PHILIPS

PHILIPS VG 8000 HOME COMPUTER

MANUALE DI RIFERIMENTO MSX - BASIC

PHILIPS MSX

PHILIPS

VG8000

MSX-BASIC
MANUALE DI RIFERIMENTO

MSX

Questo manuale è pubblicato da Philips Export B.V.,
Corporate Centre Video, Eindhoven, Olanda.

Introduzione

Questo manuale di riferimento contiene le descrizioni di tutti i comandi, le istruzioni e le funzioni MSX-BASIC, disposti per comodità in ordine alfabetico.

Le descrizioni sono state strutturate come segue:

Formato Qui si troverà l'appropriato formato dell'istruzione, del comando o della funzione. (Vedere più avanti in questa pagina "Notazione di Formato").

Scopo Qui si troverà lo scopo del comando, dell'istruzione o della funzione. In altre parole, il motivo per cui vengono usati.

Categoria Qui si troverà la categoria di appartenenza: istruzione, comando, o funzione.

Note Qui si imparerà ad usare le istruzioni efficacemente.

Esempio Per ciascuna istruzione viene fornito un esempio per illustrarne l'applicazione.

Notazione di formato

1. I vari elementi che all'interno del formato sono indicati in lettere maiuscole, devono essere battuti esattamente nello stesso modo in cui sono indicati.

2. Gli elementi indicati tra parentesi angolate < > devono essere definiti dall'utente.

3. Gli elementi indicati tra parentesi quadre [ ] sono facoltativi.

4. Gli elementi seguiti da tre punti (. . .) possono essere ripetuti tante volte quante lo consente la lunghezza della riga di programma.

5. Tutti i segni di punteggiatura ad eccezione delle parentesi quadre ed angolate devono essere battuti esattamente come indicato! Ciò si applica ai punti, alle virgole, alle parentesi, ai due punti, ai punti e virgola, ecc. (. , ( ) : ;).

6. "X", "Y" e "Z" rappresentano espressioni numeriche.

7. "X$", "Y$" e "Z$" rappresentano espressioni alfanumeriche.

1. ABS

Formato: ABS(⟨X⟩)

Scopo: Dà il valore assoluto di ⟨X⟩

Categoria: Funzione

Note: Il risultato finale di questa funzione sarà sempre una cifra positiva.

Esempio:

```
10 PRINT ABS(7*(-5))
20 END
RUN
```

2. ASC

Formato: ASC(⟨X$⟩)

Scopo: Dà il codice carattere del primo carattere di ⟨X$⟩

Categoria: Funzione

Note: Se ⟨X$⟩ non contiene un carattere, sullo schermo compare il messaggio di errore "Illegal function call" (Richiamo di funzione illecita).

Esempio:

```
10 X$="TEST"
20 PRINT ASC(X$)
30 END
RUN
```

3. ATN

Formato: ATN(⟨X⟩)

Scopo: Dà l'arcotangente di ⟨X⟩ espresso in radianti

Categoria: Funzione

Note: Il risultato di questa funzione avrà un valore compreso tra $-\pi/2$ e $+\pi/2$ (π = 3.1415926535).

Esempio:

```
10 X=5
20 PRINT ATN(X)
30 END
RUN
```

4. AUTO

Formato: AUTO [[⟨X⟩][,[⟨Y⟩]]]

Scopo: Generare numeri di riga

Categoria: Comando

Note: ⟨X⟩ e ⟨Y⟩ devono avere valori interi positivi

Dopo il comando AUTO la numerazione di riga inizia con il numero indicato da ⟨X⟩, che successivamente si incrementa del numero indicato da ⟨Y⟩.

Se non viene immesso alcun valore per ⟨X⟩ o ⟨Y⟩, viene usato automaticamente il numero 10.

Se ⟨X⟩ è seguito da una virgola ma non viene indicato alcun valore per ⟨Y⟩, viene assunto l'ultimo valore di ⟨Y⟩ utilizzato.

Se viene generato un numero di riga che è già stato usato e che è già caricato nella memoria del computer, immediatamente dopo il numero di riga compare un asterisco per avvertire l'utente che sta per immettere una nuova riga che sostituirà quella originale. Quando l'utente preme il tasto RETURN subito dopo la comparsa dell'asterisco, la riga originale verrà salvata.

La numerazione automatica di riga viene interrotta premendo simultaneamente i tasti CTRL e C o premendo simultaneamente i tasti CTRL e STOP. Così facendo l'ultimo numero di riga non verrà immesso nella memoria del computer.

Esempio:

```
AUTO 100,50
```

(genera i numeri di riga 100, 150, 200 ecc.)

```
AUTO
```

(genera i numeri di riga 10, 20, 30, 40, ecc.)

5. BASE

Formato: BASE(⟨X⟩)

Scopo: Contiene il primo indirizzo delle tabelle del processore video (VDP)

Categoria: Variabile di sistema

Note: ⟨X⟩ deve sempre avere un valore intero positivo compreso tra 0 e 19

I numeri hanno il seguente significato:

0 = Il primo indirizzo della tabella dei nomi nel modo testo 1
1 = Nessun significato
2 = Il primo indirizzo della tabella dei profili nel modo testo 1
3 = Nessun significato
4 = Nessun significato
5 = Il primo indirizzo della tabella dei nomi nel modo testo 2
6 = Il primo indirizzo della tabella dei colori nel modo testo 2
7 = Il primo indirizzo della tabella dei profili nel modo testo 2
8 = Il primo indirizzo della tabella degli attributi degli sprites nel modo testo 2
9 = Il primo indirizzo della tabella dei profili degli sprites nel modo testo 2
10 = Il primo indirizzo della tabella dei nomi nel modo grafico 1
11 = Il primo indirizzo della tabella dei colori nel modo grafico 1
12 = Il primo indirizzo della tabella dei profili nel modo grafico 1
13 = Il primo indirizzo della tabella degli attributi degli sprites nel modo grafico 1
14 = Il primo indirizzo della tabella dei profili degli sprites nel modo grafico 1
15 = Il primo indirizzo della tabella dei nomi nel modo grafico 2
16 = Il primo indirizzo della tabella dei colori nel modo grafico 2
17 = Il primo indirizzo della tabella dei profili nel modo grafico 2
18 = Il primo indirizzo della tabella degli attributi degli sprites nel modo grafico 2
19 = Il primo indirizzo della tabella dei profili degli sprites nel modo grafico 2

USARE QUESTA VARIABILE SOLTANTO SE SI HA COMPLETA FAMILIARITÀ CON IL FUNZIONAMENTO DEL VDP (PROCESSORE DELLO SCHERMO VIDEO)

Esempio:

```
10 SCREEN 0
20 PRINT BASE(2)
30 END
RUN
```

6. BEEP

Formato BEEP

Scopo: Generare un segnale acustico

Categoria: Istruzione

Note: Questa istruzione ha lo stesso effetto di "PRINT CHR$(7)"

Il suono può anche essere ottenuto premendo simultaneamente i tasti CTRL e G.

Esempio:

```
10 BEEP
20 FOR I=0 TO 1000:NEXT
30 PRINT "ORA CON CHR$(7)"
40 PRINT CHR$(7);
50 END
RUN
```

7. BIN$

Formato: BIN$(<$>)

Scopo: Dà una rappresentazione binaria del numero decimale <X>.

Categoria: Funzione

Note: Il risultato di questa funzione è un valore alfanumerico.

<X> deve essere un intero con un valore compreso fra -32768 e +65535.

Se <X> è negativo, viene usata la forma del complemento a due. Ciò significa che BIN$(-1) = BIN$(65536-1).

Esempio:

```
10 A=53
20 A$=BIN$(A)
30 PRINT A;A$
40 END
RUN
```

8. BLOAD

Formato: BLOAD "⟨periferica⟩:⟨nome file⟩"[,R][,⟨X⟩]

Scopo: Caricare nella memoria del computer un programma in linguaggio macchina da una periferica.

Categoria: Istruzione

Note: ⟨periferica⟩ può essere:
CAS = registratore dati
A = unità a dischetti 1
B = unità a dischetti 2

⟨nome file⟩ è il nome del programma in linguaggio macchina.

Se viene usata l'opzione R, il programma viene eseguito immediatamente dopo che è stato caricato nella memoria del computer.

Il programma viene registrato nella memoria del computer nella locazione specificata, aumentata del valore di ⟨X⟩ quando il programma è scritto sulla periferica con il comando "BSAVE".

Quando ⟨X⟩ non viene indicato, il suo valore s'intende zero.

Esempio:
```
BLOAD "CAS:TEST",R,&H20
```

"BLOAD CAS" PUO' ESSERE USATO SOLTANTO QUANDO AL VG8000 È STATO COLLEGATO UN REGISTRATORE DATI.

"BLOAD A" E "BLOAD B" POSSONO ESSERE USATI SOLTANTO QUANDO AL VG8000 SONO STATI COLLEGATI MSX-DOS E L'UNITA' A DISCHETTI 1 O 2.

9. BSAVE

Formato: BSAVE "⟨periferica⟩:⟨nome file⟩",⟨X⟩,⟨Y⟩,[,⟨Z⟩]

Scopo: Scrivere su una periferica un programma in linguaggio macchina.

Categoria: Istruzione

Note: ⟨periferica⟩ può essere:
CAS = registratore dati
A = unità a dischetti 1
B = unità a dischetti 2

⟨nome file⟩ è il nome del programma in linguaggio macchina.

⟨X⟩ è l'indirizzo di memoria al quale inizia il programma.

⟨Y⟩ è l'indirizzo di memoria al quale termina il programma.

⟨Z⟩ è l'indirizzo di memoria al quale deve iniziare l'esecuzione del programma. Quando ⟨Z⟩ viene omesso, ⟨X⟩ viene automaticamente considerato come il punto in cui deve iniziare l'esecuzione del programma.

Esempio:
```
BSAVE "CAS:TEST",&HC000,&HE0FF,$HC020
```

"BSAVE CAS" PUÒ ESSERE USATA SOLTANTO SE AL VG8000 È STATO COLLEGATO UN REGISTRATORE DATI.

"BSAVE A" E "BSAVE B" POSSONO ESSERE USATI SOLTANTO SE AL VG8000 SONO STATI COLLEGATI MSX-DOS E L'UNITÀ A DISCHETTI 1 O 2.

10. CALL

Formato: `CALL ⟨assegnazione⟩[(⟨A$⟩[,⟨B$⟩]...)]`

Scopo: Richiamare un'istruzione supportata da una cartuccia ROM

Categoria: Istruzione

Note: Le istruzionI dell'MSX-BASIC possono essere estese attraverso una cartuccia ROM. Nella cartuccia si deve determinare ciò che COMPORTA la costante esatta dell'istruzione comporta e come deve funzionare.

L'istruzione può essere richiamata con "CALL".

⟨A$⟩ e ⟨B$⟩ sono costanti alfanumeriche che possono essere aggiunte come argomenti all'istruzione.

Invece della parola "CALL", può anche essere usato un trattino di sottolineatura (_).

Esempio:

```
CALL TALK("A","B")
```

oppure

```
_TALK("A","B")
```

11. CDBL

Formato: `CDBL(⟨X⟩)`

Scopo: Attribuisce una variabile o un'espressione numerica ad un numero a precisione doppia.

Categoria: Funzione

Note: Il risultato di questa funzione è un numero con un massimo di 14 cifre. Esso richiede 8 byte della memoria del computer.

Esempio:

```
10 A#=CDBL(9/7)
20 PRINT A#
30 END
RUN
```

12. CHR$

Formato: CHR$(⟨X⟩)

Scopo: Dà il carattere indicato dal codice carattere di ⟨X⟩.

Categoria: Funzione

Note: ⟨X⟩ deve essere un numero compreso tra 0 e 255.

I caratteri alternativi hanno il codice carattere 1 + il codice del carattere.

Esempio:

```
10 SCREEN 1
20 FOR I=32 TO 255
30 PRINT CHR$(I);
40 NEXT:PRINT:PRINT
50 FOR I=65 TO 95
60 PRINT CHR$(1)+CHR$(I);
70 NEXT
80 END
RUN
```

13. CINT

Formato: CINT(⟨X⟩)

Scopo: Converte una variabile o un'espressione numerica in un numero intero.

Categoria: Funzione

Note: Il risultato di questa funzione è un numero con un massimo di 5 cifre senza porzione decimale. Esso occupa 2 byte della memoria del computer.

Se X<-32768 oppure se X>32767, sullo schermo compare il messaggio "overflow" (supero di capacità).

Esempio:

```
10 A%=CINT(9/7)
20 PRINT A%
30 END
RUN
```

14. CIRCLE

Formato: CIRCLE [STEP](⟨X⟩,⟨Y⟩),⟨Z⟩,[,⟨XX⟩][,⟨YY⟩][,⟨ZZ⟩] [,⟨XXX⟩]

Scopo: Disegnare un cerchio o un'ellisse sullo schermo nel modo grafico 1 o 2.

Categoria: Istruzione

Note: ⟨X⟩ è la coordinata X del punto centrale del cerchio e deve essere un intero tra 0 e 255.

⟨Y⟩ è la coordinata Y del centro del cerchio e deve essere un intero tra 0 e 191.

Quando viene usata la parola "STEP", i valori ⟨X⟩ e ⟨Y⟩ sono interpretati relativamente alla posizione del cursore. In questo caso ⟨X⟩ e ⟨Y⟩ possono anche essere interi negativi.

⟨Z⟩ rappresenta il raggio del cerchio e deve essere un intero tra 0 e 32767.

⟨XX⟩ è il numero di un colore ed indica in quale colore il cerchio deve essere disegnato. ⟨XX⟩ deve essere un intero tra 0 e 15. I numeri rappresentano i seguenti colori:

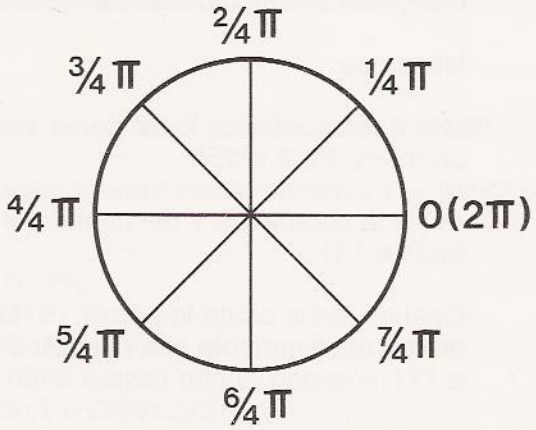
0 = trasparente	8 = rosso
1 = nero	9 = rosso chiaro
2 = verde	10 = giallo scuro
3 = verde chiaro	11 = giallo chiaro
4 = blu scuro	12 = verde scuro
5 = azzurro	13 = magenta
6 = rosso scuro	14 = grigio
7 = blu verde	15 = bianco

Se non viene indicato alcun numero di colore, viene usato l'ultimo colore indicato per il primo piano.
Il valore assunto standard di ⟨XX⟩ è 15.

⟨YY⟩ rappresenta l'angolo al quale deve iniziare il cerchio ed è indicato in radianti.
Quando per ⟨YY⟩ non viene indicato nulla, il suo valore sarà zero.

⟨ZZ⟩ è l'angolo al quale termina il cerchio, sempre indicato in radianti.
Quando per ⟨ZZ⟩ non è indicato alcun valore, viene assunto automaticamente 2 volte π.

⟨YY⟩ e ⟨ZZ⟩ devono avere un valore compreso tra 0 e 2π (π = 3.1415926535898). Il significato e la posizione approssimativa sono indicati nella seguente illustrazione:



Quando vengono usati valori negativi per ⟨YY⟩ e ⟨ZZ⟩, il punto iniziale ed il punto finale del cerchio saranno collegati al punto centrale da una linea retta.

⟨XXX⟩ è un numero che indica la relazione tra il raggio orizzontale e verticale. Se non viene indicato alcun valore per ⟨XXX⟩, si assume automaticamente 1.

Esempio:

```
10 SCREEN 2
20 A=3.1415926535898
30 CIRCLE (80,80),20,1,-(A*1/4),-(A*6/4)
40 FOR I=0 TO 3000:NEXT
50 END
RUN
```

15. CLEAR

Formato: CLEAR [⟨X⟩][,⟨Y⟩]

Scopo: Impostare tutte le variabili numeriche a zero, impostare tutte le variabili alfanumeriche ad un valore nullo e definire in memoria lo spazio per le variabili alfanumeriche.

Categoria: Istruzione

Note: ⟨X⟩ è un intero positivo che indica il numero di byte di tutte le variabili alfanumeriche.
Il valore standard preprogrammato per ⟨X⟩ è 200.

⟨Y⟩ è un intero positivo che indica l'indirizzo di memoria più elevato che può essere usato da MSX-BASIC.

Tutte le funzioni preprogrammate, ad esempio "DEF", "FN", "DEFINT", "DEFSNG", "DEFDBL", "DEFSTR", "DEFUSR" vengono eliminate dopo l'istruzione "CLEAR".

Tutte le tabelle definite con l'istruzione "DIM" sono a loro volta cancellate dopo l'istruzione "CLEAR".

Tutti i file aperti verranno chiusi dopo l'istruzione "CLEAR".

Esempio:

```
10 A=10:B$="TEST"
20 PRINT A,B$
30 CLEAR
40 PRINT A,B$
50 END
RUN
```

16. CLOAD/CLOAD?

Formati:

```
CLOAD ["<nome programma>"]
CLOAD? ["<nome programma>"]
```

Scopo: Caricare un programma da una cassetta nella memoria del computer.

Categoria: Comando

Note: Il programma deve essere scritto sulla cassetta in formato binario con il comando "CSAVE".

Il nome del programma è una costante alfanumerica specificata quando il programma è stato scritto sulla cassetta con il comando "CSAVE". Se non viene indicato alcun nome di programma, verrà caricato il primo programma trovato sulla cassetta.

Quando nella cassetta viene trovato il programma, MSX-BASIC fa comparire sullo schermo il testo seguente: FOUND(trovato), seguito dal nome del programma.

Quando il nome del programma non corrisponde a quello indicato nel comando, sullo schermo si vede comparire il messaggio SKIP:(salta), seguito dal nome del programma ed MSX-BASIC continua la sua ricerca del programma specificato.

Il comando "CLOAD" chiude tutti i file aperti e rimuove l'eventuale programma prevalente dalla memoria del computer.

"CLOAD?" confronta il programma sulla cassetta con il programma caricato nella memoria.
Quando i programmi corrispondono, MSX-BASIC risponde con "Ok".
In caso contrario, MSX-BASIC emette il messaggio "Verify error" (Ricercare errore).

Esempio:

```
CLOAD "DEMO"
```

"CLOAD" O "CLOAD?" POSSONO ESSERE USATI SOLTANTO QUANDO AL VG8000 È STATO COLLEGATO UN REGISTRATORE DATI.

17. CLOSE

Formato: `CLOSE [[#]⟨X⟩[,[#]⟨Y⟩...]`

Scopo: Chiudere un file e rilasciare il buffer di memoria ad esso associato.

Categoria: Istruzione

Note: Per aprire un file e riservare il necessario spazio del buffer, viene usata l'istruzione "OPEN".

⟨X⟩ e ⟨Y⟩ sono i numeri con i quali i file sono stati aperti con l'istruzione "OPEN".
Quando non viene indicato alcun numero di file, tutti i file aperti vengono chiusi.

Esempio:

```
10 MAXFILES=1
20 OPEN "CAS:TEST" FOR OUTPUT AS#1
25 A$="TEST FILE"
30 PRINT #1,A$
40 CLOSE #1
50 END
RUN
```

18. CLS

Formato: `CLS`

Scopo: Pulire lo schermo

Categoria: Istruzione

Note: Questa istruzione è identica a "PRINT CHR$(12)".

Lo stesso effetto può anche essere ottenuto premendo simultaneamente il tasto SHIFT ed il tasto HOME o premendo simultaneamente il tasto CTRL ed il tasto L.

Quando l'istruzione è stata eseguita, il cursore viene spostato alla prima posizione della prima riga.

Esempio:

```
10 CLS
20 PRINT "ORA CON CHR$(12)"
30 FOR I=0 TO 1000:NEXT
40 PRINT CHR$(12);
50 END
RUN
```

19. COLOR

Formato: COLOR [⟨X⟩][,⟨Y⟩][,⟨Z⟩]

Scopo: Definire il colore dello schermo

Categoria: Istruzione

Note: ⟨X⟩ è un intero tra 0 e 15 e definisce il colore del primo piano.

⟨Y⟩ è un intero tra 0 e 15 e definisce il colore dello sfondo.

⟨Z⟩ è un intero tra 0 e 15 e definisce il colore delle aree di bordo nel modo testo 2 e nei modi grafici 1 e 2.

⟨X⟩, ⟨Y⟩ e ⟨Z⟩ sono i numeri dei colori:

0 = trasparente	8 = rosso
1 = nero	9 = rosso chiaro
2 = verde	10 = giallo scuro
3 = verde chiaro	11 = giallo chiaro
4 = blu scuro	12 = verde scuro
5 = azzurro	13 = magenta
6 = rosso scuro	14 = grigio
7 = verde	15 = bianco

Quando non viene attribuito alcun valore per ⟨X⟩, ⟨Y⟩ o ⟨Z⟩, MSX-BASIC userà le ultime istruzioni di colore utilizzate per le aree di primo piano, di fondo e dei bordi.
I valori standard di ⟨X⟩, ⟨Y⟩ e ⟨Z⟩ sono rispettivamente 15, 4 e 7.

Esempio:

```
10 SCREEN 1
20 COLOR 1,1,1
30 FOR I=0 TO 15
40 PRINT "COLORE";I;",1,1"
50 COLOR I,1,1
60 FOR K=0 TO 300:NEXT K
70 NEXT I
80 COLOR 1,1,1
90 FOR I=0 TO 15
100 PRINT "COLORE 1,";I;",1"
110 COLOR 1,I,1
120 FOR K=0 TO 300:NEXT K
130 NEXT I
140 COLOR 15,1,1
145 FOR I=0 TO 15
150 PRINT "COLORE 15,1,";I
160 COLOR 15,1,I
170 FOR K=0 TO 300:NEXT K
180 NEXT I
190 COLOR 15,4,7
200 END
RUN
```

20. CONT

Formato: CONT

Scopo: Continuare l'esecuzione di un programma che è stato interrotto

Categoria: Comando

Note: L'esecuzione del programma continua iniziando dalla riga in cui è stato interrotto.

L'esecuzione del programma può essere interrotta con l'istruzione "STOP" o "END" ma anche premendo simultaneamente i tasti CTRL e STOP.

Quando il programma è interrotto dopo un'istruzione "INPUT", la sua esecuzione verrà ripresa con l'istruzione "INPUT" dopo il comando "CONT".

"CONT" è più spesso usato in combinazione con l'istruzione "STOP" per individuare l'errore in un programma.
Quando il programma è interrotto dall'esecuzione di "STOP", il contenuto di una variabile può essere esaminato e modificato nel modo diretto.

Esempio:

```
10 PRINT "PROVA"
20 STOP
30 PRINT "# STATO USATO CONT"
40 END
RUN
```

21. COPY

Formato: COPY "⟨periferica⟩:⟨nome file-1⟩"
[TO "⟨periferica⟩:⟨nome file-2⟩"]

Scopo: Copiare un file su dischetto

Categoria: Istruzione

Note: ⟨periferica⟩ può essere:
A = unità a dischetti numero 1
B = unità a dischetti numero 2

⟨nome file-1⟩ è il nome del file che deve essere copiato.

⟨nome file-2⟩ sarà il nome del file copiato.

Se non viene indicato "TO", il file verrà trasferito all'altra unità a dischetti sotto lo stesso nome.

Esempio:
```
COPY "A:TEST" TO "B:TEST2"
```

"COPY" PUÒ ESSERE USATO SOLTANTO SE AL VG8000 SONO STATI COLLEGATI MSX-DOS E LE UNITÀ A DISCHETTI 1 O 2.

22. COS

Formato: COS(⟨$⟩)

Scopo: Dà il coseno di ⟨X⟩ espresso in radianti

Categoria: Funzione

Note: Nessuna

Esempio:
```
10 X=5
20 PRINT COS(X)
30 END
RUN
```

23. CSAVE

Formato: CSAVE "⟨nome programma⟩"[,⟨X⟩]

Scopo: Scrivere sulla cassetta un programma caricato nella memoria del computer.

Categoria: Comando

Note: Il programma è scritto sulla cassetta in forma binaria e può essere letto nella memoria del computer con il comando "CLOAD".

Il nome del programma è una costante alfanumerica che deve essere definita di nuovo con il comando "CLOAD".

⟨X⟩ indica la velocità di trasmissione e deve essere 1 o 2:
1 = 1200 baud
2 = 2400 baud

Quando ⟨X⟩ non viene indicato, viene riutilizzata l'ultima velocità di trasmissione impiegata. Il valore assunto standard di ⟨X⟩ è 1.

La velocità di trasmissione può anche essere indicata con l'istruzione "SCREEN".

Esempio: CSAVE "DEMO"

"CSAVE" PUÒ ESSERE USATO SOLTANTO SE AL VG8000 È COLLEGATO UN REGISTRATORE DATI.

24. CSNG

Formato: CSNG(⟨X⟩)

Scopo: Converte una variabile o un'espressione numerica in numero a precisione singola.

Categoria: Funzione

Note: Il risultato di questa funzione è un numero con un massimo di 6 cifre. Esso occupa 4 byte della memoria del computer.

Esempio:
```
10 A!=CSNG(9/7)
20 PRINT A!
30 END
RUN
```

25. CSRLIN

Formato: CSRLIN

Scopo: Dà la coordinata Y(numero di riga) del cursore.

Categoria: Funzione

Note: Il risultato di questa funzione è un numero compreso tra 0 e 23. La riga superiore è la riga zero.

Questa funzione può essere usata soltanto nei modi testo 1 o 2.

Vedere "POS(0)"

Esempio:

```
5 SCREEN 0
10 LOCATE 10,20
20 PRINT CSRLIN
30 END
RUN
```

26. CVI/CVS/CVD

Formato:

```
CVI(<X$>)
CVS(<Y$>)
CVD(<Z$>)
```

Scopo: Converte valori alfanumerici nei corrispondenti valori numerici

Categoria: Funzione

Note: I valori numerici che vengono usati dal buffer per un file casuale su dischetto, devono essere convertiti da alfanumerici a numerici.

"CVI" converte una variabile alfanumerica a 2 byte in un intero.

"CVS" converte una variabile alfanumerica a 4 byte a precisione singola.

"CVD" converte una variabile alfanumerica a 8 byte a precisione doppia.

Esempio: Vedere "FIELD"

"CVI", "CVS" E "CVD" POSSONO ESSERE USATI SOLTANTO SE AL VG8000 SONO STATI COLLEGATI MSX-DOS E L'UNITÀ A DISCHETTI 1 O 2.

27. DATA

Formato: DATA <costante>[,<costante>...]

Scopo: Memorizzare delle costanti

Categoria: Istruzione

Note: Le istruzioni "DATA" possono essere usate ovunque in un programma.

Un'istruzione "DATA" può contenere tante costanti quante ne può accogliere una riga di programma, separate da virgole. In un programma è possibile usare qualsiasi numero di istruzioni "DATA".

L'istruzione "READ" accederà alle informazioni immesse attraverso le istruzioni "DATA" in sequenza di numero di riga.
Queste informazioni possono essere intese come una lista continua di costanti, indipendentemente da quante costanti sono state inserite in una istruzione "DATA" o dal punto in cui sono inserite nel programma.

La lista di costanti può contenere costanti numeriche (senza espressioni numeriche) e costanti alfanumeriche.
Le costanti alfanumeriche devono essere inserite tra virgolette quando aprono o chiudono con virgole, due punti o un numero notevole di spazi.

Il tipo di variabile, indicato nell'istruzione "READ" deve coincidere con la corrispondente costante nell'istruzione "DATA".

Le istruzioni "DATA" possono essere lette a partire da una data riga di programma o dall'inizio di programma per mezzo dell'istruzione "RESTORE".

Esempio:

```
10 DATA 123,ABC,48.5
20 READ A%,A$,A!
30 PRINT A%;A$;A!
40 RESTORE
50 READ A%:PRINT A%
60 END
RUN
```

28. DEF FN

Formato: DEF FN⟨nome⟩[⟨variabile⟩[,⟨variabile⟩...])]=⟨formula⟩

Scopo: Definire una funzione, scritta dall'utente.

Categoria: Istruzione

Note: Il nome deve essere un nome lecito e diventa, preceduto da "FN" il nome della funzione.

Quando riguarda una funzione alfanumerica, l'ultimo carattere del nome deve essere il segno del dollaro ($).

La variabile è la variabile nella formula della funzione che deve essere sostituita dal valore dato con il richiamo della funzione.

La formula è un'espressione delle prestazioni della funzione. La lunghezza massima è limitata a quella di una riga di programma.

Una variabile in una formula può anche comparire nella lista di variabili dopo il nome della funzione. In questo caso questa variabile verrà sostituita dal valore definito quando la funzione viene richiamata; nell'altro caso va usato l'ultimo contenuto dato della variabile.

Le variabili che seguono il nome di funzione rappresentano le variabili nella formula su base uno-uno.

Quando la natura del nome della funzione (alfanumerica o numerica) non è conforme alla formula, sullo schermo compare il messaggio di errore "Type mismatch" (tipo errato).

Esempio:

```
10 DEF FNC(A,B)=A*2+B*3
20 DEF FND$(I)=CHR$(1)+CHR$(I)
30 X=5:Z=2
40 E=FNC(X,Z)
50 PRINT E
55 FOR J=0 TO 2000:NEXT
60 SCREEN 1
70 FOR J=65 TO 95
80 PRINT FND$(J)
90 NEXT J
100 END
RUN
```

29. DEFINT/DEFSNG/DEFDBL/DEFSTR

Formati:

```
DEFINT <lettera-1>[,<lettera-2>...]
DEFSNG <lettera-1>[,<lettera-2>...]
DEFDBL <lettera-1>[,<lettera-2>...]
DEFSTR <lettera-1>[,<lettera-2>...]
```

Scopo: Determinare il tipo di variabile

Categoria: Istruzione

Note: <lettera-1> e <lettera-2> può essere qualsiasi lettera dell'alfabeto, ad esempio "K" o una serie di lettere, ad esempio "K-P". L'importante è che tutte le variabili, iniziando con la lettera definita, siano dello stesso tipo.

Con "DEFINT" tutte le variabili che seguono sono dichiarate intere.

Con "DEFSNG" tutte le variabili che seguono sono dichiarate a precisione singola.

Con "DEFDBL" tutte le variabili che seguono sono dichiarate a precisione doppia.

Con "DEFSTR" tutte le variabili che seguono sono dichiarate alfanumeriche.

Se, per contro, ad una variabile definita dall'istruzione "DEFxxx" in un programma viene attribuito un carattere di tipo (%,!,#,$), il tipo predefinito per quel carattere viene annullato.

Esempio:

```
10 DEFINT I
20 DEFSNG J
30 DEFDBL K
40 DEFSTR L-N
50 I=1.6:PRINT I
60 J=1.6:PRINT J
70 K=1/3:PRINT K
80 L="ABC":M="DEF":N="GHF"
90 PRINT L;M;N
100 END
RUN
```

30. DEFUSER

Formato: DEFUSR[⟨X⟩]=⟨Y⟩

Scopo: Specificare l'indirizzo di inizio di una subroutine in linguaggio macchina.

Categoria: Istruzione

Note: ⟨X⟩ è un numero intero che va da 0 a 9.
Quando ⟨X⟩non viene indicato, il suo valore si presume zero.

⟨Y⟩ è l'indirizzo iniziale della subroutine.

È possibile usare in un programma un numero illimitato di istruzioni "DEFUSER" per ridefinire gli indirizzi espressi inizialmente.

La subroutine in linguaggio macchina è eseguita dalla funzione "USR".

Esempio:

```
10 CLEAR 200,&HEFFF
20 AA=&HF000:DEFUSR3=AA
30 PRINT "L'INDIRIZZO INIZIALE DELLA SUBROUTINE E'";
40 PRINT AA
50 END
RUN
```

31. DELETE

Formato: DELETE [⟨X⟩][-⟨Y⟩]

Scopo: Rimuovere una riga di programma

Categoria: Comando

Note: ⟨X⟩ e ⟨Y⟩ sono numeri di riga

Quando ⟨X⟩ e ⟨Y⟩ sono definiti, tutti i numeri di riga da ⟨X⟩ fino a ⟨Y⟩compreso sono cancellati dal programma.

Quando viene definito soltanto ⟨Y⟩, verranno cancellate tutte le righe di programma, iniziando dalla prima, fino alla riga ⟨Y⟩compresa.

Quando non vengono specificati nè ⟨X⟩ nè ⟨Y⟩, compare il messaggio di errore "Illegal function call" (Richiamo di funzione illecita).

Esempio:

```
DELETE 10
```
(Rimuove la riga di programma 10)

```
DELETE 50-100
```
(Rimuove le righe di programma da 5 a 100)

```
DELETE -100
```
(Rimuove tutte le righe di programma fino a 100)

32. DIM

Formato: DIM ⟨variabile⟩(⟨X⟩[,⟨Y⟩[,⟨Z⟩]])

Scopo: Specificare il numero massimo di elementi di una tabella.

Categoria: Istruzione

Note: Quando viene usata una variabile senza l'istruzione "DIM", il numero massimo di elementi è sempre 11 (da 0 a 10).

⟨variabile⟩ è il nome della variabile

⟨X⟩ rappresenta il numero di elementi

⟨Y⟩ è il numero degli elementi bidimensionali

⟨Z⟩ è il numero degli elementi tridimensionali

Quando un indice usato con una variabile è maggiore del numero di elementi indicati nell'istruzione "DIM", sullo schermo compare il messaggio di errore "Subscript out of range" (Indice fuori campo).

Con l'istruzione "DIM", nella memoria del computer viene riservato lo spazio per l'intera tabella.

Una variabile non può essere usata in un'istruzione "DIM" se è già stata dimensionata, a meno che non venga usata l'istruzione "ERASE".

Esempio:

```
10 FOR I=0 TO 10
20 PRINT A(I)
30 NEXT
40 ERASE A:DIM A(25):DIM A$(25)
50 FOR I=0 TO 25
60 A(I)=I:A$(I)=CHR$(65+I)
70 NEXT
80 FOR I=0 TO 25
90 PRINT A(I);A$(I)
100 NEXT
110 END
RUN
```

33. DRAW

Formato: DRAW ⟨X$⟩

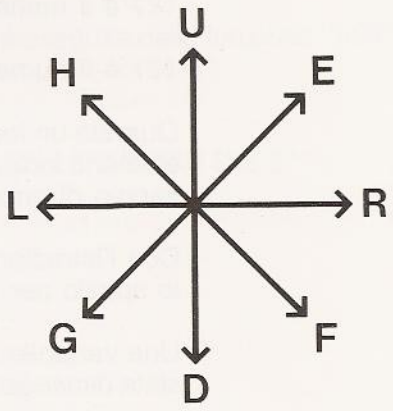
Scopo: Disegnare nel modo grafico 1 o 2

Categoria: Istruzione

Note: ⟨X$⟩ è una costante alfanumerica che può rappresentare uno dei seguenti subcomandi:

1. **Disegno di una linea**
   I seguenti subcomandi disegneranno una linea retta iniziando dalla posizione corrente del cursore nella direzione indicata:

   U⟨X⟩ = Nord
   D⟨X⟩ = Sud
   L⟨X⟩ = Ovest
   R⟨X⟩ = Est
   E⟨X⟩ = Nord-Est
   F⟨X⟩ = Sud-Est
   G⟨X⟩ = Sud-Ovest
   H⟨X⟩ = Nord-Ovest



   ⟨X⟩ indica la lunghezza della linea in termini di punti di immagine (pixels).

   Il successivo subcomando disegna una linea dalla posizione corrente del cursore ad un'altra posizione sullo schermo, indicata dalle coordinate ⟨X⟩ e ⟨Y⟩:
   M⟨X⟩,⟨Y⟩
   ⟨X⟩ rappresenta la coordinata X e ⟨Y⟩la coordinata Y.
   Quando la coordinata X è preceduta da un segno meno (-) o da un segno più (+) la coordinata X s'intende rispetto alla posizione del cursore.
   Quando la coordinata Y è preceduta da un segno meno (-) o da un segno più (+), la coordinata Y s'intende rispetto alla posizione del cursore.

   Dopo l'esecuzione dei subcomandi sopraindicati, il cursore si sarà spostato al termine della riga.

   I suddetti subcomandi possono essere preceduti dalla lettera "B". Ciò significa che il cursore si sposta al punto indicato senza disegnare nulla.

   Se è preceduto dalla lettera "N", il cursore ritorna al suo punto di partenza originale dopo aver disegnato la linea.

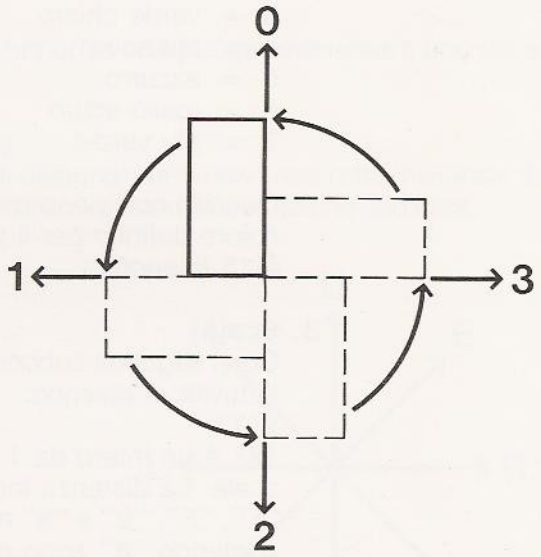
2. **Colore**
Con i seguenti subcomando si indica in quale colore verrà eseguita l'attività di disegno:
C⟨X⟩
⟨X⟩ è il numero del colore, che ha le seguenti corrispondenze:

0 = trasparente	8 = rosso
1 = nero	9 = rosso chiaro
2 = verde	10 = giallo scuro
3 = verde chiaro	11 = giallo chiaro
4 = blu scuro	12 = verde scuro
5 = azzurro	13 = magenta
6 = rosso scuro	14 = grigio
7 = blu verde	15 = bianco

Quando non viene usato il subcomando "C", verrà usato l'ultimo colore definito per il primo piano. Il valore standard per il colore è 15 (bianco).

3. **Scala**
Con i seguenti subcomandi si indica in quale scala verrà eseguita l'attività di disegno:
S⟨X⟩
⟨X⟩ è un intero da 1 a 255. ⟨X⟩ diviso per quattro è il fattore di scala. La distanza indicata nei subcomandi "U", "D", "L", "R", "E", "F", "G" e "H" nonché le coordinate relative X- e Y- nel subcomando "M" sono moltiplicate per il fattore di scala.
Quando non viene usato il subcomando "S", verrà usato l'ultimo fattore di scala definito. Il valore standard assunto per la scala è 4 (fattore di scala 1).

4. **Angolo**
   Con il seguente subcomando si indica sotto quale angolo verrà eseguita l'attività di disegno:
   `A<X>`
   `<X>` è un intero tra 0 e 3 ed ha il significato seguente:
   0 = 0 gradi
   1 = 90 gradi
   2 = 180 gradi
   3 = 270 gradi



   Quando non viene usato il subcomando "A", si presume l'ultimo angolo definito.
   Il valore standard per l'angolo è zero (0 gradi).

5. **Esecuzione**
   Le attività di disegno in una variabile alfanumerica possono essere eseguite con il subcomando:
   `X<Z$>;`
   Non dimenticare il punto e virgola!
   `<Z$>` è il nome della variabile contenente i subcomandi di disegno.

   Tutti i valori numerici nei subcomandi possono essere sostituiti da variabili numeriche. Le variabili numeriche possono essere precedute da un segno di uguale (=) e terminate da un punto e virgola (;), per esempio: `X1=40:DRAW "U=X1;"`

Esempio:

```
10 SCREEN 2
20 X1=80:X2=90
30 DRAW "BM=X1;,=X2;"
40 FOR I=0 TO 3
50 AB$="A"+STR$(I)+"S"+STR$(I*5+4)+"C1U20L10D20R10"
60 DRAW "XAB$;"
70 FOR J=0 TO 300:NEXT J
80 NEXT I
90 FOR J=0 TO 3000:NEXT J
100 DRAW "A0S4C15"
110 END
RUN
```

34. DSKF

Formato: DSKF(⟨X⟩)

Scopo: Dà la quantità di spazio libero su un dischetto nell'unità a dischetti specificata.

Categoria: Funzione

Note: ⟨X⟩ è il numero dell'unità a dischetti.

Esempio:

```
10 PRINT DSKF(1)
20 END
RUN
```

"DSKF" PUÒ ESSERE USATO SOLTANTO SE AL VG8000 SONO COLLEGATI MSX-DOS E L'UNITÀ A DISCHETTI 1 O 2.

35. END

Formato: END

Scopo: Interrompere l'esecuzione del programma

Categoria: Istruzione

Note: Quando viene eseguita l'istruzione "END" tutti i file aperti vengono chiusi e l'MSX-BASIC ritorna a livello comandi.

L'istruzione "END" può essere usata ovunque in un programma.

L'istruzione "END" è facoltativa come istruzione finale in un programma.

Esempio:

```
10 INPUT "NUMERO";K
20 IF K=0 THEN END
30 PRINT K*K
40 GOTO 10
RUN
```

36. EOF

Formato: EOF(⟨X⟩)

SCOPO: Controlla se è stata raggiunta la fine di un file sequenziale.

Categoria: Funzione

Note: ⟨X⟩ è il numero col quale il file è stato aperto con l'istruzione "OPEN".

Il file deve essere stato aperto nel modo INPUT.

Il risultato di questa funzione sarà -1 quando viene raggiunta la fine del file. In caso contrario il risultato sarà zero.

Esempio:

```
10 MAXFILES=1
20 OPEN "CAS:DEMO" FOR INPUT AS#1
30 IF EOF(1)=-1 GOTO 70
40 INPUT #1,A$
50 PRINT A$
60 GOTO 30
70 CLOSE #1
80 END
RUN
```

37. ERASE

Formato: ERASE ⟨variabile-1⟩[,⟨variabile-2⟩...]

Scopo: Eliminare da un programma le tabelle riservate

Categoria: Istruzione

Note: ⟨variabile-1⟩ e ⟨variabile-2⟩ sono i nomi delle variabili che sono state dimensionate con l'istruzione "DIM".

I nomi delle variabili usati in un programma devono essere per prima cosa cancellati con la funzione "ERASE" prima di poterli usare in un'istruzione "DIM".

Quando una variabile in un'istruzione "DIM" viene di nuovo dimensionata senza aver usato precedentemente l'istruzione "ERASE" compare il messaggio di errore: "Redimensioned array" (Matrice ridimensionata).

Esempio:

```
10 DIM A(15)
20 DIM B(13)
30 ERASE A,B
40 DIM A(80)
50 DIM B(20)
60 END
RUN
```

38. ERL

Formato: ERL

Scopo: Dà il numero della riga di programma in cui MSX-BASIC ha scoperto un errore.

Categoria: Funzione

Note: In una routine di gestione degli errori iniziata dalla istruzione "ON ERROR GOTO", la funzione "ERL" dà la riga di programma che ha attivato la routine di gestione degli errori.

Quando viene scoperto un errore, lavorando nel modo diretto, la funzione "ERL" dà il valore di 65535.

Esempio:

```
10 ON ERROR GOTO 50
20 A=25:PRINT A
30 B=A/0
40 END
50 PRINT ERL
60 RESUME NEXT
RUN
```

39. ERR

Formato: ERR

Scopo: Dà il codice di errore pertinente quando MSX-BASIC scopre un errore.

Categoria: Funzione

Note: In una routine di gestione degli errori attivata dall'istruzione "ON ERROR GOTO", la funzione "ERR" dà il codice dell'errore che ha attivato la routine di gestione degli errori.

Esempio:

```
10 ON ERROR GOTO 50
20 A=25:PRINT A
30 B=A/0
40 END
50 PRINT ERR
60 RESUME NEXT
RUN
```

40. ERROR

Formato: ERROR ⟨X⟩

Scopo: Definire i propri codici di errore e simulare errori.

Categoria: Istruzione

Note: ⟨X⟩ è un codice di errore. Deve essere un intero compreso fra 0 e 255.

Se ⟨X⟩ è un codice di errore esistente, l'istruzione "ERROR" simulerà quel particolare errore e produrrà il pertinente messaggio di errore.

È possibile usare i codici di errore da 60 a 255 per definire propri codici di errore. I codici di errore così definiti possono essere usati per la ricerca e la correzione degli errori di programma.

Se ⟨X⟩ è un codice per il quale non è stato definito alcun messaggio di errore, compare il messaggio "Unprintable error" (Errore non stampabile).

Esempio:

```
10 ON ERROR GOTO 50
20 INPUT "TESTO";A$
30 IF LEN(A$)>5 THEN ERROR 250
40 END
50 IF ERR=250 THEN PRINT "TESTO TROPPO LUNGO":RESUME 20
60 ON ERROR GOTO 0
70 END
RUN
```

41. EXP

Formato: EXP(⟨X⟩)

Scopo: Dà l'esponente cui va elevato ⟨X⟩

Categoria: Funzione

Note: ⟨X⟩ deve essere minore di o uguale a 145.06286058562

Esempio:

```
10 FOR I=1 TO 4
20 PRINT USING "##.####";EXP(I);LOG(EXP(I))
30 NEXT
40 END
RUN
```

42. FIX

Formato: FIX(⟨X⟩)

Scopo: Dà la parte intera di ⟨X⟩

Note: Questa funzione è identica a:
SGN(⟨X⟩)*INT(ABS(⟨X⟩))

La differenza rispetto alla funzione "INT" è che con "FIX" i valori negativi non sono arrotondati al successivo intero inferiore.

Esempio:
```
10 A=-1.7345
20 B=FIX(A):PRINT B
30 B=INT(A):PRINT B
40 END
RUN
```

43. FIELD

Formato: FIELD [#]⟨X⟩,⟨Y⟩ AS ⟨Y$⟩[,⟨Z⟩ AS ⟨Z$⟩...]

Scopo: Dividere il buffer per un file casuale su dischetto

Categoria: Istruzione

Note: Prima di poter usare un'istruzione "GET" o "PUT" in un programma, occorre eseguire l'istruzione "FIELD".

Prima di poter usare un'istruzione "FIELD", il file deve essere aperto.

⟨X⟩ è il numero col quale il file è stato aperto con l'istruzione "OPEN".

⟨Y⟩ e ⟨Z⟩ indicano il numero di caratteri per ciascun elemento nel buffer.

⟨Y$⟩ e ⟨Z$⟩ corrispondono ai nomi degli elementi nel buffer.

Il numero totale di caratteri nel buffer non può superare i 256.

L'istruzione "FIELD" non inserisce alcuna informazione nel buffer. A questo scopo vanno usate le istruzioni "LSET" e "RSET".

Per ciascun file casuale, possono essere usate parecchie istruzioni "FIELD" per ridividere il buffer.

Esempio:

```
10 MAXFILES=1
20 OPEN "A:TEST" AS #1
30 FIELD #1,2 AS N1$,4 AS N2$,8 AS N3$,20 AS N4$
40 INPUT "AX";AX
50 INPUT "B!";B!
60 INPUT "C#";C#
70 INPUT "D$";D$
80 RSET N1$=MKI$(AX):RSET N2$=MKS$(B!):RSET
N3$=MKD(C#):LSET N4$=D$
90 PUT #1,1
100 AX=0:B!=0:C#=0:D$=""
110 PRINT AX;B!;C#;D$
120 GET #1,1
130 AX=CVI(N1$):B!=CVS(N2$):C#=CVD(N3$):D$=N4$
140 PRINT AX;B!;C#;D$
150 CLOSE #1
160 END
RUN
```

"FIELD" PUÒ ESSERE USATO SOLTANTO SE AL VG8000 SONO STATI COLLEGATI MSX-DOS E L'UNITÀ A DISCHETTI 1 O 2.

44. FILES

Formato: FILES ["⟨periferica⟩:⟨nome file⟩"]

Scopo: Fornire l'indice di un dischetto

Categoria: Comando

Note: ⟨periferica⟩ può essere:
A = unità a dischetti 1
B = unità a dischetti 2

Se vengono omessi ⟨periferica⟩ e ⟨nome file⟩, tutti i file sul dischetto nell'unità correntemente selezionata saranno proiettati sullo schermo.

Quando vengono indicati ⟨periferica⟩ e ⟨nome file⟩, il nome del file sarà proiettato sullo schermo se quel file viene trovato sul dischetto. In caso contrario, compare in sua vece il messaggio "File not found" (file non trovato).

Esempio:

```
FILES "A:DEMO"
```

"FILES" PUÒ ESSERE USATO SOLTANTO SE AL VG8000 SONO STATI COLLEGATI MSX-DOS E L'UNITÀ A DISCHETTI 1 O 2.

45. FOR-NEXT

Formato:

```
FOR ⟨XX⟩=⟨X⟩ TO ⟨Y⟩ [STEP Z]
_
_
_
NEXT [⟨XX⟩[,⟨YY⟩]...]
```

Scopo: Eseguire una serie di istruzioni per un certo numero di volte in una iterazione.

Categoria: Istruzione

Note: ⟨XX⟩ e ⟨YY⟩ sono variabili numeriche.

⟨XX⟩ serve da contatore e ⟨X⟩ è il primo valore del contatore mentre ⟨Y⟩è l'ultimo valore del contatore.

Le righe di programma, che seguono l'istruzione "FOR" saranno eseguite fino all'istruzione "NEXT". Il contatore viene quindi incrementato del numero specificato in ⟨Z⟩. MSX-BASIC si accerterà che il valore del contatore non sia più elevato del valore finale ⟨Y⟩. In caso contrario, MSX-BASIC ritorna all'istruzione che segue "FOR" e l'intero processo viene ripetuto. Se risulta più alto, MSX-BASIC salta all'istruzione che segue "NEXT".

Se non viene indicato "STEP", ogni volta il contatore viene incrementato di 1.

Se il valore di "STEP" è negativo, il contatore è ridotto del valore di "STEP" e l'iterazione viene eseguita fino a che il contatore è minore del valore finale.

L'iterazione verrà eseguita almeno una volta quando il valore iniziale del contatore moltiplicato per il segno del valore "STEP" è maggiore del valore finale del contatore moltiplicato per il segno del valore "STEP".

Le iterazioni "FOR-NEXT" possono essere disposte all'interno di un'altra iterazione "FOR-NEXT". In tal caso, ogni iterazione "FOR-NEXT" deve avere come contatore una variabile numerica esclusiva.
L'istruzione "NEXT" per l'iterazione interna deve essere inserita prima dell'istruzione "NEXT" dell'iterazione esterna.
Quando le iterazioni interna ed esterna hanno lo stesso punto finale, la variabile può essere inserita all'interno di un'istruzione "NEXT".

Quando non vengono indicate variabili con un'istruzione "NEXT", questa si applicherà al "FOR" indicato più di recente.

Quando deve essere eseguita l'istruzione "NEXT" senza un precedente "FOR", sullo schermo compare il messaggio di errore "Next without for" (Next senza For).

Esempio:

```
10 FOR I=0 TO 4
20 PRINT I;
30 NEXT:PRINT
40 FOR I=0 TO 5 STEP 5
50 FOR J=I TO I+2
60 PRINT J;
70 NEXT J,I:PRINT
80 END
RUN
```

46. FRE

Formati:

```
FRE(0)
FRE("")
```

Scopo: Trovare quanti byte di memoria non sono stati ancora usati da MSX-BASIC.

Categoria: Funzione

Note: "FRE(0)" dà il numero di byte della memoria che non sono ancora stati usati da MSX-BASIC.

"FRE("")" dà il numero di byte della memoria che sono ancora disponibili per le variabili alfanumeriche.
Questo numero può essere modificato con l'istruzione "CLEAR".

Esempio:

```
10 CLEAR
20 PRINT FRE(0);FRE("")
30 END
RUN
```

47. GET

Formato:

```
GET [#]⟨X⟩[,⟨Y⟩]
```

Scopo: Leggere nel buffer un record da un file casuale su dischetto.

Categoria: Istruzione

Note: ⟨X⟩ è il numero con il quale il file è stato aperto con l'istruzione "OPEN".

⟨Y⟩ è il numero del record che deve essere letto. Deve essere un intero compreso tra 0 e 32767. Quando ⟨Y⟩ non viene indicato, verrà letto il successivo record, il che significa il record che segue l'ultimo record che è stato letto o scritto.

Il buffer nella memoria deve essere risevato con l'istruzione "FIELD".

Usare l'istruzione "PUT" per scrivere su un dischetto un record in un file casuale dal buffer.

Esempio: Vedere "FIELD".

"GET" PUÒ ESSERE USATO SOLTANTO SE AL VG8000 SONO STATI COLLEGATI MSX-DOS E L'UNITÀ A DISCHETTI 1 O 2.

48. GOSUB-RETURN

Formato:

```
GOSUB <X>
-
-
RETURN [<Y>]
```

Scopo: Saltare ad una subroutine e dalla subroutine ritornare di nuovo al programma principale.

Categoria: Istruzione

Note: <X> è il numero della prima riga di programma della subroutine.

In un programma, una subroutine può essere richiamata un qualsiasi numero di volte.

Una subroutine può anche essere richiamata dall'interno di una altra subroutine.

L'istruzione "RETURN" fa sì che MSX-BASIC ritorni all'istruzione che segue l'istruzione "GOSUB" più recente. Quando un'istruzione "RETURN" è seguita da un numero di riga (<Y>), MSX-BASIC ritorna al numero di riga indicato.

Le subroutine possono essere usate ovunque nel programma. È consigliabile inserire davanti al programma principale le subroutine che vengono richiamate più spesso in quel programma. Per contro, le subroutine che non vengono richiamate spesso, possono essere più utilmente disposte dopo il programma principale.

Esempio:

```
10 GOSUB 90
20 PRINT "RIGA 20"
30 GOSUB 90
40 PRINT "RIGA 40"
50 I=1:GOSUB 90
60 PRINT "RIGA 60"
70 PRINT "END"
80 END
90 PRINT "I=";I;
100 PRINT "SUBROUTINE"
110 IF I=1 THEN RETURN 70
120 RETURN
RUN
```

49. GOTO

Formato: GOTO ⟨X⟩

Scopo: Saltare ad una data riga di programma

Categoria: Istruzione

Note: ⟨X⟩ rappresenta un numero di riga di programma. L'esecuzione del programma continua con il numero di riga indicato.

Esempio:

```
10 GOTO 40
20 PRINT "SUBROUTINE"
30 RETURN
40 PRINT "PROGRAMMA PRINCIPALE"
50 GOSUB 20
60 END
RUN
```

50. HEX$

Formato: HEX$(⟨X⟩)

Scopo: Dà una rappresentazione esadecimale del numero decimale ⟨X⟩

Categoria: Funzione

Note: Il risultato di questa funzione è un valore alfanumerico.

⟨X⟩ deve essere un intero, con un valore da -32768 a 65535.

Se ⟨X⟩ è un numero negativo, viene usata la forma dei complementi a due. Ciò significa che HEX$(-1) = HEX$(65536-1).

Esempio:

```
10 A=63
20 A$=HEX$(A)
30 PRINT A;A$
40 END
RUN
```

51. IF-THEN-ELSE

Formati:

```
IF (espressione) THEN (istruzione)
 [ELSE (istruzione)]
IF (espressione) GOTO (X)
 [ELSE (istruzione)]
```

Scopo: Prendere una decisione riguardante l'esecuzione di un programma sulla base del risultato ritornato da un'espressione.

Categoria: Istruzione

Note: Se il risultato dell'espressione è vero, le istruzioni che seguono la parola "THEN" o l'istruzione "GOTO" verranno eseguite.
Se, per contro, il risultato dell'espressione non è vero, le istruzioni che seguono la parola "ELSE" verranno eseguite. Se "ELSE" non viene indicato, l'esecuzione continua con la riga di programma successiva.

Se la parola "THEN" è seguita da un'istruzione "GOTO", la parola "GOTO" può essere omessa in quanto basterà il numero di riga pertinente.

In un'espressione, il contenuto di una variabile può essere confrontato con una costante o con il contenuto di un'altra variabile.

Sono possibili i seguenti confronti:

=	UGUALE A
<>	NON UGUALE A
<	MINORE DI
>	MAGGIORE DI
<=	MINORE DI O UGUALE A
>=	MAGGIORE DI O UGUALE A

I confronti possono essere invertiti usando la parola "NOT".

Possono essere inoltre inseriti diversi confronti all'interno di una espressione usando le parole "OR", "AND", "XOR", "EQV" o "IMP"

Un'istruzione "IF-THEN-ELSE", usata in combinazione con l'istruzione "GOTO" nel modo diretto, si traduce in un messaggio di errore "Undefined line number" (Numero riga indefinito). Lo stesso succede quando il numero di riga non esiste nel modo indiretto.

Un'istruzione "IF-THEN-ELSE" può essere usata nell'ambito di una altra istruzione "IF-THEN-ELSE".

Esempio:

```
10 INPUT "NUMERO (5-25)";I
20 IF I<5 OR I>25 THEN 70 ELSE IF I<20 AND I>10 GOTO 50
30 PRINT "IL NUMERO E' 5-10 OR 20-25"
40 GOTO 10
50 PRINT "IL NUMERO E' 10-20"
60 GOTO 10
70 PRINT "IL NUMERO NON E' CORRETTO"
80 GOTO 10
RUN
```

52. INKEY$

Formato: INKEY$

Scopo: Dà il carattere corrispondente al tasto premuto sulla tastiera.

Categoria: Funzione

Note: Il carattere che è stato impostato non compare sullo schermo.

Se non viene premuto alcun tasto, viene indicato il carattere cui corrisponde il codice zero.

Tutti i caratteri sono passati al programma a meno che non vengano premuti simultaneamente i tasti CTRL e STOP. In tal caso il programma termina.

Esempio:

```
10 REM USA I TASTI CONTROLLO CURSORE
20 SCREEN 2:COLOUR 1,4,15:X=127:Y=95
30 I$=INKEY$:IF I$="" THEN 30
40 B=ASC(I$)
50 IF B<28 OR B>31 THEN 30
60 A=B-27:ON A GOTO 70,90,110,130
70 X=X+1:IF X>255 THEN X=255
80 GOTO 140
90 X=X-1:IF X<0 THEN X=0
100 GOTO 140
110 Y=Y-1:IF Y<0 THEN Y=0
120 GOTO 140
130 Y=Y+1:IF Y>191 THEN Y=191
140 PSET (X,Y)
150 GOTO 30
RUN
```

53. INP

Formato: INP(<X>)

Scopo: Dà un byte, letto da un connettore di ingresso <X> della macchina

Categoria: Funzione

Nota: <X> deve essere un intero da zero a 255.

Usare l'istruzione "OUT" per trasmettere un byte ad un connettore di uscita della macchina.

Esempio:

```
10 A=INP(&HA8)
20 A$="00000000"+BIN$(A)
30 PRINT RIGHT$(A$,8)
40 END
RUN
```

54. INPUT

Formato:

```
INPUT [<domanda>;]<variabile>,<variabile>...
INPUT #<X>,<variabile>[,<variabile>...]
```

Scopo: Aggiungere un valore ad una variabile attraverso la tastiera o leggendolo da un file sequenziale.

Categoria: Istruzione

Note: Il primo formato è usato per consentire all'utente di immettere informazioni attraverso la tastiera durante l'esecuzione di un programma.

Eseguendo un'istruzione "INPUT", il programma si interrompe. Sullo schermo compare un punto interrogativo per indicare che l'utente deve immettere i dati tramite la tastiera.

Se è stata inserita una <domanda>, la frase di domanda precederà automaticamente il punto interrogativo.
Possono essere usate sia variabili numeriche che alfanumeriche.
Le informazioni fornite dall'utente sono inserite nelle variabili dall'istruzione "INPUT". Il numero di elementi di informazione deve essere uguale al numero delle variabili. Le virgole devono essere aggiunte come segni di separazione. Quando l'utente immette più informazioni di quelle ammesse dal numero delle variabili, sullo schermo compare il messaggio di errore "? Extra ignored" (Ignorati gli extra). Se l'utente imette meno elementi del numero di variabili, sullo schermo compaiono due punti interrogativi ed il computer attende fino a che non viene battuta l'informazione mancante.

La natura dell'informazione (numerica o alfanumerica) deve corrispondere al tipo di variabile. In caso contrario, sullo schermo compare il messaggio "?Redo from start" (Rifare da capo) e l'istruzione "INPUT" deve essere eseguita di nuovo.
Le variabili accettano soltanto le informazioni immesse dall'utente quando al punto di domanda è stato risposto correttamente.

Il secondo formato dell'istruzione "INPUT" è usato per leggere le informazioni da un file sequenziale ed immetterle in una variabile.

<X> è il numero col quale il file è stato aperto con l'istruzione "OPEN".

Il file deve essere stato aperto nel modo input.

Le informazioni che vi vengono lette sono inserite in ogni variabile.

Per scrivere informazioni in un file sequenziale, viene usata l'istruzione "PRINT#".

L'istruzione "INPUT" non può essere usata nel modo diretto.

Esempio:

```
10 INPUT "A E B";A,B
20 PRINT A+B
30 END
RUN
```

55. INPUT$

Formati:

```
INPUT$(<X>)
INPUT$(<X>),[#]<Y>)
```

Scopo: Dà un valore alfanumerico degli <X> caratteri che sono stati immessi dalla tastiera o sono stati letti da un file sequenziale.

Categoria: Funzione

Note: Il primo formato è usato per i caratteri che sono immessi attraverso la tastiera.

<X> è il numero dei caratteri.

I caratteri che vengono immessi non compaiono sullo schermo.

Quando vengono premuti simultaneamente i tasti CTRL e STOP, il programma viene terminato.

Il secondo formato è usato per leggere i caratteri da un file sequenziale.

<Y> è il numero col quale il file è stato aperto con l'istruzione "OPEN". Il file deve essere stato aperto nel modo input.

Esempio:

```
10 REM IL CODICE E' MSX + TASTO SELECT
20 B$="MSX"+CHR$(24)
30 PRINT "CODICE";
40 A$=INPUT$(4):PRINT
50 IF A$<>B$ THEN PRINT "IL CODICE E' SBAGLIATO":GOTO 30
60 BEEP
70 PRINT "BENVENUTO NEL MONDO DI MSX"
80 END
RUN
```

56. INSTR

Formato: INSTR([⟨X⟩,]⟨Y$⟩,⟨Z$⟩)

Scopo: Dà la posizione in cui si trova la combinazione di caratteri ⟨Z$⟩ nella variabile ⟨Y$⟩.

Categoria: Funzione

Note: Quando viene indicato ⟨X⟩, la ricerca della combinazione di caratteri indicata in ⟨Z$⟩ inizia alla posizione ⟨X⟩ nella variabile ⟨Y$⟩.

⟨X⟩ è un numero intero da 0 a 255.

Se la combinazione di caratteri indicata non viene trovata, la funzione dà come risultato zero.

Esempio:

```
10 A$="ABCDEABCDE"
20 FOR I=1 TO 15 STEP 5
30 PRINT INSTR(I,A$,"BCD")
40 NEXT
50 END
RUN
```

57. INT

Formato: INT(⟨X⟩)

Scopo: Dà l'intero più piccolo di ⟨X⟩ o uguale a ⟨X⟩

Categoria: Funzione

Note: Nessuna

Esempio:

```
10 A=-2.7
20 PRINT INT(A)
30 A=2.7
40 PRINT INT(A)
50 END
RUN
```

58. INTERVAL/ON/OFF/STOP

Formati:

```
INTERVAL ON
INTERVAL OFF
INTERVAL STOP
```

Scopo: Attivare il controllo di un intervallo di tempo trascorso.

Categoria: Istruzione

Note: Con l'istruzione "`ON INTERVAL GOSUB`" deve essere indicato l'intervallo di tempo.

Dopo l'istruzione "`INTERVAL ON`", MSX-BASIC controllerà con ogni istruzione se è trascorso un intervallo di tempo. In caso affermativo, MSX-BASIC eseguirà una subroutine come indicato con l'istruzione "`ON INTERVAL GOSUB`".

Dopo l'istruzione "`INTERVAL OFF`", MSX-BASIC non controlla più se è trascorso un intervallo di tempo.

Dopo l'istruzione "`INTERVAL STOP`", MSX-BASIC controlla se è trascorso un intervallo di tempo ma non esegue la subroutine indicata con l'istruzione "`ON INTERVAL GOSUB`". Esso ricorda comunque se è trascorso un intervallo di tempo. In tal caso la subroutine indicata verrà eseguita immediatamente dopo aver impartito l'istruzione "`INTERVAL ON`".

Esempio: Vedere "`ON INTERVAL GOSUB`".

59. KEY

Formato: KEY ⟨X⟩,⟨X$⟩

Scopo: Attribuire una funzione specifica ad uno dei dieci tasti di funzione.

Categoria: Istruzione

Note: ⟨X⟩ è il numero del tasto di funzione (un numero da 0 a 10).

⟨X$⟩ è il testo da aggiungere al tasto di funzione indicato (lunghezza massima 15 caratteri).

Esempio:

```
10 A$="LOAD"
20 KEY 1,A$+CHR$(13)
30 END
RUN
```

60. KEY LIST

Formato: KEY LIST

Scopo: Ottenere un listato sullo schermo dei testi aggiunti ai 10 tasti di funzione.

Categoria: Istruzione

Note: Il primo testo che compare corrisponde al tasto di funzione F1, il secondo al tasto di funzione F2, ecc.

I caratteri di controllo sono battuti come spazi.

Esempio:

```
10 KEY LIST
20 END
RUN
```

61. KEY ON/OFF

Formati:

```
KEY ON
KEY OFF
```

Scopo: Attivare e disattivare la visualizzazione dei tasti di funzione sulla 24ª riga dello schermo nei modi testo 1 e 2.

Categoria: Istruzione

Note: Con l'istruzione "KEY OFF" i testi dei tasti di funzione non compaiono più sullo schermo.

L'istruzione "KEY ON" li fa ricomparire di nuovo.

Esempio:

```
10 FOR I=1 TO 5
20 KEY OFF
30 FOR J=1 TO 300:NEXT
40 KEY ON
50 FOR J=1 TO 300:NEXT
60 NEXT I
70 END
RUN
```

62. KEY(X) ON/OFF/STOP

Formati:

```
KEY(<X>) ON
KEY(<X>) OFF
KEY(<X>) STOP
```

Scopo: Controllare che sia stato premuto un tasto di funzione

Categoria: Istruzione

Note: <X> è il numero del tasto di funzione (1-10).

Dopo l'istruzione "KEY(X)ON", MSX-BASIC controlla in ogni istruzione se è stato premuto il tasto di funzione indicato. In caso affermativo, MSX-BASIC esegue la subroutine indicata con l'istruzione "ON KEY GOSUB".

Dopo l'istruzione "KEY(X) OFF", MSX-BASIC non controlla più se è stato premuto il tasto di funzione indicato.

Dopo l'istruzione "KEY(X) STOP", MSX-BASIC controlla se il dato tasto di funzione è stato premuto ma non esegue la subroutine indicata con l'istruzione "ON KEY GOSUB". Esso ricorda comunque se è stato premuto il tasto di funzione indicato. Iin tal caso la subroutine data verrà eseguita immediatamente dopo aver impartito l'istruzione "KEY(X) ON".

Esempio: Vedere "ON KEY GOSUB".

63. KILL

Formato: KILL "⟨periferica⟩:⟨nome file⟩"

Scopo: Rimuovere un file da un dischetto

Categoria: Istruzione

Note: ⟨periferica⟩ può essere:
A = unità a dischetti 1
B = unità a dischetti 2

⟨periferica⟩ è l'unità in cui è inserito il dischetto contenente il file che deve essere rimosso.

Esempio:
```
KILL "A:DEMO"
```

"KILL" PUÒ ESSERE USATO SOLTANTO SE AL VG8000 SONO STATI COLLEGATI MSX-DOS E L'UNITÀ A DISCHETTI 1 O 2.

64. LEFT$

Formato: LEFT$(⟨X$⟩,⟨X⟩)

Scopo: Dà un'espressione alfanumerica composta dai caratteri ⟨X⟩ più a sinistra di ⟨X$⟩.

Categoria: Funzione

Note: ⟨X⟩ è un numero intero da zero a 255.

Quando ⟨X⟩ è maggiore della lunghezza di ⟨X$⟩, verrà indicato il contenuto completo di ⟨X$⟩.

Quando ⟨X⟩ è zero, questa funzione darà un'espressione alfanumerica vuota.

Esempio:
```
10 A$="BASIC"
20 FOR I=1 TO LEN(A$)
30 PRINT LEFT$(A$,I)
40 NEXT
50 END
RUN
```

65. LEN

Formato: LEN(⟨X$⟩)

Scopo: Dà il numero di caratteri in ⟨X$⟩

Categoria: Funzione

Note: Vengono contati anche gli spazi ed i codici dei caratteri da 1 a 31.

Esempio:

```
10 A$="BASIC"+CHR$(13)
20 PRINT LEN(A$)
30 END
RUN
```

66. LET

Formati:

```
[LET] ⟨X⟩=⟨Y⟩
[LET] ⟨X$⟩=⟨Y$⟩
```

Scopo: Assegnare un valore ad una variabile

Categoria: Istruzione

Note: La parola "LET" è facoltativa

Esempio:

```
10 LET A=10
20 LET B=A
30 LET C$="HELLO"
40 PRINT A;B;C$
50 A=10:B=A:C$="HELLO"
60 PRINT A;B;C$
70 END
RUN
```

67. LINE

Formato: LINE [[STEP](⟨X1⟩,⟨Y1⟩)]-[STEP](⟨X2⟩,⟨Y2⟩)[,⟨Z⟩][,B[F]]

Scopo: Disegnare una linea nel modo grafico 1 o 2

Categoria: Istruzione

Note: ⟨X1⟩ è la coordinata X del punto di inizio della linea e deve essere un intero tra 0 e 255.

⟨Y1⟩ è la coordinata Y del punto di inizio della linea e deve essere un intero tra 0 e 191.

Quando viene usata la parola "STEP", i valori ⟨X1⟩ e ⟨Y1⟩ sono interpretati rispetto alla posizione del cursore. In questo caso ⟨X1⟩ e ⟨Y1⟩ possono anche essere interi negativi.

Se ⟨X1⟩, ⟨Y1⟩ o entrambi vengono omessi, vengono usate la coordinata corrente X, la coordinata corrente Y o entrambe.

⟨X2⟩ e ⟨Y2⟩ sono le coordinate X e Y del punto finale della linea. Anche in questo caso, le coordinate saranno relative all'ultima posizione del cursore quando viene usata la parola "STEP".

⟨Z⟩ è il numero del colore in cui verrà disegnata la linea. Deve essere un intero da 0 a 15 ed ha il seguente significato:

0	= trasparente	8	= rosso
1	= nero	9	= rosso chiaro
2	= verde	10	= giallo scuro
3	= verde chiaro	11	= giallo chiaro
4	= blu scuro	12	= verde scuro
5	= azzurro	13	= magenta
6	= rosso scuro	14	= grigio
7	= blu verde	15	= bianco

Quando non viene indicato alcun valore ⟨Z⟩, verrà usato di nuovo l'ultimo colore di primo piano precedentemente utilizzato. Il valore assunto standard di ⟨Z⟩ è 15.

Quando viene indicato "B" verrà disegnato un rettangolo di cui la linea data è la diagonale.

Quando viene usato "F" dopo "B", il rettangolo sarà riempito con il colore indicato.

Esempio:

```
10 SCREEN 2
20 FOR I=0 TO 95 STEP 2
30 LINE (128-I,95-I)-(128+I,95+I),1,B
40 NEXT
50 FOR K=0 TO 2000:NEXT
60 END
RUN
```

68. LINE INPUT

Formati:

```
LINE INPUT [(domanda);](X$)
LINE INPUT #(X),(X$)
```

Scopo: Aggiungere un valore ad una variabile alfanumerica che viene immessa attraverso la tastiera o che viene letta da un file sequenziale.

Categoria: Istruzione

Note: Il primo formato è usato per dare all'utente la possibilità di immettere una riga completa (massimo 254 caratteri) attraverso la tastiera, durante l'esecuzione del programma.

Eseguendo tale istruzione "LINE INPUT", il programma si interrompe. Non compare alcun punto interrogativo sullo schermo per indicare che l'utente deve immettere ulteriori informazioni.

Se viene indicato (domanda), questa compare sullo schermo.

Il secondo formato dell'istruzione "LINE INPUT" è usato per leggere un record completo da un file sequenziale e di aggiungerlo ad una data variabile.

(X) è il numero col quale il file è stato aperto con l'istruzione "OPEN". Il file deve essere aperto nel modo input.

Per leggere soltanto un elemento da un file, anziché un record completo, occorre usare l'istruzione "INPUT#".

L'istruzione "LINE INPUT#" è spesso usata per leggere un programma MSX-BASIC che è stato salvato, con l'istruzione "SAVE", come file di dati in un altro programma.

Esempio:

```
10 LINE INPUT "TESTO";A$
20 PRINT A$
30 END
RUN
```

69. LIST

Formato: LIST [⟨X⟩[-[⟨Y⟩]]]

Scopo: Visualizzare sullo schermo un programma completo o parte di esso, caricato nella memoria del computer.

Categoria: Comando

Note: ⟨X⟩ e ⟨Y⟩ sono numeri di righe di programma.

Quando vengono indicati ⟨X⟩ e ⟨Y⟩, sullo schermo compariranno tutte le righe da ⟨X⟩ a ⟨Y⟩.

Quando viene indicato soltanto ⟨Y⟩, compariranno sullo schermo tutte le righe iniziando dalla prima fino alla riga ⟨Y⟩.

Quando viene indicato soltanto ⟨X⟩, comparirà soltanto la riga ⟨X⟩

Quando non vengono indicati nè ⟨X⟩ nè ⟨Y⟩, sullo schermo comparirà l'intero programma.

Quando vengono indicati ⟨X⟩ e "-", sullo schermo compariranno la riga ⟨X⟩ e tutte le righe di programma seguenti.

Esempio:

```
LIST 10
```
(visualizza la riga di programma 10)

```
LIST 50-100
```
(visualizza le righe di programma da 50 a 100)

```
LIST -100
```
(visualizza tutte le righe di programma dalla prima fino alla 100)

```
LIST
```
(visualizza tutte le righe del programma)

70. LLIST

Formato: LLIST [⟨X⟩[-[⟨Y⟩]]]

Scopo: Stampare su carta un programma completo o parte di esso utilizzando una stampante.

Categoria: Comando

Note: Questo comando funziona allo stesso modo del comando "LIST".

Esempio: Vedere "LIST".

"LLIST" PUÒ ESSERE USATO SOLTANTO SE AL VG8000 È STATA COLLEGATA UNA STAMPANTE.

71. LOAD

Formato: LOAD "⟨periferica⟩:⟨nome programma⟩"[,R]

Scopo: Caricare un programma da cassetta o da dischetto nella memoria del computer.

Categoria: Comando

Note: ⟨periferica⟩ può essere:
CAS = cassetta
A = unità a dischetti 1
B = unità a dischetti 2

Il programma deve essere scritto sulla cassetta o sul dischetto in formato ASCII con il comando "SAVE".

Il nome del programma è una costante alfanumerica specificata quando il programma è stato scritto sulla cassetta o sul dischetto con il comando "SAVE".

Quando viene indicata l'opzione "R", il programma verrà immediatamente eseguito dopo che è stato caricato.

Esempio: LOAD "CAS:DEMO"

"LOAD CAS" PUÒ ESSERE USATO SOLTANTO SE AL VG8000 È COLLEGATO UN REGISTRATORE DATI.

"LOAD A" O "LOAD B" PUÒ ESSERE USATO SOLTANTO SE AL VG8000 SONO STATI COLLEGATI MSX-DOS O L'UNITÀ A DISCHETTI 1 O 2.

72. LOC

Formato: LOC(⟨X⟩)

Scopo: Dà la locazione corrente nel file indicato.

Categoria: Funzione

Nota: Prima che venga usata la funzione "LOC", il file deve essere stato aperto.

⟨X⟩ è il numero del file con il quale il file è stato aperto con l'istruzione "OPEN".

Questa funzione dà il numero dell'ultimo record letto o scritto nei file casuali. Nei file sequenziali, questa funzione dà il numero di settori (blocchi da 256 byte) letti o scritti sul file dal momento in cui è stato aperto.

Esempio:

```
10 MAXFILES=1
20 OPEN "A:TEST" AS #1
30 FIELD #1,2 AS N1$,4 AS N2$,8 AS N3$,20 AS N4$
40 GET #1,1
50 PRINT LOC(1)
60 CLOSE #1
RUN
```

"LOC" PUÒ ESSERE USATO SOLTANTO SE AL VG8000 SONO STATI COLLEGATI MSX-DOS E L'UNITÀ A DISCHETTI 1 O 2.

73. LOCATE

Formato: LOCATE [⟨X⟩][,⟨Y⟩][,⟨Z⟩]

Scopo: Inviare il cursore ad una data posizione sullo schermo nei modi testo 1 e 2.

Categoria: Istruzione

Note: ⟨X⟩ è la coordinata X e deve essere un numero intero da 0 a 39. Quando ⟨X⟩ non viene indicato, viene usata la coordinata X della posizione corrente del cursore.

⟨Y⟩ è la coordinata Y e deve essere un numero intero da 0 a 23. Quando ⟨Y⟩ non viene indicato, viene usata la coordinata Y della posizione corrente del cursore.

⟨Z⟩ è l'interruttore di visualizzazione del cursore e deve essere zero o 1.
Zero significa che il cursore diventa visibile sullo schermo mentre 1 significa che il cursore sarà invisibile.
Quando ⟨Z⟩ non viene indicato, viene assunto di nuovo l'ultimo valore usato per ⟨Z⟩.
Il valore standard di ⟨Z⟩ è zero.

Esempio:

```
5 SCREEN 0
10 WIDTH 36:CLS
20 FOR I=0 TO 10
30 X=INT(RND(1)*36)
40 Y=INT(RND(1)*23)
50 LOCATE X,Y:PRINT "*"
60 LOCATE 10,23:PRINT "X=";X;"Y=";Y;
70 FOR J=0 TO 1000:NEXT
80 NEXT I
90 CLS
100 END
RUN
```

74. LOF

Formato: LOF(⟨X⟩)

Scopo: Dà la lunghezza di un dato file in termini di numero di byte.

Categoria: Funzione

Note: Prima di poter usare la funzione "LOF", il file pertinente deve essere stato aperto.

⟨X⟩ è il numero col quale il file è stato aperto con l'istruzione "OPEN".

Esempio:

```
10 MAXFILES=1
20 OPEN "A:TEST" FOR INPUT AS #1
30 PRINT LOF(1)
40 CLOSE #1
50 END
RUN
```

"LOF" PUÒ ESSERE USATO SOLTANTO SE AL VG8000 SONO STATI COLLEGATI MSX-DOS E L'UNITÀ A DISCHETTI 1 O 2.

75. LOG

Formato: LOG(⟨X⟩)

Scopo: Dà il logaritmo logico di ⟨X⟩

Categoria: Funzione

Note: ⟨X⟩ deve essere maggiore di zero.

Esempio:

```
10 FOR I=1 TO 4
20 PRINT LOG(I)
30 NEXT
40 END
RUN
```

76. LPOS

Formato: LPOS(0)

Scopo: Dà un'indicazione della posizione di stampa nel buffer di stampa.

Categoria: Funzione

Note: Il risultato di questa funzione non dà necessariamente la posizione fisica della testina di stampa.

Esempio:

```
10 FOR I=1 TO 1000
20 LPRINT "HELLO";
30 IF LPOS)38 THEN LPRINT CHR$(13)
40 NEXT
50 END
RUN
```

"LPOS" PUÒ ESSERE USATO SOLTANTO SE AL VG8000 È STATA COLLEGATA UNA STAMPANTE.

77. LPRINT

Formato: LPRINT [[USING ⟨formato stampa⟩;]⟨espressione⟩...]

Scopo: Battere informazioni su carta utilizzando la stampante

Categoria: Istruzione

Note: L'istruzione funziona allo stesso modo dell'istruzione "PRINT".

Esempio: Vedere "PRINT".

"LPRINT" PUÒ ESSERE USATO SOLTANTO SE AL VG8000 È STATA COLLEGATA UNA STAMPANTE.

78. LSET

Formato: LSET ⟨X$⟩=⟨Y$⟩

Scopo: Riempire, iniziando da sinistra, la variabile ⟨X$⟩ con il contenuto della variabile ⟨Y$⟩ o con l'espressione alfanumerica ⟨Y$⟩.

Categoria: Istruzione

Note: Deve essere usata l'istruzione "LSET" o "RSET" per inserire i dati nel buffer per un file casuale su dischetto.

I dati numerici possono essere convertiti in forma alfanumerica con le funzioni "MKI$", "MKS$" e "MKD$" quando sono inserite nel buffer per un file casuale.

I dati numerici devono essere stati convertiti da alfanumerici in numerici con le funzioni "CVI", "CVS" e "CVD" quando sono presi dal buffer per un file casuale.

Esempio: Vedere "FIELD"

"LSET" PUÒ ESSERE USATO SOLTANTO SE AL VG8000 SONO STATI COLLEGATI MSX-DOS E L'UNITÀ A DISCHETTI 1 O 2.

79. MAXFILES

Formato: MAXFILES=⟨X⟩

Scopo: Specificare il numero dei file che verranno aperti simultaneamente in un programma.

Categoria: Istruzione

Note: ⟨X⟩ è il numero dei file e deve essere un numero intero tra 0 e 15.

Esempio:

```
10 MAXFILES=2
20 OPEN "CAS:DEMO" FOR INPUT AS #1
30 OPEN "LPT:" FOR OUTPUT AS #2
40 INPUT #1,A$
50 PRINT #2,A$
60 CLOSE
70 END
RUN
```

80. MERGE

Formato: MERGE "⟨periferica⟩:⟨nome programma⟩"

Scopo: Aggiungere righe da un programma su cassetta o su dischetto al programma che è correntemente caricato nella memoria del computer.

Categoria: Comando

Note: ⟨periferica⟩ può essere:
CAS = cassetta
A = unità a dischetti 1
B = unità a dischetti 2

Le righe che devono essere aggiunte al programma presente in memoria devono essere state scritte su cassetta o su dischetto nel formato ASCII con il comando "SAVE".

Se ci sono delle nuove righe i cui numeri corrispondono a righe già presenti nella memoria del computer, queste ultime vengono sostituite dalle nuove che vengono aggiunte.

Esempio:
```
MERGE "CAS:DEMO"
```

"MERGE CAS" PUÒ ESSERE USATO SOLTANTO SE AL VG8000 È STATO COLLEGATO UN REGISTRATORE DATI.

"MERGE A" O "MERGE B" PUÒ ESSERE USATO SOLTANTO SE AL VG8000 SONO STATI COLLEGATI MSX-DOS E L'UNITÀ A DISCHETTI 1 O 2.

81. MID$

Formato: MID$(⟨X$⟩,⟨X⟩[,⟨Y⟩])

Scopo: Dà un'espressione alfanumerica composta di caratteri ⟨Y⟩, iniziando dalla posizione ⟨X⟩ di ⟨X$⟩.

Categoria: Funzione

Note: ⟨X⟩ e ⟨Y⟩ devono essere numeri interi da 0 a 255.

Quando ⟨Y⟩ non viene indicato o quando alla destra di ⟨X⟩ ci sono meno caratteri di quelli indicati da ⟨Y⟩, vengono dati tutti i caratteri alla destra di ⟨X⟩.

Se viene indicato X>LEN(X$) questa funzione dà un'espressione alfanumerica vuota.

Esempio:
```
10 A$="BASIC MSX COMPUTER"
20 PRINT MID$(A$,7,3)
30 END
RUN
```

82. MID$

Formato: MID$(⟨X$⟩,⟨X⟩[,⟨Y⟩])=⟨Y$⟩

Scopo: Sostituire parti di una variabile alfanumerica con un'altra variabile alfanumerica o con una costante alfanumerica.

Categoria: Istruzione

Note: I caratteri in ⟨X$⟩, iniziando dalla posizione ⟨X⟩, sono sostituiti dai caratteri di ⟨Y$⟩.

Quando viene indicato ⟨Y⟩, segnala quanti caratteri di ⟨X$⟩ verranno sostituiti dai caratteri da ⟨Y$⟩.

Se ⟨Y⟩ non viene indicato, saranno usati tutti i caratteri di ⟨Y$⟩, almeno per quanto lo consente la lunghezza di ⟨X$⟩.

Esempio:

```
10 A$="ABCDEFG"
20 PRINT A$
30 MID$(A$,4)="XYZ"
40 PRINT A$
50 END
RUN
```

83. MKI$/MKS$/MKD$

Formati:

```
MKI$(⟨X⟩)
MKS$(⟨Y⟩)
MKD$(⟨Z⟩)
```

Scopo: Convertire valori numerici in valori alfanumerici.

Categoria: Funzione

Note: I valori numerici che vengono inseriti nel buffer per un file casuale su un dischetto devono essere convertiti dai rispettivi valori numerici in forma alfanumerica.

"MKI$" converte un intero in una variabile alfanumerica a 2 byte

"MKS$" converte una variabile a precisione singola in una variabile alfanumerica a 4 byte.

"MKD$" converte una variabile a precisione doppia in una variabile alfanumerica a 8 byte.

Esempio: Vedere "FIELD"

"MKI$", "MKS$" e "MKD$" POSSONO ESSERE USATE SOLTANTO SE AL VG8000 SONO STATI COLLEGATI MSX-DOS E L'UNITÀ A DISCHETTI 1 O 2.

84. MOTOR ON/OFF

Formati:

```
MOTOR [ON]
MOTOR [OFF]
```

Scopo: Cambiare lo stato dell'interruttore del motore del registratore dati.

Categoria: Istruzione

Nota: Quando non vengono aggiunti "ON" o "OFF", lo stato dell'interruttore cambia. Ciò significa che se il motore del registratore dati è acceso, viene spento e viceversa.

Se viene indicato "ON", il motore viene acceso; se viene indicato "OFF" il motore viene spento.

Esempio:

```
10 FOR I=1 TO 10
20 MOTOR
30 NEXT
40 END
RUN
```

"MOTOR" PUÒ ESSERE USATO SOLTANTO SE AL VG8000 È STATO COLLEGATO UN REGISTRATORE DATI.

85. NAME

Formato: `NAME "⟨periferica⟩:⟨nome file-1⟩" AS "⟨nome file-2⟩"`

Scopo: Cambiare il nome di un file su dischetto.

Categoria: Istruzione

Note: ⟨periferica⟩ può essere:
A = unità a dischetti 1
B = unità a dischetti 2

⟨nome file-1⟩ è il nome originale, mentre
⟨nome file-2⟩ è il nuovo nome per quel file

Esempio:

```
NAME "A:TEST" AS "HELP"
```

"NAME" PUÒ ESSERE USATO SOLTANTO SE AL VG8000 SONO STATI COLLEGATI MSX-DOS E L'UNITÀ A DISCHETTI 1 O 2.

86. NEW

Formato: NEW

Scopo: Rimuovere un programma dalla memoria

Categoria: Comando

Note: Il comando "NEW" è usato comunemente per cancellare la memoria del computer prima di iniziare con un nuovo programma.

Esempio: NEW

87. OCT$

Formato: OCT$(⟨X⟩)

Scopo: Dà una rappresentazione ottale del numero decimale ⟨X⟩.

Categoria: Funzione

Note: Il risultato di questa funzione è un valore alfanumerico.

⟨X⟩ deve essere un intero con un valore da -32768 a 65535.
Se ⟨X⟩ è negativo, viene usata la forma del complemento a due. Ciò significa che OCT$(-1) = OCT$(65536-1).

Esempio:

```
10 A=57
20 A$=OCT$(A)
30 PRINT A;A$
40 END
RUN
```

88. ON ERROR GOTO-RESUME

Formato:

```
ON ERROR GOTO <X>
_
_
RESUME [<Y>]
```

Scopo: Iniziare una routine di gestione degli errori e ritornare da questa routine al programma principale.

Categoria: Istruzione

Nota: <X> è il numero della prima riga della routine di gestione degli errori.

Quando viene impartita l'istruzione "ON ERROR GOTO", MSX-BASIC salterà, ad ogni errore che incontra, alla routine di gestione errori indicata. "ERR" dà il codice dell'errore pertinente e "ERL" dà il numero della riga nella quale è stato trovato l'errore.

Quando viene impartita l'istruzione "ON ERROR GOTO 0", MSX-BASIC non salta alla routine di gestione degli errori indicata. In questo caso l'errore viene segnalato dal solito messaggio. È consigliabile usare l'istruzione "ON ERROR GOTO 0" per gli errori non coperti da una procedura di gestione errori nella routine.

Quando MSX-BASIC trova un errore eseguendo una routine di gestione degli errori, viene emesso il normale messaggio di errore.

È possibile ritornare al programma principale dopo una routine di gestione degli errori usando l'istruzione "RESUME".

<Y> è il numero di riga del programma principale alla quale l'esecuzione deve essere ripresa dopo la routine di gestione degli errori. Se <Y> non viene indicato oppure se <Y> è zero, l'esecuzione del programma riprende alla riga in cui è stato trovato l'errore.

Quando viene usata la parola "NEXT" invece di <Y>, l'esecuzione del programma principale riprende alla riga che segue quella in cui si è trovato l'errore.
Una data routine di gestione degli errori si applica anche agli errori trovati nel modo diretto (ad eccezione degli errori di sintassi.

Esempio: Vedere "ERL", "ERR" e "ERROR".

89. ON GOSUB

Formato: ON ⟨espressione⟩ GOSUB ⟨X⟩[,⟨Y⟩...]

Scopo: Saltare ad una subroutine specificata in relazione all'espressione

Categoria: Istruzione

Note: ⟨X⟩ e ⟨Y⟩ sono le prime righe di programma della subroutine.

Il valore dell'espressione determina a quale delle subroutine indicate l'MSX-BASIC deve saltare. Ad esempio, quando quel valore è 3, il salto avverrà alla terza delle subroutine date.

Quando il valore dell'espressione non è un intero, la parte frazionaria verrà eliminata.

Il valore dell'espressione non può superare 255 e non può essere negativo.

Esempio:

```
10 FOR I=1 TO 3
20 ON I GOSUB 50,70,90
30 NEXT I
40 END
50 PRINT "I=";I;"SUBROUTINE 50"
60 RETURN
70 PRINT "I=";I;"SUBROUTINE 70"
80 RETURN
90 PRINT "I=";I;"SUBROUTINE 90"
100 RETURN
RUN
```

90. ON GOTO

Formato: ON ⟨espressione⟩ GOTO ⟨X⟩[,⟨Y⟩...]

Scopo: Saltare ad un numero di riga specificato in relazione all'espressione.

Categoria: Istruzione

Note: ⟨X⟩ e ⟨Y⟩ sono numeri di righe di programma

Il valore dell'espressione determina a quale dei numeri di riga indicati verrà effettuato il salto. Ad esempio, quando il valore è 3, il salto verrà effettuato al terzo dei numeri di riga indicati.

Quando il valore dell'espressione non è un intero, la parte frazionaria viene eliminata.

Il valore dell'espressione non può superare 255 e non può essere negativo.

Esempio:

```
10 FOR I=1 TO 3
20 ON I GOTO 50,70,90
30 NEXT I
40 END
50 PRINT "I=";I;"NUMERO RIGA = 50"
60 GOTO 30
70 PRINT "I=";I;"NUMERO RIGA = 70"
80 GOTO 30
90 PRINT "I=";I;"NUMERO RIGA = 90"
100 GOTO 30
RUN
```

91. ON INTERVAL GOSUB

Formato: ON INTERVAL=⟨X⟩ GOSUB ⟨Y⟩

Scopo: Impostare un intervallo di tempo ed indicare una subroutine che deve essere eseguita quando tale intervallo di tempo è scaduto.

Categoria: Istruzione

Note: ⟨X⟩ è il numero di secondi, diviso per 50.

⟨Y⟩ è il primo numero di riga della subroutine dell'intervallo di tempo.

Prima di poter eseguire una subroutine dell'intervallo di tempo, deve essere attivata l'interruzione con l'istruzione "INTERVAL ON", dopo di che l'MSX-BASIC esegue automaticamente la subroutine indicata ad intervalli regolari di X/50 secondi.

Eseguendo una routine di gestione degli errori, MSX-BASIC eseguirà automaticamente l'istruzione "INTERVAL OFF".

Esempio:

```
10 ON INTERVAL=300 GOSUB 60
20 INTERVAL ON
30 FOR I=0 TO 10000:NEXT
40 INTERVAL OFF
50 END
60 K=K+6:PRINT K;"SECONDI"
70 RETURN
RUN
```

92. ON KEY GOSUB

Formato: ON KEY GOSUB ⟨X⟩[,⟨Y⟩...]

Scopo: Indicare quale subroutine deve essere eseguita quando viene premuto uno dei tasti di funzione.

Categoria: Istruzione

Note: ⟨X⟩ e ⟨Y⟩ sono i primi numeri di riga delle subroutine.
⟨X⟩ è il primo numero di riga della subroutine che verrà eseguita quando viene premuto il tasto di funzione 1; ⟨Y⟩ è il primo numero di riga della subroutine che verrà eseguita quando viene premuto il tasto di funzione 2, ecc.

Prima di poter eseguire la subroutine del tasto di funzione, deve essere attivata l'interruzione con l'istruzione "KEY(X) ON",dopo di che l'MSX-BASIC eseguirà automaticamente la subroutine indicata quando viene premuto il relativo tasto di funzione.

Eseguendo una routine di gestione degli errori, MSX-BASIC eseguirà automaticamente l'istruzione "KEY(X) OFF".

Esempio:

```
10 CLS
20 LOCATE 5,5:PRINT "F1=SUBROUTINE 1"
30 LOCATE 5,7:PRINT "F3=SUBROUTINE 2"
40 LOCATE 5,9:PRINT "F5=END"
50 ON KEY GOSUB 80,,100,,120
60 KEY(1) ON:KEY(3) ON:KEY(5) ON
70 GOTO 70
80 LOCATE 10,11:PRINT "SUBROUTINE 1"
90 RETURN
100 LOCATE 10,11:PRINT "SUBROUTINE 2"
110 RETURN
120 LOCATE 10,11:PRINT "END          "
130 END
RUN
```

93. ON SPRITE GOSUB

Formato: ON SPRITE GOSUB (X)

Scopo: Indicare la subroutine che deve essere eseguita quando due sprites entrano in collisione.

Categoria: Istruzione

Note: (X) è la prima riga di programma della subroutine di collisione.

Prima di poter eseguire una subroutine di collisione, deve essere attivata l'interruzione con l'istruzione "SPRITE ON", dopo di che l'MSX-BASIC eseguirà automaticamente la data subroutine quando due sprite entrano in collisione.

Eseguendo una routine di gestione degli errori, MSX-BASIC eseguirà automaticamente l'istruzione "SPRITE OFF".

Esempio:

```
10 DATA 60,66,165,129,165,153,66,60
20 DATA 60,120,219,255,255,219,102,60
30 A$=""
40 FOR I=1 TO 8
50 READ A:A$=A$+CHR$(A)
60 NEXT
70 B$=""
80 FOR I=1 TO 8
90 READ A:B$=B$+CHR$(A)
100 NEXT
110 SCREEN 2,1:COLOR 15,4,1
120 ON SPRITE GOSUB 210
130 SPRITE$(0)=A$:SPRITE$(1)=B$
140 SPRITE ON
150 A=INT(RND(1)*256)
160 B=INT(RND(1)*256)
170 FOR I=0 TO 191
180 PUT SPRITE 0,(A,I),1
190 PUT SPRITE 1,(B,191-I),15
200 NEXT:GOTO 140
210 SPRITE OFF
220 PLAY "L4CDEFEDCREFGAGFER"
230 PUT SPRITE 0,(0,208)
240 PUT SPRITE 1,(0,208)
250 I=191:RETURN
RUN
```

94. ON STOP GOSUB

Formato: ON STOP GOSUB <X>

Scopo: Indicare la subroutine che deve essere eseguita quando vengono premuti simultaneamente i tasti CTRL e STOP.

Categoria: Istruzione

Note: <X> è il primo numero di riga della subroutine.

Prima di poter eseguire la subroutine, deve essere attivata la interruzione con l'istruzione "STOP ON", dopo di che l'MSX-BASIC eseguirà automaticamente la subroutine indicata quando vengono premuti simultaneamente i tasti CTRL e STOP.

Eseguendo una routine di gestione degli errori, MSX-BASIC eseguirà automaticamente l'istruzione "STOP OFF".

Esempio:

```
10 ON STOP GOSUB 50
20 STOP ON
30 INPUT A$
40 IF A$="END" THEN STOP OFF:END ELSE GOTO 30
50 PRINT "BATTERE END, QUINDI PREMERE TASTO RETURN":RETURN
RUN
```

95. ON STRIG GOSUB

Formato: `ON STRIG GOSUB <X>[,<Y>...]`

Scopo: Indicare quale subroutine deve essere eseguita quando viene premuto uno dei pulsanti di azione dei comandi manuali.

Categoria: Istruzione

Note: `<X>` e `<Y>` sono le prime righe di programma delle subroutine. `<X>` è il primo numero di riga della subroutine che verrà eseguita quando viene premuta la barra di spazio, mentre `<Y>` è il primo numero di riga della subroutine che verrà eseguita quando viene premuto il pulsante di azione del comando manuale 1, ecc.

Prima di poter eseguire la subroutine di azione, deve essere attivata l'interruzione con l'istruzione "`STRIG(X) ON`", dopo di che l'MSX-BASIC eseguirà automaticamente la subroutine indicata quando viene premuto il pertinente pulsante di azione.

Eseguendo una routine di gestione degli errori, MSX-BASIC eseguirà automaticamente l'istruzione "`STIG(X) OFF`".

Esempio:

```
10 CLS
20 ON STRIG GOSUB 50
30 STRIG(0) ON
40 GOTO 40
50 LOCATE 5,5
60 PRINT "PREMUTA BARRA SPAZI"
70 FOR I=1 TO 300:NEXT
80 LOCATE 5,5:PRINT SPC(20)
90 RETURN
RUN
```

96. OPEN

Formato: OPEN "⟨periferica⟩:⟨nome file⟩" [FOR ⟨modo⟩] AS [#]⟨X⟩

Scopo: Allocare un buffer per l'input e l'output ed impostare il modo che verrà usato con il buffer.

Categoria: Istruzione

Note: ⟨periferica⟩ può essere:
CTR = schermo di testo
GRP = schermo grafico
LPT = stampante
CAS = cassetta
A = unità a dischetti 1
B = unità a dischetti 2

⟨nome file⟩ è il nome col quale quel particolare file è stato o sarà registrato sulla periferica. ⟨nome file⟩ non deve essere usato per lo schermo di testo, lo schermo grafico e la stampante.

⟨modo⟩ indica se i dati della periferica verranno letti nel buffer o se i dati nel buffer verranno scritti sulla periferica.

Sono disponibili i seguenti modi:
INPUT = lettura sequenziale
OUTPUT = scrittura sequenziale

Se viene omessa la clausola FOR ⟨modo⟩, si suppone che il file sia di tipo casuale. In tal caso il file verrà creato se non lo si trova nel dischetto nell'unità a dischetti indicata.

Non è possibile usare ogni modo per tutte le periferiche. Le possibilità sono limitate alle seguenti combinazioni:

Periferica	Modo		omessa
	OUTPUT	INPUT	
CRT	*		
GRP	*		
LPT	*		
CAS	*	*	
A	*	*	*
B	*	*	*

⟨X⟩ è il numero del file purché il file sia stato aperto. Questo numero è usato per indicare il file con altre istruzioni di I/O, ad esempio, "INPUT#", "PRINT#", "CLOSE#", ecc.

Un file deve sempre essere aperto prima di poter eseguire qual siasi altra istruzione di I/O.

Il numero dei file che possono essere aperti contemporaneamente è determinato dall'istruzione "MAXFILES".

Esempio:

```
10 SCREEN 2:COLOR 15,4,7
20 OPEN "GRP:" FOR OUTPUT AS #1
30 LINE (32,32)-(120,120),6,B
40 CIRCLE (120,120),56,1
50 PRESET (40,8)
60 PRINT #1,"TESTO IN SCHERMO GRAFICO"
70 FOR I=1 TO 2000:NEXT
80 END
RUN
```

"OPEN LPT" PUÒ ESSERE USATO SOLTANTO SE AL VG8000 È STATA COLLEGATA UNA STAMPANTE.

"OPEN CAS" PUÒ ESSERE USATO SOLTANTO SE AL VG8000 È STATO COLLEGATO UN REGISTRATORE DATI.

"OPEN A" E "OPEN B" POSSONO ESSERE USATI SOLTANTO SE AL VG8000 SONO STATI COLLEGATI MSX-DOS E L'UNITÀ A DISCHETTI 1 O 2.

97. OUT

Formato: OUT ⟨X⟩,⟨Y⟩

Scopo: Trasmettere il valore ⟨Y⟩ ad un connettore di uscita di macchina ⟨X⟩

Categoria: Istruzione

Note: ⟨X⟩ e ⟨Y⟩ devono essere degli interi nel campo da 0 a 255

Usare la funzione "INP" per leggere un byte da un connettore di ingresso della macchina.

Esempio:

```
10 OUT &HA8,INP(&HA8)
20 END
RUN
```

98. PAD

Formato: PAD(⟨X⟩)

Scopo: Dà lo stato di un touch pad

Categoria: Funzione

Note: È possibile collegare uno o due touch pad al computer attraverso le prese di collegamento dei joystick.

⟨X⟩ è un intero che va da 0 a 7

Quando ⟨X⟩ ha un valore tra zero e 3, si presume che il touch pad sia collegato alla presa 1 del connettore per joystick.

Quando ⟨X⟩ ha un valore tra 4 e 7, si presume che il touch pad sia collegato alla presa 2 del connettore per joystick.

Quando ⟨X⟩ è zero o 4, viene indicato lo stato del touch pad.
Il risultato di questa funzione è -1 quando il pad è stato premuto e zero se non è stato ancora premuto.

Quando ⟨X⟩ è 1 o 5, viene indicata la coordinata X

Quando ⟨X⟩ è 2 o 6, viene indicata la coordinata Y

Quando ⟨X⟩ è 3 o 7, viene indicato lo stato dell'interruttore sul touch pad.
Il risultato di questa funzione è -1 quando è stato premuto l'interruttore e zero quando non è stato premuto.

Esempio:

```
10 SCREEN 2
20 AA=0
30 IF PAD(0)=0 THEN 20
40 X=PAD(1):Y=PAD(2)
50 IF AA=0 THEN PSET (X,Y) ELSE LINE -(X,Y)
60 AA=1
70 GOTO 30
RUN
```

"PAD" PUÒ ESSERE USATO SOLTANTO SE AL VG8000 È STATO COLLEGATO UN TOUCH PAD.

99. PAINT

Formato: PAINT [STEP](⟨X⟩,⟨Y⟩),⟨Z⟩[,⟨XX⟩]

Scopo: Riempire una figura grafica con un dato colore nel modo grafico 1 o 2.

Categoria: Istruzione

Note: ⟨X⟩ è la coordinata X del punto iniziale dello schermo e deve essere un intero tra 0 e 255.

⟨Y⟩ è la coordinata Y del punto iniziale dello schermo e deve essere un intero tra 0 e 191.

Quando viene usata la parola "STEP", i valori ⟨X⟩ e ⟨Y⟩ sono interpretati relativamente alla posizione del cursore. In questo caso ⟨X⟩ e ⟨Y⟩ possono anche essere interi negativi.

⟨Z⟩ è il numero del colore usato per dipingere la figura.

⟨XX⟩ è il numero del colore della linea di bordo della figura.

Nel modo grafico 1, il colore usato per dipingere deve essere lo stesso della linea del bordo. In questo caso non deve essere indicato ⟨XX⟩. Nel modo grafico 2, il colore con cui dipingere deve essere diverso dal colore della linea del bordo.

⟨Z⟩ e ⟨XX⟩ devono essere interi da 0 a 15:

0	= trasparente	8	= rosso
1	= nero	9	= rosso chiaro
2	= verde	10	= giallo scuro
3	= verde chiaro	11	= giallo chiaro
4	= blu scuro	12	= verde scuro
5	= azzurro	13	= magenta
6	= rosso scuro	14	= grigio
7	= blu verde	15	= bianco

Esempio:

```
10 SCREEN 2:COLOR 15,4,7
20 CIRCLE (80,80),20,8
30 PAINT (80,80),8
40 FOR I=1 TO 2000:NEXT
50 SCREEN 3:COLOR 15,4,7
60 LINE (10,10)-(100,100),8,B
70 PAINT (45,45),2,8
80 FOR I=1 TO 2000:NEXT
90 END
RUN
```

100. PDL

Formato: PDL(⟨X⟩)

Scopo: Dà lo stato di una paletta

Categoria: Funzione

Note: È possibile collegare una o due palette per joystick al computer attraverso le relative prese.

Questa funzione dà un valore che va da 0 a 255.

⟨X⟩ è un intero che va da 1 a 12.

Quando ⟨X⟩ ha un valore di 1, 3, 5, 7, 9 o 11, si presume che la paletta sia collegata alla presa del connettore per joystick 1. Quando il valore è 2, 4, 6, 8, 10 o 12, si presume che la paletta sia collegata alla presa per connettore per joystick 2.

Esempio:

```
10 PRINT PDL(1),PDL(2)
20 GOTO 10
RUN
```

"PDL" PUÒ ESSERE USATO SOLTANTO SE AL VG8000 È COLLEGATA UNA PALETTA.

101. PEEK

Formato: PEEK(⟨X⟩)

Scopo: Dà il contenuto di un byte della memoria.

Categoria: Funzione

Note: Questa funzione dà il valore decimale del contenuto di un indirizzo di memoria e pertanto dà un valore che va da 0 a 255.

⟨X⟩ è l'indirizzo di memoria e deve avere un valore da -32768 a 65535.

Se ⟨X⟩ è negativo, viene usata la formula del complemento a due. Significa cioè che PEEK(-1) = PEEK(65536-1).

Per riempire un indirizzo di memoria con un valore specificato, viene usata l'istruzione "POKE".

Esempio:

```
10 A=PEEK(65535)
20 A$="00"+HEX$(A)
30 PRINT RIGHT$(A$,2)
40 END
RUN
```

102. PLAY

Formato: PLAY ⟨X$⟩[,⟨Y$⟩][,⟨Z$⟩]

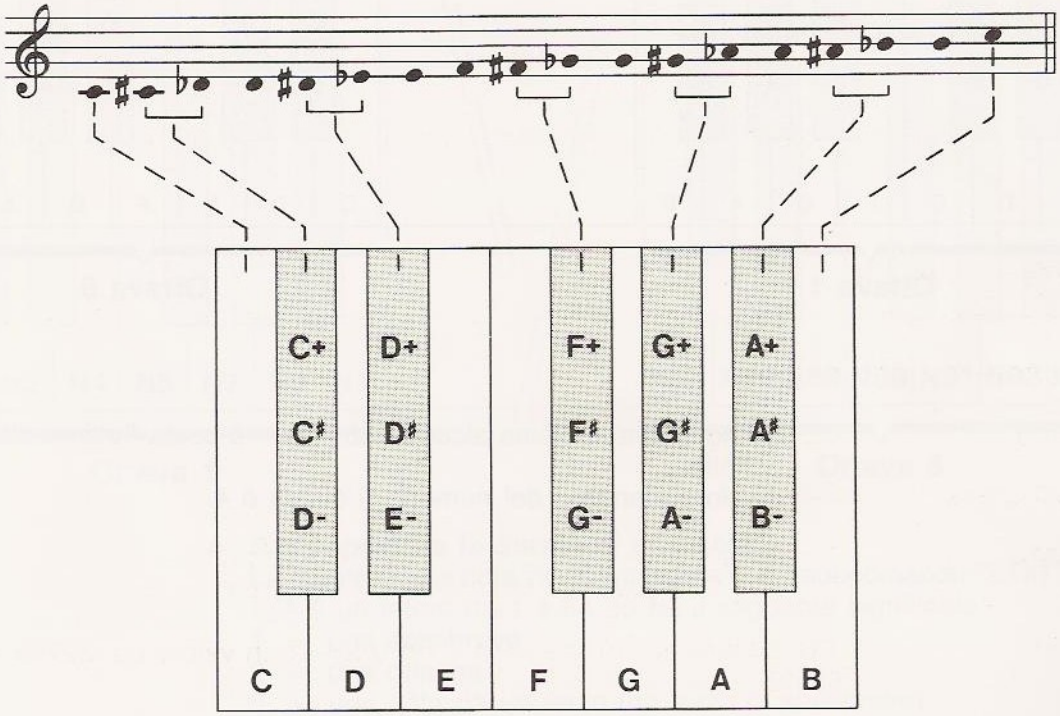
Scopo: Eseguire musica

Categoria: Istruzione

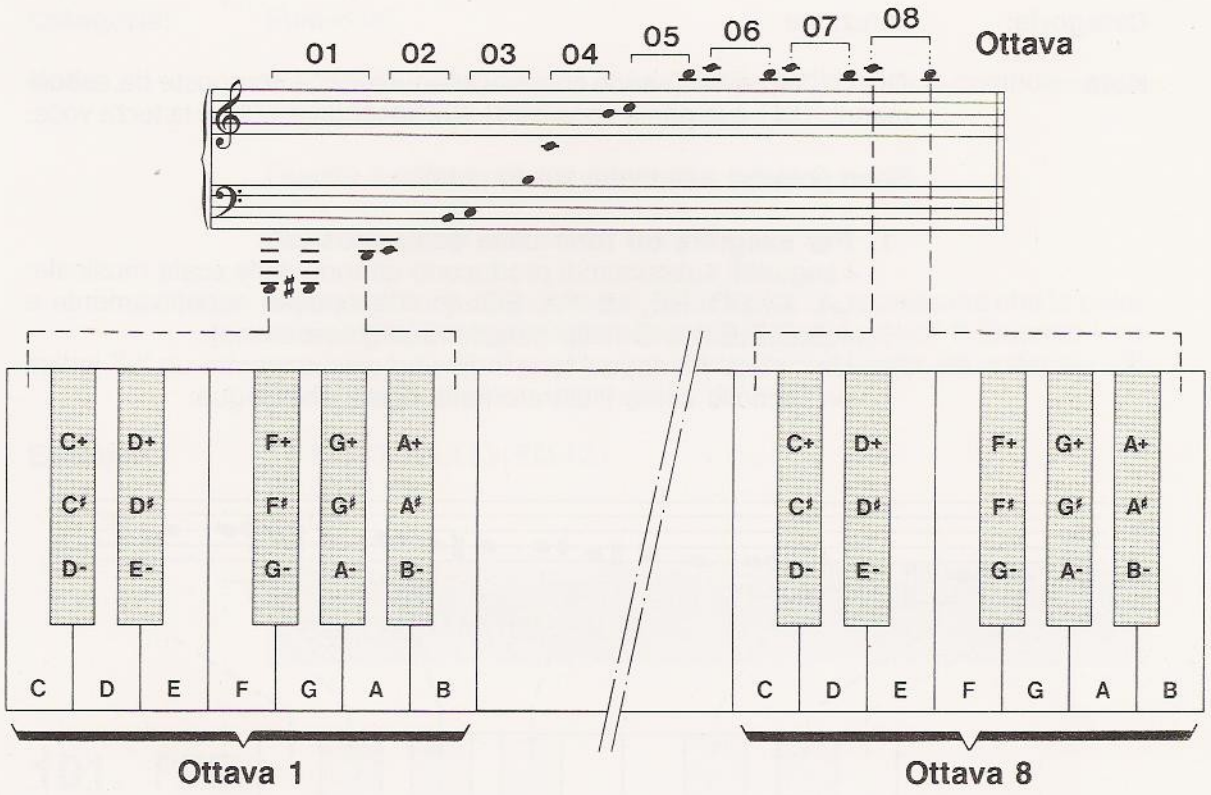
Note: ⟨X$⟩, ⟨Y$⟩ e ⟨Z$⟩ sono costanti alfanumeriche composte da subcomandi. ⟨X$⟩ è la prima voce, ⟨Y$⟩ è la seconda e ⟨Z$⟩ è la terza voce.

Sono possibili i seguenti subcomandi:

1. **Per eseguire un tono dalla scala musicale**
   I seguenti subcomandi producono un tono dalla scala musicale: LA, SI, DO, RE, MI, FA, SOL (corrispondenti rispettivamente a A,B,C,D,E,F e G nella notazione anglossassone).
   Un "+" o "#" dopo il tono indica un diesis, mentre un "-" indica un bemolle come illustrato nella figura che segue:

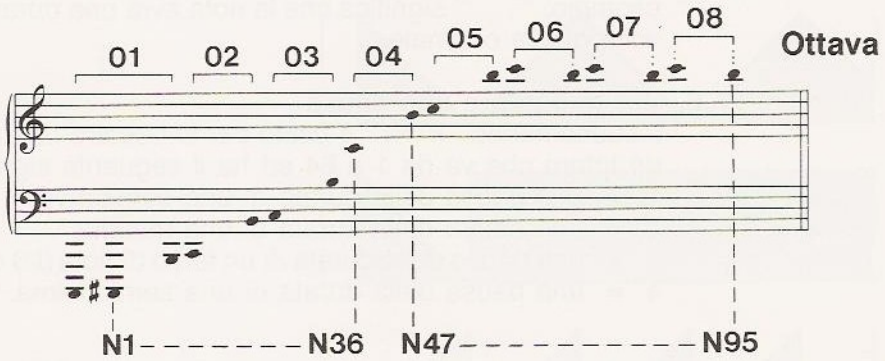


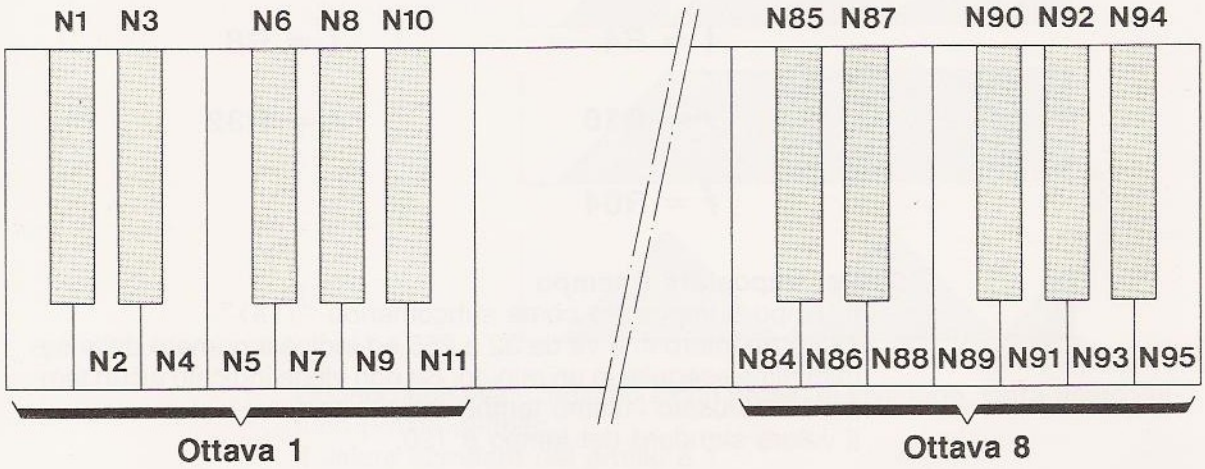
2. **Per impostare un'ottava**
Il subcomando "@⟨X⟩" imposta il numero dell'ottava della nota che deve essere eseguita.
⟨X⟩ è un intero che va da 1 a 8.
L'illustrazione che segue indica quale numero viene usato per le diverse ottave:



Se non viene indicata alcuna ottava, verrà usata l'ultima ottava definita.
Il valore standard del numero di ottava è 4.

3. **Per suonare una nota servendosi di un numero**
Invece di usare un numero di ottava ed un numero di tono dalla scala musicale, è possibile anche usare i numeri delle note musicali.
Il subcomando "N⟨X⟩" è usato per attribuire un numero ad una nota.
⟨X⟩ è un intero che va da 0 a 95.
Quando ⟨X⟩ è uguale a zero, non viene prodotto alcun tono. In tal caso viene eseguita una pausa.





4. **Per impostare la durata di una nota**
La durata di una nota viene impostata con il subcomando "L⟨X⟩".
⟨X⟩ è un intero da 1 a 64 ed ha il seguente significato:
1 = una semibreve
2 = una minima
3 = una nota da un terzo (un terzo di semibreve)
4 = una semiminima ecc.

𝅝 = **L1** 𝅗𝅥 = **L2**

♩ = **L4** ♪ = **L8**

𝅘𝅥𝅯 = **L16** 𝅘𝅥𝅰 = **L32**

𝅘𝅥𝅱 = **L64**

Se non viene indicata alcuna durata, viene usata l'ultima durata definita. Il valore standard per la durata delle note è 4.
La durata di una nota può essere aggiunta immediatamente dopo la nota stessa, senza usare la lettera "L"; esempio: "L4A" equivale a scrivere "A4".

5. **Per cambiare la durata di una nota**
   Aggiungendo un punto (.) immediatamente dopo la nota data, la sua lunghezza viene moltiplicata per 3/2.
   È inoltre possibile aggiungere parecchi punti dopo una nota; ad esempio: ". . ." significa che la nota avrà una durata pari a 27/8 volte quella originale.

6. **Per impostare una pausa**
   Il subcomando "R⟨X⟩" è usato per impostare una pausa. ⟨X⟩ è un intero che va da 1 a 64 ed ha il seguente significato:
   1 = una pausa della durata di una semibreve
   2 = una pausa della durata di una minima
   3 = una pausa della durata di un terzo di nota (l/3 di semibreve)
   4 = una pausa della durata di una semiminima, ecc

   R1 **R2**

   R4 **R8**

   R16 **R32**

   R64

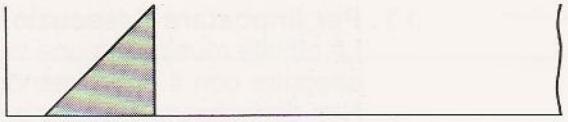
7. **Per impostare il tempo**
   Il tempo è impostato come subcomando "T⟨X⟩".
   ⟨X⟩ è un intero che va da 32 a 255 ed indica il numero delle semiminime eseguite in un minuto. Se non viene indicato alcun tempo, viene usato l'ultimo tempo definito.
   Il valore standard del tempo è 120.

8. **Per impostare il volume**
   Il volume è impostato con il subcomando "V⟨X⟩".
   ⟨X⟩ è un intero che va da 0 a 15.
   Se non viene indicato alcun volume, verrà usato l'ultimo volume definito.
   Il valore standard del volume è 8.

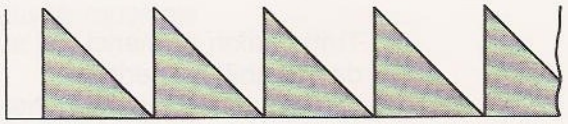
9. **Per impostare il profilo di inviluppo del suono**
   Il profilo di inviluppo o forma d'onda, è impostato con il subcomando "S⟨X⟩".
   ⟨X⟩ è un intero che va da 0 a 15.
   Questi numeri rappresentano i seguenti profili:

   0 — 3 e 9

4 — 7 e 15



8

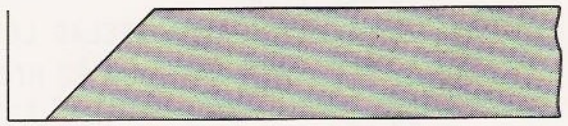


10

11

12

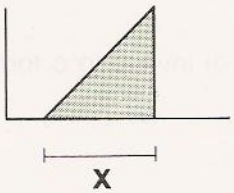
13



14

Quando non viene indicato alcun profilo di inviluppo, verrà usato l'ultimo profilo definito.
Il valore standard del profilo è 1.

10. **Modulazione del profilo**
    Il periodo di inviluppo del profilo è determinato dal subcomando "M⟨X⟩".
    ⟨X⟩ è un intero che va da 1 a 65535 ed imposta il periodo dell'inviluppo.



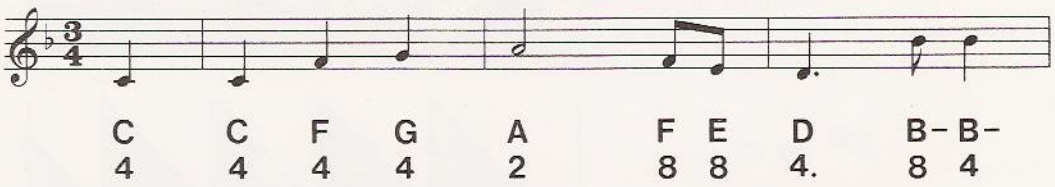
Quando non viene indicata alcuna modulazione, viene usata l'ultima modulazione definita.
Il valore standard della modulazione è 255.

11. **Per impostare l'esecuzione**
Le attività musicali in una variabile alfanumerica possono essere eseguite con il subcomando "X(A$);".
Non dimenticarsi di aggiungere il punto e virgola (;)! (A$) è il nome della variabile contenente i subcomandi musicali.

Tutti i valori numerici nei subcomandi possono essere sostituiti da variabili numeriche.
Queste variabili numeriche devono essere precedute da un segno di uguale (=) e terminate con un punto e virgola (;); per esempio: AB=10:PLAY "N=AB;"

Quando da MSX-BASIC viene generato il suono beep, tutti i valori dei subcomandi sono impostati ai valori standard.

Esempio:




```
10 A$="T128"
20 B$="L4CCFGL2AL8FEL4D.L8B-L4B-"
30 PLAY "XA$;"
40 PLAY "XB$;"
50 END
RUN
```

103. PLAY

Formato: PLAY(⟨X⟩)

Scopo: Dà lo stato dell'attività musicale

Categoria: Funzione

Note: ⟨X⟩ è un intero che va da 0 a 3 ed ha il seguente significato:
0 = voce 1, voce 2 e voce 3
1 = voce 1
2 = voce 2
3 = voce 3

Questa funzione dà il valore -1, mentre l'attività musicale per la voce in questione viene eseguita.
Quando l'attività è stata completata, la funzione mantiene il valore zero.

Quando viene usata questa funzione, immediatamente dopo l'istruzione "PLAY", si ottiene sempre il valore -1.

Esempio:

```
10 A$="":BEEP
20 FOR I=1 TO 6
30 READ AA$:A$=A$+AA$
40 NEXT
50 PLAY "XA$;"
60 SCREEN 0:WIDTH 35:CLS
70 IF PLAY (0)=-1 THEN PRINT "ORA C'E' MUSICA":GOTO 70
80 PRINT "LA MUSICA SI E' FERMATA"
90 END
100 DATA CCGGAAGR
110 DATA FFEEDDCR
120 DATA GGFFEEDR
130 DATA GGFFEEDR
140 DATA CCGGAAGR
150 DATA FFEEDDCR
RUN
```

104. POINT

Formato: POINT(⟨X⟩,⟨Y⟩)

Scopo: Dà il numero del colore di un dato punto di immagine nel modo grafico 1 o 2.

Categoria: Funzione

Note: ⟨X⟩ è la coordinata X e deve essere un intero che va da zero a 255.

⟨Y⟩ è la coordinata Y e deve essere un intero che va da zero a 191.

Esempio:

```
10 SCREEN 2:COLOR 15,1,7
20 OPEN "GRP:" FOR OUTPUT AS #1
30 FOR I=1 TO 10
40 PRESET (I*16,8)
50 C=INT(RND(1)*13)+2
60 COLOR C
70 PRINT #1, CHR$(1)+CHR$(66)
80 NEXT
90 COLOR 15
100 PRESET(40,24)
110 PRINT #1,"NUMERI DEI COLORI"
120 FOR I=1 TO 10
130 C=POINT(I*16+4,12)
140 PRESET (I*16,I*8+56),4
150 PRINT #1,USANDO "##";C
160 NEXT
170 FOR I=0 TO 2000:NEXT
180 COLOR 15,4,7
190 END
RUN
```

107. PRESET

Formato: PRESET [STEP](⟨X⟩,⟨Y⟩)[,⟨Z⟩]

Scopo: Dare ad uno specifico punto di immagine un colore nei modi grafici 1 e 2.

Categoria: Istruzione

Note: ⟨X⟩ è la coordinata X del punto di immagine e deve essere un intero nel campo da zero a 255.

⟨Y⟩ è la coordinata Y del punto di immagine e deve essere un intero nel campo da zero a 191.

Quando viene usata la parola "STEP", i valori per ⟨X⟩ e ⟨Y⟩ sono relativi alla posizione corrente del cursore. In questo caso, ⟨X⟩ e ⟨Y⟩ possono anche essere numeri negativi.

⟨Z⟩ è il numero del colore e deve essere un intero da zero a 15:

0	= trasparente	8	= rosso
1	= nero	9	= rosso chiaro
2	= verde	10	= giallo scuro
3	= verde chiaro	11	= giallo chiaro
4	= blu scuro	12	= verde scuro
5	= azzurro	13	= magenta
6	= rosso scuro	14	= grigio
7	= blu verde	15	= bianco

Quando non viene indicato ⟨Z⟩, verrà usato l'ultimo colore di fondo utilizzato.
Il valore standard di ⟨Z⟩ è 4.

Vedere "PSET".

Esempio:

```
10 SCREEN 2:COLOR 15,4,7
20 LINE (40,40)-(215,151),15,BF
30 FOR I=0 TO 1000
40 A=INT(RND(1)*173)+41
50 B=INT(RND(1)*109)+41
60 PRESET (A,B)
70 FOR K=0 TO 300:NEXT
80 NEXT I
90 END
RUN
```

108. PRINT

Formati:

```
PRINT [[USING <formato stampa>;]<espressione>...]
PRINT #<X>, [USING <formato stampa>;]<espressione>
```

Scopo: Proiettare dati sullo schermo o scrivere tali informazioni in un file sequenziale.

Categoria: Istruzione

Note: Il primo formato viene usato per proiettare i dati sullo schermo.
Se non ci sono ulteriori aggiunte dopo la parola "PRINT", MSX-BASIC proietterà una riga vuota.
Quando viene aggiunta una espressione, il valore dell'espressione è proiettato sullo schermo. L'espressione può essere numerica o alfanumerica.
Quando vengono usate costanti alfanumeriche, devono essere disposte tra virgolette.

Posizione di stampa
La posizione in cui compare l'espressione sullo schermo dipende dalla punteggiatura tra le varie espressioni. MSX-BASIC divide una riga dello schermo in zone di 14 posizioni ciascuna.
Se dopo l'espressione viene inserita una virgola (,), la successiva espressione compare all'inizio della zona successiva.
Se viene inserito un punto e virgola (;) o uno spazio dopo una espressione, la successiva espressione viene visualizzata sullo schermo immediatamente dopo quella che precede.
Se un'istruzione "PRINT" termina con una virgola (,) o con un punto e virgola (;), l'espressione della successiva istruzione "PRINT" compare sulla stessa riga dello schermo.
Se un'istruzione "PRINT" non termina con una virgola (,) o con un punto e virgola (;), la successiva istruzione "PRINT" inizia sulla riga successiva.
Se il numero delle posizioni nell'espressione è maggiore del numero di posizioni sulla riga dello schermo (determinata dalla istruzione "WIDTH"), MSX-BASIC continuerà automaticamente sulla riga successiva.

Numeri
I numeri che vengono proiettati sullo schermo saranno sempre seguiti da uno spazio. I numeri positivi sono preceduti da uno spazio, i numeri negativi dal segno meno (-).

Il secondo formato dell'istruzione "PRINT" è usato per immettere dati in un file sequenziale.

<X> è il numero col quale il file è stato aperto con la istruzione "OPEN". Il file deve essere stato aperto nel modo output.
Dopo ogni istruzione "PRINT #", nel file viene scritto un codice "CR" e "LF".
MSX-BASIC inserisce automaticamente una virgola (,) tra espressioni numeriche in una singola istruzione "PRINT #" come segno di separazione quando viene scritto nel file. Occorre inserire una virgola come segno di separazione tra espressioni alfanumeriche.

Per leggere dati da un file sequenziale, viene usata l'istruzione "INPUT #".

Formati di stampa
È possibile formattare le espressioni quando sono precedute dall'opzione "USING".

Le espressioni **alfanumeriche** possono essere formattate come segue:

USING "!"
Indica che sarà stampato soltanto il primo carattere delle espressioni date.
Esempio:

```
10 A$="EUROPE"
20 PRINT USING "!";A$
30 END
RUN
E
Ok
```

USING "& &"
Indica che il numero di caratteri stampati sarà pari al numero di spazi tra i due caratteri & aumentato di 2.
Esempio:

```
10 A$="EUROPE"
20 PRINT USING "&  &";A$
30 END
RUN
EURO
Ok
```

USING "@"
Indica che occorre sostituire i simboli "@" con le espressioni date.
Esempio:

```
10 A$="I":B$="EUROPE"
20 PRINT USING "@ LOVE @";A$,B$
30 END
RUN
I LOVE EUROPE
Ok
```

Le espressioni **numeriche** possono essere formattate come segue:

USING "#"
Indica quante cifre dell'espressione verrano stampate.
Se l'espressione ha meno cifre del numero di posizioni indicato, MSX-BASIC farà precedere l'espressione con degli spazi.
Se l'espressione contiene più cifre del numero dato di posizione, davanti all'espressione verrà stampato il segno di percentuale (%).
Se necessario, MSX-BASIC arrotonderà la cifra.
Esempio:

```
10 A=109:B=7:C=1198
20 PRINT USING "###";A,B,C
30 END
RUN
109 7%1198
Ok
```

USING "."

Indica dove deve essere stampato il punto decimale.
È ammesso un solo punto decimale.
Esempio:

```
10 A=10.21:B=5.5:C=.245:D=3
20 PRINT USING "##.##";A,B,C,D
30 END
RUN
10.21 5.5 0.25 3.00
Ok
```

USING ","

Quando viene inserita una virgola alla sinistra del punto decimale, verrà stampata una virgola davanti ad ogni gruppo di tre cifre, alla sinistra del punto decimale.
Se viene indicata una virgola al termine del formato di stampa, la virgola verrà stampata immediatamente dopo l'espressione.
Esempio:

```
10 B=1234.5
20 PRINT USING "####,.##";B
30 PRINT USING "####.##,";B
40 END
RUN
1,234.50
1234.50,
Ok
```

USING "+"

Indica che verrà stampato un segno più (+) con espressioni positive ed un segno meno (-) con espressioni negative.
Se il segno più è indicato come primo elemento nel formato di stampa, sarà stampato davanti all'espressione.

Se è indicato al termine del formato di stampa, il segno sarà stampato immediatamente dopo l'espressione.
Esempio:

```
10 A=1.25:B=-1.25
20 PRINT USING "+#.##";A,B
30 PRINT USING "#.##+";A,B
40 END
RUN
+1.25-1.25
1.25+1.25-
Ok
```

USING "-"

Questo simbolo può essere usato soltanto al termine di un formato di stampa.
Il risultato sarà la stampa di uno spazio dopo un'espressione positiva ed un segno meno (-) dopo un'espressione negativa.
Esempio:

```
10 A=1.25:B=-1.25
20 PRINT USING "#.##-";A,B
30 END
RUN
1.25 1.25-
Ok
```

USING "**"

I due asterischi possono essere disposti soltanto all'inizio di un formato di stampa, facendo sì che gli spazi di testa vengano riempiti con asterischi anziché con zeri.
Esempio:

```
10 A=10.25:B=1.25:C=-1.25
20 PRINT USING "**#.##";A,B,C
30 END
RUN
*10.25**1.25*-1.25
Ok
```

USING "$$"

I due segni del dollaro ($) possono essere usati soltanto all'inizio di un formato di stampa.
Ciò si traduce nella stampa di un segno del dollaro davanti all'espressione.
Esempio:

```
10 A=12.35:B=-12.35
20 PRINT USING "$$###.##";A,B
30 PRINT USING "$$###.##-";A,B
40 END
RUN
 $12.35 -$12.35
 $12.35   $12.35-
Ok
```

USING "**$"

Questi simboli possono essere usati soltanto all'inizio di un formato di stampa.
Si tratta di una combinazione di entrambi i precedenti formati di stampa.
Esempio:

```
10 A=12.35
20 PRINT USING "**$#.##";A
30 END
RUN
*$12.35
Ok
```

USING "^^^^"

Questi simboli sono indicati al termine di un formato di stampa e fanno sì che l'espressione venga stampata in formato esponenziale.
Il punto decimale può essere disposto ovunque.
La prima posizione sarà uno spazio a meno che nel formato di stampa non sia indicato un segno più o un segno meno.
Esempio:

```
10 A=234.56:B=12.34:C=-12.34
20 PRINT USING "##.##^^^^";A
30 PRINT USING "#.##^^^^-";C
40 PRINT USING "+#.##^^^^";B,C
50 END
RUN
 2.35E+02
1.23E+01-
+1.23E+01-1.23E+01
Ok
```

Invece della parola "PRINT" è possibile usare un punto di domanda (?).

Esempio:

```
10 CLS
20 FOR I=-1 TO 1 STEP .5
30 PRINT USING "IL NUMERO E' #.#-";I
40 NEXT
RUN
```

109. PSET

Formato: PSET [STEP](⟨X⟩),(⟨Y⟩)[,⟨Z⟩]

Scopo: Attribuire ad un punto di immagine specificato un colore nel modo grafico 1 e 2.

Categoria: Istruzione

Note: ⟨X⟩ è la coordinata X del punto di immagine e deve essere un intero compreso tra zero e 255.

⟨Y⟩ è la coordinata Y del punto di immagine e deve essere un intero compreso tra zero e 191.

Se viene usata la parola "STEP", i valori ⟨X⟩ e ⟨Y⟩ sono interpretati relativamente alla posizione del cursore. In questo caso ⟨X⟩ e ⟨Y⟩ possono anche essere interi negativi.

⟨Z⟩ è il numero del colore e deve essere un intero compreso tra zero e 15:

0	= trasparente	8	= rosso
1	= nero	9	= rosso chiaro
2	= verde	10	= giallo scuro
3	= verde chiaro	11	= giallo chiaro
4	= blu scuro	12	= verde scuro
5	= azzurro	13	= magenta
6	= rosso scuro	14	= grigio
7	= blu verde	15	= bianco

Quando non viene indicato ⟨Z⟩, verrà usato l'ultimo colore usato in primo piano.
Il valore standard di ⟨Z⟩ è 15.

Vedere "PRESET".

Esempio:

```
10 SCREEN 2:COLOR 15,4,7
20 LINE (40,40)-(215,151),15,B
30 FOR I=1 TO 1000
40 A=INT(RND(1)*173)+41
50 B=INT(RND(1)*109)+41
60 PSET (A,B)
70 FOR K=0 TO 300:NEXT
80 NEXT I
90 END
RUN
```

110. PUT

Formato: PUT [#]⟨X⟩[,⟨Y⟩]

Scopo: Scrivere un record dal buffer in un file casuale su dischetto.

Categoria: Istruzione

Note: ⟨X⟩ è il numero col quale il file è stato aperto con l'istruzione "OPEN".

⟨Y⟩ è il numero del record che deve essere scritto. Deve essere un intero tra zero e 32767.
Quando non viene indicato ⟨Y⟩, verrà scritto il successivo record, il che significa il record che segue l'ultimo che è stato letto o scritto.

Il buffer nella memoria deve essere riservato con l'istruzione "FIELD".

Usare l'istruzione "GET" per leggere nel buffer un record da un file casuale su un dischetto nel buffer.

Esempio: Vedere "FIELD".

"PUT" PUÒ ESSERE USATO SOLTANTO SE AL VG8000 SONO STATI COLLEGATI MSX-DOS E L'UNITÀ A DISCHETTI 1 O 2.

111. PUT SPRITE

Formato: PUT SPRITE ⟨Z⟩[,[STEP](⟨X⟩,⟨Y⟩)][,⟨XX⟩][,⟨YY⟩]

Scopo: Inserire un dato sprite sullo schermo nei modi grafici 1 o 2.

Categoria: Istruzione

Note: ⟨Z⟩ indica la priorità dello sprite e deve essere un intero nel campo tra zero e 31.

⟨X⟩ è la coordinata X della posizione sullo schermo e deve essere un intero nel campo da -32 a 255.

⟨Y⟩ è la coordinata Y della posizione sullo schermo e deve essere un intero nel campo fra -32 e 191 oppure 208 o 209.
Se ⟨Y⟩ ha un valore di 208 tutti gli sprites con priorità minore scompariranno dallo schermo.
Se ⟨Y⟩ ha un valore di 209 scomparirà dallo schermo soltanto questo sprite.

Quando viene usata la parola "STEP", i valori ⟨X⟩ e ⟨Y⟩ sono interpretati relativamente alla posizione del cursore. In questo caso ⟨X⟩ e ⟨Y⟩ possono anche essere interi negativi.

Se ⟨X⟩, ⟨Y⟩ o entrambe vengono omesse, vengono usate la coordinata corrente X, la coordinata corrente Y o entrambe.

⟨XX⟩ è il numero del colore per lo sprite e deve essere un intero nel campo tra zero e 15.

0 = trasparente	8 = rosso
1 = nero	9 = rosso chiaro
2 = verde	10 = giallo scuro
3 = verde chiaro	11 = giallo chiaro
4 = blu scuro	12 = verde scuro
5 = azzurro	13 = magenta
6 = rosso scuro	14 = grigio
7 = blu verde	15 = bianco

Quando ⟨XX⟩ non è indicato, verrà usato l'ultimo colore utilizzato in primo piano.
Il valore standard di ⟨XX⟩ è 15.

⟨YY⟩ è il numero dello sprite così come è stato registrato nella variabile "SPRITE$(X)".
Quando ⟨YY⟩ non è indicato, il numero dello sprite corrisponderà al numero che indica la priorità.

La dimensione dello sprite è determinata dall'istruzione "SCREEN".

Esempio: Vedere "ON SPRITE GOSUB".

112. READ

Formato: READ <variabile>[,<variabile>...]

Scopo: Leggere una costante da un'istruzione "DATA" ed assegnarla ad una variabile.

Categoria: Istruzione

Note: Un'istruzione "READ" può essere usata soltanto in combinazione con un'istruzione "DATA".

L'istruzione "READ" dà accesso ai dati, memorizzati con l'istruzione "DATA" in sequenza di numeri di riga di programma.

La variabile in un'istruzione "READ" può essere numerica o alfanumerica. Il tipo di variabile deve concordare con la corrispondente costante nell'istruzione "DATA".

Una singola istruzione "READ" può dare accesso ad una o più istruzioni "DATA". È anche possibile ottenere accesso ad una singola istruzione "DATA" con parecchie istruzioni "READ".

Se il numero di variabili nell'istruzione "READ" è maggiore del numero di costanti nell'istruzione "DATA", MSX-BASIC produrrà il messaggio di errore "Out of data" (Mancanza di dati).

Quando il numero delle variabili nell'istruzione "READ" è minore del numero delle costanti nell'istruzione "DATA", la seguente istruzione "READ" continuerà con la lettura delle costanti nella stessa istruzione "DATA".
Quando non ci sono altre istruzioni "READ", le costanti che non sono state lette saranno ignorate.

Le istruzioni "DATA" possono essere impartite dall'inizio o possono iniziare ad una data riga di programma per mezzo dell'istruzione "RESTORE".

Esempio: Vedere "DATA".

113. REM

Formato: REM <commento>

Scopo: Inserire commenti in un programma

Categoria: Istruzione

Note: Le istruzioni "REM" non verranno eseguite durante lo svolgimento di un programma. Esse verranno comunque stampate quando il programma viene listato.

Il programma può essere fatto saltare alle istruzioni "REM" attraverso un'istruzione "GOTO" o "GOSUB". In tal caso l'esecuzione del programma verrà continuata all'istruzione immediatamente successiva alla "REM".

È possibile usare un apice invece della parola "REM".

Non usare questa istruzione in un'istruzione "DATA".

Esempio:

```
10 REM CALCOLO
20 FOR I=1 TO 10
30 SUM=SUM+V(I)
40 NEXT I
50 SUM=SUM/10
```

oppure:

```
10 FOR I=1 TO 10 'CALCOLO
20 SUM=SUM+V(I)
30 NEXT I
40 SUM=SUM/10
```

114. RENUM

Formato: RENUM [[⟨X⟩][,⟨Y⟩][,⟨Z⟩]]

Scopo: Rinumerare righe di programma

Categoria: Comando

Note: ⟨X⟩ è la prima riga di programma che verrà usata nella nuova sequenza.
Se non viene indicato ⟨X⟩, la nuova sequenza inizierà alla riga 10.

⟨Y⟩ è la riga di programma esistente dalla quale deve iniziare la rinumerazione.
Se non viene indicato ⟨Y⟩, la rinumerazione inizierà dalla prima riga di programma.

⟨Z⟩ è il valore del quale la riga di programma deve essere incrementata nella nuova sequenza.
Se non viene indicato ⟨Z⟩, MSX-BASIC supporrà automaticamente che ⟨Z⟩ sia 10.

Il comando "RENUM" rinumererà automaticamente tutti i riferimenti ed i numeri di riga in un programma nelle istruzioni "GOTO", "GOSUB", "IF THEN ELSE", "ON GOTO" e "ON GOSUB".

Quando si incontra un numero di riga inesistente, viene emesso il messaggio di errore "Unidentified line numer" (numero di riga non identificato).
Il numero di riga inesistente non viene modificato dal comando "RENUM".

Il comando "RENUM" non può essere usato per cambiare la sequenza dei numeri di riga.

Esempio:

```
RENUM
```

(Ciò rinumera l'intero programma iniziando dalla prima riga. La prima riga avrà il numero 10. Le righe successive saranno numerate ad incrementi di 10).

```
RENUM 1000,900,50
```

(Ciò rinumera il programma iniziando dalla riga 900. La riga 900 viene cambiata in 1000 e le righe successive verrano numerate a incrementi di 50).

115. RESTORE

Formato: RESTORE [⟨X⟩]

Scopo: Leggere le costanti da un'istruzione "DATA" a partire da un dato punto per mezzo di un'istruzione "READ".

Categoria: Istruzione

Note: ⟨X⟩ è un numero di riga di programma di un'istruzione "DATA".

Dopo l'esecuzione dell'istruzione "RESTORE", la prima costante nell'istruzione "DATA" della riga data verrà letta con la successiva istruzione "READ".

Quando ⟨X⟩ non è indicato, la prima costante nella prima istruzione "DATA" del programma verrà letta insieme alla successiva istruzione "READ".

Esempio: Vedere "DATA".

116. RIGHT$

Formato: RIGHT$(⟨X$⟩,⟨X⟩)

Scopo: Fornisce un'espressione alfanumerica composta dai caratteri ⟨X⟩ più a destra di ⟨X$⟩.

Categoria: Funzione

Note: ⟨X⟩ è un intero nel campo da 0 a 255

Quando ⟨X⟩ è maggiore della lunghezza di ⟨X$⟩, viene indicato il contenuto completo di ⟨X$⟩.

Quando ⟨X⟩ è zero, si ottiene un'espressione alfanumerica vuota.

Esempio:

```
10 A$="BASIC"
20 FOR I=1 TO LEN(A$)
30 PRINT RIGHT$(A$,I)
40 NEXT
50 END
RUN
```

117. RND

Formato: RND(⟨X⟩)

Scopo: Dà un numero casuale nel campo tra zero e 1.

Categoria: Funzione

Note: Ogni volta che viene avviato un programma con il comando "RUN", viene generata la stessa sequenza di cifre casuali.

Quando ⟨X⟩ è maggiore di zero, viene indicato il successivo numero nella sequenza.

Quando ⟨X⟩ è uguale a zero, viene indicato l'ultimo numero nella sequenza.

Quando ⟨X⟩ è minore di zero, la sequenza di numeri casuali viene generata da capo per il dato valore negativo di ⟨X⟩.

Per ottenere un numero assolutamente casuale, il generatore deve scegliere una nuova sequenza ogni volta che il programma inizia l'esecuzione.
Ciò può essere effettuato usando la variabile "TIME".

Esempio:

```
10 R=RND(-TIME)
20 FOR I=1 TO 10
30 SCREEN 2:COLOR 15,4,7
40 C=INT(RND(1)*15+1)
50 IF C=4 GOTO 40
60 CIRCLE (80,80),20,C
70 PAINT (80,80),C
80 FOR J=0 TO 300:NEXT
90 NEXT I
100 END
RUN
```

118. RSET

Formato: RSET ⟨X$⟩=⟨Y$⟩

Scopo: Riempire, iniziando da destra, la variabile ⟨X$⟩ con il contenuto della variabile ⟨Y$⟩ o dell'espressione alfanumerica ⟨Y$⟩.

Categoria: Istruzione

Note: L'istruzione "RSET" o "LSET" deve essere usata per inserire i dati nel buffer per un file casuale sul dischetto.

I dati numerici devono essere convertiti in dati alfanumerici con le funzioni "MKI$", "MKS$" e "MKD$" quando sono inseriti nel buffer per un file casuale.

I dati numerici devono essere convertiti da alfanumerici a numerici con le funzioni "CVI", "CVS" e "CVD" quando sono estratti dal buffer per un file casuale.

Esempio: Vedere "FILED"

"RSET" PUÒ ESSERE USATO SOLTANTO SE AL VG8000 SONO STATI COLLEGATI MSX-DOS E L'UNITÀ A DISCHETTI 1 O 2.

119. RUN

Formato: RUN [⟨periferica⟩:⟨nome programma⟩] [⟨X⟩]

Scopo: Avviare l'esecuzione del programma che è caricato nella memoria del computer o in una periferica.

Categoria: Comando

Note: ⟨periferica⟩ può essere:
CAS = cassetta
A = unità a dischetti 1
B = unità a dischetti 2

Il programma deve essere scritto su cassetta o su dischetto in formato ASCII con il comando "SAVE".

Il nome del programma è una costante alfanumerica che viene specificata quando il programma viene scritto su cassetta o su dischetto con il comando "SAVE".

Quando ⟨periferica⟩:⟨nome programma⟩ viene omesso, ha inizio l'esecuzione del programma caricato nella memoria del computer.

⟨X⟩ è un numero di riga di programma.

Quando viene indicato ⟨X⟩, l'esecuzione del programma inizierà alla riga di programma indicata.

Quando ⟨X⟩ non è indicato, l'esecuzione del programma inizierà alla prima riga.

Esempio: RUN 100

(In questo caso MSX-BASIC inizia l'esecuzione alla riga numero 100).

"RUN CAS" PUÒ ESSERE ESEGUITO SOLO SE AL VG8000 È COLLEGATO UN REGISTRATORE A CASSETTA.

"RUN A" o "RUN B" POSSONO ESSERE USATI SOLO SE AL VG8000 SONO COLLEGATI L'MSX-DOS E LE UNITÀ A DISCHETTI 1 O 2.

120. SAVE

Formato: SAVE "⟨periferica⟩:⟨nome programma⟩"

Scopo: Scrivere un programma dalla memoria del computer sulla cassetta o sul dischetto.

Categoria: Comando

Note: ⟨periferica⟩ può essere:
CAS = cassetta
A = unità a dischetti 1
B = unità a dischetti 2

Il programma viene scritto sulla cassetta o sul dischetto in formato ASCII e può essere letto nella memoria del computer con il comando "LOAD" o "MERGE".

Il nome del programma è una costante alfanumerica che deve essere definita di nuovo con il comando "LOAD" o "MERGE".

Esempio: SAVE "CAS:DEMO"

"SAVE CAS" PUÒ ESSERE USATO SOLTANTO SE AL VG8000 È STATO COLLEGATO UN REGISTRATORE DATI.

"SAVE A" O "SAVE B" PUÒ ESSERE USATO SOLTANTO SE AL VG8000 SONO STATI COLLEGATI MSX-DOS E L'UNITÀ A DISCHETTI 1 O 2.

121. SCREEN

Formato: SCREEN [⟨X⟩][,⟨Y⟩][,⟨Z⟩][,⟨XX⟩][,⟨YY⟩]

Scopo: Assegnare il modo schermo, la dimensione dello sprite, il suono della battuta dei tasti, la velocità di trasmissione della cassetta ed il tipo di stampante usata.

Categoria: Istruzione

Note: ⟨X⟩ identifica il modo dello schermo.
Deve essere un intero tra 0 e 3 ed ha il seguente significato:

0 = modo testo 1
1 = modo testo 2
2 = modo grafico 1
3 = modo grafico 2

Quando ⟨X⟩ non viene indicato, viene riutilizzato l'ultimo modo schermo usato.
Il valore standard di ⟨X⟩ è zero.

Le istruzioni grafiche "PUT SPRITE", "CIRCLE", "DRAW", "LINE", "PAINT", "PSET", "PRESET", "ON SPRITE GOSUB", "SPRITE ON/OFF/STOP" e "POINT" possono essere usate soltanto quando è attivato uno dei modi grafici.

⟨Y⟩ determina la dimensione degli sprites che devono essere usati.
Deve essere un intero che va da zero a 3 ed ha il seguente significato:

0 = sprites piccoli (8x8 pixel)
1 = sprites piccoli ingranditi a 16x16 pixel
2 = sprites grandi (16x16 pixel)
3 = sprites grandi ingranditi a 32x32 pixel

Quando ⟨Y⟩ non è indicato, viene usata la dimensione dell'ultimo sprite visualizzato.
Il valore standard di ⟨Y⟩ è zero.

⟨Z⟩ determina se deve essere prodotto il suono ticchettante quando viene premuto un tasto.
Deve essere zero o 1 ed ha il seguente significato:

0 = nessun suono ticchettante
1 = suono ticchettante

Quando ⟨Z⟩ non è indicato, viene adottato l'ultimo valore usato.
Il valore standard di ⟨Z⟩ è 1.

⟨XX⟩ dà la velocità di trasmissione per il registratore dati.
Deve essere 1 o 2 ed ha il seguente significato:

1 = velocità di trasmissione 1200 baud
2 = velocità di trasmissione 2400 baud

Quando ⟨XX⟩ non è indicato, deve essere usata l'ultima velocità di trasmissione utilizzata.
Il valore standard di ⟨XX⟩ è 1.

⟨YY⟩ serve per dire al computer se viene usata una specifica stampante MSX o meno.
Deve essere zero o 1 ed ha il seguente significato:

0 = stampante MSX
1 = stampante non-MSX

Quando ⟨YY⟩ non viene indicato, viene riutilizzata l'indicazione dell'ultima stampante usata.
Il valore standard di ⟨YY⟩ è 1.

Esempio:

```
10 FOR I=2 TO 3
20 SCREEN I:COLOR 15,4,7
30 LINE (32,32)-(200,132),6
40 FOR K=0 TO 3000:NEXT
50 NEXT I
60 END
RUN
```

122. SGN

Formato: SGN(⟨X⟩)

Scopo: Dà il valore 1 quando X>0, il valore zero quando X=0 ed il valore -1 quando X<0.

Categoria: Funzione

Note: Nessuna

Esempio:

```
10 INPUT "NUMERO";N
20 H=SGN(N):H=H+2
30 ON H GOSUB 60,70,80
40 PRINT B$
50 END
60 B$="IL NUMERO E' NEGATIVO":RETURN
70 B$="IL NUMERO E' ZERO":RETURN
80 B$="IL NUMERO E' POSITIVO":RETURN
RUN
```

123. SIN

Formato: SIN(⟨X⟩)

Scopo: Dà il seno di ⟨X⟩ in radianti

Categoria: Funzione

Note: Nessuna

Esempio:

```
10 X=5
20 PRINT SIN(X)
30 END
RUN
```

124. SOUND

Formato: SOUND ⟨X⟩,⟨Y⟩

Scopo: Inserire un valore in uno dei registri del generatore di suoni programmabile (PSG).

Categoria: Istruzione

Note: ⟨X⟩ è il numero di registro e deve essere un intero nel campo tra zero e 15.

⟨Y⟩ è il valore che deve essere inserito nel registro indicato. Deve trattarsi di un intero nel campo tra zero e 255.

USARE QUESTA ISTRUZIONE SOLTANTO SE SI HA COMPLETA FAMILIARITÀ CON IL FUNZIONAMENTO DEL PSG (GENERATORE PROGRAMMABILE DI SUONI).

Esempio:

```
10 FOR I=0 TO 13
20 SOUND I,0
30 NEXT I
40 SOUND 7,62:SOUND 8,15
50 FOR I=0 TO 255
60 SOUND 0,I
70 NEXT I
80 SOUND 8,0
90 END
RUN
```

125. SPACE$

Formato: SPACE$(⟨X⟩)

Scopo: Dà un valore alfanumerico con un numero ⟨X⟩ di spazi

Categoria: Funzione

Note: ⟨X⟩ deve essere un intero nel campo tra zero e 255.

Esempio:

```
10 FOR I=0 TO 5
20 A$=SPACE$(I)
30 PRINT A$;I
40 NEXT I
50 END
RUN
```

126. SPC

Formato: SPC(⟨X⟩)

Scopo: Stampare un numero ⟨X⟩ di spazi

Categoria: Funzione

Note: Questa funzione può essere usata soltanto in combinazione con un'istruzione "PRINT" o "LPRINT".

⟨X⟩ deve essere un intero nel campo tra zero e 255.

Esempio:

```
10 PRINT "QUI";SPC(5);"LA'"
20 END
RUN
```

127. SPRITE ON/OFF/STOP

Formati:

```
SPRITE ON
SPRITE OFF
SPRITE STOP
```

Scopo: Attivare il controllo di una collisione tra gli sprites.

Categoria: Istruzione

Note: Dopo l'istruzione "SPRITE ON", MSX-BASIC controllerà in ogni istruzione se è avvenuta una collisione tra due sprites.
In caso affermativo, MSX-BASIC eseguirà la subroutine indicata con l'istruzione "ON SPRITE GOSUB".

Dopo l'istruzione "SPRITE OFF", MSX-BASIC non controllerà più se è avvenuta una collisione tra due sprites.

Dopo l'istruzione "SPRITE STOP", MSX-BASIC controllerà se si è verificata una collisione tra due sprites ma non eseguirà la subroutine indicata con l'istruzione "ON SPRITE GOSUB".

Esso ricorderà comunque se è avvenuta una collisione. In tal caso la subroutine indicata verrà eseguita immediatamente dopo l'istruzione "SPRITE ON".

Esempio: Vedere "ON SPRITE GOSUB".

128. SPRITE$

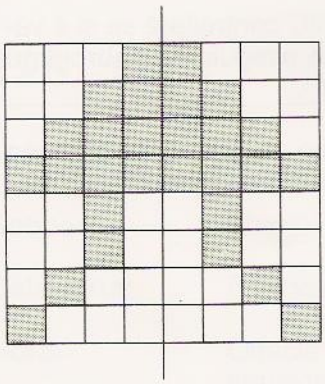
Formato: SPRITE$(⟨X⟩)

Scopo: Contiene la definizione degli sprites nel modo grafico 1 o 2.

Categoria: Variabile di sistema

Note: ⟨X⟩ è il numero dello sprite.
Quando la dimensione degli sprites indicata con l'istruzione "SCREEN" è zero o 1, ⟨X⟩ deve essere un intero nel campo da zero a 255.
Quando la dimensione degli sprites è indicata come 3 o 4 con l'istruzione "SCREEN", ⟨X⟩ deve essere un intero nel campo tra zero e 63.

Il contenuto della variabile "SPRITE$" può essere un valore binario esadecimale o decimale come illustrato dall'esempio che segue:



Binario		Esadecimale		Decimale
&B00011000	=	&H18	=	24
&B00111100	=	&H3C	=	60
&B01111110	=	&H7E	=	126
&B11111111	=	&HFF	=	255
&B00100100	=	&H24	=	36
&B00100100	=	&H24	=	36
&B01000010	=	&H42	=	66
&B10000001	=	&H81	=	129

Definizione binaria

```
10 SCREEN 2,0
20 B$=""
30 FOR I=1 TO 8
40 READ A$:B$=B$+CHR$(VAL("&B"+A$))
50 NEXT I
60 SPRITE$(0)=B$
70 DATA 00011000,00111100,01111110,11111111,00100100,
00100100,01000010,10000001
```

Definizione esadecimale

```
10 SCREEN 2,0
20 B$=""
30 FOR I=1 TO 8
40 READ A$:B$=B$+CHR$(VAL("&H"+A$))
50 NEXT I
60 SPRITE$(0)=B$
70 DATA 18,3C,7E,FF,24,24,42,81
```

Definizione decimale

```
10 SCREEN 2,0
20 B$=""
30 FOR I=1 TO 8
40 READ A:B$=B$+CHR$(A)
50 NEXT I
60 SPRITE$(0)=B$
70 DATA 24,60,126,255,36,36,66,129
```

Vedere inoltre "PUT SPRITE"

Esempio: Vedere "ON SPRITE GOSUB".

129. SQR

Formato: SQR(⟨X⟩)

Scopo: Dà la radice quadrata di ⟨X⟩

Categoria: Funzione

Note: ⟨X⟩ deve essere maggiore o uguale a zero.

Esempio:
```
10 X=10:Y!=SQR(X)
20 PRINT Y!
30 END
RUN
```

130. STICK

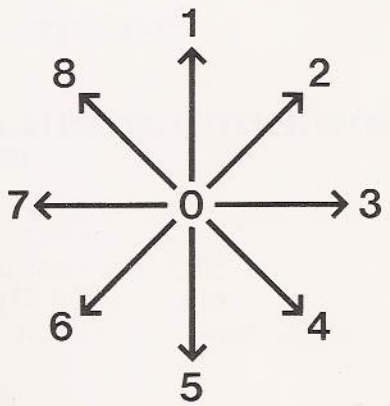
Formato: STICK(⟨X⟩)

Scopo: Dà la posizione di un joystick o lo stato dei tasti di controllo del cursore.

Categoria: Funzione

Note: ⟨X⟩ può rappresentare una delle seguenti indicazioni:
- 0 = i tasti di controllo del cursore servono da joystick
- 1 = riguarda il joystick collegato al connettore N.1
- 2 = riguarda il joystick collegato al connettore N.2

Questa funzione dà i seguenti valori:
- 0 = neutro
- 1 = Nord
- 2 = Nord-Est
- 3 = Est
- 4 = Sud-Est
- 5 = Sud
- 6 = Sud-Ovest
- 7 = Ovest
- 8 = Nord-Ovest



Esempio:
```
10 PRINT "PREMI I TASTI CONTROLLO CURSORE"
20 PRINT STICK(0);
30 FOR I=0 TO 300:NEXT I
40 GOTO 20
RUN
```

131. STOP

Formato: STOP

Scopo: Interrompere l'esecuzione di un programma

Categoria: Istruzione

Note: L'istruzione "STOP" può essere usata ovunque in un programma.

Incontrando l'istruzione "STOP", l'esecuzione del programma verrà interrotta e sullo schermo comparirà il messaggio: "BREAK IN xxxxx" (interruzione in xxxxx). "xxxxx" rappresenta il numero di riga alla quale si è verificata l'interruzione del programma.

L'esecuzione può essere ripresa con il comando "CONT".

Esempio: Vedere "CONT".

132. STOP ON/OFF/STOP

Formati:
```
STOP ON
STOP OFF
STOP STOP
```

Scopo: Controllare che i tasti CTRL e STOP siano stati premuti simultaneamente.

Categoria: Istruzione

Note: Dopo l'istruzione "STOP ON", MSX-BASIC controllerà in ogni istruzione se sono stati premuti simultaneamente i tasti CTRL/STOP.
In caso affermativo MSX-BASIC eseguirà una subroutine indicata nell'istruzione "ON STOP GOSUB".

Dopo l'istruzione "STOP OFF", MSX-BASIC non controllerà più se i tasti CTRL e STOP sono stati premuti simultaneamente.

Dopo l'istruzione "STOP STOP", MSX-BASIC controllerà se i tasti CTRL e STOP sono stati premuti ma non eseguirà la subroutine indicata nell'istruzione "ON STOP GOSUB".
Esso ricorderà comunque se i tasti CTRL e STOP sono stati premuti. In tal caso la subroutine indicata verrà eseguita immediatamente dopo l'istruzione "STOP ON".

Esempio: Vedere "ON STOP GOSUB".

133. STRIG

Formato: STRIG(⟨X⟩)

Scopo: Dà lo stato del pulsante di azione dei comandi manuali o della barra spazi.

Categoria: Funzione

Note: ⟨X⟩ è un intero che va da zero a 4 ed ha il seguente significato:

0 = barra spazi
1 = pulsante di azione del comando manuale collegato alla presa N. 1
2 = pulsante di azione del comando manuale collegato alla presa N. 2
3 = pulsante di azione del comando manuale collegato alla presa N. 1
4 = pulsante di azione del comando manuale collegato alla presa N. 2

Quando viene premuto il pulsante di azione o la barra spazi, questa funzione produce il valore di -1. In caso contrario, si ottiene il valore zero.

Esempio:

```
10 PRINT "PREMI BARRA SPAZI"
20 P=STRIG(0)
30 IF P=-1 THEN BEEP
40 GOTO 20
RUN
```

134. STRIG(X) ON/OFF/STOP

Formati:

```
STRIG(<X>)
ON STRIG(<X>) OFF
STRIG(<X>) STOP
```

Scopo: Attivare il controllo dello stato di un pulsante di azione o della barra spazi.

Categoria: Istruzione

Note: <X> è un intero che va da zero a 4 ed ha il seguente significato:

- 0 = barra spazi
- 1 = pulsante di azione del comando manuale collegato alla presa N. 1
- 2 = pulsante di azione del comando manuale collegato alla presa N. 2
- 3 = pulsante di azione del comando manuale collegato alla presa N. 1
- 4 = pulsante di azione del comando manuale collegato alla presa N. 2

Dopo l'istruzione "STRIG(X) ON", MSX-BASIC controllerà in ogni istruzione se è stato premuto il pulsante di azione indicato.
In caso affermativo, MSX-BASIC eseguirà una subroutine indicata nell'istruzione "ON STRIG GOSUB".

Dopo l'istruzione "STRIG(X) OFF", MSX-BASIC non controllerà se è stato premuto il pulsante di azione indicato.

Dopo l'istruzione "STRIG(X) STOP", MSX-BASIC controllerà se è stato premuto il pulsante di azione indicato ma non eseguirà la subroutine indicata con l'istruzione "ON STRIG GOSUB".
Esso ricorderà comunque se è stato premuto il pulsante di azione indicato.
In questo caso, verrà eseguita immediatamente la subroutine indicata dopo l'istruzione "STRIG(X) ON".

Esempio: Vedere "ON STRIG GOSUB".

135. STR$

Formato: STR$(⟨X⟩)

Scopo: Dà una rappresentazione alfanumerica dell'espressione numerica ⟨X⟩.

Categoria: Funzione

Note: Usare l'istruzione "VAL" per la rappresentazione numerica di un'espressione alfanumerica.

Esempio:

```
10 AMOUNT=12.34216:PRINT QUANTITA'
20 A$=STR$(AMOUNT)
30 HZ=INSTR(A$,".")
40 IF HZ=<1 GOTO 80
50 MID$ (A$,HZ,1)=","
60 H$=LEFT$(A$,HZ+2)
70 PRINT H$
80 END
RUN
```

136. STRING$

Formati:

```
STRING$(⟨X⟩,⟨Y⟩)
STRING$(⟨X⟩,⟨Y$⟩)
```

Scopo: Dà un valore alfanumerico di lunghezza ⟨X⟩, contenente solo caratteri con il codice ⟨Y⟩ o il primo carattere dell'espressione alfanumerica ⟨Y$⟩.

Categoria: Funzione

Note: ⟨X⟩ è la lunghezza del valore alfanumerico e deve essere un intero nel campo tra zero e 255.

⟨Y⟩ è un codice di carattere e deve essere un intero nel campo tra 32 e 255.

Esempio:

```
10 SCREEN1:COLOR 15,4,7
20 A$=CHR$(42)
30 FOR K=1 TO 22
40 C=INT(RND(1)*20)
50 LOCATE 3,K
60 PRINT USING "##";C
70 LOCATE 6,K
80 PRINT STRING$(C,A$)
90 NEXT K
100 END
RUN
```

137. SWAP

Formato: SWAP ⟨variabile⟩,⟨variabile⟩

Scopo: Scambiare i contenuti di due variabili

Categoria: Istruzione

Note: Può essere usato qualsiasi tipo di variabile (intera, a precisione singola, a doppia precisione, alfanumerica).
La sola condizione è che entrambe le variabili siano dello stesso tipo.

Esempio:

```
10 X=3:Y=7
20 GOSUB 60
30 SWAP X,Y
40 GOSUB 60
50 END
60 PRINT "X=";X;"Y=";Y
70 RETURN
RUN
```

138. TAB

Formato: TAB(⟨X⟩)

Scopo: Spostare il cursore sulla stessa riga ad una data posizione ⟨X⟩

Categoria: Funzione

Note: Questa funzione può essere usata soltanto in combinazione con un'istruzione "PRINT" o "LPRINT".

⟨X⟩ deve essere un intero nel campo fra zero e 255.

Se il cursore ha già superato la posizione ⟨X⟩ sulla riga, non succede nulla.

Esempio:

```
10 PRINT "0";TAB(9);"9"
20 END
RUN
```

139. TAN

Formato: TAN(⟨X⟩)

Scopo: Dà la tangente di ⟨X⟩ in radianti

Categoria: Funzione

Note: Nessuna

Esempio:

```
10 X=5
20 PRINT TAN(X)
30 END
RUN
```

140. TIME

Formato: TIME

Scopo: Contiene il tempo interno del sistema

Categoria: Variabile di sistema

Note: Ogni volta che il processore video (VDP) genera un'interruzione (50 volte al secondo), la variabile "TIME" è aumentata di 1.

Quando non viene generata alcuna interruzione (ad esempio quando un programma viene scritto sulla cassetta o letto dalla cassetta), la variabile "TIME" rimane inalterata.

Esempio:

```
10 CLS
20 LOCATE 10,6
30 PRINT "HH:MM:SS INIZIO"
40 TIME=0
50 T=TIME
60 H=INT(T/180000)
70 T=T-(H*180000)
80 M=INT(T/3000)
90 T=T-(M*3000)
100 S=INT(T/50)
110 LOCATE 10,8
120 PRINT USING "##:##:##";H;M;S
130 GOTO 50
RUN
```

141. TROFF

Formato: TROFF

Scopo: Terminare la funzione di traccia o analisi del programma

Categoria: Comando

Note: La funzione di traccia è attivata con il comando "TRON".

La funzione di traccia viene anche terminata con il comando "NEW".

Esempio: Vedere "TRON".

142. TRON

Formato: TRON

Scopo: Attivare la funzione di traccia o di analisi del programma

Categoria: Comando

Note: Quando viene impartito il comando "TRON", MSX-BASIC stampa tra parentesi tutti i numeri delle righe di programma che saranno eseguite durante il normale svolgimento del programma.

Esempio:

```
10 FOR I=1 TO 3
20 PRINT I
30 NEXT
40 END TRON Ok
RUN
[10][20] 1
[30][20] 2
[30][20] 3
[30[]40]
Ok
TROFF
Ok
```

143. USR

Formato: USR[⟨X⟩](⟨Y⟩)

Scopo: Richiama una subroutine in linguaggio macchina

Categoria: Funzione

Note: ⟨X⟩ è un intero nel campo da zero a 9, che indica quale subroutine in linguaggio macchina deve essere eseguita.
Quando ⟨X⟩ non viene indicato, si suppone il valore zero.

Il primo indirizzo di ogni subroutine in linguaggio macchina deve essere indicato con l'istruzione "DEFUSR".

⟨Y⟩ è il valore fornito con la subroutine in linguaggio macchina.
Questo valore è indicato in uno dei modi seguenti:

1. **Per i valori alfanumerici**
   L'indirizzo di memoria &HF663 contiene il valore 3.
   Gli indirizzi di memoria &HF7F8 e &HF7F9 contengono l'indirizzo del descrittore del valore alfanumerico. Il descrittore si compone di 3 byte. Il primo byte indica la lunghezza del valore alfanumerico; il secondo ed il terzo byte contengono l'indirizzo di memoria del valore alfanumerico.

2. **Per i valori numerici interi**
   L'indirizzo di memoria &HF663 contiene il valore 2.
   Gli indirizzi di memoria &HF7F8 e &HF7F9 contengono il valore dell'intero.

3. **Per i valori a precisione singola**
   L'indirizzo di memoria &HF663 contiene il valore 4.
   Gli indirizzi di memoria da &HF7F6 fino a &HF7F9 contengono il valore della precisione singola.

4. **Per i valori a precisione doppia**
   L'indirizzo di memoria &HF663 contiene il valore 8.
   Gli indirizzi di memoria da &HF7F6 a &HF7FD contengono il valore della precisione doppia.

   I valori che sono ritornati all'MSX-BASIC dalla subroutine in linguaggio macchina devono essere inseriti nella memoria allo stesso modo sopradescritto ad opera della subroutine in linguaggio macchina.

   Viene usata l'istruzione "CLEAR" per riservare il necessario spazio di memoria per le subroutine in linguaggio macchina.

Esempio:

```
10 CLEAR 200,&HEFFF
20 AB=&HF000
30 FOR I=AB TO AB+9
40 READ A$:A=VAL("&H"+A$)
50 POKE I,A
60 NEXT I
70 DEFUSR=&HF000
80 INPUT "IMMETTI UN NUMERO INTERO";AX
90 PRINT "NUMERO INTERO=";AX
100 R=USR(AX)
110 PRINT "RISULTATO E' IL NUMERO INTERO PIU' 1";R
120 END
130 DATA 23,23,4E,23,46,03,70,2B,71,C9
RUN
```

144. VAL

Formato: VAL(⟨X$⟩)

Scopo: Dà il valore numerico dell'espressione alfanumerica ⟨X$⟩.

Categoria: Funzione

Note: La funzione trascura gli spazi ed i caratteri di controllo nell'espressione alfanumerica.

Usare "STR$" per ottenere una rappresentazione alfanumerica di un'espressione numerica.

Esempio:

```
 10 AMOUNT=12.34216:PRINT AMAOUNT
20 A$=STR$(AMOUNT)
30 H%=INSTR(A$,".")
40 IF H%<1 GOTO 80
50 H%=H%+2
60 B$=LEFT$(A$,H%)
70 AMOUNT=VAL(B$):PRINT AMOUNT
80 END
RUN
```

145. VARPTR

Formati:

```
VARPTR(<variabile>)
VARPTR(#<X>)
```

Scopo: Dà l'indirizzo di memoria del primo byte di una variabile o del primo byte del blocco di controllo file di un file.

Categoria: Funzione

Note: Il primo formato dà l'indirizzo di memoria della variabile.

Alla variabile deve essere per prima cosa attribuito un valore, altrimenti viene generato il messaggio "Illegal function call" (Richiamo di funzione illecita).

Nella funzione "VARPTR" può essere incluso qualsiasi tipo di variabile (intera, a precisione singola, a precisione doppia, alfanumerica) nonché variabili dimensionate (definite con la istruzione "DIM").

A tutte le variabili non dimensionate deve essere assegnato un valore prima che venga richiamato l'indirizzo di una variabile dimensionata con la funzione "VARPTR". Ciò è necessario dato che gli indirizzi delle variabili dimensionate vengono variati non appena viene usata una nuova variabile non dimensionata.

La funzione "VARPTR" è più spesso usata per ottenere l'indirizzo di memoria di una variabile che deve essere trascritta in una subroutine in linguaggio macchina.

Il secondo formato della funzione dà il primo indirizzo del blocco di controllo file di un file.

<X> è il numero col quale il file è stato aperto con l'istruzione "OPEN".

Il risultato della funzione "VARPTR" è un numero compreso tra -32768 e 32767.
Se viene indicato un numero negativo, occorre aggiungergli dapprima 65536 per poter ottenere l'indirizzo effettivo.

Esempio:

```
10 A=10
20 B=VARPTR(A)
30 IF B<0 THEN B=B+65536
40 C$="0000"+HEX$(B)
50 PRINT RIGHT$(C$,4)
60 END
RUN
```

146. VDP

Formato: VDP(⟨X⟩)

Scopo: Contiene il contenuto dei registri del processore video (VDP)

Categoria: Variabile di sistema

Note: ⟨X⟩ è il numero di registro e deve essere un intero nel campo tra zero e 8.

I contenuti del registro 8 non possono essere cambiati. Possono essere soltanto richiamati.

USARE QUESTA VARIABILE SOLTANTO SE SI HA COMPLETA FAMILIARITÀ CON IL FUNZIONAMENTO DEL VDP (PROCESSORE VIDEO).

Esempio:

```
10 FOR I=0 TO 8
20 A=VDP(I)
30 B$="00000000"+BIN$(A)
40 PRINT RIGHT$(B$,8)
50 NEXT
60 END
RUN
```

147. VPEEK

Formato: VPEEK(⟨X⟩)

Scopo: Dà il contenuto di un byte dalla memoria del video.

Categoria: Funzione

Note: Questa funzione dà il valore decimale dei contenuti di un indirizzo di memoria del video e pertanto avrà un valore nel campo fra 0 e 255.

⟨X⟩ è l'indirizzo della memoria del video e deve avere un valore nel campo fra zero e 16383.

Per inserire in un indirizzo di memoria del video un valore specifico, viene usata l'istruzione "VPOKE".

USARE QUESTA FUNZIONE SOLTANTO SE SI HA COMPLETA FAMILIARITÀ CON IL FUNZIONAMENTO DEL VDP (PROCESSORE VIDEO).

Esempio:

```
10 A=VPEEK(0)
20 A$="00"+HEX$(A)
30 PRINT RIGHT$(A$,2)
40 END
RUN
```

148. VPOKE

Formato: VPOKE ⟨X⟩,⟨Y⟩

Scopo: Inserire in un byte dalla memoria del video un valore dato.

Categoria: Istruzione

Note: ⟨X⟩ è l'indirizzo della memoria del video e deve avere un valore nel campo fra zero e 16383.

⟨Y⟩ è il valore che deve essere inserito nel byte indicato e deve avere un valore compreso fra zero e 255.

Per ottenere il valore di un indirizzo dalla memoria del video, viene usata la funzione "VPEEK".

USARE QUESTA ISTRUZIONE SOLTANTO SE SI HA PIENA FAMILIARITÀ CON IL FUNZIONAMENTO DEL VDP (PROCESSORE VIDEO).

Esempio:

```
10 VPOKE 0,(VPEEK(0))
20 END
RUN
```

149. WAIT

Formato: WAIT ⟨X⟩,⟨Y⟩[,⟨V⟩]

Scopo: Controllare lo stato di un connettore di ingresso ⟨X⟩ della macchina.

Categoria: Istruzione

Note: Durante l'esecuzione di questa istruzione, i dati dal connettore ⟨X⟩ sono confrontati con l'espressione intera ⟨Y⟩ (su base XOR) e con l'espressione intera ⟨Z⟩ (su base AND).
Se il risultato di questo confronto è uguale a zero, MSX-BASIC confronta di nuovo i dati dal connettore ⟨X⟩.
In tutti gli altri casi continua con il programma.

Quando ⟨Z⟩ non viene indicato, si presume automaticamente il valore zero.

Esempio: Vedere "INP" e "OUT".

150. WIDTH

Formato: WIDTH ⟨X⟩

Scopo: Indicare il numero di posizione su una riga di schermo nei modi testo 1 e 2.

Categoria: Istruzione

Note: ⟨X⟩ è il numero di posizioni per riga e deve essere un intero nel campo fra 1 e 40 nel modo testo 1 e nel campo da 1 a 32 nel modo testo 2.

Quando MSX-BASIC viene avviato, il modo testo 1 funziona secondo l'istruzione "WIDTH 37" ed il modo testo 2 secondo la istruzione "WIDTH 29".

Esempio:

```
5 SCREEN 0
10 FOR I=37 TO 10 STEP -1
20 WIDTH I
30 PRINT "QUESTA E' UNA RIGA DI TESTO SULLO SCHERMO"
40 NEXT I
50 END
RUN
```

Fotocomposizione: ARPe/Milano

PHILIPS