VHS - Lüdenscheid

Q B A S I C

für

Anfänger

Inhaltsverzeichnis :

Fachbegriffe :

ASCII	amerikanische Norm für die Darstellung des Alphabetes im Computer
Hardware	Computergeräte, Laufwerke, Bildschirme, Drucker usw. (alles was man anfassen kann)
Software	Programme
IC	Integrierte Schaltung, Baustein aus dem Computer hergestellt werden
Mikroprozessor	Die Schaltung, die die eigentliche Arbeit macht (rechnen, vergleichen, speichern, usw.)
Speicher	Die Schaltungen in denen der Mikroprozessor Daten und Programme speichert
RAM	Speicherschaltung die vom Prozessor beschrieben und gelesen werden kann. Verliert beim Ausschalten alle Daten und Programme
ROM	Speicherschaltung die nur vom Hersteller beschrieben werden kann und vom Prozessor nur gelesen werden kann
Peripherie	Zusatzgeräte z.B. Monitor, Drucker, Laufwerk
Bit	kleinste Informationseinheit im Computer (0 oder 1 bzw. ja oder nein)
Byte	ein 'Computerwort' besteht aus 8 Bit

Cursor	Blinkende Schreibmarke auf dem Bildschirm, die anzeigt wo die nächste Eingabe stattfindet
Betriebssystem	Programm im ROM, das die Arbeit des Computers organisiert
Maschinensprache	Befehle, die der Mikroprozessor direkt verstehen kann. Bestehen aus 1-5 Bytes
Assembler	Programm, das Befehlsworte in Maschinen-sprache-Bytes übersetzt
Programmiersprache	Befehle, die für den Menschen einfach zu lernen sind und in viele einzelne Maschinen-befehle übersetzt werden müssen. (BASIC, PASCAL, FORTRAN, COBOL usw.)
BASIC	Programmiersprache die für Anfänger leicht zu erlernen und universell verwendbar ist
Interpreter	Programm in Maschinensprache, das Befehle einer höheren Programmiersprache in Maschinensprache übersetzt
Compiler	Programm in Maschinensprache, das ein gesamtes Programm einer höheren Sprache in Maschinen-sprache übersetzt

Der Commodore 64 :

Der Commodore 64 ist ein Heimcomputer, der in BASIC programmierbar ist. Dazu enthält das Betriebssystem einen Basicinterpreter, der die vom Benutzer eingegebenen Basicprogramme in die Maschinensprache übersetzt. Insgesamt stehen 64kByte (das heißt 64*1024 = 65536 Byte) RAM-Speicher zur Verfügung. Davon benötigen das Betriebssystem und der Bildschirmspeicher einige Kilobytes. Daher sind für den Benutzer nur etwa 39kByte frei. Zur Eingabe von Daten und Programmen hat der C 64 eine eingebaute Tastatur. Diese entspricht weitgehend der amerikanischen Schreibmaschinentastatur. Zusätzlich sind noch einige Sondertasten vorhanden.
Diese haben folgende Bedeutung :

RETURN Wagenrücklauf bei der Schreibmaschine, beendet eine Zeile und startet die Bearbeitung

SHIFT Tastenumschaltung auf obere Zeichen bzw. Graphiksymbole

SHIFT LOCK dient zum Feststellen der SHIFT-Funktion

CRSR Steuerung des Cursors in die angegebene Pfeilrichtung

INST/DEL Einfügen bzw. Löschen eines Zeichens an der Cursorposition

CLR/HOME Löschen des Bildschirms bzw. Setzen des Cursors in die linke obere Ecke

CTRL zweite Tastenumschaltungsmöglichkeit

RUN/STOP dient zum Anhalten und Weiterlaufenlassen von Programmen

C= Commodoretaste dient zum Erreichen der linken Graphikzeichen sowie mit SHIFT zum Umschalten zwischen Großbuchstaben/Graphikzeichen und Groß-/Kleinbuchstaben

RESTORE dient zusammen mit der RUN/STOP-Taste zum Neustarten des Betriebssystems

Einige der Tasten haben eine Wiederholfunktion, d.h. wenn man sie festhält wird die jeweilige Funktion dauernd wiederholt.

Wenn man einen Befehl oder eine Programmzeile eingegeben hat, muß man immer die RETURN-Taste drücken, damit der Computer die Funktion ausführt.

Als Befehle versteht der C 64 nur die BASIC-Befehle, die im Prinzip normale englische Wörter sind. Diese bezeichnen, was der Rechner machen soll. Wenn der Computer einen Befehl nicht versteht, dann gibt er eine Fehlermeldung aus.

Variablen :

Variablen sind Namen für Speicherplätze, in denen sich der Computer Zahlen oder Wörter merkt, sogenannte 'Platzhalter'. In Programmen benutzt man solche Variablennamen zur allgemeinen Darstellung der Rechnung oder Aufgabe für den Rechner. Während das Programm läuft setzt der Computer für die Namen die entsprechenden Zahlen oder Wörter ein.

Variablennamen dürfen beim Commodore 64 ein oder zweistellig sein und müssen mit einem Buchstaben beginnen. Das zweite Zeichen darf auch eine Zahl sein.

Mögliche Variablennamen wären zum Beispiel:
A, B, AD, F6, XY usw.

Nicht erlaubt ist folgendes:
_3B ,A/ ,T& usw.

Prinzipiell dürfen die Namen zwar mehr als zwei Stellen haben, aber es werden nur die ersten beiden Stellen unterschieden. Daher kann es zu Fehlern führen, wenn man längere Namen verwendet.

So sind zum Beispiel folgende Variablen für den Computer gleich, obwohl sie unterschiedliche Namen haben :

ALTER und ALPHABET , KILOMETER und KISTEN , usw._

Es gibt grundsätzlich drei verschiedene Variablentypen beim C 64 :

INTEGER Integervariablen sind Variablen, die nur ganze Zahlen zwischen -32768 und +32767 enthalten dürfen.Diese Variablen werden durch ein angehängtes Prozentzeichen dargestellt. Man benutzt diesen Typ z.B. für Zähler.

Beispiel:

A% = 5 , BS% = 1243 usw.

REAL Realvariablen sind der Standardtyp für die Speicherung von Zahlen. Sie dürfen ganze oder gebrochene Zahlen im Bereich von -1.70141183 E+38 bis +1.70141183 E+38 enthalten. Dabei werden Zahlen, die kleiner sind als 2.93873588 E-39 als Null angesehen. Dieser Variablentyp wird ohne Kennzeichen dargestellt.

Beispiel:

A = 2.345 E+12 , XY = 3 , WI = -2.5613 usw.

STRING Stringvariablen enthalten keine Zahlen, sondern Zeichen. Sie werden zur Speicherung von Buchstaben, Zeichen oder Wörtern gebraucht. Sie können auch Ziffern enthalten, aber der Computer kann dann nicht mit diesen Ziffern rechnen. Der Variablentyp wird dargestellt durch ein angehängtes Dollarzeichen.

Beispiel:

A$ = "Hallo" , BX$ = "12 Uhr" , XY$ = "a,b;c:d!"

Aus- und Eingabe-Befehle :

PRINT Der Print-Befehl dient zur Ausgabe von Texten, Zahlen oder Graphiksymbolen

Beispiel:

PRINT 5 schreibt die Zahl 5 auf den Bildschirm

PRINT A schreibt den Inhalt der Variablen A

PRINT "Hallo" schreibt das Wort Hallo

_
Mehrere Ausgaben können durch Kommata oder
Semikoli getrennt hinter einen PRINT-Befehl
geschrieben werden. Ein Semikolon hängt die
Ausgaben direkt aneinander und ein Komma
erzeugt einen Zwischenraum.

_
Beispiel:

PRINT 5;6 schreibt 5 6 auf den Bildschirm

PRINT 5,6 schreibt 5 6

PRINT "Hallo";"Du" schreibt HalloDu

PRINT "Hallo","Du" schreibt Hallo Du

_
? das Fragezeichen kann als Abkürzung für PRINT eingegeben werden

TAB(x) setzt das nächste Zeichen auf die Stelle Nummer 'x' in der Zeile

Beispiel:

PRINT TAB(30);"Hallo" schreibt Hallo an die 30. Stelle

SPC(x) schreibt 'x' Leerzeichen

Beispiel:

PRINT "Hallo";SPC(5);"Du" schreibt 5 Leerzeichen zwischen Hallo und Du

INPUT dient zur Eingabe von Daten an Variablen von der Tastatur

Beispiel:

INPUT X fragt mit einem Fragezeichen nach einer Zahl, die der Benutzer eingeben soll. Die Zahl wird unter dem Namen 'X' gespeichert.

INPUT "Zahl = ";X schreibt zunächst 'Zahl = ' auf den Bildschirm und erwartet dann die Eingabe

Will man Buchstaben oder Wörter eingeben, so kann man dies auch mit INPUT machen.

_
Beispiel:

INPUT A$ erwartet die Eingabe eines Buchstaben oder Wortes, das dann als A$ gespeichert wird

GET GET dient zur Abfrage der Tastatur. Es muß ebenfalls eine Stringvariable angegeben werden.

_
Beispiel:

GET A$ holtein einzelnes Zeichen von der
Tastatur und speichert es in A$

_
Der GET-Befehl fragt die Tastatur immer nur ein einziges Mal ab. Deshalb muß er in einer Schleife wiederholt werden bis eine Taste gedrückt wurde.
Man kann auch den Befehl GET A benutzen, aber dann darf nur eine Zifferntaste gedrückt werden, da es sonst eine Fehlermeldung gibt.

READ READ dient auch zur Eingabe von Daten, aber nicht von der Tastatur, sondern von DATA-Zeilen im Programm. Dies wird benötigt, wenn man viele Daten fest im Programm speichern will, weil man sie bei jedem Programmlauf benötigt. Der Befehl braucht immer eine oder mehrere DATA-Zeilen.

_
Beispiel:

READ A,B
DATA 5,8,4,7
diese beiden Befehle lesen die Zahlen 5 und 8 und speichern sie als A und B. Beim nächsten READ-Befehl würden die 4 und die 7 gelesen.

_
DATA gehört zu READ und beinhaltet die Daten

RESTORE dient zum Zurücksetzen des READ-Zeigers

: Mit dem Doppelpunkt können mehrere Befehle in einer Zeile eingegeben werden.

_
Beispiel:

```
PRINT A : PRINT 'TEXT" : INPUT "ZAHLEN = ";A,B
```
_

Mathematische Funktionen:

= dient zur Zuweisung eines Wertes an eine Variable

_Beispiel:

A = 5 speichert 5 in A

B = A speichert den Inhalt von A (also 5) in B

X$ = "Hallo" speichert das Wort Hallo in X$

_

+ dient zur Addition zweier Werte

_Beispiel:

PRINT 5 + 6 zeigt 11 an

_

- dient zur Subtraktion zweier Werte

_Beispiel:

PRINT 23 - 12 zeigt 11 an

_

* dient zur Multiplikation zweier Werte

_Beispiel:

A = 5 * 3 speichert 15 in A

_

/ dient zur Division zweier Werte

_Beispiel:

D = 18 / 6 speichert 3 in D

_

^ dient zur Potenzierung einer Zahl mit einer anderen

_Beispiel:

PRINT 4 ^ 2 zeigt 16 an (4 hoch 2)

_

SIN(x) berechnet den Sinus der Zahl 'x' in Radian
_Beispiel:

PRINT SIN(1.570796) zeigt 1 an (sin(pi/2))
_
COS(x) berechnet den Cosinus der Zahl 'x' in Radian
_Beispiel:

A = COS(3.141592) speichert -1 in der Variablen A
_
TAN(x) berechnet den Tangens der Zahl 'x' in Radian
_Beispiel:

Y = TAN(.7853981) speichert 1 in Y (tan(pi/4))
_
SQR(x) berechnet die Quadratwurzel von 'x'
_Beispiel:

PRINT SQR(9) zeigt 3 an
_
EXP(x) berechnet die Potenz 'x' zur Eulerschen Zahl e (2.7182818)
_Beispiel:

PRINT EXP(3) zeigt 20.085536 an (e hoch 3) _
ATN(x) berechnet den Arcustangens der Zahl 'x'
_Beispiel:

PRINT ATN(1) zeigt .78539816 an (pi/4)
_
LOG(x) berechnet den natürlichen Logarithmus zur Basis e
_Beispiel:

PRINT LOG(5) zeigt 1.609437 an
_

ABS(x) bildet den Betrag von 'x'
_Beispiel:

PRINT ABS(-3),ABS(5) zeigt 3 5 an

_
INT(x) bestimmt die nächstkleinere Zahl von 'x'
_Beispiel:

PRINT INT(-3.12),INT(5.27) zeigt -4 5 an

_
SGN(x) bestimmt das Vorzeichen von 'x'
_Beispiel:

SGN(-4.35) = -1 ; SGN(0) = 0 ; SGN(3.25) = 1

_
RND(1) erzeugt eine Zufallszahl zwischen 0 und 1
Die Zahl in der Klammer ist unwichtig, sie dient nur der einheitlichen Schreibweise des Befehls. Es kann jede positive Zahl eingesetzt werden.Eine negative Zahl ergibt eine konstante 'Zufallszahl'. Eine Null ergibt immer eine Neue.
_Beispiel:

PRINT RND(1) zeigt .1234567 an

_
PEEK(x) liest den gespeicherten Zahlenwert aus der Speicherstelle mit der Nummer 'x'. Es sind Nummern zwischen 0 und 65535 möglich.
_Beispiel:

PRINT PEEK(1060) zeigt wahrscheinlich 32 an, die Nummer des Leerzeichens auf dem Bildschirm.

_
FN xxx definiert eine neue Funktion
_Beispiel:

DEF FN F(X) = SIN(X)/COS(X)

Die Funktion FN F(X) ist jetzt eine neue Funktion
PRINT FN F(5) würde SIN(5)/COS(5) berechnen und
anzeigen

_
FRE(0) ermittelt den noch freien Speicherplatz
_Beispiel:

PRINT FRE(0) schreibt z.B. 37652

_
POS(0) ermittelt die Position des Cursors in der Zeile
_Beispiel:

PRINT POS(0) schreibt 10

_

Befehle und Kommandos :
---

1 - 65000 diese Zahlen sind als Zeilennummern zugelassen.
Unabhängig von der Eingabe werden die Zeilen immer in der Reichenfolge der Nummern sortiert.

RUN startet ein Basic-Programm in der ersten vorhanden Zeile. Bei einem Start mit RUN
werdenimmer zuerst alle Variablen gelöscht.

RUN x startet das Programm in Zeile Nummer 'x'

LIST zeigt das gesamte vorhandene Programm an. Mit
der CTRL-Taste kann die Ausgabe gebremst werden.

LIST x-y listet das Programm von Zeile 'x' bis Zeile 'y'

LIST -y listet das Programm vom Anfang bis Zeile 'y'

LIST x- listet das Porgramm von Zeile 'x' bis zum Ende

CONT läßt das Programm nach einem STOP oder nach Betätigung der RUN/STOP-Taste weiterlaufen, aber nur wenn keine Änderung vorgenommen wurde.

STOP als Befehl im Programm hält das Programm an,
z.B. um es auf Fehler zu untersuchen.

END bezeichnet das Ende des Programmes

NEW löscht das Programm aus dem Speicher

CLR löscht alle Variablen

REM markiert eine Kommentarzeile, alle Befehle
hinter REM innerhalb der Zeile werden ignoriert.

OPEN a,b,c öffnet den Kanal mit der Nummer 'a' zum Gerät Nummer 'b' mit der Sekundäradresse 'c'. Es kann ein Filename angegeben werden.

_Beispiel: OPEN 1,8,2,"Daten,S,W" öffnet den Kanal zur Floppy um Daten in eine sequentielle Datei zu schreiben

_
CLOSE a schließt den Kanal 'a'

PRINT# schreibt Daten in einen Kanal

INPUT# liest Daten aus einem Kanal

POKE a,b schreibt den Wert 'b' direkt in den Speicher an die Speicherstelle Nummer 'a'

SYS a startet ein Maschinenprogramm an der Speicherstelle 'a'

USR(a) holt den Wert aus einem Maschinenprogramm

WAIT a,b,c wartet bis in Speicherstelle 'a' ein bestimmter
Wert vorliegt

Schleifen :

Häufig müssen bestimmte Rechnungen oder Befehle mehrmals durchgeführt werden. Um dabei nicht mehrmals den gleichen Befehl eingeben zu müssen verwendet man sogenannte Schleifen. Dabei wiederholt der Rechner einen Programmteil solange, bis eine bestimmte Bedingung erfüllt ist.

Eine Schleife kann folgendermaßen aussehen :

```
10 A = 1 Anfangswert
20 PRINT A A anzeigen
30 A = A + 1 A um 1 erhöhen
40 IF A < 10 THEN GOTO 20 wenn A kleiner als 20 ist, soll bei
 20 weitergemacht werden
50 PRINT "ENDE"sonst ist das Programm zu Ende
60 END
```

Die gleiche Funktion kann auch mit folgenden Befehlen erreicht werden :

```
10 FOR A = 1 TO 20 Schleifenzähler A von 1 bis 20
20 PRINT A A anzeigen
30 NEXT A Da nächste A ist A plus 1
40 PRINT "ENDE"fertig
50 END
```

Will man den Schleifenzähler nicht um 1 weiterzählen, so kann man die Schrittweite angeben.

```
10 FOR A = 1 TO 20 STEP .5 Zähler von 1 bis 20 um 0.5 erhöhen
20 PRINT A A anzeigen
30 NEXT A Nächstes A ist A plus 0.5
40 PRINT "ENDE"fertig
50 END
```

Es können mehrere FOR...NEXT-Schleifen ineinander geschachtelt werden, sie dürfen sich jedoch nicht überschneiden.

Hinter dem NEXT-Befehl kann man die Variable weglassen, dann wird die zuletzt geöffnete Schleife geschlossen.

Außerdem können mehrere Variablen gleichzeitig angegeben werden.
Beispiel: NEXT A,B oder NEXT A : NEXT B oder NEXT : NEXT

Bedingungen :

Manchmal dürfen einige Befehle nur ausgeführt werden, wenn eine bestimmte Bedingung erfüllt ist. Für solche Bedingungen gibt es den folgenden Befehl :

IF bedingnung THEN befehle

Als Bedingung können Variablenvergleiche eingesetzt werden. Hinter dem THEN dürfen beliebige Befehle stehen, die dann ausgeführt werden sollen, wenn die Bedingung erfüllt ist.Falls die Bedingung nicht erfüllt ist, werden die Befehle hinter THEN ignoriert und der Rechner arbeitet in der nächsten Zeile weiter.

_ Beispiele:

```
10 INPUT A
20 IF A = 5 THEN PRINT "Erraten" : GOTO 10
30 IF A < 5 THEN PRINT "zu klein" : GOTO 10
40 IF A > 5 THEN PRINT "zu groß" : GOTO 10

100 GET A$ : IF A$="" THEN 100
```
_

Als Vergleichsbefehle können folgende Zeichen eingesetzt werden :

a = b prüft auf Gleichheit

a < b prüft ob 'a' kleiner ist als 'b'

a > b prüft ob 'a' größer ist als 'b'

a <> b prüft ob 'a' und 'b' ungleich sind

a <= b prüft ob 'a' kleiner oder gleich 'b' ist

a >= B prüft ob 'a' größer oder gleich 'b' ist
.pa
Mehrere Bedingunge können mit AND oder OR verknüpft werden.

_Beispiel: IF a < 4 OR a> 8 THEN ....
Die Bedingung ist nur dann erfüllt, wenn 'a' kleiner ist als 4 oder größer als 8

IF b > 3 AND b < 7 THEN ....
Die Bedingung ist erfüllt, wenn 'b' zwischen 3 und 7 liegt

IF a = 3 AND b = 2 THEN ....
Die Bedingung ist erfüllt, wenn 'a' gleich 3 ist und 'b' gleich 2
_
Eine weiterer bedingter Befehl ist ON ... GOTO/GOSUB ....
Bei diesem Befehl wird in Abhängigkeit von dem Zahlenwert der Bedingung ein Sprung ausgeführt.
_Beispiel:

ON a GOTO 100,200,300 wenn 'a' gleich 1 ist wird nach 100 gesprungen, wenn 'a' gleich 2 ist nach 200 usw.

_

Felder :

Zusätzlich zu den 'normalen' Variblentypen gibt es auch noch die Möglichkeit sogenannte 'Felder' zu verwenden.Felder sind Variablen, die zusammengehören und deshalb den gleichen Namen haben. Zur Unterscheidung der einzelnen Feldelemente werden diese durchnummeriert. Man kann sich zum Beispiel ein Schachbrett als zweidimensionales Feld vorstellen. Es hat den Namen 'Schachbrett' und die einzelnen Elemente A1,A2,A3,...,B1,B2,B3,... usw.
Ein eindimensionales Feld wäre z.B. ein Regal mit dem Namen
'Regal' und den Elementen 'Fach1,Fach2,Fach3,...'.
Ebenso kann man Variablenfelder erzeugen. Zum Beispiel das Feld
A, es hätte als zweidimensionales Feld die Elemente
A(1,1),A(1,2),A(1,3),...,A(2,1),A(2,2),... usw.
Da der Computer wissen muß, wie groß ein Feld ist, müssen wir es vor der ersten Verwendung dimensionieren. Dazu gibt es den Befehl DIM.
_Beispiele:
---
10 DIM A(30,20) dimensioniert ein 2-D-Feld von 30*20
20 FOR I = 1 TO 30 zählt Spalten von 1 bis 30
30 FOR J = 1 TO 20 zählt Zeilen von 1 bis 20
40 PRINT A(I,J), zeigt Feldelemente an
50 NEXT J,I schließt Schleifen ab
60 END

_
Felder die weniger als 10 Elemente pro Dimension haben, brauchen nicht dimensioniert zu werden. Jede Variablenart kann als Feld strukturiert werden (z.B. A$(12,35)).

Zeichenketten :

Zeichenketten, auch Strings genannt, werden in Stringvariablen gespeichert, die den Zusatz '$' an dem Variablennamen tragen. In solchen Strings können bis zu 255 Zeichen gespeichert werden. Wenn man nun aus einem String Zeichen herauslösen will gibt es dafür mehrere Befehle:

LEFT$(x$,a) löst aus x$ von links a Zeichen heraus
_ Beispiel:

PRINT LEFT$("ABCDEF",3) schreibt ABC
_
RIGHT$(x$,a) löst aus x$ von rechts a Zeichen heraus
_ Beispiel:

PRINT RIGHT$("ABCDEF",3) schreibt DEF
_
MID$(x$,a,b) löst aus x$ an der Stelle a b Zeichen aus
_ Beispiel:

PRINT MID$("ABCDEF",3,2) schreibt CD
_
Weitere Befehle für Strings :

LEN(x$) bestimmt die Länge von x$
_ Beispiel:

PRINT LEN("ABCDEF") schreibt 6
_
ASC(x$) bestimmt die ASCII-Nummer des ersten
Zeichens von x$
_ Beispiel:

PRINT ASC("A") schreibt 65
_

CHR$(a) bildet einen String aus dem Zeichen Nummer a
_ Beispiel:

A$ = CHR$(5) : PRINT A$ schreibt A

_
STR$(a) bildet einen String aus der Zahl a
_ Beispiel:

A$ = STR$(A) wandelt die Zahlenvariable A
in eine Stringvariable A$ um

_
VAL(x$) bildet eine Zahl aus den Ziffern eines
Stringes
_ Beispiel:

A = VAL("123ABC") speichert 123 in A

_

Unterprogramme :

Programmteile die häufig gebraucht werden, schreibt man nicht jedesmal neu an die Stelle, wo sie benötigt werden, sondern nur einmal als Unterprogramm, welches man mit GOSUB xxx aufrufen kann. Unterprogramme enden mit dem Rücksprungbefehl RETURN.
Innerhalb von Unterprogrammen darf der Befehl CLR nicht benutzt werden, da er nicht nur die Variablen, sondern auch die Rücksprungadresse löscht.

_Beispiel:

```
10 GOSUB 1000
20 PRINT A
30 GOSUB 1000
40 PRINT A^2
50 END

1000 REM Unterprogramm
1010 PRINT CHR$(147) : REM Bildschirm löschen
1020 INPUT "Zahl = ";A
1030 RETURN
```

_
Ein Unterprogramm darf auch ein weiteres Unterprogramm aufrufen. Der RETURN-Befehl kehrt immer zu dem Befehl zurück, der hinter dem GOSUB steht.

Abspeichern und Laden :

Da der Speicher beim Ausschalten des Rechners alle Daten und Programme verliert muß man die Programme vor dem Ausschalten auf einer Diskette oder Cassette sichern.

Dazu dient der Befehl SAVE "programmname"

_Beispiele:

SAVE "name" speichert das Programm aus dem Speicher auf Cassette unter dem Namen 'name'

SAVE "name",8 speichert das Programm auf Diskette
_

Zum Einladen in den Rechner gibt es den Befehl LOAD.

_Beispiele:

LOADlädt das nächste Programm von Cassette

LOAD "name" lädt das Programm 'name' von Cassette

LOAD "$",8 lädt das Inhaltsverzeichnis von Diskette
(anzeigen mit LIST)

LOAD "name",8 lädt das Programm 'name' von Diskette
_

Mit dem Befehl VERIFY kann geprüft werden, ob das Programm richtig abgespeichert wurde.

_Beispiel:

VERIFY "name" prüft das Programm 'name' auf Cassette

VERIFY "name",8 prüft das Programm 'name' auf Diskette

_
Zum Abspeichern von Daten gibt es den Befehl PRINT#. Vorher muß allerdings ein Kanal geöffnet werden.

_Beispiel:

OPEN 1,8,2,"DATEN,S,W" öffnet den Kanal 1 zu Diskette zum Schreiben von Daten
PRINT#1,A,B,C schreibt die Variablen A,B und C auf Diskette
CLOSE 1 schließt den Kanal wieder
_

Gelesen werden die Daten mit INPUT#.

_Beispiel:

OPEN 1,8,2,"DATEN,S,R" öffnet den Kanal 1 zu Diskette zum Lesen
INPUT#1,A,B,C liest die Daten von Diskette in A,B und C
CLOSE 1 schließt den Kanal wieder
_

Drucken :

Die Ausgabe auf den Drucker wird ebenfalls mit dem PRINT#-Befehl ausgeführt.

_Beispiel:

OPEN 1,4 öffnet den Kanal 1 zum Drucker
PRINT#1,A$;B gibt A$ und B an den Drucker aus
CLOSE 1 schließt den Kanal wieder

_

Will man ein Programmlisting auf Papier drucken so geht das folgendermaßen :

_Beispiel:

OPEN 1,4 öffnet den Kanal 1 zum Drucker
CMD 1 schickt alle Kommandos zum Kanal 1
LIST listet alle Zeilen auf dem Drucker
CLOSE 1 schließt den Kanal wieder

_

Fehlermeldungen :

BAD DATA Es wurden Zahlen erwartet aber Strings eingegeben

BAD SUBSCRIPT Feldelementnummer liegt außerhalb des Bereiches

CAN'T CONTINUE Vor CONT wurde das Programm geändert

DEVICE NOT PRESENT Angesprochenes Gerät nicht vorhanden

DIVISION BY ZERO Division durch Null ist nicht erlaubt

EXTRA IGNORED überzählige Daten bei INPUT ignoriert

FILE NOT FOUND gesuchtes File existiert nicht

FILE NOT OPEN Kanal wurde nicht geöffnet

FILE OPENKanal wurde mehrmals geöffnet

FORMULA TOO COMPLEX Stringbefehl zu umfangreich

ILLEGAL DIREKT INPUT nur im Programm

ILLEGAL QUANTITY Funktionswert außerhalb des Bereiches

LOAD ERRORLadefehler bei Cassette oder Diskette

NEXT WITHOUT FOR Schleifen falsch verschachtelt

NOT INPUT FILE Lese-File wurde zum Schreiben geöffnet

NOT OUTPUT FILE Schreib-File wurde zum Lesen geöffnet

OUT OF DATA keine Daten mehr in DATA-Zeile für READ

OUT OF MEMORY kein Speicherplatz mehr vorhanden

OVERFLOW Zahl größer als 1.70141183 E+38

REDIM'D ARRAY Feld nur einmal dimensionieren

REDO FROM START wiederhole INPUT mit richtigem Datentyp

RETURN WITHOUT GOSUB ein RETURN ohne GOSUB gefunden

STRING TOO LONG String hat mehr als 255 Zeichen

SYNTAX ERROR Befehl falsch geschrieben oder unbekannt

TYPE MISMATCH Zahl statt String, oder umgekehrt, benutzt

UNDEF'D FUNKTION Funktion wurde nicht definiert

UNDEF'D STATEMENT Zeile hinter GOTO, GOSUB, RUN nicht da

VERIFY ERROR Programm nicht korrekt auf Diskette
