ad 80

SPECTRAVIDEO MSX SVI-728

colonne e per poter far girare i programmi sotto CP/M; l'SVI-737 MSX che è un modem con la velocità di trasmissione di 300 Baud, comprendente anche una interfaccia seriale standard RS 232; l'SVI-757 che invece è solamente l'interfaccia RS 232; l'SVI-747 che consente la espansione di 64K di RAM utente, oppure uno dei tanti MSX game su cartridge, dalle sorprendenti e sofisticate elaborazioni. Sul lato posteriore del computer sono predisposte: la presa TV, la presa monitor, una porta per dispositivi di 1/0 quali l'SVI-707 MSX Disk Drive, una interfaccia parallela Centronics per il collegamento di una stampante e, per finire, una normale presa per il collegamento di un qualsiasi registratore a cassetta. Inoltre bisogna dire che tra la uscita monitor e la porta di 1/0 vi è una piccola, ma comoda, presa della massa del sistema. Ora guardando sul lato sinistro della macchina si vedono due porte per l'alloggio di due joystick già disponibili con la sigla SVI-101. Ovviamente vi è una nona presa su lato destro dello SVI-728 che è comune a tutti i computer: la presa dell'alimentazione.

Prima di passare a vedere le capacità operative della macchina, vorremmo ricordare un'altra particolarità strutturale dell'SVI-728 per sottolineare l'accuratezza con cui è stato progettato questo computer e cioè la predisposizione, nella parte inferiore a sinistra della tastiera, di un tasto di accensione del sistema corredato di relativa lucina rossa di power on.

Lo schermo, in forma di testo, può assumere una lunghezza massima di 40 caratteri per linea attraverso l'istruzione WIDTH n. In modo grafico, invece, lo schermo può presentarsi in alta e bassa risoluzione. Cioè quando la griglia è di 64x48 punti si è in bassa risoluzione, mentre quando è di 256x192 punti, si è in alta risoluzione.

I comandi grafici sono numerosi e sarebbe troppo lungo elencarli tutti; ricorderemo semplicemente alcuni particolari come DRAW stringa, che consente tramite delle macro istruzioni grafiche (codici GML) di eseguire qualsiasi figura sullo schermo; la VPEEK e la VPOKE per la lettura e la scrittura diretta in memoria video e il comando SPRITE$ per definire un modello sprite.

Come detto all'inizio, l'SVI-728 offre anche un'alta qualità del suono che può essere gestito in modo molto semplice tramite la programmazione dei registri del PSG. Questa programmazione è, come nel caso della grafica, aiutata da una serie di macro istruzioni dedicate al suono.

Per agevolare l'introduzione dei comandi Basic è possibile definire ben 10 tasti funzione dislocati nella parte alta della tastiera. Inizialmente questi tasti sono predefiniti con funzioni di colore, di autonumerazione delle linee di programma, salto, lista, run, caricamento e continua prosecuzione programma.

La manipolazione dei dati può essere suddivisa in quattro tipi: alfanumerici, interi, singola precisione e doppia precisione. Inoltre è possibile specificare un dato numerico nella forma decimale, binaria, esadecimale ed ottale (funzioni BIN$, HEX$, OCT$).

Le istruzioni di input/output sono anch'esse numerose ed alcune sono esclusivamente dedicate al registratore a cassetta come MOTOR ON/OFF e SOUND ON/OFF.

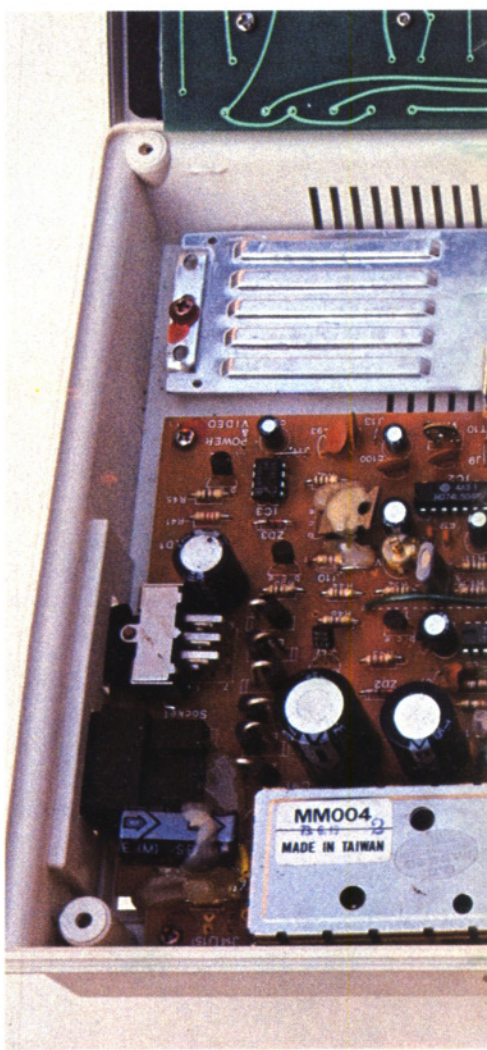
Particolare attenzione va rivolta anche alle istruzioni di controllo dell'esecuzione del programma Basic tra le quali spicca di utilità l'istruzione ON SPRITE GOSUB; con essa è possibile controllare la collisione sul video di sprites agevolando parecchio i programmi grafici e di video games.

Abbiamo provato e collaudato la macchina che ci appare dunque veramente completa: la Comtrad, importatrice per l'Italia, ci ha assicurato che sono già disponibili moltissime periferiche e straordinari programmi. Cercheremo di parlarne una prossima volta.

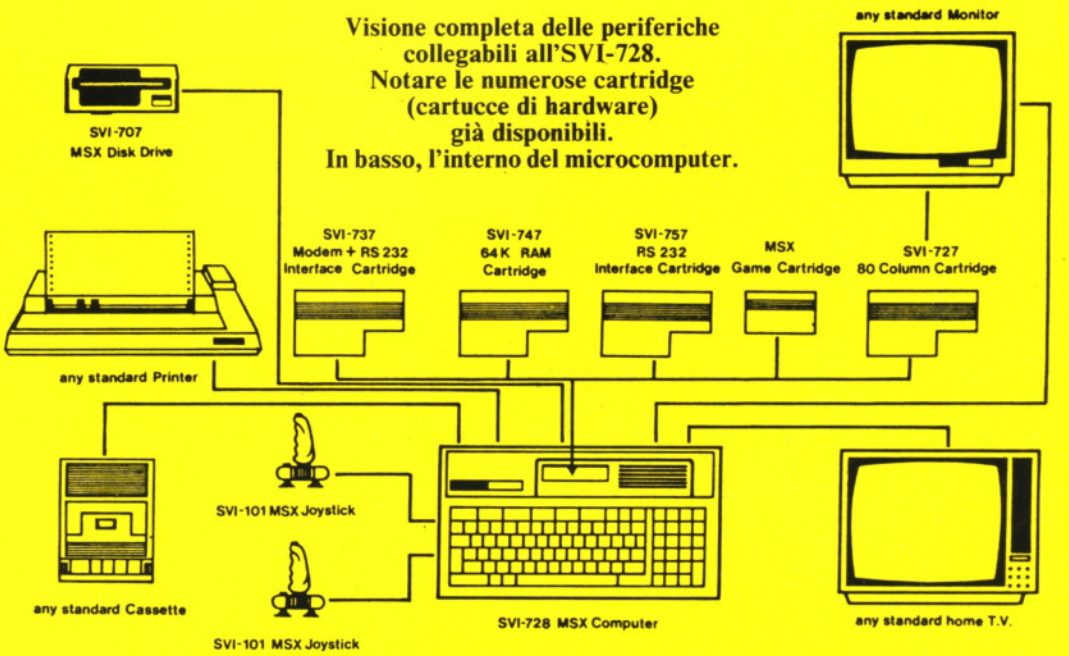
LE CARATTERISTICHE

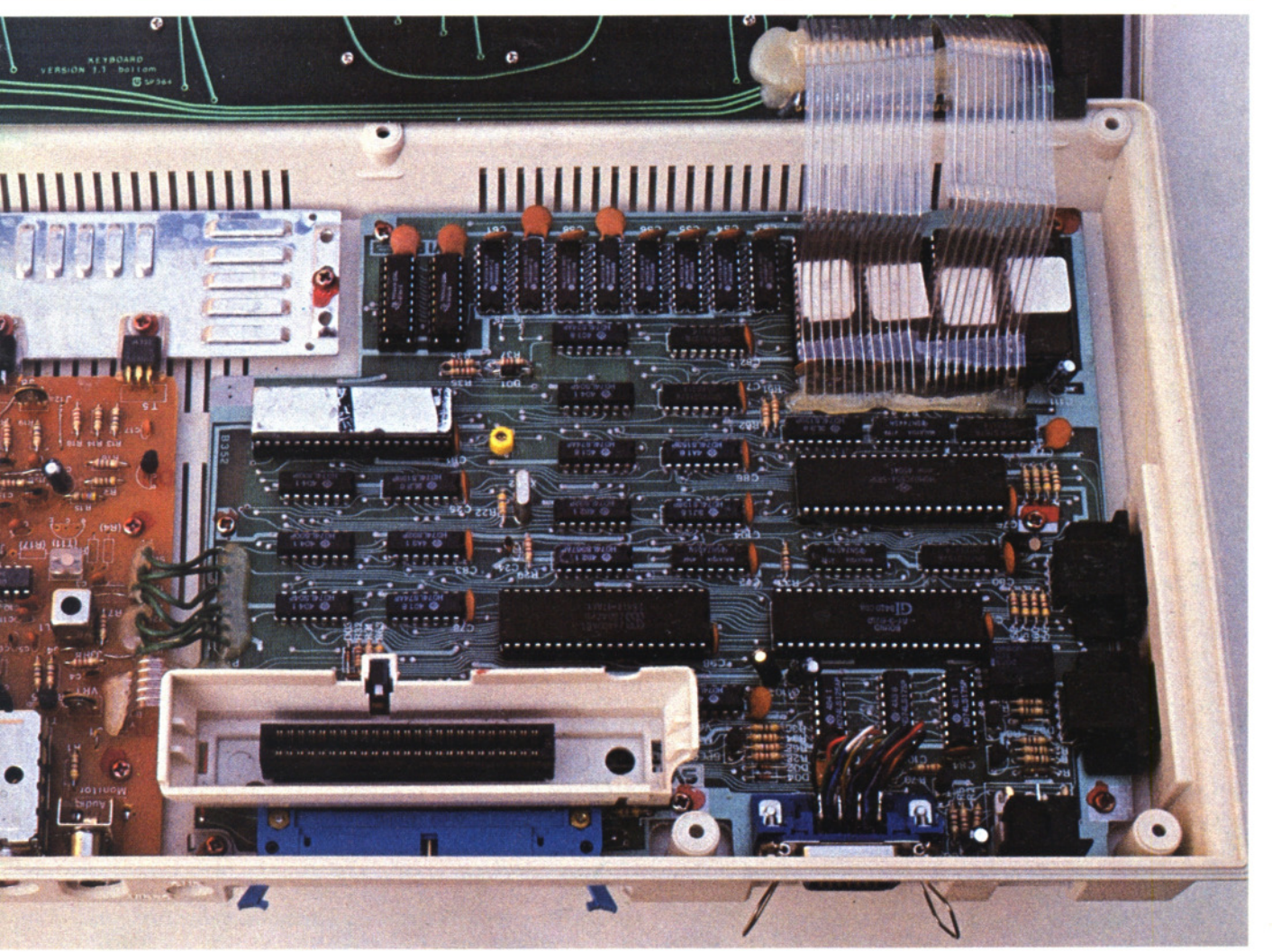
- **Linguaggio: MSX BASIC**
- **Processore: Z80A (inoltre possiede un processore video e un processore sonoro)**
- **Memoria RAM: 16K video e 64K utente espandibile a 128K**
- **Memoria ROM: 32K**
- **Risoluzione: 64x48 oppure 256x192**
- **Colori: 16**
- **Sprites: 32**
- **Uscita video: monitor o TV**
- **Interfacce: cassette e parallela**
- **Prese: 1 slot per cartridge, 2 per joystick, 1 per Disk Drive**

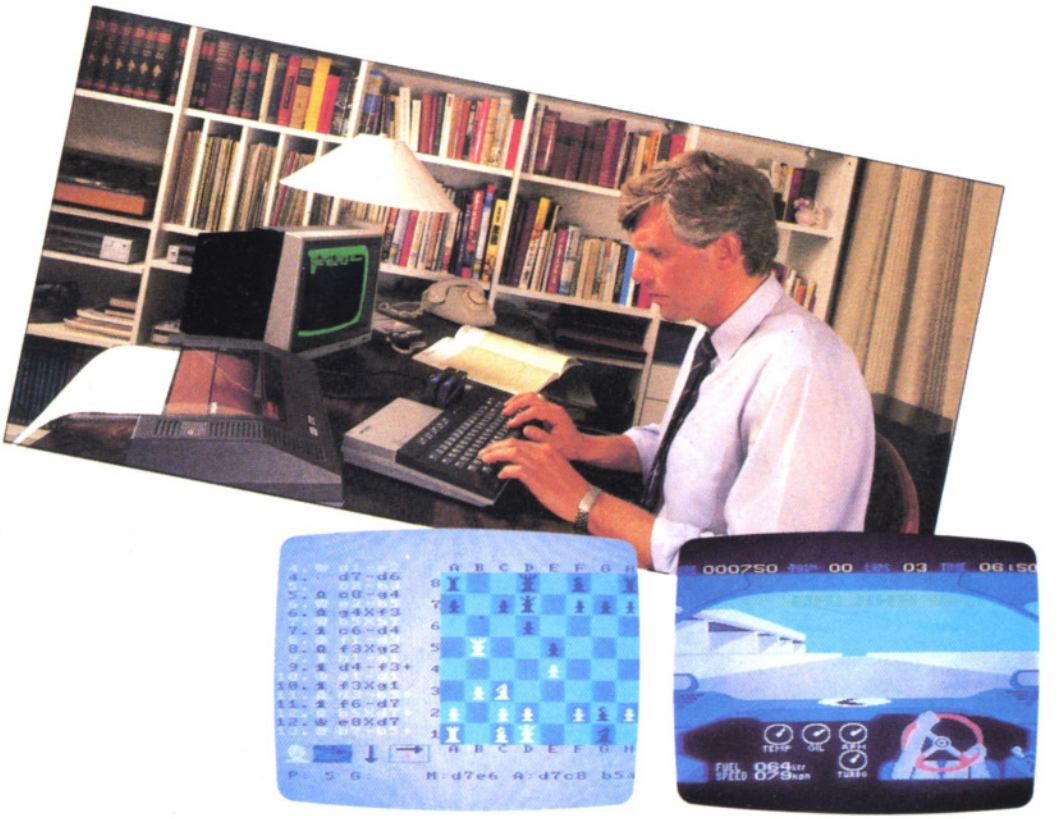
Importatore:
COMTRAD Div. Computers,
Tel. (0586) 424348.



Visione completa delle periferiche collegabili all'SVI-728. Notare le numerose cartridge (cartucce di hardware) già disponibili. In basso, l'interno del microcomputer.







MSX SPRITE STORY

PASSO DOPO PASSO LE FANTASTICHE POSSIBILITÀ GRAFICHE CHE IL VOSTRO SISTEMA MSX PUÒ DARVI

Con il basic MSX è naturalmente possibile creare effetti di animazione, comunemente detti sprite.

Vediamo come l'argomento è trattato nell'ottimo manuale Philips dato in dotazione insieme al sistema MSX VG-8000.

Una delle tipiche proprietà degli sprites è la possibilità di apparire in movimento in posizione più o meno avanzata sullo schermo.

Definizione degli sprites

Per usare gli sprites, occorre soddisfare due condizioni:

— Lo sprite deve essere definito, ossia occorre determinare in altre parole a quale oggetto deve assomigliare.

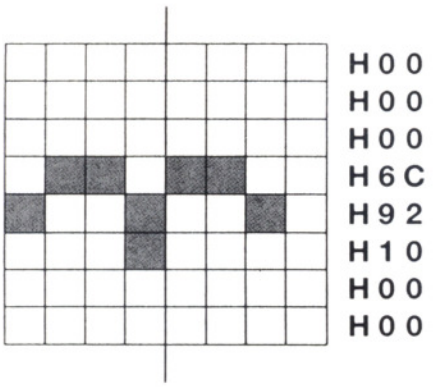
— Occorre decidere dove compariranno gli sprites sullo schermo.

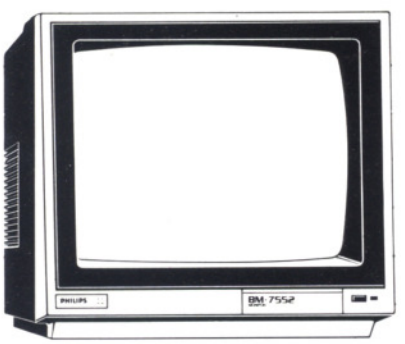
Gli sprites possono essere definiti in formato grande o piccolo. Un formato piccolo significa che gli sprites saranno composti da 8x8 punti di immagine. Usando sprites di formato piccolo è possibile definirne un massimo di 256. La definizione di uno sprite è espressa nella variabile «SPRITE$».

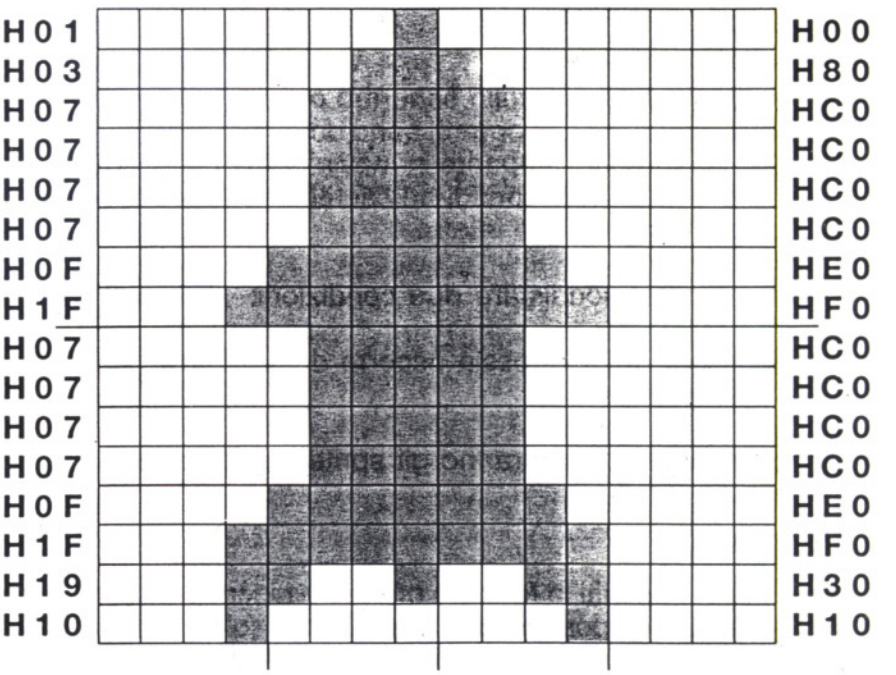
Si supponga di voler definire lo sprite della immagine piccola in alto.

Ogni otto punti di immagine orizzontali formano un byte, che viene diviso in due gruppi di 4 bit. Questi sono definiti in forma esadecimale come indicato alla destra delle illustrazioni. La dimensione dello sprite è indicata nell'istruzione «SCREEN» che segue il modo dello schermo.

Il suddetto sprite potrebbe essere codificato come segue (qui appresso il primo programma listato per sicurezza direttamente da stampante).







```
10 SCREEN 2,0
20 A$=CHR$(&H00)+CHR$(&H00)+CHR$(&H00)+CHR$(&H6C)+
CHR$(&H92)+CHR$(&H10)+CHR$(&H00) +CHR$(&H00)
30 SPRITE$(1)=A$
```

Gli sprites possono anche essere definiti in formato grande, nel qual caso ognuno di tali sprites si comporrà di 16x16 punti di immagine.

Usando sprites di formato grande, è possibile definirne un massimo di 64. Si supponga di volere creare uno sprite come indicato nella illustrazione grande.

Ogni 16 punti orizzontali di immagine formano 2 byte, ognuno dei quali viene diviso in due gruppi di 4 bit ciascuno. Questi sono codificati in formato esadecimale.

I byte sono in sequenza verticale. Lo sprite dell'illustrazione potrebbe essere codificato come segue:

```
10 SCREEN 2,2
20 A$=CHR$(&H01)+CHR$(&H03)+CHR$(&H07)+CHR$(&H07)+
CHR$(&H07)+CHR$(&H07)+CHR$(&H0F)+CHR$(&H1F)
30 B$=CHR$(&H07)+CHR$(&H07)+CHR$(&H07)+CHR$(&H07)+
CHR$(&H0F)+CHR$(&H1F)+CHR$(&H19)+CHR$(&H10)
40 C$=CHR$(&H00)+CHR$(&H80)+CHR$(&HC0)+CHR$(&HC0)+
CHR$(&HC0)+CHR$(&HC0)+CHR$(&HE0)+CHR$(&HF0)
50 D$=CHR$(&HC0)+CHR$(&HC0)+CHR$(&HC0)+CHR$(&HC0)+
CHR$(&HE0)+CHR$(&HF0)+CHR$(&H30)+CHR$(&H10)
60 SPRITE$(1)=A$+B$+C$+D$
```

Gli sprites possono anche essere codificati in un modo diverso, ad esempio per lo sprite in formato piccolo:

```
10 SCREEN 2,0
20 DATA 0,0,0,108,146,16,0,0
30 A$=""
40 FOR I=1 TO 8
```

```
50 READ A:A$=A$+CHR$(A)
60 NEXT I
70 SPRITE$(1)=A$
```

In questo esempio il valore decimale di otto byte è memorizzato in un'istruzione «DATA». Questi valori sono letti nella riga 50 e memorizzati nella variabile «A$» come valori alfanumerici.

Lo sprite di formato grande potrebbe essere codificato come segue:

```
10 SCREEN 2,2
20 DATA 1,3,7,7,7,7,15,31
30 DATA 7,7,7,7,15,31,25,16
40 DATA 0,128,192,192,192,192,224,240
50 DATA 192,192,192,192,224,240,48,16
60 A$=""
70 FOR I=1 TO 32
80 READ A:A$=A$+CHR$(A)
90 NEXT I
100 SPRITE$(1)=A$
```

In questo esempio, il valore decimale di tutti i 32 byte è memorizzato in 4 istruzioni «DATA». Questi valori sono letti nella riga 80 e memorizzati nella variabile «A$» come valori alfanumerici.

Inserimento degli sprites sullo schermo

Nell'istruzione «SCREEN» viene inserito un codice immediatamente dopo il modo schermo, che indica quali tipi di sprites saranno usati. Il codice usato ha il seguente significato:

0 = piccoli sprites, formato 8x8
1 = piccoli sprites, allargati al formato 16x16
2 = grandi sprites, formato 16x16
3 = grandi sprites, allargati al formato 32x32

Nella seguente riga di programma si indica per prima cosa il modo dello schermo (2), seguito dall'informazione relativa al formato degli sprites, in questo caso 32x32.

10 SCREEN 2,3

Non lo si è finora detto ma è evidente da quanto sopra che gli sprites possono essere allargati al doppio della dimensione originariamente definita.

Ci sono alcune limitazioni che riguardano l'uso degli sprites:

a. Gli sprites non possono essere usati nei modi testo 1 e 2
b. Non è possibile usare una combinazione di dimensioni di sprites diverse
c. È possibile inserire simultaneamente su una riga orizzontale dello schermo un massimo di 4 sprites.

Uno sprite viene inserito sullo schermo attraverso l'istruzione «PUT SPRITE», seguita dalle seguenti informazioni:

a. Numero priorità
b. Posizione sullo schermo
c. Colore dello sprite
d. Numero dello sprite.

Il numero di priorità deve essere compreso fra 0 e 31 ed ha il seguente significato: quando due sprites sono inseriti nella stessa posizione sullo schermo, lo sprite con la priorità minore precederà quello con il numero di priorità maggiore.

La posizione sullo schermo è indicata dalle coordinate X e Y. La coordinata X può essere un numero da —32 a 255 e la coordinata Y un numero da —32 a 191. Ciò consente di disporre uno sprite al di fuori dello schermo. Le coordinate X e Y indicano la posizione del punto più in alto e più a sinistra dello sprite.

Anche in questo caso le coordinate possono avere un valore relativo o assoluto. Quando la coordinata Y ha un valore di 209 lo sprite scompare dallo schermo.

Il colore dello sprite è indicato dal numero di codice del colore.

Il numero dello sprite infine determina quale sarà lo sprite che verrà disposto sullo schermo.

Interrompere il programma corrente (premere i tasti CTRL + STOP) ed impostare il seguente esempio:

```
NEW
10 SCREEN 2,0
20 DATA 0,3,31,63,255,255,63,7
30 DATA 0,128,248,255,255,254,252,192
40 DATA 0,0,0,108,146,16,0,0
50 FOR I=1 TO 3
60 A$=""
70 FOR J=1 TO 8
80 READ A:A$=A$+CHR$(A)
90 NEXT J
100 SPRITE$(I)=A$
110 NEXT I
120 PUT SPRITE 4,(180,50),15,1
130 PUT SPRITE 5,(188,50),15,2
140 FOR I=-32 TO 212
150 PUT SPRITE 3,(I,50),1,3
160 NEXT I
170 PUT SPRITE 3,(I,209)
180 FOR I=212 TO -32 STEP -1
190 PUT SPRITE 3,(I,50),1,3
200 NEXT I
210 GOTO 140
RUN
```

Se il programma è stato compilato correttamente, si vedrà comparire un uccello nero che vola attraverso una nube bianca. La nube bianca è stata definita con due sprites (sprite numero 1 e 2) mentre l'uccello nero è stato definito con lo sprite numero 3.

La nube bianca compare sullo schermo come conseguenza delle istruzioni nelle righe di programma 120 e 130.

L'uccello nero si muove da sinistra verso destra come indicato dall'iterazione «FOR... NEXT» nelle righe di programma da 140 a 160. L'iterazione «FOR... NEXT» nelle righe da 180 a 200 fa sì che l'uccello torni indietro, spostandosi da destra verso sinistra, ecc.

Interrompere il programma (premere contemporaneamente i tasti CTRL + STOP) e modificare la riga 10 in:

```
10 SCREEN 2,1
```

Ora eseguire di nuovo il programma. La dimensione degli sprites si è raddoppiata, come è facile constatare. Interrompere di nuovo il programma (premere i tasti CTRL+STOP) e modificare la riga 190 in:

```
190 PUT SPRITE 6,(I,50),1,3
```

Il volatile scomparirà dietro la nuvola.

SANDY

PRODOTTI PER HOME E PERSONAL COMPUTER

SINCLAIR ZX SPECTRUM & ACCESSORI

QL	L. 1.150.000
SPECTRUM 48K:	L. 395.000
INTERFACE 1: inter RS232 indispensabile per il collegamento del microdrive.	L. 165.000
MICRODRIVE: drive per micro cartucce originale Sinclair.	L. 155.000
SUPERFACE: sint. vocale + gen. di suoni ampl. sonoro + interfaccia joystick e registratore.	L. 145.000
TAVOLETTA GRAFICA: consente di costruire immagini grafiche in alta risoluzione.	L. 165.000
TASTIERA: con pad. numerico può alloggiare alim. ed eventuali interfacce.	L. 140.000
MODEM: rivoluzionario strumento di comunicazione tramite linea telefonica.	L. 155.000

EPROM PROGRAMMER: può programmare 2716/ 2732/ 2764/ 27128 completo di software.	L. 270.000
INTERF. RS232: adatta per collegare stampanti modem, plotter ect...	L. 90.000
INTERF. CENTRONICS: adatta per collegare qualsiasi stampante professionale.	L. 120.000
INTERF. JOYSTICK: programm. senza ausilio di software ne hardware.	L. 69.000
JOYSTICK:	L. 23.000
ESPANSIONI 48K:	L. 75.000

Per tutto il materiale non elencato (monitor, stampanti, software... ect) richiedere il catalogo.

IVA 18% ESCLUSA

VENDITA PER CORRISPONDENZA PRESSO:

SANDY

PERSONAL COMPUTER PRODUCTS S.R.L.
Via Monterosa 22 Senago (MI) tel. 02-9989407

VENDITA DIRETTA PRESSO:

SANDY COMPUTER CENTER
VIA ORNATO 14 – TEL. 02-6473621
MILANO

NOVITÀ!!!
FLOPPY DISK DRIVE PER SPECTRUM

CARATTERISTICHE PRINCIPALI

- Versione da 3" e 5" da 100 a 800 kbytes
- Sistema operativo in rom non utilizza spazio in ram
- Possibilità di collegare fino a quattro drive con una interfaccia (3,2 megabytes)
- Facile conversione di programmi. Modello da 100 kbytes **L. 610.000**

BELLUNO – COL COMPUTERS P.zza S. Stefano, 1 tel. 0437-212204
NAPOLI – LAMPITELLI Vico Acitlio, 71 tel. 081-657365
NOVARA – SYELCO Via S.F. d'Assisi, 20 tel. 0321-27786
TRIESTE – C.G.S. GASPARINI Via Paolo Reti, 6 tel. 040-61602

HARDWARE

MSX PHILIPS COMPUTER

32K ROM E 16K RAM CON POSSIBILITÀ DI ESPANDERE LA MEMORIA FINO A 128K.

a cura della Redazione

La prima macchina disponibile in redazione: uno splendido MSX Philips immediatamente funzionante. La sorpresa di una nuova tastiera e di nuove immagini soft made in Japan dalla notissima Konami (vedi giochi vari...).

Una prova? No, solo un primo sguardo nell'ancora misterioso mondo degli MSX che peraltro si annuncia foriero di novità, comunque affascinante.

modo normale

LE CARATTERISTICHE

● Linguaggio: MSX ● Processore: Z80A (3,6 MHz) ● Memoria: 32K (di cui 16 riservati al video) espandibile a 128K ● ROM: 16K ● Processore video: TMS9929 ● Processore audio: AY-3-8910 ● Video: 24 linee/40 caratteri ● Risoluzione: 49.152 pixel (256 x 192) ● Colori: 16 ● Sprites: 32 (con un massimo di 4 su ciascuna riga) ● Generatore sonoro: 3 toni su 8 ottave ● Uscita video: monitor o TV (canale 32) ● Tastiera: professionale a 72 tasti ● Interfaccia cassette: 1200 o 2400 baud con controllo remote ● Prese: 2 slot per cartuccia, prese joystick, registratore, monitor ● Prezzo al pubblico: 600 mila lire circa.

LO STANDARD MSX

Nato nel 1982 per iniziativa della Nec/Matsushita e della Ascii/Microsoft, lo standard denominato MSX (da MicroSoft Extended Basic) dovrebbe finalmente portare un po' d'ordine nel caotico mondo degli home e personal computer dove tutte le case hanno da sempre operato in maniera completamente indipendente le une dalle altre col risultato di avere delle macchine che non presentano la ben che minima compatibilità tra loro. Il successo del sistema MSX, per tutti questi motivi, è fuori di dubbio; un'altra conferma viene dal numero di Case che hanno deciso di adottare il sistema: al momento in cui scriviamo sono oltre 30 le aziende che hanno aderito.

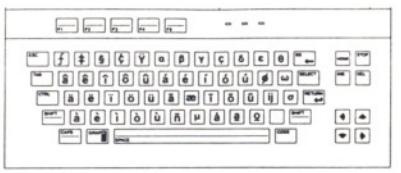
Sembra addirittura che anche la Sinclair stia per adottare questo standard per le sue future macchine. I computer che utilizzano questo standard sono compatibili

La macchina presenta una linea sobria ed ha un aspetto quasi professionale dovuto in gran parte alla tastiera i cui tasti a corsa ridotta presentano una spaziatura identica a quella delle tastiere professionali (19 mm).

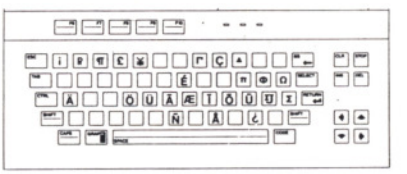
Sulla consolle, in alto a destra, sono previsti due slot per le espansioni e le cartucce con i programmi; sul retro troviamo, da sinistra a destra, le prese per l'alimentatore (esterno), per il registratore, per i joystick (due), per il monitor e per il TV (canale 32). La tastiera comprende 72 tasti di cui 5 di funzione ai quali possono essere assegnate 10 funzioni basic; sono previsti (vedi disegni) sei differenti modi di funzionamento selezionabili tramite le funzioni SHIFT, GRAPH e CODE. Tre led situati sulla destra dei tasti funzione indicano il modo di funzionamento della tastiera. Nell'apposito riquadro sono riportate le principali caratteristiche hardware della macchina che, come il software, sono compatibili con lo standard MSX.

Fare parte della grande famiglia MSX consente al VG8000 di utilizzare un numero incredibilmente vasto di programmi. Il Basic MSX, e quindi anche quello del VG8000, è basato sulla versione da 16K dell'MBASIC che comprende 84 istru-

modo codice

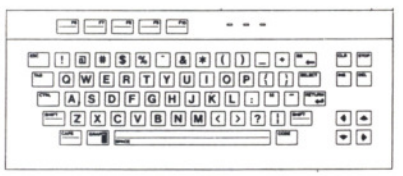


modo codice + shift

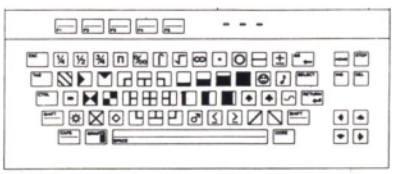


Il sistema MSX prevede sei modi di funzionamento della tastiera, nel caso del VG8000 i tasti sono di tipo professionale a corsa ridotta.

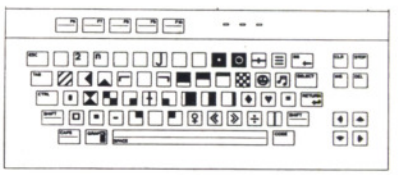
modo shift

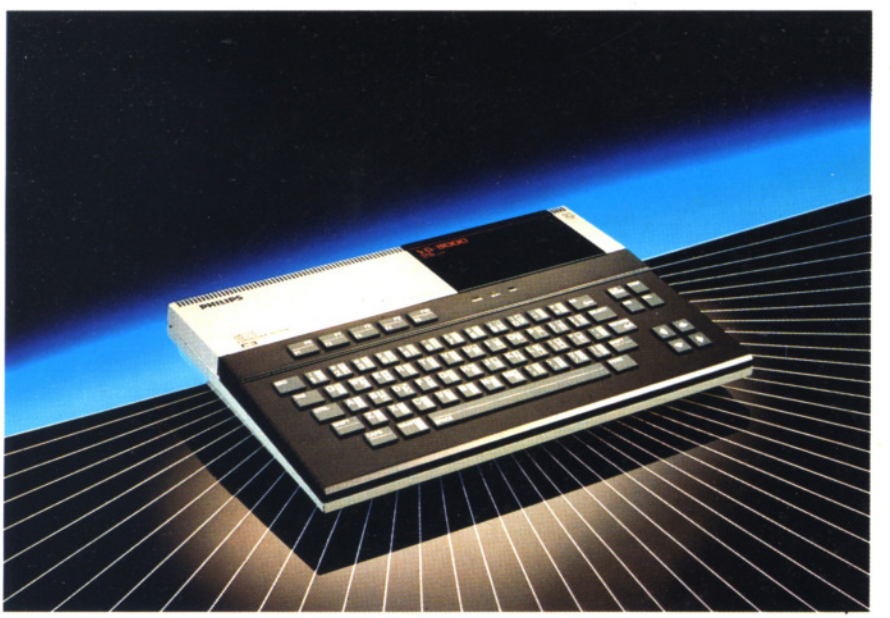


modo grafico



modo grafico + shift





sia dal punto di vista hardware che da quello software; ovviamente ci sono alcune differenze tra computer MSX per l'Europa, l'America e il Giappone, dovute principalmente alla frequenza di rete, al sistema di modulazione TV ed al set di caratteri. L'hardware MSX è basato su un'architettura di sistema aperta e flessibile che usa i seguenti componenti chiave: processore Z80, processore video TI, processore suono GI. Il sistema minimo MSX ha le seguenti caratteristiche: 24K RAM (di cui 16 sotto il controllo del processore video); 32K ROM contenente il sistema operativo e l'interprete basic, tastiera alfanumerica, interfacce video, cassette, joystick ed un connettore per cartuccia. Grazie alla natura «aperta» del concetto MSX questo sistema minimo può essere ampliato a piacere. Il software del sistema MSX è contenuto in una ROM da 32K e si compone del BIOS (Basic Input/Output System, circa 8K) e dell'interprete MSX-Basic (circa 24K, comprendenti 16K per il MicroSoft Extended Basic, più funzioni extra e grafici). Ciascun computer MSX usa pertanto lo stesso software di sistema che può essere ampliato esattamente come l'hardware. Attualmente tutte le macchine che adottano lo standard MSX utilizzano componentistica standard; tuttavia, entro il 1985, è prevista l'adozione di un integrato VLSI appositamente realizzato che svolgerà le funzioni dei tre processori attualmente utilizzati.

Cercheremo nei limiti del possibile di annotare su queste pagine ogni nuova notizia sul sistema MSX: a presto risentirci.

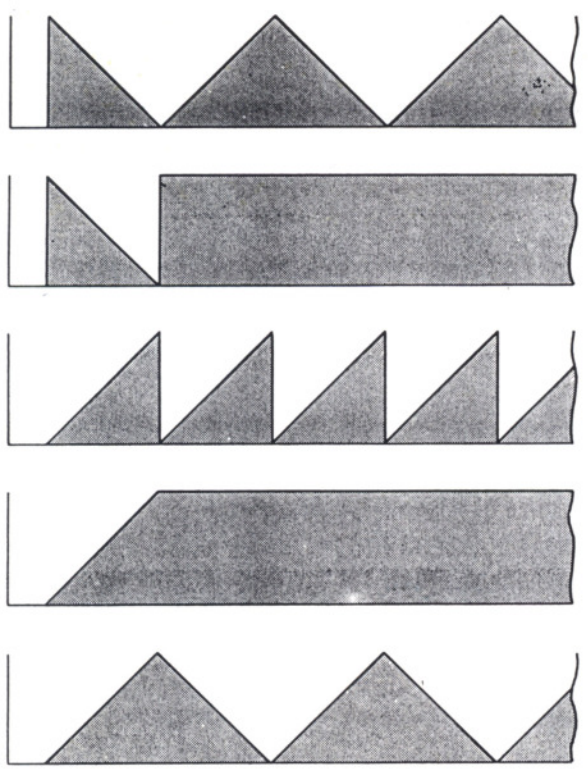
Nei prossimi fascicoli si vedrà anche di collaudare le vere possibilità del microprocessore.

LE ISTRUZIONI DEL BASIC MSX

Comandi:
AUTO, CONT, DELETE, LIST/LLIST, NEW, RENUM, RUN, SAVE/-LOAD/MERGE, BLOAD/BSAVE, CLOAD/CSAVE, TRON/TROFF
Istruzioni standard:
BASE, CALL, CLEAR, DATA, DIM, DEFINIT/SNG/DBL/STR, DEFFN, DEFUSR, END, ERROR, FOR-NEXT, GOSUB-RETURN, GOTO, IF-THEN/IF-GOTO, INPUT, KEY/KEY LIST, LINE/INPUT, LET, MAX-FILES, ON ERROR GOTO, ON GOTO/GOSUB, -ON/OFF/STOP, ON/GOSUB, OPEN/CLOSE, OUT, POKE, PRINT/LPRINT, PRINT/L-PRINT USING, PRINT # /INPUT #, READ, REM, RESTORE, RESUME, STOP, TIME
Grafica e suono:
BEEP, CIRCLE, CLS, COLOR, DRAW, LINE, LOCATE, PUT SPRITE, PAINT, PLAY, PSET/PRESET, SCREEN, SPRITES, VPOKE, SOUND, WIDTH
Funzioni:
BIS $, CDBL, CINT, CSNG, CSRLIN, EOF, PAD, PDL, PLAY, POINT, STICK, STRIG, TIME, VPEEK
Funzioni speciali:
ABS, ASC, ATN, CHR $, COS, EXP, ERR/ERL, FRE(0)/FRE(" "), INKEY $, INP, INSTR, INT, LEFT $, LEN, LOG, LPOS, MID $, PEEK, POS, RIGHT $, RND, SGN, SIN, SPACE $, SQR, STR $, STRING $, TAB, TAN, USR, VAL, VARPTR

FORMA D'ONDA DEI TONI

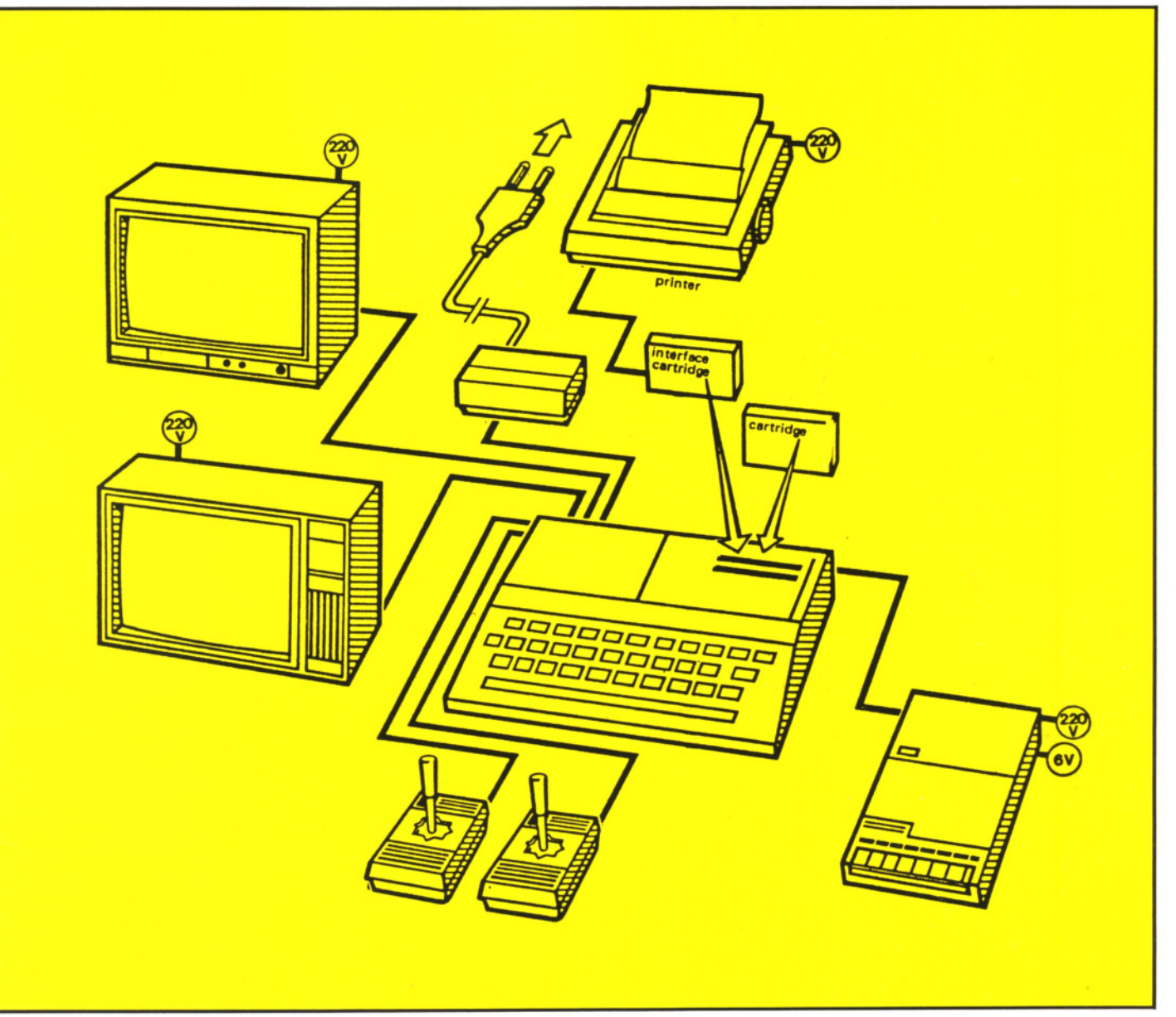
Il Basic MSX offre la possibilità di determinare la forma d'onda del tono che viene eseguito, utilizzando il subcomando "Sn"; dove n può assumere un valore compreso tra 0 e 15. Qui di seguito le forme d'onda ottenute con valori di n compresi tra 10 e 14.



LE POSSIBILI CONFIGURAZIONI

Le possibili configurazioni del sistema MSX VG8000 della Philips sono numerosissime. La configurazione standard prevede la consolle, l'alimentatore esterno, i due joystick, il registratore a cassette, la stampante e il monitor o il TV. I componenti di questo sistema sono già disponibili. Per la stampante esistono due possibilità: o il modello a 40 colonne (del tutto simile alla Seikosha GP-50) oppure il modello a 80 colonne con possibilità di foglio singolo. L'unità a dischetti sarà disponibile nel 1985 e utilizzerà dischetti da 3 1/2 pollici. Per quanto riguarda l'uscita RS232 questa, nel sistema MSX Philips, non è standard. Tuttavia, anche questa interfaccia, nel giro di pochi mesi sarà disponibile. Per quanto riguarda i joystick sono attualmente disponibili due modelli. Concludiamo con il registratore a cassette. Questo, al contrario di quanto accade con alcuni computer (leggi Commodore), non è dedicato; in altre parole, nonostante la Philips proponga diversi modelli, è possibile utilizzare un qualsiasi registratore a cassette.

zioni standard tipo RENUM, AUTO, DELETE ecc.; l'MSX basic comprende pertanto tutte queste istruzioni ma è stato ulteriormente esteso. Le estensioni riguardano le seguenti istruzioni: PLAY e SOUND per quanto riguarda la sezione musicale, CIRCLE, LINE, DRAW, PAINT per quanto riguarda le istruzioni grafiche, SCREEN e COLOR per quanto riguarda la gestione video, LOCATE, PRESET e PSET per quanto riguarda il controllo del cursore. Il VG8000 consente di visualizzare simultaneamente sullo schermo sino a 32 sprites ciascuno con il proprio colore e la propria priorità. Tra le altre particolarità segnaliamo la funzione cro-



nometro e l'editing per la correzione dei programmi.

Complessivamente (vedi elenco) l'MSX basic comprende 130 tra comandi, istruzioni e funzioni. Il sistema operativo DOS (l'MSX-DOS) è contenuto in una cartuccia per un totale di 8K. L'MSX-DOS è in gran parte compatibile con il CP/M80 e pertanto un computer MSX con MSX-DOS può eseguire virtualmente tutti i programmi CP/M80 quali DBASE 2, WordStar ecc. La formattazione dei dischi con MSX-DOS è identica a quella con MS-DOS; un computer MSX può pertanto leggere dati da un disco che è stato formattato con MS-DOS. Per contro un computer MSX non può eseguire programmi MS-DOS, ad esempio VisiCalc. Oltre al software di sistema su cartuccia, sono già disponibili alcuni programmi di utilità quali il FORMAT e l'M80 ASSEMBLER indispensabile quest'ultimo per sviluppare programmi in assembler. In conclusione ci sembra che la prima macchina MSX disponibile sul mercato non abbia deluso le attese di quanti, già da parecchio tempo, auspicavano uno standard universale.

L'ARCHIVIO ELETTRONICO

NON È INDISPENSABILE AVERE UNA UNITÀ DRIVE PER LA GESTIONE DI UN ARCHIVIO: È SUFFICIENTE DISPORRE DI UN REGISTRATORE A CASSETTE!

Avere un computer ed essere felici di averlo: perché può essere utilizzato in mille modi. Vediamo questa volta un programmino utile per archiviare un'agenda telefonica. Ringraziamo per la gentile collaborazione la Philips.

Parleremo ovviamente in linguaggio basic che, ben usato, è abbastanza potente come fra poco si vedrà. Vediamo come i dati possono essere registrati, salvati, e poi riletti nella memoria del computer.

Un programma può essere trasferito ad una cassetta attraverso il comando «CSAVE». La procedura è la seguente:

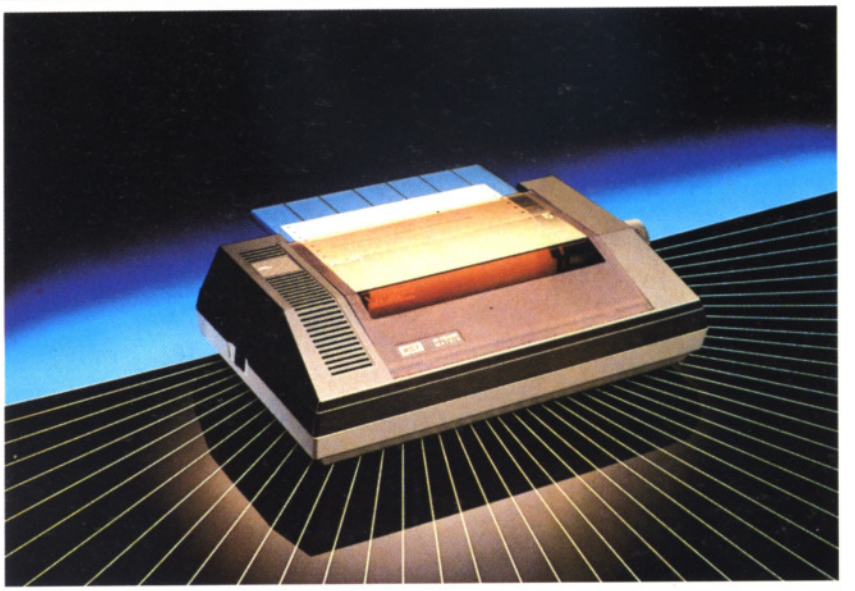
1. Inserire una cassetta vuota nel registratore e riavvolgerla completamente.
2. Far avanzare il nastro contenuto nella cassetta fino a che non si è superata la striscia di guida.
3. Premere simultaneamente i tasti PLAY e REC del registratore.
4. Battere il comando «CSAVE» seguito immediatamente dal nome del programma che si sta per immettere nella cassetta. Inserire il nome tra virgolette (CSAVE «nome del programma»).
5. Premere il tasto RETURN.

Il programma sarà trasferito alla cassetta.

Lettura

Un programma può essere letto nella memoria del computer dalla cassetta attraverso il comando «CLOAD».

Inserire la cassetta contenente il programma nel registratore e riavvolgerla completamente.



Qui sopra la stampante a 80 colonne VW0020 della Philips.

PRINTER INTERFACCIA E STAMPANTI

L'interfaccia parallela VU0040 (Standard Centronics), consente di collegare entrambe le stampanti del sistema VG-8000. La interfaccia si inserisce in una delle slot per cartucce, mentre il cavo della stampante si inserisce direttamente nell'interfaccia.
La VW0010 è una stampante a 40 colonne del sistema VG-8000. Il metodo di stampa è di tipo ad impatto a matrice di punti. Oltre a stampare tutti i caratteri disponibili sui computer MSX, consente anche la stampa di grafici. La sua velocità di stampa è di 35 caratteri al secondo. Inoltre è possibile programmare, via software, la spaziatura dell'interlinea.
La VW0020 è una stampante, come la VW0010, ad impatto a matrice di punti. Anche essa consente la stampa di tutti i caratteri del set MSX disposti fino ad un massimo di 80 per linea. La sua velocità di stampa è di 37 caratteri al secondo.

Un altro modello di stampante: VW0010 Philips a 40 colonne.

Premere il tasto PLAY del registratore.
Battere il comando: CLOAD «nome del programma».
Premere il tasto RETURN.
Se non è stato attribuito alcun nome al programma, unitamente all'istruzione «CLOAD», nella memoria del computer verrà letto il primo programma registrato sul nastro.
Quando MSX-BASIC trova un nome di programma diverso da quello chiesto, sullo schermo compare il testo che segue:
SKIP (PROGRAM NAME)
(Salta (nome programma)
Quindi MSX-BASIC continua la ricerca del programma giusto. Se il programma viene trovato, sullo schermo compare il testo che segue ed il programma viene letto nella memoria del computer.
FOUND (PROGRAM NAME)
(Trovato (nome programma))
Al termine del lavoro, MSX-BASIC ritorna a livello comandi.

Registrazione di dati

Le informazioni raccolte usando un programma possono a loro volta essere conservate su una cassetta.
Le informazioni sono registrate in sequenza. Questo insieme di informazioni è detto «file». Ad ogni file viene attribuito un nome ed il file deve essere «aperto» per potervici lavorare. Il numero dei file all'interno di un programma deve essere specificato con l'istruzione «MAXFILES». Ecco un esempio:

```
5 MAXFILES=1
10 OPEN "CAS:ADDR" FOR OUTPUT AS#1
20 A=15:B=23.5:C=10:D=25.2:E=13
30 PRINT #1,A;B;C
40 PRINT #1,D;E
50 CLOSE #1
```

Alla riga 10, viene aperto il file denominato «ADDR» (si sta qui facendo riferimento ad un file di indirizzi, ma può funzionare con qualsiasi nome, naturalmente in relazione al tipo di file).

La parola «CAS» indica che il file è registrato su una cassetta.

Le informazioni sono effettivamente registrate sulla cassetta con l'istruzione «PRINT #».

Dopo ogni istruzione «PRINT #» sul nastro viene inserito un codice «CR» e «LF». Tra le variabili numeriche viene automaticamente inserita una virgola. L'esempio suddetto avrebbe la seguente «immagine» sul nastro:

Battendo una variabile alfanumerica, la virgola deve essere aggiunta manualmente con l'istruzione «PRINT #» come nell'esempio che segue:

```
5 MAXFILES =1
10 OPEN "CAS:NAME" FOR OUTPUT AS#1
20 A$="JOHN":B$="PETE"
30 PRINT #1,A$","B$
40 CLOSE #1
```

Questa produrrà sul nastro il seguente risultato:

Notare che le fasi procedurali da 1 a 3, come spiegato a proposito della registrazione di un programma, devono essere eseguite prima di trasferire i dati sulla cassetta!

Lettura di informazioni

I dati che sono stati registrati su una cassetta possono essere riletti nella memoria del computer attraverso l'istruzione «INPUT #».

Per leggere le variabili numeriche dell'esempio precedente occorre usare la seguente istruzione:

```
5 MAXFILES=1
10 OPEN "CAS:ADDR" FOR INPUT AS#1
20 INPUT #1;V,W,X
30 INPUT #1;Y,Z
40 CLOSE #1
```

Per le variabili alfanumeriche occorrono ad esempio le seguenti istruzioni:

```
5 MAXFILES=1
10 OPEN "CAS:NAME" FOR INPUT AS#1
20 INPUT #1,Y$,Z$
30 CLOSE #1
```

Un'istruzione «INPUT #» riempie le variabili con le informazioni fino alla prima virgola o fino al codice «CR» e «LF».

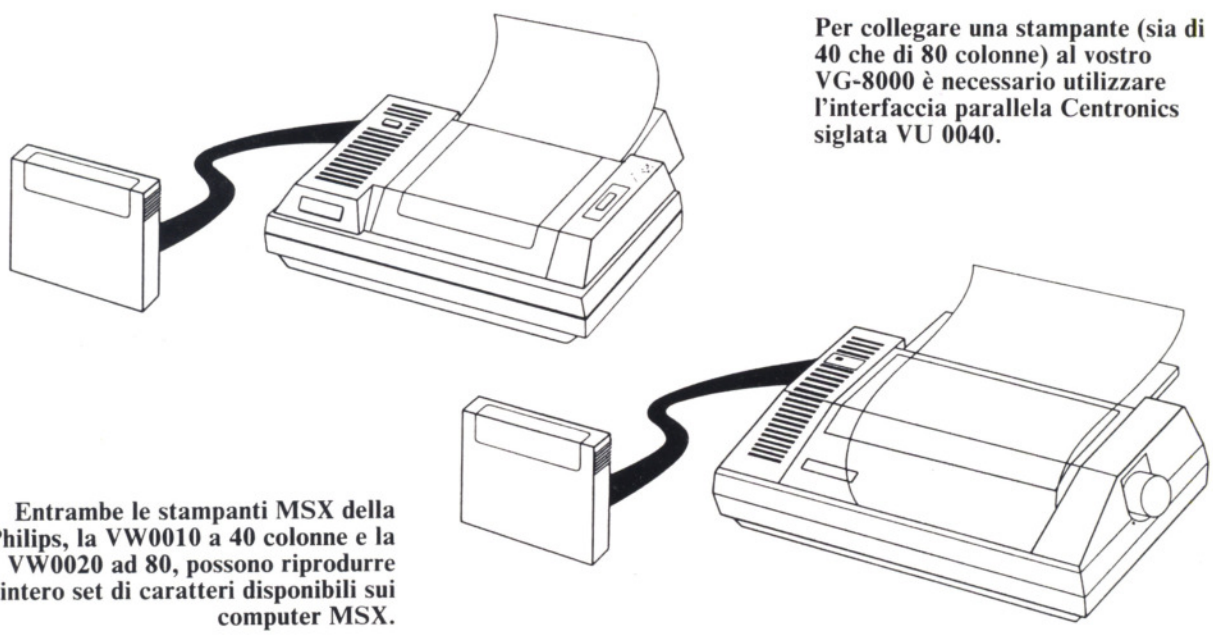
Un'istruzione «LINE INPUT #» riempie le variabili indicate con le informazioni fino al successivo codice «CR» e «LF», come in questo esempio:

```
NEW
5 MAXFILES=1
10 OPEN "CAS:NAME" FOR INPUT AS#1
20 LINE INPUT #1,X$
30 CLOSE #1
```

In questo esempio «X$» contiene «JOHN, PETE».

Con la funzione «EOF» si può verificare se si è raggiunta la fine di un file.

È necessario attenersi rigorosamente alle fasi procedurali da 1 a 2, indicate nelle istruzioni per la lettura di



Per collegare una stampante (sia di 40 che di 80 colonne) al vostro VG-8000 è necessario utilizzare l'interfaccia parallela Centronics siglata VU 0040.

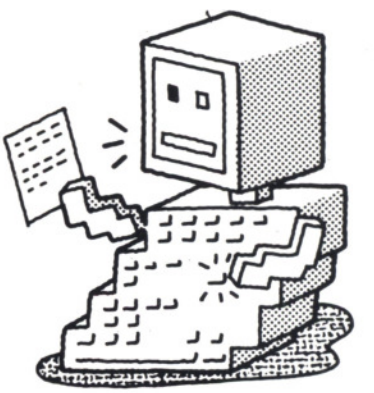
Entrambe le stampanti MSX della Philips, la VW0010 a 40 colonne e la VW0020 ad 80, possono riprodurre l'intero set di caratteri disponibili sui computer MSX.

una cassetta prima di leggere le informazioni nella memoria del computer.

Un file di indirizzi

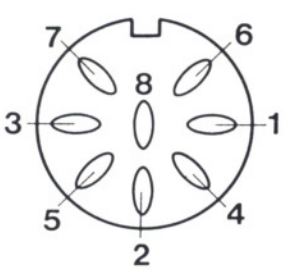
Segue ora un programma che è possibile usare per inserire nomi, indirizzi, ecc. Le informazioni vengono trasferite sulla cassetta e possono essere lette in qualsiasi momento.

```
NEW
10 SCREEN 0:KEY OFF:WIDTH 39:MAXFILES=1
20 CLS:LOCATE 10,2:PRINT "MENU"
30 ON KEY GOSUB 100,400,700
40 LOCATE 5,5:PRINT "F1=IMMETTI INDIRIZZO"
50 LOCATE 5,7:PRINT "F2=SCRIVI INDIRIZZO"
60 LOCATE 5,9:PRINT "F3=FINE PROGRAMMA"
70 KEY (1) ON:KEY (2) ON:KEY (3) ON
80 GOTO 80
100 CLS
110 PRINT:PRINT "PREDISPONI REGISTRATORE
E PREMI SIMULTANEAMENTE PLAY E RECORD"
120 PRINT:PRINT:INPUT "PRONTO (S/N)";F$
130 IF F$="N" GOTO 120
140 IF F$="n" GOTO 120
150 OPEN "CAS:ADDRESS" FOR OUTPUT AS#1
160 CLS
170 LOCATE 1,5:INPUT "NOME";A$
180 LOCATE 1,7:INPUT "INDIRIZZO";B$
190 LOCATE 1,9:INPUT "CITTA'";C$
200 LOCATE 1,11:INPUT "TELEFONO";D$
210 LOCATE 1,20:INPUT "ESATTO (S/N)";F$
220 IF F$="N" GOTO 160
230 IF F$="n" GOTO 160
240 PRINT #1,A$;",";B$;",";C$;",";D$
250 LOCATE 1,22:INPUT "ALTRO INDIRIZZO (S/N)";F$
260 IF F$="N" GOTO 290
270 IF F$="n" GOTO 290
280 GOTO 160
290 CLOSE #1:RETURN 20
400 CLS
410 PRINT:PRINT "PREDISPONI REGISTRATORE E PREMI PLAY
KEY"
420 PRINT:PRINT:INPUT "PRONTO (S/N)";F$
430 IF F$="N" GOTO 420
440 IF F$="n" GOTO 420
450 OPEN "CAS:ADDRESS" FOR INPUT AS#1
460 CLS:PRINT
470 IF EOF(1) THEN 570
480 INPUT# 1,A$,B$,C$,D$
490 PRINT "NOME      :";A$:PRINT
500 PRINT "INDIRIZZO:";B$:PRINT
510 PRINT "CITTA'    :";C$:PRINT
520 PRINT "TELEFONO  :";D$
530 LOCATE 1,22:INPUT "ALTRO INDIRIZZO (S/N)";F$
540 IF F$="N" GOTO 570
550 IF F$="n" GOTO 570
560 GOTO 460
570 CLOSE #1:RETURN 20
700 CLS:PRINT "FINE PROGRAMMA"
710 END
```



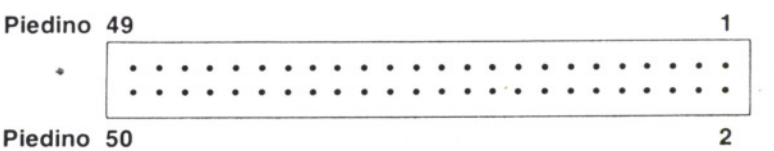
Eseguire il programma e provare ad immettere alcuni indirizzi. Questi verranno registrati sul nastro quando si preme il tasto F1 e saranno riletti nella memoria del computer quando si preme il tasto F2. Non dimenticarsi di riavvolgere la cassetta prima di accingersi a leggere gli indirizzi.

Se il programma non funziona correttamente, controllare che non ci siano errori di battitura.



Connettore per registratore dati

Piedino	Nome	I/O
1	GND	
2	GND	
3	GND	
4	CMTOUT	O
5	CMTIN	I
6	REM+	O
7	REM—	O
8	GND	



Connettori per cartuccia

Piedino	Nome	I/O	Piedino	Nome	I/O	Piedino	Nome	I/O
1	CS1	O	2	CS2	O	3	CS12	O
4	SLTSL	O	5	Riservato		6	RFSH	O
7	WAIT	I	8	INT	I	9	M1	O
10	BusDIR	I	11	IORQ	O	12	MERQ	O
13	WR	O	14	RD	O	15	RESET	O
16	Riservato		17	A9	O	18	A15	O
19	A11	O	20	A10	O	21	A7	O
22	A6	O	23	A12	O	24	A8	O
25	A14	O	26	A13	O	27	A14	O
28	AO	O	29	A3	O	30	A2	O
31	A5	O	32	A4	O	33	D1	I/O
34	D0	I/O	35	D3	I/O	36	D2	I/O
37	D5	I/O	38	D4	I/O	39	D7	I/O
40	D6	I/O	41	GND		42	CLOCK	O
43	GND		44	SW1		45	+ 5V	
46	SW2		47	+ 5V		48	+ 12V	
49	SOUNDIN	I	50	—12V				

MSX BASIC

INTRODUZIONE AL LINGUAGGIO DEI MICROCOMPUTER MSX STANDARD

di E. DASSI

In ogni numero sarà presente questa rubrica che più che un corso vuole essere un'attenta analisi del BASIC MSX, al fine di poterlo sfruttare al meglio nei nostri programmi.

In questa prima puntata parleremo delle costanti, delle variabili, degli array e quindi delle istruzioni di assegnazione. Per coloro che sono all'inizio con la programmazione, diciamo che un programma normalmente deve elaborare dei dati. Questi ultimi possono presentarsi, nella stesura del listato, o come costanti o come variabili.

COSTANTI E VARIABILI

La costante è un dato, numerico o alfanumerico, che non varia durante l'esecuzione del programma. La variabile, invece, individua una zona di memoria, con un determinato nome, soggetta a contenere informazioni, o per meglio dire dati, variabili durante l'esecuzione del programma.
Facciamo un esempio:

```
10 A=10
20 PRINT 10
30 PRINT A
```

La linea 20 stampa il numero 10, che è rappresentato come una costante, mentre la linea 30 stampa ancora 10 rappresentato, però, da una variabile identificata con il nome A.

Nel BASIC MSX una costante alfanumerica viene descritta da una serie di caratteri racchiusi tra le virgolette ("..."). Questa serie non deve superare i 255 caratteri.

Per quanto riguarda le costanti numeriche, vi è da dire che possono essere positive o negative e di differenti tipi, precisamente 6: intera, in virgola fissa, in virgola mobile, esadecimale, ottale e binaria.

Una costante intera è un numero intero compreso tra −32768 e 32767, per esempio −9213, 4, −2, 1000 ecc. Alcune funzioni, per esempio PEEK, prevedono costanti intere senza segno. In questo caso il loro range può andare da 0 a 65535.

Una costante in virgola fissa è un numero reale positivo o negativo con cifre dopo il punto decimale, come per esempio 18.33 oppure −22.144.

Una costante in virgola mobile, invece, prevede un numero, positivo o

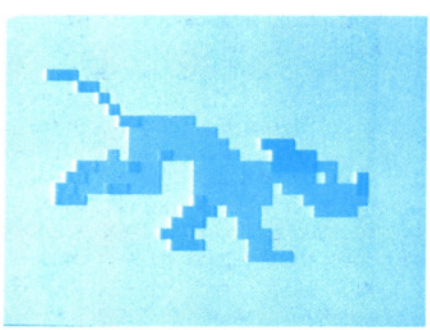
negativo, simile alla notazione scientifica. Essa consiste di un eventuale numero intero o in virgola fissa (mantissa), seguito dalla lettera D o E e da un numero intero (esponente). Esempio: 235.1E−7 (= .00002351) oppure 2994E6 (=2994000000). Una costante esadecimale è un numero esadecimale (base 16) compreso nel range degli interi e preceduto dal prefisso "&H" oppure "&h". Esempio: &H3F oppure &h3F50.

Una costante ottale è un numero in base 8 compreso, anch'esso, nel range degli interi e preceduto da "&0" oppure semplicemente da "&" come per esempio &31.

Una costante binaria è un numero in base due nel range degli interi e preceduto dal prefisso "&B"; esempio &B10010000 oppure &b10001111.

Quando abbiamo parlato di virgola mobile abbiamo accennato alla singola e alla doppia precisione. Una costante, o una variabile numerica, nel BASIC MSX può essere rappresentata fino a 6 cifre per la singola precisione e fino a 14 cifre per la doppia precisione. Per default le costanti sono in doppia precisione. Una costante in virgola fissa e a precisione singola è contraddistinta da un punto esclamativo (22.5!), mentre a virgola mobile dalla "E" (−1.09E−06).

Una costante in virgola fissa, ma a doppia precisione, è contraddistinta dal simbolo "#" (3490.0#), mentre a virgola mobile dalla "D" (−1.09433D−06). Anche un numero come 3450 è considerato per default in doppia precisione in quanto nessuna specificazione è stata fatta.



Così come per le costanti, anche le variabili possono essere numeriche o alfanumeriche. Il loro nome è scelto dal programmatore e può avere lunghezza a piacere anche se vengono considerati solo i primi due caratteri. I caratteri "$", "%", "!" e "#" posti alla fine del nome della variabile la definiscono, rispettivamente, alfanumerica, intera, a singola precisione e a doppia precisione.

È possibile determinare il tipo di variabile anche attraverso le istruzioni DEFINT, DEFSNG, DEFDBL e DEFSTR.

Queste istruzioni, seguite da una lettera o da una serie di lettere, definiscono tutte quelle variabili inizianti con la lettera specificata di un certo tipo (intere, singola precisione ecc.). Vediamo subito un esempio:

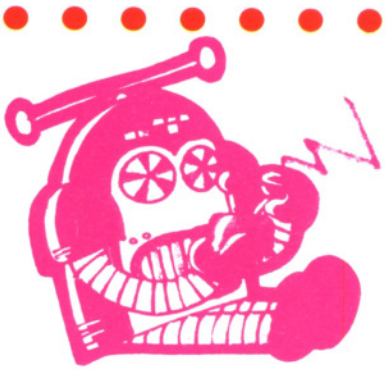
```
10 DEFINT A, L-N
```

```
20 DEFSNG B
30 DEFDBL C-E
40 DEFSTR G-J
50 H="MSX"
60 H%=2
```

L'istruzione 10 definisce tutte le variabili che iniziano con la lettera A, L, M ed N intere; la 20 definisce tutte le variabili che iniziano per B in singola precisione; la 30 definisce tutte le variabili che iniziano per C, D ed E come variabili in doppia precisione e la 40 definisce le variabili inizianti per G, H, I e J come alfanumeriche. La linea 50 assegna ad H il valore "MSX" e la linea 60 ridefinisce H come variabile intera, annullando il tipo predefinito per quel carattere.

A proposito delle variabili alfanumeriche va detto che il BASIC MSX consente di adoperare in totale fino a 200 caratteri; l'istruzione CLEAR consente di variare tale valore.

Se una costante numerica di un tipo è assegnata ad una variabile numerica di differente tipo, il numero memorizzato sarà del tipo dichiarato dalla variabile. Esempio:



```
10 B% = 23.3
20 PRINT B%
```

L'esecuzione delle due istruzioni produrrà la stampa del numero 23 in quanto è stato assegnato ad una variabile intera un numero reale. Le cifre dopo il punto decimale vengono perse. Consideriamo invece le seguenti istruzioni:

```
10 D = 6/7!
20 PRINT D
```

Il risultato della stampa sarà un numero in doppia precisione perché, pur essendo l'operando 7! in singola precisione, il risultato viene assegnato ad una variabile a doppia precisione. Conversione dei tipi di variabile può essere anche fatta tramite le funzioni CBDL, CINT, CSNG e STR$. Per queste ultime, il cui significato è già stato visto in parte negli esempi appena descritti, rimandiamo il lettore alla consultazione del proprio ma-

nuale MSX-BASIC.

Dopo tutto questo discorso sulle variabili e sulle costanti, concludiamo dicendo che una variabile intera occupa 2 byte, una a singola precisione 4 byte, una a doppia precisione 8 byte ed infine un'alfanumerica 3 byte più quelli necessari a contenere i caratteri della variabile.

Dopo queste ultime informazioni vogliamo darvi un consiglio: studiate attentamente i dati che dovrete elaborare e quindi scegliete il tipo di variabile più economico, cioè che occupi minor spazio in memoria.

ARRAY

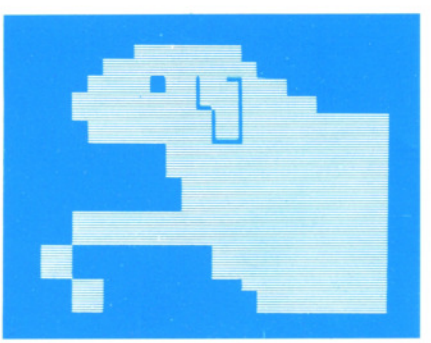
In alcuni problemi è logico attribuire ad una serie di dati una struttura simile a tante caselle in successione o a matrice. Questa struttura di dati è identificata sotto il nome di array. Gli array non sono che un insieme di più variabili individuate da un unico nome ad indice.

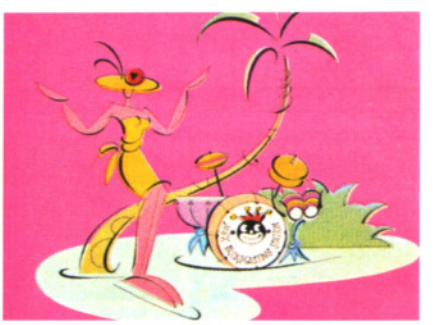
Supponiamo di voler creare un blocco dati contenente i nomi dei giorni della settimana:

```
10 DIM G$ (6)
20 DATA LUNEDÌ, MARTEDÌ,
   MERCOLEDÌ, GIOVEDÌ,
   VENERDÌ, SABATO,
   DOMENICA
30 RESTORE 20
40 FOR I=0 T0 6
50 READ G$ (I)
60 NEXT
```

La linea 10 dichiara, con l'istruzione DIM, una variabile alfanumerica G$ a 7 posizioni, essendo anche considerata la posizione 0. La linea 20 utilizza l'istruzione DATA che ha la funzione di dichiarare delle costanti, numeriche o alfanumeriche, all'interno di un programma. La linea 30 è stata volutamente messa per spiegare l'istruzione RESTORE. Infatti anche senza tale linea il programma funzionerebbe lo stesso. L'istruzione RESTORE consente di posizionare ad una determinata linea (nell'esempio la 20) il lettore delle costanti inserite nella istruzione DATA. Tale lettore è l'istruzione READ presente alla linea 50. Essendo l'istruzione READ inserita in un ciclo FOR-NEXT, essa leggerà tutti i valori della DATA e li porrà nelle differenti posizioni dell'array G$. Questo tipo di array monodimensione viene detto vettore.

L'MSX BASIC considera ogni sua variabile come vettore di indice 10, cioè a 11 posizioni. Quindi, rifacendoci all'esempio visto, avremmo potuto lo stesso eseguire il programma senza dover definire G$ alla linea 10, perché l'MSX BASIC lo considera già a 11 posizioni.





È possibile anche dichiarare array a più dimensioni (fino a 255). L'indice massimo consentito è però relativo alla capacità di memoria disponibile.

ISTRUZIONI DI ASSEGNAZIONE

Abbiamo visto con READ/DATA delle istruzioni di assegnazione. Esistono anche altre due istruzioni che svolgono la funzione di assegnazione ad una variabile; esse sono LET e INPUT.

Vediamo l'esempio:

```
10 LET A=10
20 A=10
30 INPUT A
```

Tutte e tre le linee svolgono la stessa funzione, cioè assegnano alla variabile A un determinato valore. La linea 20 e identica alla linea 10, infatti l'istruzione LET può essere omessa e la variabile assume lo stesso il valore posto a destra del segno uguale.

L'istruzione INPUT, così come INPUT$ o LINE INPUT (per le tre differenti istruzioni rimandiamo il lettore alla consultazione del manuale del proprio computer MSX), invece, ferma l'elaborazione e attende, da tastiera, il valore d'assegnare alla variabile in gioco. Nel nostro esempio si tratta della variabile numerica A.

Vogliamo concludere questa prima puntata presentando anche un'altra istruzione di assegnazione molto comoda: l'istruzione SWAP. Tale comando consente di scambiare il contenuto di due variabili, purché dello stesso tipo, senza doverne adoperare una terza. Consigliamo a tutti di farne un uso frequente in quanto fa risparmiare memoria, tempo di esecuzione e rende il listato più elegante.

Le variabili, così come le costanti, sono tutto per un programma; un loro utilizzo corretto rende più leggibile il programma ma anche lo fa più veloce, elemento, questo ultimo, di rilevante importanza. ■

by LOAD'N'RUN

Suppl. N. 13 - FEB./MARZO 1985 - L. 10.000
Sped. in abb. post. Gr. III/70

COM 64

RACCOLTA DI PROGRAMMI SU CASSETTA PER COMPUTER

11 GIOCHI & UTILITY 11 PER IL TUO COMMODORE 64

U.F.O.

ZAX LASER

TORNADO JET

FIND & NUMBER

BREAK LIST

CRUCIPUZZLE

LA MONGOLFIERA

COPIA DISCO

FUNAMBOLO

SPRITE DRAWER

MAGIC FLIPPER

in tutte le edicole!

GRATIS
A COLORI
LA COPERTINA
DELLA
CASSETTA

A CACCIA DELL'ERRORE

PERCHÉ UN PROGRAMMA NON FUNZIONA

di S. ROCCHI

Pur essendo il Basic un ottimo linguaggio di programmazione, sia per il suo facile uso che per la sua versatilità nelle più svariate applicazioni, i programmatori che ne fanno uso non possono sempre sfuggire ai «bugs», ovvero agli errori interni ad un programma.

In questo articolo vogliamo dare una serie di consigli di come fare a scovare i bugs. Certo non è l'unico procedimento, ma senz'altro sarà un'ottima lettura per tutti quei programmatori che si arrendono fin dall'inizio quando un loro programma non funziona.

Il primo, e il più comune metodo di trovare gli errori, è quello di controllare costantemente il listato che appare sul video. Sembrerà palese, ma gli errori più «evidenti» affioreranno velocemente. Spesso, però, questo metodo è inefficiente

perché il bug si nasconde anche ad una attenta osservazione del listato. Bisogna così passare ad un debugging strutturato secondo delle regole ben precise. Se non riuscite a trovare il bug in pochi minuti, non continuate a fissare e a meditare sul listato, perché esso rappresenta semplicemente un'informazione statistica, mentre invece un programma in esecuzione è lo sviluppo di una serie di informazioni dinamiche. Per cui il debugging dovrà essere un'attiva, piuttosto che passiva, esaminazione del programma.

Il salto corretto

Cosa bisogna allora esaminare? La risposta è semplice: ci sono solo due fattori che determinano il modo di funzionamento di un programma: i valori memorizzati nelle variabili e l'ordine dell'esecuzione delle istruzioni; sono quindi questi i responsabili dell'inesatta esecuzione del programma stesso ed è in essi che si nasconde il bug.

Un metodo semplice ed efficace è quello quindi di predisporre temporaneamente nel programma di particolari PRINT che stampino il valore delle variabili e la linea di esecuzione. L'arte di questa procedura è però il saper controllare la giusta quantità di informazioni e non di più. Infatti se un programma si presenta all'utente con una scelta da menù ma non esegue il corretto salto, relativo alla scelta, allora bisognerà controllare quella variabile utilizzata per gestire la scelta, e non altre; ma se il valore di quest'ultima fosse corretto, allora dovreste esaminare l'ordine di esecuzione delle istruzioni che gestiscono la sezione del menù, seguendo questi criteri: sono eseguite le istruzioni?; il programma esegue sempre le stesse operazioni?; il programma salta alla corretta opzione e poi continua da un'altra parte?, e così via.

Vi sono comunque alcune particolarità di un programma Basic che spesso sono responsabili di una scorretta esecuzione del programma, per esempio, i numeri di linea.

I casi difficili

Dovendo obbligatoriamente (tranne una o due versioni del Basic) assegnare ad ogni linea un proprio numero, è possibile sbagliare l'inserzione, in macchina, di anche una sola cifra, provocando così una rilocazione della linea in una parte di programma non assegnatale e quindi, un inevitabile bug. Questo errore può essere molto difficile da scovare, se la linea viene inserita in una parte di programma già provata e quindi al di sopra di ogni sospetto. Consigliamo perciò di introdurre piccoli blocchi di linee e poi testare che esse siano tutte presenti e corrette.

La rinumerazione di alcune parti del programma potrebbe provocare degli errori quando i valori numerici assegnati alle istruzioni GOTO e GOSUB non vengono aggiornati, provocando un risultato scorretto

per quanto riguarda l'esecuzione del programma.

Dove si nascondono

A questo proposito consigliamo sempre di controllare i parametri delle istruzioni GOTO e GOSUB, così come altre istruzioni, come per esempio la RESTORE (solo in alcune versioni Basic).

Un'istruzione GOTO o GOSUB, rispetto alla posizione in cui si trova, può eseguire un salto indietro o avanti nel programma. È proprio in quest'ultimo caso che si verifica spesso un bug. Se non si conosce esattamente la nuova destinazione dopo il salto, è facile commettere un errore nel trascrivere un valore.

Per ricordare

Per evitare ogni problema vi consigliamo di porre inizialmente accanto alle GOTO e alle GOSUB degli identificatori particolari, per esempio «*» o «?», che possono ricordarvi di dover porre al loro posto il numero di linea esatto.

Questo potrà essere fatto qualora

sia già stato scritto, o si preveda con certezza, il numero di linea a cui dovranno far riferimento le istruzioni sopra dette.

Anche i cicli FOR-NEXT, a seconda delle versioni Basic, possono nascondere «misteriosi» bug. Per esempio consideriamo il problema di dover scandire una stringa per trovare il primo carattere uguale a quello di un'altra stringa. Il problema viene spesso risolto così:

```
10 FOR I=1 TO LEN (S$)
20 IF MID$ (S$,I,1)=T$ THEN
   GOTO 40
30 NEXT I
40 continua il programma...
```

Questo è naturalmente soltanto un semplice esempio.

L'esecuzione delle istruzioni viste può provocare, in alcune versioni Basic, il crash del sistema perché non è permesso un numero illimitato di interruzione dei cicli FOR-NEXT.

Una soluzione a questo bug è quella di forzare la fine del ciclo. Questo avviene nel modo seguente:

```
20 IF MID$ (S$,I,1)=T$
   THEN L=I: I=LEN S$
```

In questa linea, L è usata per salvare la posizione del carattere da trovare e I viene posta al valore che causerà la fine del ciclo FOR-NEXT.

Decimo non omettere

Ma i bugs nascono anche quando si tralasciano istruzioni del tipo END, STOP o RETURN. Questo tipo di istruzioni permette di fermare il programma o ritornare da un sottoprogramma, evitando così la ripetizione di alcune elaborazioni già eseguite. Addirittura accade di omettere dei GOTO quando si vuole fare eseguire la lista 1 d'istruzioni, se A è negativo, e la lista 2 d'istruzioni

by Elettronica 2000

N. 13 - FEBBRAIO 1985 - L. 9.000
Sped. in abb. post. Gr. III/70

LOAD'N'RUN

RACCOLTA DI PROGRAMMI SU CASSETTA PER IL TUO COMPUTER

Spectrum DRIVE IN

- LO SBARCO
- ROTATE 3 D
- LA CITTÀ FANTASMA
- GALAXIAN
- BATTAGLIA NAVALE
- A TUTTO GAS
- LA SCOPA
- ENIGMISTICA
- CUBO MAGICO
- SOMMERGIBILE
- WORMY LABIRINTO
- BEETHOVEN

in tutte le edicole!

12 GIOCHI & UTILITY PIÙ UN MILIONE PER TE

GRATIS LA COPERTINA DELLA CASSETTA

se A è positivo. La corretta procedura di controllo sarà:

```
    10 IF A<0 THEN GOTO 1000
    :
    lista 2
    :
  990 GOTO 2000
1000  . .
      :
      lista 1
      :
2000 resto del programma...
```

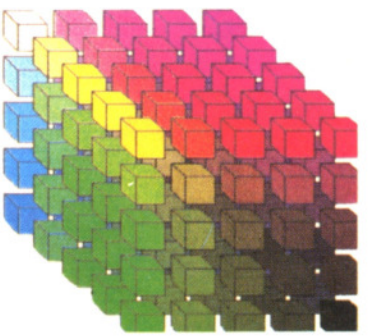
L'errore più comune è proprio quello di dimenticare l'istruzione 990 così che la lista 1 viene eseguita anche dopo l'esecuzione della lista 2.

Quindi attenzione alle GOTO e servitevi, per il controllo del listato, dei diagrammi di flusso.

Qualche esempio

Vogliamo concludere questa breve panoramica sui bugs più comuni, presentandovi la trattazione in Basic di particolari espressioni e condizioni logiche.

Perfino dopo tanti anni di programmazione ci si può ancora confondere tra lo scrivere le espressioni aritmetiche in Basic e normalmente.



Per esempio l'espressione:

$$\frac{1}{y\,(1+x)}$$

nel Basic non deve essere scritta come:

```
1/y*1+x
```

ma spesso è programmata come:

```
1/y*(1+x)
```

anche se la forma corretta è naturalmente:

```
1/y/(1+x)
```

Quest'ultima spesso è inutilizzata proprio perché l'uso della doppia barra mette in dubbio la sua validità.

In conclusione

Nella trattazioni delle condizioni logiche accade, per esempio, di testare il responso di una domanda che doveva essere «SI» o «NO». La forma corretta è:

```
IF A$ <> «SI» AND A$ <> «NO»
      THEN GOTO errore
```

Molti programmatori invece costantemente scrivono OR al posto di AND, provocando chiaramente un bug.

In altre parole la pratica del debugging è da fare molto seriamente e sempre, anche quando siete convinti che il vostro programma sia perfetto.

MSX, LE ISTRUZIONI

COME CONVERTIRE LE CODIFICAZIONI BASIC DEI COMPUTER PIÙ DIFFUSI

```
190 GX=GX+DX:GY=GY+DY:CO=CO+1:PUTSPRITE0,(GX,GY),10,COMOD2:C=C+1:
200 IFDY<>0THENIFLX<27ANDLX>4THEN1020
210 LX=LX+SGN(DX):LY=LY+SGN(DY):DY=0:C=0:IFLY=21THEN960
220 D=VPEEK(BASE(5)+LX+LY*32+65+(DX<0)):IFD<>219ANDD<>35THENDY=CY
230 IFD<>219ANDD<>35THENDY=CY
```

Ancora oggi molte aziende produttrici di computer non sono arrivate a sfornare macchine aventi tra di loro lo stesso linguaggio. Finalmente però sono arrivati i primi MSX: noi perciò abbiamo pensato di realizzare una comoda tabella (vedi pagine successive) che raggruppi le parole e i formati dei Basic più conosciuti. La presentazione dei differenti Basic è in riferimento al Basic MICRO-SOFT.

Nella tabella (in questo fascicolo le prime istruzioni) non sono stati inseriti i comandi di suono e grafica, perché sarebbe stato troppo complicato per un elenco così rapido; inoltre, la tabella è preceduta da una serie di istruzioni comuni nella sintassi ai differenti computers presentati.

In pratica

Ed ora vediamo come utilizzarla. Per prima cosa bisogna vedere se il computer che noi vogliamo è presente tra i nomi indicati in cima alle colonne. Se sì, allora, seguendo la colonna scelta si possono leggere le istruzioni del suo Basic e le informazioni indicate nel riquadro. Va precisato che non sempre, per motivi di spazio, queste informazioni illustrano l'intera possibilità operativa dell'istruzione, ma semplicemente la sua funzione principale.

In secondo luogo, se volessimo trovare l'equivalente istruzione su un altro computer, dovremmo spostarci orizzontalmente, dalla posi-

zione precedentemente trovata, finché non puntiamo alla colonna del computer di cui ci serve l'equivalente istruzione. Facciamo un esempio per chiarire quanto detto.

Subito un esempio

Supponiamo di voler tradurre un programma scritto in linguaggio Basic Sinclair ZX Spectrum per il Commodore 64 o VIC 20. Trovando nel programma l'istruzione CODE a$, e volendola tradurre per il Commodore, dovremo trovare la colonna dello ZX Spectrum: successivamente trovare la casella con indicata l'istruzione CODE e, spostandoci orizzontalmente, raggiungere la colonna riservata al Commodore 64 e VIC 20.

A questo punto abbiamo puntato ad una casella con indicato ASC (str). Quindi, l'istruzione CODE a$ diventa per il Commodore ASC a$.

In giallo abbiamo naturalmente evidenziato la colonna che ci interessa di più, cioè quella degli MSX.

La casella vuota

Se nelle nostre conversioni dovessimo trovare delle caselle vuote, ciò vuol dire che per quel computer non è stata implementata quel tipo di istruzione.

Sicuri dell'utilità di questa tabella, vi auguriamo le migliori conversioni e, nel qual caso queste dovessero essere particolarmente interessanti, vi invitiamo ad inviarcele.

LE ISTRUZIONI COMUNI

ABS (exp)
COS (exp)
END (N.B. non disponibile sui Sinclair)
FOR var=exp TO exp STEP exp
LEN (str)
LET var=exp
REM testo
SIN (exp)
SQR (exp)
STOP
TAN (exp)
VAL (exp) N.B. non disponibile su QL

Abbreviazioni usate:
addr = indirizzo; cost = costante; dis = dispositivo; exp = espressione; inc = incremento; len = lunghezza; nl = numero linea; nr = numero record; par(s) = parametro (i); stmt = statement; str = stringa; val = valore; var = variabile; [] indica il codice opzionale.

STANDARD MICROSOFT	ZX SPECTRUM	COMMODORE 64 e VIC 20	SINCLAIR QL	MSX BASIC	ZX 81
ASC (str)	CODE (s*r)	ASC (str)	CODE (str)	ASC (str)	CODE (str) Nota: ZX81 non usa codici ASCII
ATN (exp)	ATN (exp)	ATN (exp)	ATN (exp)	ATN (exp)	ATN (exp)
AUTO [nl, val]			AUTO [nl] [,inc]	AUTO [nl] [,inc]	
CALL var [(var, var...)]	LET var = USR (addr) RAND USR (addr)	SYS (addr)	CALL addr [exp] [pars]	USR (addr)	LET var = USR (addr) RAND USR (addr)
CHAIN "filename"			MERGE dis		
CHR$ (exp)	CHR$ (exp)	CHR$ (exp)	CHR$ (exp)	CHR$ (exp)	CHR$ (exp) Nota: ZX81 non usa codici ASCII
CLEAR [exp, exp]	CLEAR	CLEAR (exp) CLR	CLEAR	CLEAR	CLEAR
CLOSE	CLOSE # n. canale	CLOSE # n. file	CLOSE # canale	DISK	
CONT	CONT	CONT	CONTINUE oppure RETRY	CONT	CONT
DATA cost [cost...]	DATA cost [,cost]	DATA cost [,cost]	DATA cost [,cost]	DATA cost [,cost]	
DEF FNvar [(var, var...)] = **exp**	DEF FN var [(var, var...)] = exp	DEF FN var = exp	DEF FN var [$ o % o" [pars]	DEF FN var [(pars)] = exp	
DELETE nl [,nl]			DLINE range [range]	DELETE [nl] [-nl]	
DIM var(val) [,var(val),...]	DIM var(val)	DIM var(val) [,var(val)...]	DIM var(val) [,var(val)...]	DIM var(val) [,var(val)...]	DIM var(val)
EDIT nl	EDIT linea cursore	cursore	EDIT nl [,step]	cursore	EDIT linea cursore
EXP (exp)	EXP (exp)	EXP (exp)	EXP (exp)	EXP (exp)	EXP (exp)
FRE [exp]		FRE (exp)		FRE (exp)	
GET [#] **nl** [,nr]	Consultare il manuale del Microdrive	GET # nl, record [,record]	INKEY$ (channel)		
GOSUB nl	GOSUB nl/ var/exp	GOSUB nl	GOSUB nl/ var/exp	GOSUB nl	GOSUB nl/ var/exp
GOTO nl	GOTO nl/ var/exp	GOTO nl	GOTO nl/ var/exp	GOTO nl	GOTO nl/ var/exp
IF exp THEN [**ELSE**] **stmt**]	IF exp THEN stmt	IF exp THEN stmt	IF exp THEN stmt ELSE stmt END	IF exp THEN stmt/nl	IF exp THEN stmt

ATARI	APPLE SOFT	IBM PC BASIC A	BBC BASIC	MZ-80 K	TRS-80 II GENIE	MEMOTECH MTX 512	AMSTRAD CPC 464
ASC (str)	ASC (str)	ASC (str)	ASC (str)	ASC (str)	ASC (str)	ASC (str)	ASC (str)
ATN (exp)	ATN (exp)	ATN (exp)	ATN (exp)	ATN (exp)	ATN (exp)	ATN (exp)	ATN (exp)
		AUTO [nl] [,inc]	AUTO [nl, val]		AUTO [nl, val]	AUTO [nl, val]	AUTO [nl, val
	CALL addr	CALL addr [var,..., var]	CALL addr [var,... var]	USR (addr)		USR (addr)	CALL addr [,pars]
RUN "C:"	CHAIN "filename"	CHAIN filename	CHAIN "filename"				CHAIN "filename" [,nl, exp]
CHR$ (exp)	CHR$ (exp)	CHR$ (exp)	CHR$ (exp)	CHR$ (exp)	CHR$ (exp)	CHR$ (exp)	CHR$ (exp)
CLR	CLEAR	CLEAR	CLEAR	CLR	CLEAR [(exp)]	CLEAR	CLEAR [all] ERASE [list of] var
CLOSE [# n. file, n. file]	CLOSE "filename"	CLOSE [#] [filename]	CLOSE # n. file	CLOSE [filename]	dipende dal sistema operativo	DISC CLOSE # n. canale	CLOSEIN CLOSEOUT
CONT	CONT	CONT		CONT	CONT	CONT	CONT
DATA cost [,cost]	DATA cost [,cost...]	DATA cost [,cost...]	DATA cost [,cost...]	DATA cost [,cost]	DATA cost [,cost]	DATA cost [,cost...]	DATA cost [,cost...]
	DEF FNvar (var) = exp	DEF FNvar [(pars)] = exp	DEF FNvar [(var, var)] = exp	DEF FNvar (var) = exp	differenti DEF	DEF FNvar [(pars)] = exp	DEF FNvar [(pars)] = exp
	DEL nl, nl	DELETE [nl] [-nl]	DELETE nl, nl		DELETE nl-nl		DELETE [nl-nl]
DIM [o COM] var(val)	DIM var(val) [,var(val)...]	DIM var(val) [,var(val)...]	DIM var(val) [,var(val)...]	DIM var(val) [,var(val)...]	DIM var(val)	DIM var(val) [,var(val)...]	DIM [list of] var(val)
cursore	cursore	EDIT nl	cursore	cursore	EDIT nl	EDIT nl	EDIT nl
EXP (exp)	EXP (exp)	EXP (exp)	EXP (exp)	EXP (exp)	EXP (exp)	EXP (exp)	EXP (exp)
FRE (exp)	FRE (exp)	FRE (exp)	HIHEM-TOP	SIZE	FRE (exp) [TRS80] oppure MEM (Genie)		FRE (exp)
GET # nl, record	INPUT var [,var...]	GET [#] filename	INPUT # n. linea ,record	INPUT/T record	INPUT # n. file, record	DISC INPUT # n. canale	LINE INPUT # [,guida] [lista variabili]
GOSUB nl/ var/exp	GOSUB nl/ var/exp	GOSUB nl	GOSUB nl/ var/exp	GOSUB nl	GOSUB nl	GOSUB nl	GOSUB nl
GOTO nl/ var/exp	GOTO nl	GOTO nl	GOTO nl/ var/exp	GOTO nl	GOTO nl	GOTO nl	GOTO nl
IF exp THEN stmt	IF exp THEN stmt	IF exp THEN stmt ELSE stmt	IF exp THEN stmt ELSE stmt	IF exp THEN stmt	IF exp THEN stmt [ELSE stmt]	IF exp THEN stmt ELSE stmt	IF exp THEN stmt ELSE stmt

STANDARD MICROSOFT	ZX SPECTRUM	COMMODORE 64 e VIC 20	SINCLAIR QL	MSX BASIC	ZX 81
INKEY$	INKEY$	GET var	INKEY$ [channel[wait]]	var $ = INKEY$	INKEY$
INPUT [str:]**var**[**var,...**]	INPUT [string;] var	INPUT [str:] var [,var...]	INPUT [channel] [var[var,...]]	INPUT [prompt] var, [var]	INPUT var
INT (exp)	INT (exp)	INT (exp)	INT (exp)	INT (exp)	INT (exp)
LEFT$ (str, len)	str (TO finish)	LEFT$ (str, len)	str (TO finish)	LEFT$ (str, len)	str (TO finish)
LIST [**nl, nl**]	LIST [nl]	LIST [nl-nl]	LIST [channel] 1 line [-last line]	LIST [1 line] [-last line]	LIST [nl]
LLIST [**nl, nl**]	LLIST [nl]	OPEN 4,4:CMD4: LIST [nl-nl] OPEN 3,4:CMD3: LIST [nl-nl]	LIST # [channel] 1 line [,last line]	LLIST [nl, nl]	LLIST [nl]
LOAD ["**filename**"]	LOAD "filename"	LOAD ["filename"	LOAD device [filename]	CLOAD filename	LOAD "[filename]"
LOG (exp)	LN (exp)	LOG (exp)	LN (exp) Nota: può fare LOG 10 (exp)	LOG (exp)	LN (exp)
MID$ (str, start [,**len**])	str (start TO finish)	MID$ (str, start [len])	str (start TO finish)	MID$ (str, start [,len])	str (start TO finish)
NAME "filename" AS "filename"		OPEN 1,8, 15, "RO:filename = filename"			
NEW	NEW	NEW	NEW	NEW	NEW
NEXT var [,**var**]	NEXT var	NEXT [var, var]	NEXT var	NEXT [var, var...]	NEXT var
ON ERROR GOTO nl				ON ERROR GOTO nl	
ON exp GOSUB nl [,**nl...**]		ON exp GOSUB nl [,nl...]	ON var GOSUB nl [,nl]	ON exp GOSUB nl	
ON exp GOTO nl [,**nl...**]		ON exp GOTO nl [,nl...]	ON var GOTO nl [,nl]	ON exp GOTO nl	
OPEN mode [#] **n. file "filename"**	Consulta il manuale del Microdrive	OPEN ' exp, n. file, mode, "filename"	OPEN channel, device [filename]	soltanto su DISK Basic	
OUT port, byte	OUT port, byte			OUT port, data	
PEEK (addr)	PEEK (addr)	PEEK (addr)	PEEK [o W o L] (addr)	PEEK (addr)	PEEK (addr)
POKE addr, byte	POKE addr, byte	POKE addr, byte	POKE [o W o L] (addr), byte	POKE (addr, byte)	POKE addr, byte
PRINT [[#] **n. file**] [**exp**] [**, exp...**]	PRINT [AT] ("exp")	PRINT # n. file record [record]	PRINT [channel,] exp [,exp]...	PRINT [n. file,] lista di argomenti	PRINT [AT] ("exp")

ATARI	APPLE SOFT	IBM PC BASIC A	BBC BASIC	MZ-80 K	TRS-80 II GENIE	MEMOTECH MTX 512	AMSTRAD CPC 464
	GET var	var $ = INKEY$	GET var oppure INKEY$ (time)	GET var	INKEY$	var $ = INKEY$	INKEY$
INPUT [exp] var [var...]	INPUT [str] var [,var...]	INPUT var [,var...]	INPUT [str[,]] var [,var...]	INPUT [str;] var	INPUT [str;] var [,var...]	INPUT var, [var...]	INPUT [var [list]]
INT (exp)	INT (exp)	INT (exp)	INT (exp)	INT (exp)	INT (exp)	INT (exp)	INT (exp)
str (start, len)	LEFT$ (str, len)	LEFT$ (str, len)	LEFT$ (str, len)	LEFT$ (str, len)	LEFT$ (str, len)	LEFT$ (str, len)	LEFT$ (str, len)
LIST [nl, nl]	LIST [nl, nl]	LIST [1 linea] [-last line]	LIST [nl, nl]	LIST [nl-nl]	LIST [nl-nl]	LIST [1 line] [,last line]	LIST [nl, nl]
LIST "P:"	LIST "P"	LIST [nl, nl]	CTRL-B poi LIST [nl-nl]	LIST/P [nl-nl]	LLIST [nl-nl]	LIST [nl, nl]	LIST [nl, nl] # 8
CLOAD filename o LOAD "filename"	LOAD filename	LOAD filename [,R]	LOAD "filename"	LOAD ["filename"]	CLOAD "[file name]"	LOAD "filename"	LOAD ["filename" [,addr]
LOG (exp)	LOG (exp)	LOG (exp)	LN (exp)	LN (exp)	LOG (exp)	LN (exp)	LOG (exp)
str (start [,len])	MID$ (str, start [,len])	MID$ (str, start [,len])	MID$ (str, start [len])	MID$ (str, start len)	MID$ (str, start len)	MID$ (str, start [,len])	MID$ (str, start, len)
	RENAME old name newname	NAME filename AS filename		solo su disk Basic	consulta il manuale OS	DISC REN str = str	
NEW	NEW	NEW	NEW	NEW	NEW	NEW	NEW
NEXT var	NEXT [var, var...]	NEXT [var, var...]	NEXT [var, var...]	NEXT [var]	NEXT [var, var...]	NEXT var	NEXT [var, var...]
TRAP nl/var/ exp	ONERR GOTO nl	ON ERROR GOTO nl	ON ERROR stmt	solo su disco Basic	ON ERROR GOTO nl		ON ERROR GOTO nl
ON exp GOSUB nl [,nl...]	ON exp GOSUB nl [,nl...]	ON [exp: COM: KEY:PEN:STRIG] GOSUB nl	ON exp/var GOSUB nl [,nl...]	ON exp GOSUB nl [,nl...]	ON exp GOSUB n. [,nl...]	ON exp GOSUB nl	ON exp GOSUB nl [,nl...]
ON exp GOTO nl [,nl...]	ON exp GOTO nl [,nl...]	ON exp GOTO nl	ON exp/var GOTO nl [,nl...]	ON exp GOTO nl [,nl...]	ON exp GOTO nl [,nl...]	ON exp GOTO nl	ON exp GOTO nl [,nl...]
OPEN # n. file, mode, c. mode, filename	OPEN filename	OPEN filename [FOR Mode] AS [#] filename [LEN = rec]	n. file = OPENIN [per leggere] o n. file=OPENOUT [per scrivere]	ROPEN-leggere WOPEN-scrivere	consulta il manuale OS	DISC OPEN # n. channel "file name", filetype, recordin	OPENIN filename OPENOUT filename
		OUT port, data		OUT (port), byte	OUT port, byte	OUT port, data	OUT port, byte
PEEK (addr)	PEEK (addr)	PEEK (addr)		PEEK (addr)	PEEK (addr)	PEEK (addr)	PEEK (addr)
POKE addr, byte	POKE addr, byte	POKE addr, byte		POKE addr, byte	POKE addr, byte	POKE addr, byte	POKE (addr), byte
PRINT # n. file record [,record...]	PRINT exp [,exp...]	PRINT [exp][;]	PRINT # filename record [,record...]	PRINT/T record [,record...]	PRINT # n. file, record [,record...]	[DISC] PRINT [# n. channel] print list	PRINT [n.], [print list]

STANDARD MICROSOFT	ZX SPECTRUM	COMMODORE 64 e VIC 20	SINCLAIR QL	MSX BASIC	ZX 81
RANDOMIZE [**exp**]	RAND (exp)	RND (-TI)	RANDOMIZE		RAND (exp)
READ var [**,var...**]	READ var [,var...]	READ var [,var]	READ var [,var...]	READ var [,var, var...]	
RENUM [**nl, val**]			RENUM [old start n.] [TO old end n.] [new start n.] [inc]	RENUM [newn.] [,old nl] [,inc]	
RESTORE	RESTORE [nl/ exp]	RESTORE	RESTORE	RESTORE [nl]	
RESUME			RETRY	RESUME	
RETURN	RETURN	RETURN	RETURN exp	RETURN [nl]	RETURN
RIGHT$ (str, len)	str (start TO) len)	RIGHT$ (str, len)	str, (start, TO)	RIGHT$ (exp, len)	str (start TO)
RND (exp)	RND	RND (exp)	RND [exp TO exp]	RND (exp)	RND
RUN [**nl**]	RUN [nl/var/ exp]	RUN [nl]	RUN [nl]	RUN [nl]	RUN [nl/var/ exp]
SAVE filename	SAVE "filename"	SAVE ["filename"]	SAVE device [nl[nl]]...	CSAVE "filename"	SAVE "filename"
SGN (exp)	SGN (exp)	SGN (exp)	SGN (exp)	SGN (exp)	SGN (exp)
STRING$ (len, str)			FILL$ (str, len)	STRING$ (len, str)	
STR$ (exp)	STR$ (exp)	STR$ (exp)	conversione automatica all'assegnamento	STR$ (exp)	STR$ (exp)
SYSTEM					
TROFF				TROFF	
TRON				TRON	
USR (par)	USR addr	USR (par)	vedi CALL oppure usa EXEC	USR (par)	USR (addr)
WAIT port, mark [**,select**]	PAUSE n. (50/secondo)	WAIT addr, exp, exp	PAUSE [delay]		PAUSE exp
WHILE exp WEND			REPEAT name IFcond EXITname End REPEAT name		
WIDTH (val)			intero video controllabile	WIDTH (exp)	

ATARI	APPLE SOFT	IBM PC BASIC A	BBC BASIC	MZ-80 K	TRS-80 II GENIE	MEMOTECH MTX 512	AMSTRAD CPC 464
RND (-exp)		RANDOMIZE (exp)	RND (-exp)	RND (-exp)	RANDOM	RAND (exp)	RANDOMIZE (exp)
READ var [,var...]	READ var [,var...]	READ var [,var...]	READ var [,var...]	READ var [,var...]	READ var [,var...]	READ var [,var...]	READ var [,var...]
		RENUM [newn] [,old nl] [,inc]	RENUMBER [start] [,inc]		RENUM start, inc		RENUM [new nl] [,old nl] [,inc]
RESTORE [nl]	RESTORE	RESTORE [nl]	RESTORE (exp)	RESTORE	RESTORE [nl/exp]	RESTORE [nl]	RESTORE [nl]
	RESUME	RESUME			RESUME [nl]		RESUME [nl] o RESUME NEXT
RETURN	RETURN	RETURN [nl]	RETURN	RETURN	RETURN	RETURN	RETURN
str (start)	RIGHT$ (str, len)	RIGHT$ (exp, len)	RIGHT$ (str, len)	RIGHT$ (str, len)	RIGHT$ (str, len)	RIGHT$ (exp, len)	RIGHT$ (exp, len)
RND (exp)	RND (exp)	RND (exp)	RND (exp)	RND (exp)	RND (exp)	RND (exp)	RND (exp)
RUN	RUN [nl]	RUN [nl]	RUN	RUN	RUN [nl]	RUN [nl]	RUN [nl]
CSAVE "filename"	SAVE filename n. file	SAVE "file name" [,A,P]	SAVE "file name"	SAVE "file name"	CSAVE "filename"	SAVE "file name"	SAVE "filename [, file type] [,binary par]
SGN (exp)	SGN (exp)	SGN (exp)	SGN (exp)	SGN (exp)	SGN (exp)	SGN (exp)	SGN (exp)
		STRING$ (len, str)	STRING$ (len, str)		STRING$ (len, str)		STRING$ (len, str)
STR$ (exp)	STR$ (exp)	STR$ (exp)	STR$ (exp)	STR$ (exp)	STR$ (exp)	STR$ (exp)	STR$ (exp)
BYE		SYSTEM		BYE			
	NOTRACE	TROFF	TRACEOFF		TROFF		TROFF
	TRACE	TRON	TRACE ON		TRON		TRON
USR (par)	USR (par)	USR (exp)	USR (par)	USR (par)	USR (par)	USR (par)	vedi CALL
	WAIT addr, exp [,exp]	WAIT port, exp [,exp]					WAIT addr, mask [,inversion]
		WHILE exp WEND	REPEAT stmt UNTIL exp				WHILE exp WEND
POKE 82, val POKE 83, val	POKE 32, val POKE 33, val	WIDTH exp	WIDTH val				WIDTH exp

Sapete quanto vale il vostro computer usato?

MICROSTAR

in occasione del lancio sul mercato italiano dell'AMSTRAD CPC 464,
ritira in permuta il tuo vecchio computer,
valutandolo al massimo.

L'AMSTRAD CPC 464 è il primo e unico personal computer completo:

Z 80A
64K RAM
32K ROM
Registratore incorporato
Tastiera professionale

Monitor a colori o a fosfori verdi
Basic esteso velocissimo
Grafica in alta risoluzione (640 x 200)
80 colonne
Suono (3 voci e 7 ottave)

Il prezzo? La più piacevole delle sorprese

Per saperne di più, scrivete o telefonateci:

MICROSTAR srl
Via Cagliero, 17 - 20125 Milano
Tel. 02/6887604

MSX BANK

FINO AD UN MILIONE PER UN PROGRAMMA TUO, CHE SIA ORIGINALE

☆

Se hai programmi originali, che siano proprio pensati e fatti da te, di giochi e utilità, mandaceli in visione. Quelli pubblicati verranno compensati con almeno 200 mila lire l'uno. S'intende che se realizzerai qualcosa di davvero favoloso potrai guadagnare addirittura 1 milione di lire! Registra un solo programma per ogni cassetta, completo di istruzioni scritte, ed invialo a MSX Computer Magazine, c.so Vittorio Emanuele 15, 20122 Milano.

MISTER KIT

Elettronica 2000

ELETTRONICA APPLICATA, SCIENZA E TECNICA

N. 70 - FEBBRAIO 1985 - L. 3.000

Sped. in abb. post. gruppo III

Canon MSX V-20

Il Canon MSX V-20 recentemente presentato dalla Canon Italia allo Smau di Milano è un interessantissimo Home Computer che si colloca nella fascia alta di questo specifico mercato.
E va detto, a merito della multinazionale nipponica, che l'alta collocazione non è conseguenza di un "alto prezzo" (dal momento che questo sarà assolutamente allineato) ma, piuttosto, è merito delle innovazioni tecnologiche e delle conseguenti elevate prestazioni della macchina.
Elemento cruciale del Canon V-20 è l'adozione del sistema MSX, che merita d'essere spiegato brevemente.

L'MSX: un solo software per tutti.

MSX vuol dire microsoft basic, e si riferisce al sistema operativo utilizzato da tutte le macchine prodotte dalle marche aderenti ad uno speciale progetto.
La finalità con cui il progetto MSX è nato, è quella di permettere una perfetta intercambiabilità dei programmi e delle periferiche (stampanti, unità floppy disk, tavoletta elettronica ecc...) così che gli utilizzatori possano avvalersi di tutto il software delle varie case senza alcun problema di compatibilità, così come è totale anche la compatibilità dell'hardware.

Microsoft, il numero 1 del software.

Lo sviluppo del sistema MSX è stato affidato al colosso americano Microsoft, leader mondiale del software, da una ventina di società, molte delle quali giapponesi: insieme, si tratta del meglio oggi esistente al mondo, destinato esclusivamente agli utenti del sistema MSX.

Software per gioco e software sul serio.

Queste premesse garantiscono all'utente una ricchissima biblioteca di programmi.
Inizialmente saranno disponibili sul mercato essenzialmente programmi di "base" (foglio elettronico, word processing, data base, grafica) e cassette per videogiochi per sfruttare le eccezionali qualità di queste macchine collegate ad un normale televisore.
Immediatamente dopo verrà introdotto del software applicativo che potrà essere sfruttato al meglio da un computer versatile e potente come il Canon MSX V-20.

Canon MSX V-20 è una potenza.

64 Kb RAM, 32 Kb ROM; microprocessore Z-80; linguaggio MSX Basic; due alloggiamenti (slots) per cartucce ROM o per espansioni; interfacce per stampante e per registratore a cassette (assolutamente standard, incorporato); tastiera professionale con tasti rigidi (72); tasti guida cursore di grandi dimensioni; due uscite per joy-stick; uscita per televisione sistema PAL; testi: 25 linee per 40 colonne; grafica: 256 punti x 192 punti; 16 colori; suono: 8 ottave, 3 toni; possibilità di interfacciamento parallela e seriale.

Una scelta intelligente.

Canon MSX V-20 è un Home Computer che vi offre tutte le garanzie: quella del numero 1 mondiale della fotografia, con il meglio della tecnologia giapponese e con il meglio del software mondiale riuniti assieme, capace di dare all'operatore - professionale o amatoriale - un'ampiezza di programmi senza uguali e di grande qualità.
Infatti il DOS (Disk Operating System) che sarà sviluppato su questa macchina nell'immediato futuro, permetterà sofisticati utilizzi tecnici e gestionali, grazie alle possibilità di collegamento ad unità floppy disk.

Canon V-20 è l'Home Computer che, comprato oggi, vale per il futuro, compatibile, senza rischi, senza cambi, senza problemi.
C'è una scelta più sicura e intelligente?

Canon

NOVITÀ

UN LOOK
COLORATO
PER
LA TUA
CASSETTA

VEDI
SUBITO A
PAGINA
SEGUENTE

PUOI
RITAGLIARE
E
PERSONALIZZARE
LA TUA
CASSETTA
CON
IL TUO
NOME

YASHICA...

benvenuta in informatica!

Con il nuovo YC64 uniformato MSX, la Kyocera-Yashica entra nel mondo dell'home-computer. Un "inizio" di grande levatura per la casa fotografica piu' amata in Italia.
Anche nel nostro Paese, a 720mila Lire circa, e' disponibile, nei migliori negozi di computer e fotografia, il computer Yashica YC64 funzionante con sistema operativo MSX. Lo Yashica YC64 e' un apparecchio estremamente versatile in quanto lavora perfettamente con qualsiasi accessorio di standard MSX, tutti i videogiochi MSX...
Impostato come home-computer completo per la fascia media del mercato, basato sul famoso microprocessore Z80, si collega al televisore domestico. Le connessioni prevedono possibilita' di collegamento con un registratore esterno per la memoria di massa (ma esiste anche un lettore di cartucce interno), una porta per la gestione di un floppy disk-driver esterno, nonche' una porta parallela per una stampante ed una presa per monitor.
Il processore TMS9929A coadiuva lo Z80 per quanto riguarda le funzioni di gestione del video: e' capace di 16 accattivanti colori, 24 linee per 40 colonne ed una risoluzione grafica pari a 256x192 punti.
Incorpora un generatore di suoni AY.3.8910, capace di una gamma di 8 ottave, con tre diversi generatori di tono piu' effetti sonori e software con funzioni "sound".
La configurazione di memoria disponibile prevede 32Kbytes di ROM e 64Kbytes di RAM: questo amplia notevolmente la possibilita' di utilizzo ben oltre il normale uso di home-computer e di videogioco.

CARATTERISTICHE TECNICHE

CPU: Z80 con clock a 3,579545 MHz
VDG: TMS9929A
Text display: 24 righe da 40 colonne
Graphic display: 256x192 punti
Colori: 16
ROM: 32Kbytes MSX Basic
RAM: 64Kbytes utente
Interfaccia cassette: modulazione FSK, velocita' 1200/2400 Baud, gestione automatica del registratore
Sound: AY.3.8910 con 3 generatori indipendenti da 8 ottave
Porte I/O: 2 porte universali per joystick
Tastiera: 72 tasti con scansione software
Slot: 1 slot per cartucce preprogrammate
Video out: video composito modulato ed RGB
Printer: interfaccia parallela standard Centronics, connettore Amphenol 14 pin
Alimentazione: 220 Volt 50 Hz

distribuito da:
FOWA S.p.A.: via Tabacchi, 29 - 10132 TORINO

Spedire a: FOWA S.p.A. - Via Tabacchi, 29 - 10132 TORINO
Gratis senza impegno inviatemi materiale illustrativo per COMPUTER HC-YC64-MSX
Nome
Indirizzo
Città