Einführung

HEIMCOMPUTER robotron Z 9001

PROGRAMMIERSPRACHE BASIC

VEB ROBOTRON-MESSELEKTRONIK >OTTO SCHÖN< DRESDEN
DDR-8012 Dresden, Lingneralleе 3, Postschließfach 211

Änderungen/Ergänzungen

zur Ausgabe 8/84a der Einführung in die Programmiersprache BASIC

Seite 3

ersetzen letzten Absatz durch:

"Es wird die Eingabe der höchsten für den BASIC-Interpreter verfügbaren RAM-Speicheradresse erwartet (s. Programmierhilfen Tabelle 5). So kann z.B. ein Speicherbereich für Maschinenprogramme reserviert werden."

Seite 16, vorletzte Zeile

ersetzen "eine Zahl"
durch "eine ganze Zahl"

Seite 24, Zeile 2

ersetzen "±2.9387E-39"
durch "∓9.40396E-39"

Seite 42

einfügen nach Zeile 8

"Das Feld, auf das durch die Anweisung CLOAD* gelesen werden soll, ist vorher so zu dimensionieren, wie das mit CSAVE* ausgelagerte. Zusätzlich ist beim Einlesen von Zeichenkettenfeldern darauf zu achten, daß der Stringspeicherbereich hinreichend groß ist (Standard 256 byte), anderenfalls ist er durch eine Anweisung CLEAR I entsprechend einzurichten."

Seite 46

ersetzen Zeile 26 durch

"löscht, dabei muß Zeilennummer 1 kleiner Zeilennummer 2 sein."

Seite 53

streichen ab "Beispiel 4:" bis Seite 54, Zeile 2

Seite 54, Zeile 3

ersetzen "13.19. GOSUB"
durch "13.19. GOSUB...RETURN"

Seite 58

einfügen nach Zeile 16

"Tritt nach einem LIST#-Kommando ein IO-Error auf, so bleibt der dort intern eingestellte Wert für LINES (65536) erhalten und muß von Hand zurückgesetzt werden."

Seite 59

einfügen nach Zeile 24

"Die eingelesenen Quellzeilen werden auf dem Bildschirm angezeigt."

Seite 60

einfügen nach Zeile 5

"Zu beachten ist, daß die zuletzt gültige Größe des Stringspeicherbereiches (s. CLEAR) nicht verändert wird."

Seite 60

streichen Zeilen 8 bis 11

Seite 60, Zeile 26

ersetzen "in der nächsten Programmzeile"
durch "mit der nächsten Anweisung"

Seite 66

einfügen nach Zeile 22

"Bemerkung: Durch PRINT AT wird die aktuelle Position des Kursors nicht beeinflußt. Ein Überschreiben des Zeichens auf der Kursorposition kann aber dazu führen, daß nach den Anweisungen PRINT oder WINDOW das Zeichen wieder gelöscht wird. Ebenso werden durch PRINT TAB(I)... bzw. PRINT SPC(I)... alle Zeichen von alter bis neuer Kursorposition gelöscht."

Seite 69

einfügen nach Zeile 10

"wird ein Programmabschnitt neu numeriert, so ist darauf zu achten, daß keine Zeilennummern doppelt auftreten und die aufsteigende Folge der Zeilennummern erhalten bleibt. Anderenfalls kann es zu Abarbeitungsfehlern kommen."

Seite 72

einfügen nach Zeile 22

"Nach STOP bleiben alle aktuellen Variablenwerte und Systemzustände erhalten, einschließlich der durch CLEAR, WINDOW, NULL oder LINES gesetzten."

Seite 78

einfügen nach Zeile 17

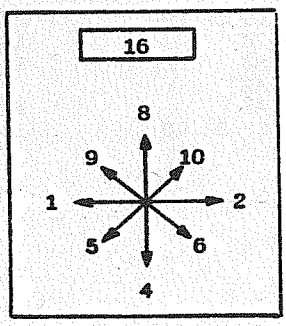
"FRE(X) liefert eine ganze Zahl zwischen -32768 und +32767. Ist der Wert kleiner als Null, so ergibt sich die Anzahl der noch freien Speicherplätze aus 65536 + FRE(X)."

Seite 81

einfügen nach Zeile 5

"Während der Abfrage der Spielhebel im Programm sollte keine Tastaturbetätigung erfolgen, da die Funktion JOYST dann Werte liefern kann, die keiner Spielhebelposition entsprechen. Außerdem ist zu beachten, daß durch die Spielhebelbedienung der Arbeitszutand der Tastatur verändert werden kann, der dann erforderlichenfalls von Hand zurückgesetzt werden muß (GRAPHIC!).

Codierung der Spielhebelposition:



Seite 82

ersetzen Zeile 10 ab "Die Werte von ..."
durch "Es muß gelten: 1 ≤ I ≤ 255
Ø ≤ J ≤ 255."

Seite 88

einfügen nach Zeile 28

"Weitere Hinweise siehe Anhang A4."

Seite 94, Zeile 10

ersetzen "NOTD#"
durch "NOT"

Seite 96 ergänzen

Hinweise zur Anwendung der Funktion USR(X)

Die Übernahme des Parameters erfolgt, indem aus dem Maschinencodeprogramm die Routine EPRVL3 des BASIC-Interpreters aufgerufen wird (CALL EPRVL3). Der Wert des Parameters steht dann im Registerpaar DE, er muß im Bereich von -32768 bis +32767 liegen. Die Übermittlung des Ergebnisses an das BASIC-Programm erfolgt durch Aufruf der Routine FRE3 (JMP FRE3). Das Ergebnis muß vorher in den Registern A und B bereitgestellt werden, ebenfalls als vorzeichenbehaftete Integerzahl. Das Maschinencodeprogramm darf andere Register nicht verändern.

Adressen

BASIC-Interpreter auf	RAM	ROM
WSP+4	2AØ4H (10756)	3Ø4H (772)
EPRVL3	ØD6FH	ØC96FH
FRE3	14B1H	ØDØB1H

Beispiel:

ab Adresse 25ØH (592)

CALL	EPRVL3	CD 6F C9	2Ø5	111	2Ø1
LD	A,E	7B	123		
ADD	E	83	131		
LD	B,A	47	71		
XOR	A	AF	175		
JMP	FRE3	C3 B1 DØ	195	177	2Ø8

```
10 DOKE 772, 592
20 A = 5
30 B = USR(A)
40 PRINT A,B

RUN

 5                    1Ø
```

Einführung in die Programmiersprache BASIC
des HEIMCOMPUTERS robotron Z9001

Inhaltsverzeichnis — Seite

1. Einleitung 2
2. Start des BASIC-Interpreters 3
3. Einführung in die Programmiersprache BASIC 4
4. Zeichensatz des Z9001 22
5. Konstanten 22
6. Variable 25
7. Ausdrücke und Operationen mit Zahlen 28
8. Operationen mit Zeichenketten 31
9. Spezielle Tasten zur BASIC-Unterstützung 31
10. Laden und Abarbeiten von BASIC-Programmen 33
11. Beschreibungsform der BASIC-Sprachelemente 34
12. Zeilenformat 35
13. BASIC-Anweisungen und -Kommandos 37
14. BASIC-Funktionen 75

Anhang

A1 Fehlermeldungen 90
A2 Bemerkungen zum Speicherplatzbedarf 93
A3 Reservierte Worte 94
A4 Spezielle Programmierhinweise 95

8/84 a

1. Einleitung

BASIC ist eine auch für ungeübte Anwender schnell erlernbare Programmiersprache, die vor allem den Dialog mit dem Rechner, z.B. die Einflußnahme auf die Programmabarbeitung, sehr gut unterstützt.

Mit Hilfe des BASIC des Heimcomputers robotron Z 9001 können sowohl einfache mathematische Operationen (im sogenannten "Kommando-Modus") sofort ausgeführt, als auch Programme eingegeben, abgearbeitet und auf Magnetbandkassette gespeichert werden. Auch die Ausgabe von Programmen, Rechenergebnissen und anderen Informationen auf einen Drucker ist möglich. Ebenso können die Heimcomputer-spezifischen Ergänzungsmodule (Farbmodul, Spielhebel u.a.) sowie der für die Gestaltung von Spielen bedeutsame Grafik-Zeichensatz (128 spezielle Grafiksymbole) vom BASIC einfach angesprochen werden. Weiterhin lassen sich digitale Informationen von angeschlossenen Meßgeräten abrufen und weiterverarbeiten und auch Steuersignale an entsprechende Geräte übertragen.

Das BASIC des Z 9001 ist in seinem Aufbau (Anweisungs- und Kommando-Vorrat, syntaktische Struktur) dem international weitverbreiteten Standard angeglichen, so daß eine Übertragung von Programmen vieler anderer Rechner auf den Z 9001 in der Regel leicht fallen wird.

Der BASIC-Interpreter kennt zwei typische Arbeitszustände:

- den Kommando-Modus, in dem die Tastatureingabe von Kommandos durch den Nutzer erfolgt. Deren Ausführung ist im wesentlichen Voraussetzung für die Programmabarbeitung bzw. beeinflußt diese (Programm starten, unterbrechen, fortsetzen, speichern, laden, anzeigen, ändern,...).

- den Programm-Modus, innerhalb dessen die Abarbeitung von Anweisungen des gestarteten BASIC-Programms stattfindet. Diese dienen der eigentlichen Problemlösung und werden interpretierend

abgearbeitet, d.h. in vorgegebener Reihenfolge einzeln analysiert und unmittelbar ausgeführt.

Nach einführenden Bemerkungen in den Abschnitten 2 und 3 wird das BASIC des Z9001 in den folgenden Abschnitten ausführlicher dargestellt.

2. Start des BASIC-Interpreters

Für den Z9001 gibt es den BASIC-Interpreter als steckbaren Modul und als ladbares Programm von der Kassette. Ist der BASIC-Interpreter als Zusatzmodul gesteckt, so wird er vom Monitor (Steuerprogramm) aus mittels der Eingabe

BASIC [ENTER]

gestartet. Die Kassettenvariante des Interpreters wird nach der Eingabe

BASIC [ENTER] in den

Hauptspeicher geladen und selbständig gestartet. Das Laden des BASIC-Interpreters ist in der Weise vorzunehmen, wie es in der Bedienungsanleitung angegeben ist. Der Wiederstart des Interpreters erfolgt mit dem Kommando

WBASIC [ENTER]

In diesem Fall bleibt ein BASIC-Programm, welches sich schon im Speicher befindet, erhalten. Die Werte der Variablen bleiben unverändert.

Nach dem Start meldet sich der Interpreter mit der Eingabeanforderung

MEMORY SIZE ?

Es wird eine Längenangabe in Bytes, die das Ende des nutzbaren Arbeitsspeicherbereiches definiert, erwartet. So kann z.B. ein Speicherbereich für Maschinenprogramme reserviert werden.

Wird nur die Taste [ENTER] gedrückt, wird vom Interpreter der gesamte freie Arbeitsspeicher (RAM-Bereich) benutzt. Es wird die für Programme nutzbare Länge in Bytes auf dem Bildschirm ausgeschrieben. Anschließend wird die Meldung OK ausgegeben. Der Interpreter befindet sich jetzt im Kommando-Modus.

Der Kommando-Modus wird stets durch das Zeichen ">" in der ersten Spalte angezeigt. Der Rechner wartet dann auf die Eingabe von Kommandos oder Anweisungen, die mit der Taste [ENTER] abgeschlossen werden. Nach [ENTER] führt der Rechner das eingegebene Kommando aus.

3. Einführung in die Programmiersprache BASIC

3.1. Einfache Rechenoperationen

Wie bereits erwähnt, können im BASIC ähnlich wie mit einem Taschenrechner einfache mathematische Rechenoperationen ausgeführt werden, wobei Aufgaben und Ergebnisse am Bildschirm (24 Zeilen, 40 Spalten) sichtbar sind.

Dazu stehen die Grundoperationen +, -, *, /, und ^ (Potenzsymbol), sowie die Standardfunktionen (vgl. auch Abschnitt 14)

ABS(X) - Absoluter Betrag
ATN(X) - Arcustangens
COS(X) - Cosinus (X in Bogenmaß)
EXP(X) - Exponentialfunktion e^X
INT(X) - Ganzer Teil
LN (X) - Natürlicher Logarithmus
RND(X) - Zufallszahl
SGN(X) - Vorzeichenfunktion
SIN(X) - Sinus (X in Bogenmaß)
SQR(X) - Quadratwurzel
TAN(X) - Tangens (X in Bogenmaß)
PI - Konstante π

zur Verfügung. Diese Funktionen können beliebig auf numerische Größen (Zahlen, Variable oder Ausdrücke) angewendet und miteinander verknüpft werden.

Dabei entsteht jeweils ein "numerischer Ausdruck", zum Beispiel

```
                5*7+SIN(PI*45/180)
oder            -7*(4+EXP(2))
```

Will man das Ergebnis einer solchen Rechenoperation am Bildschirm anzeigen, so muß dem Ausdruck das Schlüsselwort PRINT ("Drucke") vorangestellt werden.

Die Eingabe des gesamten Befehls erfolgt zeichenweise über die Tastatur, wobei jede Befehlszeile durch Betätigung der [ENTER] - Taste abgeschlossen wird.

Der Rechner arbeitet die Aufgabe sofort nach [ENTER] ab und zeigt das Ergebnis an.

Beispiel: Es wird eingegeben

```
PRINT 5*7+8*6 [ENTER]
```

Auf dem Bildschirm sind dann Aufgabe und Ergebnis in folgender Weise dargestellt:

```
PRINT 5*7+8*6
 83
OK
>■
```

"OK" quittiert dabei die vollständige Abarbeitung der Rechenoperation. Ab der Position des Kursors (■) kann nun die Eingabe einer weiteren Aufgabe beginnen.

Zur Verkürzung der Eingabe kann das Schlüsselwort PRINT auch durch ein Fragezeichen ("?") ersetzt werden. Eine Folge von Aufgaben und Ergebnissen ergibt am Bildschirm dann

```
?SIN(PI*45/180)
 .707107
OK
?-5*7
-35
OK
?0.5*LN(10)/(5-3*5)
-.115129
OK
>
```

wobei die Zeilen 1, 4 und 7 jeweils über die Tastatur eingegeben und mit [ENTER] abgeschlossen werden.

3.2. Variable und Bezeichnungen

Bevor wir zur Erläuterung einiger einfacher BASIC-Befehle übergehen können, muß noch vorangestellt werden, daß im BASIC des Z 9001 die Verwendung von zwei Variablen-Typen möglich ist.

Zunächst gibt es die numerischen Variablen, die jeweils einen konkreten Zahlenwert (4-byte-Gleitkommazahl) beinhalten, der der Variablen durch eine Eingabe- oder eine Ergibtanweisung (vgl. Abschnitt 2.3.) zugewiesen wird. Außerdem können solche Variablen auch mit den im voranstehenden Abschnitt erläuterten Rechenoperationen zu numerischen Ausdrucken verknüpft werden.
Im Gegensatz zur üblichen Schreibweise in der Mathematik, in der Variable jeweils durch einen Buchstaben (z. B.: x, y, a, b, i, j, ...) bezeichnet werden, kann man im BASIC des Z 9001 bis zu zwei Zeichen (Großbuchstabe und Großbuchstabe bzw. Ziffer) zur Bezeichnung nutzen. Dadurch ist es möglich, ausreichend viele Variable zu unterscheiden.

Beispiele: A, A1, AS, XX, R7, K9

Neben den numerischen Variablen gibt es beim Z 9001 noch den Typ der Zeichenkettenvariablen (auch String-Variablen), die beliebigen Text bzw. Zeichenfolgen (Groß- und Kleinbuchstaben, Sonderzeichen, Grafikzeichen) bis zu 255 Zeichen beinhalten können. Die Variablenbezeichnung erfolgt wie für numerische Variablen, nur wird noch das Zusatzzeichen "$" (Dollarzeichen) angefügt.

Beispiele: B$, A1$, Z$, T5$, KK$

Eine Wertzuweisung für alle Variablen erfolgt wie üblich durch das Gleichheitszeichen, wobei auf der rechten Seite auch Ausdrücke stehen dürfen.

Beispiele:
```
A=5.7
C1=-12.6*4
XX=3*C1
A$="WORT1"
ZK$="Zeichenkette"
```

Außer den bereits beschriebenen einfachen Variablen können mittels der DIM-Anweisung Variablen-Felder (Matrizen) für numerische und Zeichenkettenvariable vereinbart werden. Zum Beispiel wird durch

```
DIM X(3,4)
```

die Matrix

```
X(0,0) ........ X(0,4)
  .              .
  .              .
  .              .
X(3,0) ........ X(3,4)
```

definiert und Speicherplatz für sie bereitgestellt.

Die Elemente X(I,J), I=0,...,3, J=0,...,4 können dann im Programm wie einfache numerische Variable benutzt werden. Ebenso kann durch

DIM T$(2,4)

eine Matrix vereinbart werden, deren Elemente Zeichenketten sind.

3.3. Einfache Programme

Nach den Vorbemerkungen im Abschnitt 3.2 wollen wir jetzt einige einfache Programme in BASIC formulieren und ihre Abarbeitung am Z 9001 beschreiben.

Dazu muß man wissen, daß jede Programmzeile mit einer Zeilennummer beginnt und eine oder mehrere Anweisungen (durch Doppelpunkt getrennt) enthält. Bei der Neuerstellung eines Programmes wird jede Programmzeile über die Tastatur eingegeben. Zur Erleichterung späterer Ergänzungen sollte dabei im Zehnerabstand numeriert werden. Nach [ENTER] wird die eingegebene Programmzeile in den Programmspeicher (RAM) übernommen.

Fehler bei der Eingabe einer Programmzeile können mit Hilfe der Kursortasten [→], [←] sowie der Editiertasten [INS], [DEL] und [CL LN] korrigiert werden, falls noch nicht [ENTER] betätigt wurde (vgl. Abschnitt 9).

Bereits mit [ENTER] übernommene Zeilen können nach Neueingabe mit der gleichen Zeilennummer geändert bzw. überschrieben werden. Eine Programmzeile kann gelöscht werden, indem nur die Zeilennummer und [ENTER] eingegeben werden. Ausführlicher wird auf Korrekturprobleme im Abschnitt 13.16 (EDIT-Kommando) eingegangen.

Nach beendeter Eingabe können die Programme durch das Kommando "RUN" gestartet werden. Hierzu steht die Taste [RUN] zu Verfügung, oder es wird

RUN [ENTER]

eingetippt. Die Abarbeitung der Programme erfolgt, wenn nicht anders angewiesen, in der Reihenfolge der Numerierung.

Der Rechner befindet sich nach RUN im Programm-Modus und führt die Anweisungen des Programmes aus. Erst bei Programmende oder bei Programmabbruch wird in den Kommando-Modus zurückgekehrt.

Einfache Programme bestehen nur aus Eingabe-, Berechnungs- und Ausgabe-Anweisungen, die durch die Schlüsselworte

INPUT - Eingabeanweisung
LET - Berechnungs- oder Ergibtanweisung (dieses Schlüsselwort kann auch entfallen)
PRINT - Ausgabeanweisung

charakterisiert werden.

<u>Beispiel:</u> Soll der Mittelwert von drei Zahlen berechnet werden, so genügt folgendes Programm:

```
10 INPUT "3 Zahlen eingeben :";A,B,C
20 LET M=(A+B+C)/3
30 PRINT "Mittelwert    :";M
```

Die Abarbeitung dieses einfachen Beispiels erfolgt in folgender Weise:

Das Programm wird durch [RUN] gestartet. Durch die Anweisung 10 wird dann der Text

```
3 Zahlen eingeben : ■
```

am Bildschirm angezeigt.

Der Rechner wartet jetzt auf die Eingabe der 3 Zahlen A,B,C, die durch Komma getrennt werden müssen, z. B.:

```
5,7,12 [ENTER]
```

Danach wird gemäß Zeile 20 der Mittelwert M berechnet und mit Hilfe der Anweisung 30 am Bildschirm ausgegeben. Insgesamt entsteht folgendes Bild:

```
3 Zahlen eingeben : 5,7,12
Mittelwert     :  8
OK
>
```

Durch "OK" wird das Ende der Programmabarbeitung angezeigt.

Diese kleine Rechnung läßt sich sicher auch auf einem Taschenrechner sehr schnell ausführen. Wollen wir aber zum arithmetischen Mittelwert gleichzeitig noch das geometrische und quadratische Mittel bestimmen, so ist das im BASIC wesentlich einfacher möglich.

Zur Realisierung dieser Programmerweiterung wird zusätzlich zum bereits gespeicherten Programm folgendes eingegeben:

a) Zur Berechnung der gesuchten Werte:

```
21 MG=(A*B*C)^(1/3) [ENTER]
22 MQ=SQR((A*A+B*B+C*C)/3) [ENTER]
```

b) Zur Ausgabe der Werte:

```
40 PRINT "Geometr.Mittel:";MG [ENTER]
50 PRINT "Quadrat.Mittel:";MQ [ENTER]
```

Die Ausführung des jetzt im Rechner gespeicherten Programms nach [RUN] liefert am Bildschirm

```
3 Zahlen eingeben : 5,7,12
Mittelwert     :  8
Geometr.Mittel:  7.48887
Quadrat.Mittel:  8.52448
OK
>■
```

Selbstverständlich kann dieses Programm beliebig oft abgearbeitet werden, wobei nach [RUN] stets nur die 3 Zahlenwerte einzugeben sind.

3.4. BASIC-Kommandos

Zur Eingabe, Abarbeitung und Änderung der BASIC-Programme stehen Kommandos zur Verfügung, die ebenfalls mit [ENTER] abgeschlossen und anschließend vom Rechner sofort ausgeführt werden. Für einige Kommandos stehen Tasten zur Verfügung, die eine sofortige Ausführung dieser Kommandos bewirken (z.B. für [RUN], [STOP], [LIST], [CONT], [PAUSE] - vgl.Tastatur). Die Handhabung der wichtigsten Kommandos wollen wir kurz beschreiben.

LIST

Mit dem LIST-Kommando kann ein Programm ausgegeben ("gelistet") werden. Dies erfolgt beim Heimcomputer standardmäßig auf dem Bildschirm.

Wird das Programm zur Mittelwertberechnung wie im Abschnitt 2.3 eingegeben und ergänzt, so wird es nach

LIST [ENTER]

am Bildschirm wie folgt angezeigt:

```
10 INPUT "3 Zahlen eingeben :",A,B,C
20 LET M=(A+B+C)/3
21 MG=(A*B*C) (1/3)
22 MQ=SQR((A*A+B*B+C*C)/3)
30 PRINT "Mittelwert     :";M
40 PRINT "Geometr.Mittel:";MG
50 PRINT "Quadrat.Mittel:";MQ
OK
>
```

Zur Ausführung des Kommandos kann ebenso die [LIST] -Taste gedrückt werden.

EDIT

Durch das EDIT-Kommando wird die Änderung von Programmen einfach möglich. Nach

EDIT 20 [ENTER]

wird vom beschriebenen Mittelwertprogramm die Zeile mit der Nummer 20 angezeigt, also

20 LET M=(A+B+C)/3

Nun kann diese Zeile mit Hilfe der Kursortasten [→], [←] sowie der Taste zum Streichen und Einfügen von Zeichen (Taste DEL/INS) korrigiert werden. Z. B. kann also das überflüssige Schlüsselwort LET herausgestrichen werden. Nach [ENTER] wird die korrigierte Zeile wieder in das Programm übernommen und anschließend Zeile 21 zur Änderung bereitgestellt. Der EDIT-Zustand kann durch [STOP] verlassen werden.

CSAVE/CLOAD

Da man vor allem längere Programme nicht mehrmals in den Rechner

eintippen möchte, sollte man diese auf einer Magnetbandkassette speichern. Mit dem Kommando

CSAVE "PROGR1" [ENTER]

erfolgt die Speicherung des gesamten BASIC-Programmes auf der Kassette unter dem Namen "PROGR1". Als Name kann hierbei jede Zeichenkette mit maximal 8 Zeichen benutzt werden.
Zu einem späteren Zeitpunkt läßt sich das gespeicherte Programm nach Positionierung des Magnetbandes vor den Programmanfang durch

CLOAD "PROGR1" [ENTER]

wieder in den Rechner laden und dann abarbeiten.

LIST#1/LOAD#1

Während die Speicherung von Programmen mit CSAVE in der internen Darstellung erfolgt, können Programme durch

LIST#1 "name" [ENTER]

in "Textform" gespeichert werden. Dies kostet zwar etwas mehr Speicherplatz, bietet aber den Vorteil vielseitigerer Anwendbarkeit. Das Laden der so gespeicherten Programme erfolgt durch

LOAD#1 "name" [ENTER]

Beim Laden der Programme mit CLOAD oder LOAD#1 sollte die in der Bedienungsanleitung (Abschnitt 5) beschriebene Handlungsvorschrift beachtet werden.

Der Vollständigkeit halber sei abschließend noch der komplette Kommandosatz des Z 9001 angeführt. Eine ausführliche Beschreibung aller Kommandos erfolgt im Abschnitt 13.

AUTO - Automatische Zeilennumerierung bei Eingabe
BYE - Verlassen des BASIC-Interpreters

CLEAR - Löschen aller Variablen
CONT - Fortsetzung unterbrochener Programme
CSAVE - Abspeichern von Programmen auf Magnetband
CLOAD - Laden von Programmen vom Magnetband
DELETE - Löschen von Programmzeilen
EDIT - Programmänderungsmodus
LINES - Anzahl der bei LIST ausgegebenen Programmzeilen
LIST - Programmausgabe (abschnittsweise, Fortsetzung jeweils nach [ENTER])
LIST#n - Programmausgabe auf Drucker oder Magnetband
LOAD - Laden von LIST-gespeicherten Programmen vom Magnetband
NEW - Löschen des gesamten BASIC-Programms
NULL - Legt Anzahl auszugebender Nullen bei LIST# fest
RENUMBER - Automatische Neunumerierung des BASIC-Programmes
RUN - Programmstart
TROFF - Ausschalten TRACE } Verfolgung des Programmlaufs
TRON - Einschalten TRACE } anhand der Zeilennummern
WIDTH - Länge der Ausgabezeilen festlegen

3.5. Weitere BASIC-Anweisungen

Neben den bereits genannten fundamentalen Anweisungen INPUT, LET und PRINT stehen eine Reihe weiterer Anweisungen zur Verfügung. Hier sollen nur einige kurz erläutert werden:

REM-Anweisung

Zur besseren Übersichtlichkeit der Programme können nach dem Schlüsselwort "REM" beliebige Kommentare in die Programme eingefügt werden, die keinen Einfluß auf deren Abarbeitung haben. "REM" kann auch durch "!" ersetzt werden.

FOR...NEXT-Anweisung

Zur mehrfachen Abarbeitung von Anweisungsgruppen (z. B. bei der Berechnung von Tabellen, Folgen, Reihen oder bei Iterationszyklen) ist eine sogenannte "Laufanweisung" erforderlich, die im BASIC durch die FOR...NEXT-Anweisung gebildet wird. Sollen z. B.

die Funktionen

$$f(x) = x \cdot e^{x} \quad , \qquad g(x) = \sqrt{3x+10}$$

im Intervall [-2,2] im Abstand von 0.5 tabelliert werden, so wird dies durch folgendes Programm erreicht:

```
40 REM Tabellenberechnung
50 FOR X=-2 TO 2 STEP .5
60 F=X*EXP(X)
70 G=SQR(3*X+10)
80 PRINT X,F,G
90 NEXT X
```

Dabei bedeutet die Anweisung 50, daß die folgenden Anweisungen mehrfach ausgeführt werden sollen, und zwar für X=-2 bis X=2 in Abständen von 0.5, d.h. für

X = -2, -1.5, -1, -0.5, 0, 0.5, 1, 1.5, 2 .

Die Anweisung 90 beendet diese Anweisungsgruppe.
Zur Zeile 80 muß noch erläutert werden, daß die durch PRINT auszugebenden Elemente durch Komma oder Semikolon getrennt werden können. Das Komma bewirkt die Ausgabe in einem festen Raster (Tabelle), während durch ein Semikolon die Werte fortlaufend ausgegeben werden.

Die Ausführung dieses Programmes liefert am Bildschirm:

```
-2          -.270671    2
-1.5        -.334695    2.34521
-1          -.367879    2.64575
-.5         -.303265    2.91548
 0           0          3.16228
 .5           .824361   3.39117
 1           2.71828     3.60555
 1.5         6.72253     3.80789
 2           14.7781     4
OK
>
```

IF...THEN-Anweisung

Häufig ist es erforderlich, einige Anweisungen nur unter gewissen Bedingungen auszuführen (z. B. Berechnung von SQR(X) nur für $X \geq 0$). Dazu dient die Anweisung

IF bedingung THEN anweisung .

Hierbei steht für "bedingung" ein logischer Ausdruck (z. B. $A > B$, $X < 5$, A$="A"), der wahr oder falsch sein kann.

Nur wenn er wahr ist, wird die auf THEN folgende "anweisung" ausgeführt. Wesentlich für die Nutzung ist, daß die "bedingung" auch Vergleiche von Zeichenketten enthalten darf.

Beispiel:

```
 90 INPUT "Gib Wert für A ein :";A
100 IF A>0 THEN PRINT "A ist positiv"
110 IF A<0 THEN PRINT "A ist negativ"
```

GOTO-Anweisung

Durch die Anweisung

GOTO zeilennummer

kann ein Programm, abweichend von der normalen Reihenfolge, in einer anderen Programmzeile entsprechend der "zeilennummer" fortgesetzt werden. Diese "zeilennummer" muß als Zahlenwert konkret angegeben werden, beispielsweise

```
210 GOTO 40
```

Ein Beispiel für die Anwendung der beschriebenen Befehle ist das folgende Ratespiel, in dem im Befehl 30 durch einen Spieler (A) eine Zahl eingegeben werden muß, die ein anderer Spieler (B) dann erraten soll.

```
 10 REM Zahlenratespiel
 20 REM Spieler A gibt vor (Zeile 30)
 30 INPUT "Vorgabezahl(Spieler A) :";A
 35 REM Loeschen des Bildschirmes in 36
 36 CLS
 40 REM Spieler B darf raten (Zeile 50)
 50 INPUT "Gib einen Tip ein :";B
 60 IF A<B THEN PRINT B,"ist zu gross"
 70 IF A>B THEN PRINT B,"ist zu klein"
 80 IF A=B THEN 100
 90 GOTO 40
100 PRINT B,"ist die gesuchte Zahl"
110 INPUT "Neues Spiel (J/N)?";A$
120 IF A$="J" THEN GOTO 30
130 PRINT "ENDE"
```

Unter Nutzung der im Abschnitt 2.1 beschriebenen Standardfunktionen kann sehr leicht auch der Rechner als "Spieler A" eingesetzt werden. Ersetzt man Zeile 30 durch

```
30 A=INT(100*RND(1))
```

so wird automatisch eine ganzzahlige zweistellige Zufallszahl erzeugt.

Der Ablauf des Programmes am Bildschirm vollzieht sich in folgender Weise:

```
Gib einen Tip ein : 50
 50 ist zu gross
Gib einen Tip ein : 10
 10 ist zu klein
Gib einen Tip ein : 25
 25 ist zu klein
Gib einen Tip ein : ■
```

Neben den mathematischen Standardfunktionen (vgl. Abschnitt 3.1) stehen im BASIC des Z 9001 einige Funktionen zur Zeichenkettenverarbeitung zur Verfügung. Von diesen sollen hier CHR$, LEN und STRING$ erläutert werden.

CHR$(N)	- Liefert das Zeichen mit der Nummer N in der ASCII-Code-Tabelle.
LEN(A$)	- Liefert die Anzahl der Zeichen, die die Zeichenkette A$ enthält.
STRING$(N,A$)	- Setzt die Zeichenkette A$ genau N-mal aneinander.

Zusätzlich sei erwähnt, daß man Zeichenketten durch "+" aneinanderfügen kann und in einer IF...THEN-Anweisung wie bei Zahlen die Vergleichsoperatoren < , > , = , >= , <= anwendbar sind.
Damit kann man mit Zeichenketten praktisch "rechnen", wobei die Vergleichsoperatoren die Zeichenketten bezüglich ihrer alphabetischen Ordnung (allgemeiner: entsprechend der ASCII-Code-Numerierung) sortieren, wie das folgende Programmbeispiel zeigt:

```
 5 ! Sortierung zweier Namen
10 INPUT "Gib 2 Namen ein :";N1$,N2$
20 IF N1$<N2$ THEN GOTO 60
30   H$=N2$
40   N2$=N1$
50   N1$=H$
60 PRINT "Alphabetische Reihenfolge :"
70 PRINT N1$;CHR$(44);N2$
80 GOTO 10
```

Werden nach [RUN] in Anweisung 10 die Namen KARL,ERICH eingegeben, so ist am Bildschirm folgendes zu sehen:

```
Gib 2 Namen ein : KARL,ERICH
Alphabetische Reihenfolge :
ERICH,KARL
Gib 2 Namen ein :■
```

Zur Erläuterung der Zeile 70 sei vermerkt, daß der Rechner alle Buchstaben, Ziffern, Sonder- und Grafikzeichen in codierter Form speichert. Jedem dieser Zeichen entspricht eine Zahl (0 bis 255). Durch die Funktion CHR$(N) kann umgekehrt einer Zahl auch das zugehörige Zeichen zugeordnet werden.
In Zeile 70 liefert der Wert CHR$(44) das Kommazeichen, welches zwischen ERICH und KARL ausgegeben wird.

Die Anwendungsmöglichkeiten der beschriebenen Zeichenkettenfunktionen sind sehr vielfältig. Neben der Analyse von Zeichenketten können besonders Probleme der Bildgestaltung zweckmäßig gelöst werden.

Soll eine Überschrift vorher unbekannter Länge unterstrichen werden, so läßt sich dies mit den obigen Funktionen leicht realisieren. Im folgenden Programmteil soll die Textzeile

ABRECHNUNG FUER monatsname

für beliebigen "monatsnamen" entsprechend der Textlänge mit dem Zeichen Nr.160 (▬) unterstrichen werden.

```
10 ! Anwendung der ZK-Funktionen
20 A$="ABRECHNUNG FUER "
30 INPUT "Gib Monatsnamen ein :";MO$
35 CLS:REM Bildschirm loeschen
40 N=LEN(MO$)+16
50 PRINT A$+MO$
60 PRINT STRING$(N,CHR$(160))
70 ! Datenauswertung fuer lfd. Monat
 .
 .
 .
```

3.6. Zusammenstellung der BASIC-Anweisungen

Eine vollständige Beschreibung aller Anweisungen und Funktionen des Z9001-BASIC erfolgt in den Abschnitten 13 und 14. An dieser Stelle soll lediglich eine Übersicht der entsprechenden Schlüsselwörter aufgeführt werden.

a) Logische Operatoren
AND, NOT, OR

b) Mathematische Funktionen
vergl. Abschnitt 3.1.

c) Zeichenketten-Funktionen

ASC	- ACSII-Code eines Zeichens
CHR$	- Zeichen entsprechend ASCII-Nummer
INSTR	- Position einer Zeichenkette in einer anderen
LEN	- Länge einer Zeichenkette
LEFT$	- Linker Teil einer Zeichenkette
MID$	- Mittlerer Teil einer Zeichenkette
RIGHT$	- Rechter Teil einer Zeichenkette
STR$	- Wandelt Zahl in Zeichenkette um
STRING$	- Vervielfacht eine Zeichenkette
VAL	- Wandelt Zeichenkette in Zahl um

d) Sonderfunktionen

FRE	- Gibt noch freien Speicherplatz an
POS	- Liefert Kursorposition
SPC	- Ausgabe von Leerzeichen
USR	- Aufruf nutzereigener Maschinenprogramme
TAB	- Setzt Kursor auf TAB-Position

e) Anweisungen

BEEP	- Tonerzeugung
BORDER	- Farbzuweisung für Randbereich
CALL	- Aufruf eines Maschinenprogramms
CLS	- Bildchirm löschen
DATA	- Datenbereitstellung
DEEK	- 2-Byte-Eingabe

DEF FN	- Definition einzeiliger Funktionen
DIM	- Feldvereinbarung
DOKE	- 2-byte-Ausgabe
END	- Physisches Programmende
FOR...NEXT	- Laufanweisung
GOSUB	- Sprung zu einem Unterprogramm
GOTO	- Sprung zu einer Zeile
IF...THEN...:ELSE	- Bedingte Anweisungen
INK	- Festlegung der Vordergrundfarbe
INKEY$	- Zeicheneingabe von Tastatur
INPUT	- Eingabe von Zahlen oder Zeichenketten
INP	- Lesen einer Port-Adresse
JOYST	- Spielhebelabfrage
LET	- Ergibtanweisung
ON...GOTO	- Verzweigungsanweisung
ON...GOSUB	- Verzweigung zu Unterprogrammen
OUT	- Ausgabe auf Port-Adresse
PAPER	- Festlegung der Hintergrundfarbe
PAUSE	- Programmunterbrechung
PEEK	- Lesen eines Bytes von einer Speicheradresse
POKE	- Schreiben eines Bytes in eine Speicheradresse
PRINT	- Standardausgabe von Zahlen und Zeichen
PRINT AT	- Ausgabe ab spezieller Bildschirmposition
READ	- Lesen der DATA-Anweisungen
REM	- Bemerkung (Kurzzeichen: !)
RESTORE	- Setzen des Zeigers für READ-Anweisung
RETURN	- Abschluß eines Unterprogrammes
STOP	- Programmabbruch
WAIT	- Zeitweilige Programmunterbrechung
WINDOW	- Festlegung des Rollbereiches am Bildschirm

4. Zeichensatz des Z9001

Der Zeichensatz des BASIC-Interpreters besteht aus alphabetischen, numerischen, grafischen Zeichen und Sonderzeichen. Die alphabetischen und numerischen Zeichen entsprechen dem ASCII-Code (Codierung 32 bis 127). Die Sonderzeichen (Codierung 0 bis 31) weichen vom ASCII-Standard etwas ab und sind für die Zwecke des Heimcomputers angepaßt worden. Die Zeichen mit einer Codierung ab 128 sind die grafischen Zeichen des Z9001.

Kleinbuchstaben werden in Großbuchstaben umgewandelt, sofern sie nicht in Zeichenketten, Eingabeanweisungen oder Kommentaranweisungen stehen. Leerzeichen sind bedeutungslos, falls durch ihr Weglassen keine neuen reservierten Worte (siehe Anhang A3) als Teil einer Folge von Zeichen in einer Anweisung entstehen. Leerzeichen in Zeichenketten dürfen nicht weggelassen werden. In der Tabellensammlung ist der Zeichenvorrat des Heimcomputers Z9001 aufgeführt.

5. Konstanten

Konstanten sind feste Größen, mit denen man im BASIC operieren kann. Der BASIC-Interpreter unterscheidet zwei Arten von Konstanten:

numerische Konstanten
Zeichenketten-Konstanten (String-Konstanten)

5.1. Numerische Konstanten

Bei numerischen Konstanten sind nur die ersten 7 Ziffern signifikant. Das liegt an der internen Zahlendarstellung des Heimcomputers. Die Mantissenlänge beträgt nur 3 byte.

Numerische Konstanten sind Zahlen, die in 3 Darstellungsarten eingegeben werden können, als

ganze Zahlen, Festkommazahlen, Gleitkommazahlen.

Beispiele für mögliche ganze Zahlen:

```
  3
-10
4000
  0
```

Beispiele für Festkommazahlen:

```
  0.1
  0.0264
 -0.0009
 26.0087
-12.5
```

Es muß darauf geachtet werden, daß anstelle des in der Mathematik üblichen Kommas der Dezimalpunkt zu verwenden ist. Sonst bringt der BASIC-Interpreter eine Fehlermeldung.

Für Gleitkommazahlen gilt folgendes Format:

```
[vorz.]  mantisse E  [vorz. exp.]  exponent
                                   └──→ 01...38
                     └──→ [+] , -
         └→max. 6 Stellen, 6. Stelle gerundet
└→[+] , - (kein Vorzeichen heißt +)
```

Beispiele:	BASIC-Darstellung	Wert im Dezimalsystem
	3E1	30
	20.5E+4	205000
	77E+2	7700
	77E-2	0.77
	1E-10	0.0000000001

Der BASIC-Interpreter verarbeitet Zahlen im Bereich von:

± 2.9387E-39 bis ± 1.70141E+38 und die 0.

Betragsmäßig größere Zahlen als 1.70141E+38 führen zu einem sogenannten Überlauf-Fehler (siehe Fehlermeldung OV). Kleinere Zahlen als die angegebene kleinste Zahl werden automatisch Null.

Die Darstellungsart bei der Ausgabe von numerischen Konstanten bzw. des Wertes von Variablen richtet sich nach dem jeweiligen Wert.

Ganze Zahlen zwischen -999999 und +999999
werden als ganzzahlige Werte ausgegeben.

Gebrochene Zahlen mit einem Betrag 0.01 ≦ Zahl ≦ 999999
werden als Festkommazahl ohne Exponent ausgegeben.

Alle übrigen Zahlenwerte werden als Gleitkommazahl ausgegeben.

Beispiele: (? steht für PRINT)

eingegebene Anweisung	Ausführung der PRINT-Anweisung
? 0.000123456789	1.23457E-04
? 0.00123456789	1.23457E-03
? 0.0123456789	0.0123457
? 0.123456789	0.123457
? 1.23456789	1.23457
? 12.3456789	12.3457
? 123.456789	123.457
? 1234.56789	1234.57
? 12345.6789	12345.7
? 123456.789	123457
? 1234567.89	1.23457E+06
? 12345678.9	1.23457E+07

5.2. Zeichenketten-Konstanten

Zeichenketten-Konstanten werden in Anführungszeichen eingeschlossen und können alphanumerische Zeichen, graphische Zeichen und Sonderzeichen aus dem Zeichenvorrat des Heimcomputers enthalten. Die maximale Länge einer Zeichenketten-Konstante beträgt 255 Zeichen.

Beispiele: "HEIMCOMPUTER robotron Z9001"
"beliebiger Text"
"13.01.1947"

6. Variable

6.1. Was ist eine Variable?

Eine Variable ist eine Dateneinheit, deren Wert verändert werden kann. Der Wert einer Variablen wird durch die Zahl der Zeichenkette bestimmt, die der Variablen zugewiesen wurde.

Gibt man im Kommando-Modus die Anweisung

```
PRINT 5;10;15
```

ein, so antwortet der Heimcomputer mit der Ausgabe von

```
5  10  15
```

auf dem Bildschirm. Mit dieser Anweisung werden immer wieder die gleichen Zahlen ausgegeben, da es sich um konstante Werte handelt. Mit den folgenden Anweisungen erhält man die gleiche Ausgabe auf dem Bildschirm:

```
A=5:B=10:C=15:PRINT A;B;C

5  10  15
```

Werden die Wertzuweisungen zu den Variablen geändert, erscheint auch ein anderes Ergebnis:

```
A=-1:B=20:C=100:PRINT A;B;C

-1  20  100
```

Variable werden in fast allen BASIC-Programmen benutzt. Es gibt zwei Arten von Variablen:

numerische Variable
Zeichenkettenvariable

Nach dem Start eines Programmes mit [RUN] erhalten numerische Variable den Wert Null, während Zeichenkettenvariable als leere Zeichenkette (Zeichenkette der Länge Null) angenommen werden.

6.2. Variablenbezeichnung

Variablennamen müssen immer mit einem Buchstaben beginnen und können beliebig lang sein. Es sind aber nur die ersten beiden Zeichen signifikant. Das zweite und die weiteren Zeichen können Buchstaben und Ziffern sein.

Ein Dollarzeichen ($) am Ende des Namens weist eine Variable als Zeichenkettenvariable aus. Der Variablenname darf keine reservierten Worte enthalten (vgl. Anhang A3).

Beispiele für syntaktisch richtige und falsche Bezeichnungen von Variablen:

NAME	richtig, aber nur die ersten zwei Buchstaben sind signifikant
X	richtig
A1	richtig
1Y	falsch, Variablenname muß mit einem Buchstaben beginnnen
B!	falsch, Sonderzeichen außer $ sind unzulässig
C X	richtig, Leerzeichen sind bedeutungslos

TOR	falsch, enthält das reservierte Wort OR (Vergleichsoperator)
NA	richtig, aber identisch der Variablen NAME
TEXT$	richtig, Zeichenkettenvariable
na	richtig, der BASIC-Interpreter wandelt die Kleinbuchstaben in Variablenbezeichnungen bei der Eingabe in Großbuchstaben um (NAME und NA)
TE$	richtig, identisch TEXT$

6.3. Feldvariable

Ein Feld ist eine Gruppe von Zahlen oder Zahlenketten, die durch einen Variablennamen repräsentiert werden. Jedes Feldelement wird mit einem Index (oder mehreren Indizes), eingeschlossen in runde Klammern, zum Variablennamen beschrieben. Der Index kann eine ganze Zahl oder ein ganzzahliger Ausdruck sein. Der niedrigste Index ist stets 0.

Felder mit bis zu 11 Elementen (0 bis 10 bei einem eindimensionalen Feld) können wie normale Variable verwendet werden. Erst wenn mehr als 11 Elemente eines Feldes benötigt werden, muß dieses Feld vorher durch eine DIM-Anweisung dimensioniert werden (siehe Abschnitt 13.14).

Eine Feldvariable und eine normale Variable mit dem gleichen Namen behandelt der BASIC-Interpreter als 2 unterschiedliche Variable.

Beispiel:

```
10 FOR I=0 TO 10
20 READ A(I)
30 NEXT I
40 DATA 1,2,3,4,5,6,7,8,9,10,11
50 A = 999
60 PRINT A(0);A(2);A(4);A(6);
70 PRINT A(8);A(10);A
RUN
1  3  5  7  9  11  999
OK
>
```

In diesem Programm wird die Feldvariable A ohne vorherige Dimensionierung verwendet. Durch die READ-Anweisung werden den 11 Elementen Werte zugewiesen. Anschließend wird jedes zweite Element des Feldes A ausgeschrieben. In der Zeile 50 wird der Variablen A ein Wert zugewiesen, der an letzter Stelle in der PRINT-Anweisung 70 mit ausgegeben wird.

7. Ausdrücke und Operationen mit Zahlen

Ein Ausdruck kann aus einer Konstanten, Variablen oder einer Kombination aus Konstanten und Variablen, die mit Operatoren verknüpft sind, bestehen.

Operatoren führen mit Werten mathematische oder logische Operationen durch.

Es wird zwischen arithmetischen Operatoren, Vergleichsoperatoren und logischen Operatoren unterschieden. Folgende Operatoren können benutzt werden:

Rang	Operator	Bedeutung	
1	^	Potenzierung	arithmetische Operatoren
2	*	Multiplikation	
	/	Division	
3	+	Addition	
	-	Subtraktion	
4	<	kleiner als	Vergleichsoperatoren
	<=	kleiner gleich	
	=	gleich	
	< >	ungleich	
	>	größer als	
	> =	größer gleich	
5	NOT	Negation	logische Operatoren
6	AND	Konjunktion	
7	OR	Disjunktion	

Die Abarbeitungsreihenfolge eines Ausdruckes liegt fest. Im allgemeinen werden die Operationen von links nach rechts ausgeführt.

Der Vorrang der Operationen ist oben angegeben. Vor den Operatoren in der Rangfolge, wobei der Rang 1 die höchste und der Rang 7 die niedrigste Priorität bedeuten, werden zuerst die geklammerten Ausdrücke und die Funktionen des BASIC-Interpreters ausgewertet.

Der logische Operator NOT ist ein einseitiger Operator, d.h., er bezieht sich nur auf einen logischen Operanden, vor dem er steht. Die übrigen logischen Operanden AND und OR sind zweiseitig und verknüpfen zwei logische bzw. arithmetische Operanden.

Trifft der BASIC-Interpreter während der Auswertung eines Ausdrucks auf eine Division durch Null, so wird die Auswertung mit einer Fehlermeldung (/0) angezeigt.

Wird der Betrag des Ergebnisses einer Berechnung größer als der maximal erlaubte Wert (1.70141E+38), reagiert der BASIC-Interpreter mit der Fehlermeldung OV.

Ausdrücke, die logische Operatoren und Vergleichsoperatoren enthalten, bezeichnet man als logische Ausdrücke. Sie liefern einen logischen Wert, der "wahr" oder "falsch" sein kann. Das Ergebnis kann in Verbindung mit der BASIC-Anweisung IF...THEN...ELSE zur Steuerung des Programmablaufs benutzt werden.

Wenn in einem Ausdruck auch arithmetische Operatoren auftreten, werden nach der oben angegebenen Reihenfolge diese zuerst abgearbeitet.

Beispiele: A+B < (X-1)/Y

Dieser Ausdruck ist "wahr", wenn der Wert von A+B kleiner ist als der Wert von X-1 dividiert durch Y.

```
IF SIN(X)<0 THEN 1000
IF I>10 OR K<0 THEN 40
```

Das sind zwei Beispiele, wie das Ergebnis von logischen Ausdrükken zur Steuerung von Programmabläufen beitragen kann.

Beispiele zur Bildung von arithmetischen Ausdrücken:

Nummer	BASIC-Darstellung	mathematische Schreibweise	Bemerkung
1	A/B/C	$\frac{A}{B \cdot C}$	richtig
2	A/(B*C)	$\frac{A}{B \cdot C}$	richtig, Nummer 1 und 2 liefern unter Umständen verschiedene Werte und sind nicht identisch (wegen unterschiedlicher Abarbeitungsreihenfolge) Rundung!
3	K*-52	$-52 \cdot K$	richtig
4	K*(-52)	$-52 \cdot K$	richtig
5	-52*K	$-52 \cdot K$	richtig, Verwendung eines einseitigen Operators
6	4*PI*R∧3/3	$\frac{4}{3}\pi \cdot R^3$	richtig
7	SQR(A∧2+B∧2)	$\sqrt{A^2+B^2}$	richtig, Verwendung einer Funktion
8	A+FELD(-3)	-	falsch, -3 ist ein unzulässiger Index-Ausdruck

Beispiele für syntaktisch richtige und falsche logische Ausdrücke:

Nummer	BASIC-Darstellung	mathematische Umschreibung	Bemerkungen
1	NOT(A>B)	$\neg(A>B)$	richtig, vgl. 2
2	A<=B	$A \leq B$	richtig, mit Bsp. 1 gleich
3	A<X AND X<=B	$A<X \leqq B$	richtig
4	A OR B * (I $ J)	-	falsch
5	X <= Y>Z	-	falsch

8. Operationen mit Zeichenketten

Zeichenketten können mit dem Additions-Zeichen (+) miteinander verknüpft werden.

Beispiel:

```
10 X$="BAD" : Y$="WETTER"
20 PRINT X$+"E"+Y$
RUN
BADEWETTER
OK
```

Außerdem können Zeichenketten mit Hilfe der Vergleichsoperatoren verglichen werden. Der Vergeich erfolgt zeichenweise von links nach rechts. Dabei wird der Code der jeweiligen Zeichen miteinander verglichen. Sind alle Zeichencodes der beiden Zeichenketten gleich, so sind auch die Zeichenketten identisch.

Wird während eines Vergleichs das Ende einer Zeichenkette erreicht, so gilt die kürzere als die kleinere Zeichenkette. Die folgenden logischen Ausdrücke liefern den Wert "wahr":

```
"AA" < "AB"
"Z9001" = "Z9001"
"X%" > "X$"
"SPACE␣"> "SPACE" (wobei ␣ für Leerzeichen steht)
"kp" > "KP"
"BIER" < "BIERE"
T$ < "15.06.84"    (wobei T$ = "14.06.84")
```

9. Spezielle Tasten zur BASIC-Unterstützung

Die Kommunikation mit dem Rechner erfolgt in der Regel über die Tastatur. Die schreibmaschinenähnliche Tastatur ist in der Bedienungsanleitung beschrieben. Hier soll auf die BASIC-spezifischen Tasten eingegangen werden.

Für den BASIC-Interpreter sind folgende Funktionstasten von besonderer Bedeutung:

[ENTER] Schließt alle Eingaben ab. Bewirkt damit die Abarbeitung der eingegebenen Kommandos/Anweisungen bzw. die Übernahme von Daten in den Rechner.

[RUN] Startet ein im Arbeitsspeicher befindliches BASIC-Programm. Hat die gleiche Wirkung wie die Tastenfolge

RUN [ENTER]

[STOP] Unterbricht die Programmabarbeitung. Der BASIC-Interpreter kehrt in den Kommando-Modus zurück.

[PAUSE] Unterbricht die Programmabarbeitung ohne Rückkehr in den Kommando-Modus.

[CONT] Es erfolgt die Fortsetzung der Abarbeitung. Entspricht der Tastenfolge

CONT [ENTER]

[LIST] Das im Arbeitsspeicher befindliche BASIC-Programm wird am Bildschirm ausgelistet.
Entspricht der Tastenfolge

LIST [ENTER]

Die Tasten [DEL], [INS], [CL LN] dienen zur Korrektur der jeweils aktuellen Eingabezeile. Das ist die Zeile, auf der sich der Kursor befindet. Sie kann max. 72 Zeichen(d. h. fast 2 Bildschirmzeilen) aufnehmen.

[DEL] (DELETE-Taste). Mit [DEL] wird das Zeichen, auf dem sich der Kursor befindet, gelöscht.

[INS] (INSERT-Taste). Mit [INS] werden in die Eingabezeile ab der Kursorposition Leerzeichen eingefügt, die anschließend durch andere Zeichen ersetzt werden können.

[CL LN] (CLEAR LINE-Taste). Durch [CL LN] wird die aktuelle Eingabezeile gelöscht. Der Kursor rückt auf den Zeilenanfang.

10. Laden und Abarbeiten von BASIC-Programmen

BASIC-Anwenderprogramme wurden entweder mit

CSAVE "Name" [ENTER]

oder mit

LIST#1 "Name" [ENTER]

auf die Kassette gespeichert (vgl. Abschnitt 3.4.).

Entsprechend können sie durch das Kommando

CLOAD "Name" [ENTER]

bzw. das Kommando

LOAD#1 "Name" [ENTER]

in den Rechner geladen werden. Für "Name" muß dabei der konkrete Progammname eingegeben werden. Z. B. wird durch

LOAD#1 "OTHELLO" [ENTER]

das Programm "OTHELLO" geladen. Natürlich muß das Magnetband vorher vor das entsprechende Programm positioniert werden. Die Kommandos CLOAD und LOAD werden in den Abschnitten 13.7 und 13.27 genauer beschrieben. Bezüglich der Bedienung des Magnetbandgerätes (auch im Fehlerfall) wird auf die Bedienungsanleitung verwiesen.

Achtung! Vom Hersteller geschützte Programme können nicht wieder mit CSAVE auf eine Kassette gespeichert werden.

Die Abarbeitung geladener Programme beginnt mit [RUN] . Danach können die Programme Eingaben anfordern, die im allgemeinen aus INPUT-Anweisungen im Programm resultieren. Diese Eingaben (Zahlen oder Zeichenketten) erfolgen über die Tastatur und müssen mit [ENTER] abgeschlossen werden. Damit wird gleichzeitig die Programmabarbeitung fortgesetzt. Die Programmabarbeitung kann auch zu beliebigem Zeitpunkt durch [PAUSE] unterbrochen und dann durch [CONT] wieder fortgesetzt werden.

Dagegen wird ein Programmlauf durch [STOP] abgebrochen, wobei für einen Abbruch beispielsweise bei der Abarbeitung von Zeile 150 eine Information gemäß

```
BREAK IN 150
OK
>
```

erscheint. Der BASIC-Interpreter befindet sich dann im Kommando-Modus. Es können alle dort möglichen Anweisungen eingegeben werden. Insbesondere ist auch ein Zugriff auf die Programmvariablen möglich. Gegebenenfalls kann die Programmabarbeitung durch

GOTO Zeilennummer [ENTER]

in einer beliebigen Programmzeile fortgesetzt werden.

11. Beschreibungsform der Sprache BASIC

Für die Darstellung des Formates der Anweisungen, Kommandos und Funktionen des BASIC-Interpreters werden folgende Schreibregeln (syntaktische Elemente) verwendet:

- Worte in Großbuchstaben
  Sie repräsentieren reservierte Worte (siehe Anhang A3) und müssen wie angegeben geschrieben werden.

- Worte in Groß- und Kleinschrift werden zur Bezeichnung von variablen Größen verwendet, die der Anwender einsetzt.
- Eckige Klammen ([ ])
  Worte in eckigen Klammern sind wahlfrei, sie können entfallen.
- Dreipunkt (...)
  Worte, auch in eckigen Klammern eingefaßt, denen ein Dreipunkt folgt, können bis zur maximalen Länge einer Anweisungzeile (72 Zeichen) wiederholt werden.
- Sonderzeichen
  Alle Sonderzeichen, außer eckige Klammern, müssen wie angegeben geschrieben werden. Der Doppelpunkt z.B. trennt Anweisungen, die sich in einer Zeile befinden, voneinander.

12. Zeilenformat

BASIC-Programmzeilen haben folgendes allgemeine Format:

n BASIC-Anweisung [: BASIC-Anweisung] ...

Der Programmierer kann mehrere Anweisungen in eine Zeile von max. 72 Zeichen Länge schreiben. Jede weitere Anweisung muß jedoch von der vorhergehenden durch einen Doppelpunkt getrennt werden. Fortsetzungszeilen sind nicht möglich.

Jede BASIC-Programmzeile beginnt mit einer Zeilennummer n. Die Zeilennummern legen die Abarbeitungsreihenfolge der einzelnen Anweisungen fest, wenn sie nicht durch Sprunganweisungen geändert wird.

Es sind nur ganzzahlige Zeilennummern zwischen 1 und 65529 erlaubt.

Beispiel:

```
1 X=10          10 X=10
2 Y=20   besser ist: 20 Y=20
3 Z=X*Y         30 Z=X*Y
4 PRINT Z       40 PRINT Z
```

Die Zeilennummern können aufeinanderfolgende Zahlen sein, z.B. 1,2,3,4. Günstiger ist es jedoch, in 10er-Schritten zu numerieren. Dadurch ist ein späteres Einfügen weiterer Anweisungen zwischen existierende leicht möglich.

Beispiel:

```
20 X1= 15
30 X2 = 2  5
5 0 PRINT X1 - X2
40IF  X1< X2 THEN PRINT X 1
10 REM KOMMENTARZEILE
```

Nach der Eingabe dieses Programms über die Tastatur sortiert der BASIC-Interpreter die Zeilen in aufsteigender Reihenfolge.

Das eingegebene Programm wird vom BASIC-Interpreter wie folgt ausgelistet:

```
10 REM KOMMENTARZEILE
20 X1= 15
30 X2 = 2  5
40 IF  X1< X2 THEN PRINT X 1
50 PRINT X1 - X2
```

Anmerkungen zur Verwendung von Leerzeichen:

Zwischen den Zeilennummern 40 und dem weiteren Text dieser BASIC-Zeile fügt der BASIC-Interpreter automatisch ein Leerzeichen ein. In Zeilennummern (hier 50) werden Leerzeichen entfernt. Leerzeichen in Variablen und Konstanten werden ignoriert. Aus diesem Grund sind für den BASIC-Interpreter X 1 und X1 bzw. 2 5 und 25 identisch. Bei Abarbeitung des Programms erscheinen folgende Angaben auf dem Bildschirm: *

* Hier und in den folgenden Abschnitten werden die Meldung "OK" und die Eingabeanforderung ">" nicht mit dargestellt. Ebenso wird [ENTER] nicht mehr dargestellt.

```
RUN
 15
-10
```

Für die gute Lesbarkeit eines Programms sollte jeweils ein Leerzeichen als Trennzeichen zwischen Worten verwendet werden. Zu viele Leerzeichen erhöhen den Speicherplatzbedarf des Programmes unnötig (s. A2).

13. BASIC-Anweisungen und -Kommandos

Im folgenden werden alle BASIC-Anweisungen und -Kommandos in alphabetischer Reihenfolge beschrieben. Es wird dabei die Notation, die im Abschnitt 11 beschrieben ist, verwendet. Im Normalfall können die BASIC-Anweisungen und -Kommandos in beiden Modi des BASIC-Interpreters (Programm- und Kommandomodus) benutzt werden. Ausnahmen sind gekennzeichnet.

13.1. AUTO

Format 1: AUTO
Format 2: AUTO Zeilennummer [,Schrittweite]

Durch das Kommando werden selbständig, d. h. vom BASIC-Interpreter, Zeilennummern bei der Eingabe von Programmen über Tastatur erzeugt. Man kann sich auf diese Weise die Eingabe von Zeilennummern ersparen. Die Parameter Zeilennummer und Schrittweite geben die erste zu erzeugende Zeilennummer bzw. die Differenz zweier aufeinanderfolgender Zeilennummern an. Wenn die so erzeugte Zeilennummer mit einer schon im Programm vorhandenen übereinstimmt, wird anschließend an die Zeilennummer das Zeichen "*" ausgegeben. Danach kann die neue Zeile eingegeben werden. Bei sofortigem Abschluß der Zeile mit [ENTER] wird eine schon vorhandene Programmzeile mit dieser Zeilennummer gelöscht. So kann eine Folge von Programmzeilen einfach gelöscht werden. Fehlen die Parameter, werden Standardwerte angenommen.

Standardwert für Zeilennummer: 10
Standardwert für Schrittweite: 10

Der AUTO-Modus wird durch Drücken der STOP -Taste verlassen.

13.2. BEEP

Format: BEEP

Mit der Anweisung BEEP wird ein Ton durch den im Heimcomputer eingebauten Summer erzeugt.

Beispiel:

```
10 DIM Z(5)
20 FOR J=0 TO 5
30     BEEP
40     INPUT "Zahl eingeben:";Z(J)
50 NEXT J
60 FOR J=0 TO 5
70     PRINT Z(J)
80 NEXT J
90 END
RUN
Zahl eingeben:
```

Vor jeder Eingabeaufforderung ertönt ein Summerton. Die eingegebenen Werte werden nach Beendigung der Eingaben ausgedruckt.

13.3. BORDER

Format: BORDER Farbcode

Mit dieser Anweisung wird die Farbe des Bildschirmrandes festgelegt. Nach Ausführung der Anweisung leuchtet der Bildschirmrand in der angegebenen Farbe.

Farbcode: Es ist die Angabe einer ganzzahligen Konstanten oder einer Variablen möglich. Den Farbcodewerten 1 bis 8 sind folgende Farben zugeordnet:

1 - schwarz
2 - rot
3 - grün
4 - gelb
5 - blau
6 - purpur
7 - zyan
8 - weiß

Die BORDER-Anweisung hat nur Wirkung, wenn die zusätzliche Farbausrüstung vorhanden ist. Anderenfalls wird sie ignoriert.

Beispiel: BORDER 3

Nach Ausführung dieser Anweisung leuchtet der Bildschirmrand grün auf.

13.4. BYE

Verlassen des BASIC-Interpreters mit einem Rücksprung in das Steuerprogramm (Monitor). Anschließend meldet sich das Steuerprogramm mit einer Eingabeanforderung.

Danach können Steuerprogrammkommandos eingegeben werden. Will man wieder mit dem BASIC-Interpreter arbeiten, muß er erneut gestartet werden.

13.5. CALL

Die CALL-Anweisung kann in zwei Formaten benutzt werden:

Format 1: CALL Adresse (dezimal)
Format 2: CALL* Adresse (hexadezimal)

Durch diese Anweisung hat man die Möglichkeit, ein Maschinencodeprogramm aufzurufen, das auf der angegebenen Adresse im Speicher beginnt. Dieses Maschinencodeprogramm muß mit dem Assemblerbefehl RET (ØC9H) abgeschlossen werden und darf die CPU-Register nicht verändern.

Die Anfangsadresse des Maschinencodeprogramms kann entweder im Format 1 dezimal oder im Format 2 hexadezimal angegeben werden.

```
Beispiel:        10 CALL  12298
          oder   10 CALL* 3ØØA
```

Im beiden Fällen führt der BASIC-Interpreter einen Sprung zur hexadezimalen Adresse 300A (dezimal 12298) aus.

13.6. CLEAR

```
Format 1:        CLEAR
Format 2:        CLEAR  I[,J]
```

Das CLEAR-Kommando ohne Parameter löscht alle bereits vereinbarten Variablen. Numerische Variable werden auf Null gesetzt, und Zeichenkettenvariable erhalten die Länge Null (leere Zeichenkette).

Zusätzlich können bei Verwendung des Formates 2 folgende Festlegungen getroffen werden:

Mit dem ersten Parameter wird zusätzlich der maximal zur Verfügung gestellte Arbeitsspeicheranteil zur Speicherung von Zeichenkettenvariablen festgelegt. Standardwert nach Neustart des BASIC-Interpreters sind 256 Zeichen (Byte). Der maximale Wert von I ist abhängig vom verfügbaren Arbeitsspeicher.

Mit dem Ausdruck J kann die obere Grenze des zugewiesenen Arbeitsspeichers neu definiert werden. Diese Anweisung kann benutzt werden, um einen Speicherbereich für andere Verwendungszwecke freizuhalten. Als Standard wird vom BASIC-Interpreter der gesamte zur Verfügung stehende Arbeitsspeicher (RAM) benutzt.

Beispiel:

```
10 Y=5
20 PRINT Y
30 CLEAR
40 PRINT Y
RUN
5
0
```

Nach CLEAR hat die Variable Y den WERT 0.

13.7. CLOAD

Diese Anweisung bzw. dieses Kommando kann in folgenden Formen eingegeben werden:

Format 1: CLOAD "Name"
Format 2: CLOAD* "Name"; Feldname

Durch diese Anweisung können Informationen vom Kassettenmagnetband gelesen werden, die mit dem CSAVE-Kommando auf die Kassette geschrieben wurden.

Format: CLOAD "Name"

Ein mit dem Kommando

CSAVE "Name"

auf eine Kassette ausgelagertes Programm wird mit

CLOAD "Name"

in den Arbeitsspeicher des Heimcomputers geladen. Die Kassette muß vorher vor den Anfang des Programms positioniert werden.

Beispiel: CLOAD "SPIEL"

Mit dem Drücken der ENTER-Taste beginnt der Einlesevorgang von der Kassette. Die Zeitdauer des Einlesens richtet sich nach der

Länge des einzulesenden Programms.

Format: CLOAD* "Name";Feldname

Das angegebene Feld eines Programms wird in den Speicher des Heimcomputers geladen. Während der Ausführung eines Programms kann auf diese Weise ein Feld mit definierten Werten belegt werden, die zu einem früheren Zeitpunkt mit der Anweisung

CSAVE* "Name";Feldname

auf einer Kassette gespeichert wurden.

Beim Dateneinlesen von Kassette können unter Umständen Lesefehler auftreten. Die Art der Fehler und die Fehlerreaktionen sind in der Bedienungsanleitung aufgeführt.

13.8. CLS

Diese Anweisung löscht den aktuellen Ausgabebereich des Bildschirms, der durch die Anweisung WINDOW definiert wird. Der Kursor befindet sich ausschließlich auf der ersten Kursorposition des aktuellen Ausgabebereichs des Bildschirms. Standardmäßig, d. h. ohne zuvor erfolgte Benutzung der Anweisung WINDOW, wird der gesamte Bildschirm gelöscht, und der Kursor befindet sich danach am linken Rand der ersten Bildschirmzeile.

Beispiel:

```
.
.
.
120 CLS
130 PRINT "1. Position des Fensters"
```

Durch die Zeile 120 wird in einem Programm der gesamte aktuelle Ausgabebereich des Bildschirms gelöscht und anschließend der Text "1. Position des Fensters" ausgegeben.

13.9. CONT (nur für Kommandomodus)

Die Abarbeitung eines Programms, die durch Drücken der STOP-Taste oder durch eine STOP-Anweisung abgebrochen wurde, kann durch die Eingabe des Kommandos CONT fortgesetzt werden.

Will man ein Programm an einer bestimmten Stelle fortsetzen, muß man im Kommandomodus

GOTO Zeilennummer

eingeben.

CONT kann im Zusammenhang mit der STOP-Anweisung zur Fehlersuche in einem Programm verwendet werden. Nach einer STOP-Anweisung befindet sich der BASIC-Interpreter im Kommandomodus. Jetzt können Zwischenergebnisse angezeigt und verändert werden. Anschließend kann die Programmausführung durch die Eingabe von CONT bzw. GOTO Zeilennummer fortgesetzt werden.

Beispiel: Siehe STOP-Anweisung (in Abschnitt 13.43).

13.10. CSAVE

Das CSAVE-Kommando bzw. die CSAVE-Anweisung gibt es in zwei Formaten.

Format 1: CSAVE "Name"
Format 2: CSAVE* "Name";Feldname

Format: CSAVE "Name"

Ein im Speicher befindliches BASIC-Programm wird unter dem angegebenen Namen als Programmdatei auf der Kassette abgespeichert. Der Name muß mindestens 4 Zeichen lang sein. Bis zu 8 Zeichen sind signifikant. Vor dem Drücken der ENTER-Taste muß das Kassettengerät auf Aufnahme geschaltet sein.

Die so abgespeicherten Programme kann man mit

CLOAD "Name"

wieder einlesen.

Format: CSAVE* "Name";Feldname

Mit dieser vorzugsweise im Programmodus verwendeten Anweisung kann ein Feld (siehe Feldvariable) eines Programms abgespeichert werden.

Hinweis: Nach erfolgter Auslagerung (eines Programms bzw. des Feldes eines Programms) fragt der Heimcomputer, ob die ausgelagerten Daten durch einen Einlesevorgang auf Lesbarkeit überprüft werden sollen oder nicht (VERIFY). Die Handlungen, die dazu notwendig sind, sind in der Bedienungsanleitung erläutert.

13.11. DATA (nur für Programmodus)

Die DATA-Anweisung muß in der folgenden Form eingegeben werden:

DATA Konstante [,Konstante] ....

Durch diese Anweisung werden numerische und Zeichenkettenkonstanten abgespeichert, die danach mit der READ-Anweisung gelesen werden können (siehe Abschnitt 13.36).

Jede DATA-Anweisung kann so viele Konstanten enthalten, wie in eine Zeile passen (72 Zeichen). DATA-Anweisungen können an beliebiger Stelle im Programm stehen. Die Zahl der DATA-Anweisungen ist ebenfalls beliebig. Die READ-Anweisung liest die einzelnen DATA-Zeilen in der Reihenfolge ihrer Zeilennummern.

Der in der READ-Anweisung angegebene Variablentyp (numerische oder Zeichenkettenvariable) muß mit dem entsprechenden Konstantentyp in der DATA-Anweisung übereinstimmen.

Beispiel:

```
10 DATA "Das ","ist ","ganz ","einfach!"
20 FOR I=1 TO 4
30 READ X$
40 PRINT X$;
50 NEXT I
RUN
Das ist ganz einfach!
```

13.12. DEF FN (nur für Programmodus)

Format: DEF FNName[(Parameter)] = Funktionsdefinition

Dabei bedeuten:

Name: Variablenname, der durch das Voranstellen von FN als Funktion deklariert wird.

Parameter: Wahlweise kann eine Variable als Parameter, das Argument der Funktion, angegeben werden. Dieser wird beim Aufruf der Funktion durch den aktuellen Parameter ersetzt.

Funktionsdefinition:

Ist ein beliebiger Ausdruck, der die Funktion darstellt, die ausgeführt werden soll. Die Länge des Ausdrucks ist auf eine Anweisungszeile begrenzt. Variablennamen in diesem Ausdruck dienen nur der normalen Funktionsdefinition und können auch als Variable im eigentlichen Programm benutzt werden. Ein Variablenname kann als Parameter auftreten, muß aber nicht. Ist der Parameter (formaler Parameter) angegeben, wird er durch den im Funktionsaufruf benutzten Variablennamen (aktueller Parameter) ersetzt.

Die "DEF FN"-Anweisung muß ausgeführt werden, bevor der Funktionsaufruf erfolgt. Ist dies nicht der Fall, reagiert der BASIC-Interpreter mit einer Fehlerausschrift (siehe Fehlermeldung UF).

Beispiel 1:

```
100 DEF FNAB(Y) = Y^2+3*Y+X
.
.
.
310 X=7
320 B=FNAB(A)
```

In der Zeile 100 wird die Funktion definiert, die in der Zeile 320 in der Form $B=A^2+3A+X$ aufgerufen wird. Dabei wird die Variable Y durch den aktuellen Wert der Variablen A ersetzt. Die Variable X erhält den Wert, den sie z. Z. des Funktionsaufrufes hat, in diesem Fall den Wert 7.

Beispiel 2:

```
20 DEF FNAS(X)=ATN(X/SQR(1-X*X))
.
.
.
150 C = FNAS(N)
```

Beim Aufruf in der Zeile 150 wird C = Arcsin (N) gebildet. Als Argument der Arcustangens-Funktion tritt hier wieder eine Funktion auf.

13.13. DELETE

Format: DELETE Zeilennummer 1 [, Zeilennummer 2]

Mit diesem Kommando können in einem BASIC-Programm Zeilen gelöscht werden. Ist nur ein Parameter angegeben, wird nur die spezifizierte Zeile gelöscht. Sind zwei Zeilennummern in der Anweisung enthalten, wird der angegebene Programmbereich gelöscht.

Beispiel: DELETE 100, 150

Die Zeilen mit Zeilennummern zwischen 100 und 150 werden in einem BASIC-Programm gelöscht, einschließlich der Zeilen 100 und 150.

13.14. DIM

Format: DIM Feldname (Feldindex [, Feldindex] ...)

Mit dieser Anweisung wird der Speicherplatz für die Elemente eines Feldes reserviert und jedes Feldelement gelöscht (Null bzw. leere Zeichenkette).

Wird die DIM-Anweisung für ein im BASIC-Programm verwendetes Feld nicht angegeben, werden standardmäßig 11 Feldelemente vereinbart (Feldindex 0 bis 10).

Feldname: Name des Feldes

Feldindex: Ausdruck, der die max. Anzahl von Feldelementen der jeweiligen Dimension festlegt. Der Ausdruck muß ganzzahlig sein bzw. wird als ganze Zahl behandelt (in diese umgeformt).

Der kleinstmögliche Index ist immer Null. Es könnten Felder (Matrizen) mit bis zu 255 Dimensionen angegeben werden, von denen jede max. 32 767 Elemente enthalten darf. Das sind aber theoretische Werte. Praktisch kommt man mit weit weniger Dimensionen aus. Außerdem ist die Mehrdimensionalität bzw. die Feldgröße durch die max. BASIC-Zeilenlänge bzw. den verfügbaren Speicher begrenzt.

Beispiel 1:

```
10 DIM F(5)
20 FOR I=0 TO 3
30 F(I)=10+I
40 NEXT I
50 FOR I=0 TO 5
60 PRINT "F("I")="F(I)
70 NEXT I
RUN
F( 0 )= 10
F( 1 )= 11
```

```
F( 2 )= 12
F( 3 )= 13
F( 4 )=  0
F( 5 )=  0
```

Beispiel für ein eindimensionales Feld. Die Feldelemente F(4) und F(5) werden in der FOR...NEXT-Anweisung nicht mit Werten belegt. Die automatische Zuweisung des Wertes Null der DIM-Anweisung bleibt erhalten.

Die Zeile 10 kann auch weggelassen werden, da dieses Feld weniger als 11 Feldelemente besitzt. Allerdings erhöht sich dann der Speicherbedarf des Programmes um den für die nichtbenötigten Feldelemente F(6) bis F(10).

Beispiel 2:

```
 10 DIM A(3,3)
 20 FOR I=1 TO 3
 30 FOR J=1 TO 3
 40 A(I,J)=10*I+J
 50 NEXT J
 60 NEXT I
 70 FOR I=1 TO 3
 80 FOR J=1 TO 3
 90 PRINT A(I,J);
100 NEXT J
110 PRINT
120 NEXT I
 RUN
 11  12  13
 21  22  23
 31  32  33
```

Beispiel für ein zweidimensionales Feld.

Die Elemente mit dem Feldindex 0 wurden in diesem Programm nicht berücksichtigt.

13.15. DOKE

Format: DOKE Adresse, Wort

Ein Wort, bestehend aus 2 Bytes, wird in den Arbeitsspeicher geschrieben. Das niederwertige Byte erscheint auf der angegebenen Adresse, während das höherwertige Byte des Wortes auf die Adresse+1 geschrieben wird.

Um den gesamten Speicher adressieren zu können, muß gegebenenfalls auch mit negativen Zahlen als Adressen gearbeitet werden (siehe Funktion DEEK).

13.16. EDIT

Format: EDIT Zeilennummern

Editieren eines im Arbeitsspeicher befindlichen Programms ab der angegebenen Zeile. Nach Eingabe dieses Kommandos wird die BASIC-Programmzeile mit der eingegebenen Zeilennummer im Eingabebereich angezeigt und bereitgestellt. Das EDIT-Kommando sollte nur im Kommandomodus eingegeben werden.

Anschließend können fortlaufend BASIC-Programmzeilen korrigiert werden. Die aktuell angezeigte Zeile kann verändert, gelöscht oder unverändert wieder im Arbeitsspeicher abgelegt werden.

Ein aktuell auftretender Stern (*) hinter der Zeilennummer zeigt an, daß die Zeile nicht vollständig in den Eingabebereich paßt. Der Fall kann eintreten, wenn eingabeseitig Abkürzungen für reservierte Worte (z.B. ? statt PRINT) verwendet wurden. Soll die zu lange Zeile unverändert erhalten bleiben, kann man auf zwei Arten reagieren. Durch sofortiges Drücken der Taste [↓] wird die Zeile im Arbeitsspeicher abgelegt und die nächste angezeigt. Durch Betätigen der [STOP] -Taste wird die Zeile abgelegt, und der BASIC-Interpreter befindet sich anschließend im Kommandomodus.

Folgende Aufbereitungsmöglichkeiten einer Zeile stehen zur Verfügung:

Zeichen einfügen:	Kursor wird bei Bedarf mit Hilfe der Tasten [→] und [←] nach rechts oder links bewegt. Durch Betätigen der [INS] -Taste wird das Einfügen ab Kursorposition ermöglicht, d. h., es erfolgt eine Rechtsverschiebung.
Zeichen streichen:	Bei Bedarf Kursorbewegungen ausführen. Durch Drücken der [DEL] -Taste wird das Zeichen, auf dem der Kursor steht, gelöscht. Der Kursor bleibt unverändert.
Zeile unverändert übernehmen:	Alle Zeilen, einschließlich der mit einem Stern gekennzeichneten, können durch Betätigen der Taste [↓] unverändert im Arbeitsspeicher abgelegt werden. Anschließend wird die nächste Zeile angezeigt.
Zeile wie angezeigt übernehmen:	Mit der [ENTER] -Taste wird die Aufbereitung einer Zeile abgeschlossen. Sie wird im Arbeitsspeicher so abgelegt, wie sie angezeigt ist. Anschließend erfolgt die Anzeige der nächsten Programmzeile.
Beenden des Editierens:	Durch Betätigung der [STOP] -Taste wird der Kommandomodus erreicht. Das ist automatisch der Fall, wenn die letzte BASIC-Programmzeile übernommen wurde.

13.17. END

Format: END

Diese Anweisung beendet den Programmablauf und setzt den Rechner in den Kommandomodus. Als physisches Ende eines Programmes kann END stehen, ist aber nicht erforderlich. END-Anweisungen können zur Programmbeendigung an beliebiger Stelle des Programms stehen.

Im Gegensatz zur STOP-Anweisung erzeugt END keine Meldung auf dem Bildschirm.

Beispiel:

```
.
.
.
100 IF K=50 THEN END
.
.
.
```

13.18. FOR...NEXT

Format:

```
FOR numerische Variable = X TO Y  STEP Z
.
.
.
NEXT   numerische Variable, ...
```

Dabei stehen X, Y und Z für numerische Ausdrücke. Diese Anweisung führt eine Gruppe von anderen Anweisungen in einer Schleife bei vorgegebener Anzahl von Durchläufen aus.

Numerische Variable: Sie dient als Durchlaufzähler der Schleife.

X : Anfangswert des Zählers
Y : Endwert des Zählers
Z : Schrittweite der Erhöhung des Zählers

Bei der ersten Ausführung erhält die numerische Variable den Wert von X, und alle Anweisungen bis zum zugehörigen NEXT werden ausgeführt. Danach wird der Zähler um die Schrittweite Z erhöht. Wird die Schrittweite nicht mit STEP vereinbart, wird standardmäßig Z = 1 gesetzt. Anschließend erfolgt der Abbruchtest für den Schleifendurchlauf. Die Anweisungen in einer Schleife werden also mindestens einmal ausgeführt.

Es erfolgt so lange eine Weiterbearbeitung der Schleife, wie die numerische Variable (Durchlaufzähler) ≤ dem Endwert Y ist. Für

negative Schrittweiten wird der Durchlaufzähler jedesmal um die Schrittweite erniedrigt. In diesem Falle erfolgt die Weiterbearbeitung der Schleife so lange, wie der Zähler ≧ dem Endwert Y ist. Anderenfalls wird die Bearbeitung mit der auf NEXT folgenden Anweisung fortgesetzt. Die numerische Variable kann innerhalb der Schleife verwendet werden, sollte aber nur gezielt verändert werden. Schleifen können beliebig oft ineinandergeschachtelt werden. Jeder Schleifenzähler muß dann jedoch einen eigenen Namen haben.

Hinweis: Enden mehrere ineinandergeschachtelte Schleifen an derselben Stelle, kann ein gemeinsamer Schleifenabschluß mittels NEXT, gefolgt von den Zählervariablen in der richtigen Reihenfolge, angegeben werden.

Fehlt die Bezeichnung der numerischen Variablen unter NEXT, so bezieht sich das NEXT auf die zuletzt bearbeitete FOR...NEXT-Anweisung.

Eine Programmschleife kann vor Erreichen der mit FOR...TO angegebenen Abbruchbedingung verlassen werden.

Beispiel 1:

```
10 N=10
20 FAK=1
30 FOR I=1 TO N
40 FAK=FAK*I
50 PRINT I;FAK
60 NEXT I
RUN
1  1
2  2
3  6
4  24
5  120
6  720
7  5040
8  40320
9  362880
10  3.6288E+06
```

Das ist ein Beispiel zur Berechnung von N-Fakultät. Der Durchlaufzähler wird hier zur Berechnung der Fakultät benutzt.

Beispiel 2:

```
10 FOR I=1 TO 3
20 FOR J=1 TO 3
30 PRINT I;J
40 NEXT J,I
RUN
1  1
1  2
1  3
2  1
2  2
2  3
3  1
3  2
3  3
```

Das ist ein Beispiel für geschachtelte Schleifen.

Beispiel 3:

```
10 K=0
20 FOR I=1 TO K
30 PRINT I
40 NEXT I
RUN
1
```

Der Anfangswert ist größer als der Endwert. Trotzdem wird die Schleife einmal durchlaufen, da der Abbruchtest erst beim Erreichen von NEXT erfolgt.

Beispiel 4:

```
10 A=5
20 FOR I=0 TO 10
30 A=A-I
40 IF A<0 GOTO 60
50 NEXT I
60 PRINT A
RUN
-1
```

Beispiel dafür, wie vor Erreichen der Abbruchbedingung eine Schleife verlassen werden kann.

13.19. GOSUB

Format:

```
GOSUB Zeilennummer
.
.
.
REM   Beginn Unterprogramm
.
.
.
RETURN
```

Verzweigung zu einem Unterprogramm, das auf der angegebenen Zeile beginnt. Das BASIC-Unterprogramm muß mit der Anweisung RETURN enden. Nach Ausführung des Unterprogramms wird die Programmabarbeitung mit der auf die GOSUB-Anweisung folgenden Anweisung fortgesetzt.

Ein BASIC-Unterprogramm kann mehrere RETURN-Anweisungen enthalten, wenn ein Rücksprung von logisch unterschiedlichen Stellen erforderlich ist.

Beispiel:

```
10 I=1
20 GOSUB 80
30 PRINT "1.Rückkehr"
40 I=2
50 GOSUB 80
60 PRINT "2. Rückkehr"
70 END
80 PRINT "Aufruf ";I
90 RETURN
RUN
Aufruf 1
1. Rückkehr
Aufruf 2
2. Rückkehr
```

13.20. GOTO

Format: GOTO Zeilennummer

Mit dieser Anweisung wird ein unbedingter Sprung zur angegebenen Zeilennummer erzwungen. Existiert die spezifizierte Zeile nicht, wird das Programm mit der nächsthöheren vorhandenen Zeile fortgesetzt.

Beispiel: Siehe Abschnitt 13.18, Beispiel 4.

13.21. IF...THEN...:ELSE

Format 1: IF Ausdruck THEN Zeilennummer [:ELSE Zeilennummer]
Format 2: IF Ausdruck THEN Anweisung [: Anweisung] ...
[:ELSE Anweisung [: Anweisung] ...]

Die IF-Anweisung wird zur Steuerung des Programmablaufs in Abhängigkeit von dem Wert eines Ausdruckes benutzt. Falls das Ergebnis des Ausdruckes den Wert "wahr" liefert, d. h. ungleich Null, wird das Programm bei der nach THEN angegebenen Zeilennummer fortgesetzt bzw. die oder mehrere folgende Anweisungen werden ausgeführt. Die Zeilennummer bzw. Anweisung(en) hinter ELSE werden in diesem Fall ignoriert.

Ist der Wert des Ausdruckes "falsch", d. h. gleich Null, wird der THEN-Zweig ignoriert, und die nach ELSE stehenden Anweisungen werden ausgeführt. Ist der Wert des Ausdruckes gleich Null, und der ELSE-Zweig ist nicht vorhanden, wird das Programm in der nächsten Zeile fortgesetzt.

Beispiel:

```
50 IF (I>=100)AND(I<=200)THEN X=3 : GOTO 150:
ELSE GOTO 60
60 PRINT "AUSSERHALB DES BEREICHES"
```

Falls der Ausdruck "wahr" ist, d.h., I liegt zwischen 100 und 200, wird X mit dem Wert 3 belegt, und das Programm wird mit der Zeile 150 fortgesetzt. Ist die Bedingung nicht erfüllt, wird das

Programm mit der Zeile 60 fortgesetzt. Die beiden Klammern in Zeile 50 können auch weggelassen werden (siehe Abschn. 7).

13.22. INK

Format: INK Farbcode

Diese Anweisung wird zur Einstellung der Vordergrundfarbe benutzt. Nach Ausführung dieser Anweisung erscheinen bei den folgenden Ein- und Ausgaben die Zeichen in der angegebenen Farbe auf dem Bildschirm. Der Farbcode ist der gleiche wie bei den Anweisungen BORDER und PAPER (siehe dazu Anweisung BORDER).

Beispiel:

```
10 INK5
20 PRINT "DIESER TEXT IST BLAU"
25 INK2
30 PRINT "JETZT SIND DIE BUCHSTABEN ROT"
RUN
DIESER TEXT IST BLAU
JETZT SIND DIE BUCHSTABEN ROT
```

Ist der HC nicht farbtüchtig, erscheinen beide Zeilen in schwarz/weiß auf dem Bildschirm.

13.23. INPUT (nur für Programmodus)

Format: INPUT ["Zeichenkette";] Variable [, Variable] ...

Die INPUT-Anweisung ist eine Aufforderung zur Dateneingabe über die Tastatur während der Programmausführung. Erkennt der BASIC-Interpreter die Eingabeanweisung, erscheint die wahlweise angebbare Zeichenkette bzw. nur ein Fragezeichen, wenn die Zeichenkette fehlt. Die anschließend einzugebenden Werte werden den Variablen zugewiesen. Die Anzahl der einzugebenden Werte, getrennt durch Komma, muß mit den Variablen der INPUT-Anweisung übereinstimmen. Es sind numerische Variable, Zeichenkettenvariable und Feldvariable möglich.

Werden zu wenig Daten eingegeben, erfolgt eine wiederholte Eingabeanforderung mit "??". Werden zu viele Daten in einer Eingabezeile angegeben, wird dies mit der Ausschrift EXTRA IGNORD angezeigt. Der Typ der eingegebenen Daten muß mit dem Typ der Variablen übereinstimmen (numerischer Wert oder Zeichenkette).

Fehlerhafte Eingaben werden mit

```
?REDO FROM START
```

quittiert und müssen wiederholt werden.

Wird eine Eingabeanforderung nur mit dem Drücken der Taste [ENTER] beantwortet, ändert sich der Wert der Variablen nicht.

Beispiel:

```
10 INPUT "EINGABE ZAHL:";X
20 PRINT X;X^2;X^3
RUN
EINGABE ZAHL: 6
 6  36  216
```

In diesem Beispiel wurde bei der Eingabeanforderung eine 6 eingegeben. Die Eingabe muß mit dem Drücken der Taste [ENTER] abgeschlossen werden.

13.24. LET

Format: [LET] Variable = Ausdruck

Einer Variablen wird der Wert eines Ausdruckes zugewiesen. Das reservierte Wort LET kann weggelassen werden.

Beispiel:

```
100 LET A=13: LETB=13*13
110 LET C=169/13
120 LET D=A+B+C
```

Eine kürzere Schreibweise mit der gleichen Bedeutung kann folgendermaßen aussehen:

```
100 A=13 : B=13*13
110 C=169/13
120 D=A+B+C
```

13.25. LINES

Format: LINES Ausdruck

Dieses Kommando legt die Anzahl der auf einmal auszugebenden Programmzeilen beim Listen auf dem Bildschirm fest. Standardmäßig wird eine feststehende Zeilenanzahl (10) erzeugt. Auf diese Weise kann man Programme mit dem LIST-Kommando durch Betätigen der [LIST] -Taste bequem auf dem Bildschirm zur Anzeige bringen.

Beispiel:

```
LINES 15          LINES 15
LIST      oder    [LIST]
```

Nach Eingabe dieser beiden Zeilen im Kommandomodus erscheinen die ersten 15 Zeilen des im Speicher befindlichen Programms. Durch Drücken der [ENTER] -Taste werden jeweils die nächstfolgenden 15 Zeilen des Programmes angezeigt.

13.26. LIST

Format 1: LIST [Zeilennummer]

Format 2: LIST # Gerätenummer "Name" [Zeilennummer]

Bei Verwendung von Format 1 wird das im Speicher befindliche Programm, beginnend mit der niedrigsten oder angegebenen Zeilennummer, auf dem Bildschirm aufgelistet. Es werden immer die Standardzahl (10) oder die mit dem LINES-Kommando festgelegte Anzahl Zeilen ausgegeben. Der nächstfolgende Zeilenblock wird durch [ENTER] zur Anzeige gebracht. Durch Drücken der [STOP] -Taste ist ein Abbruch der Listenausgabe über den Bildschirm möglich.

Beispiel: siehe Abschnitt 13.25.

Das Format 2 dient zur Ausgabe des im Speicher befindlichen

Programms im externen Code (ASCII-Code). Das Ausgabegerät wird durch die Gerätenummer spezifiziert. Folgende Gerätenummern sind zulässig:

0 Bildschirm
1 Kassette
2 Drucker
3 nicht belegt

Für das LIST#-Kommando gilt das LINES-Kommando nicht. Der BASIC-Interpreter arbeitet intern so, als hätte man LINES 65536 gegeben. Mit dem LIST#-Kommando können BASIC-Programme im externen Code, d. h. ASCII-Code, auf einer Kassette als Quelltextdaten abgespeichert werden.

Bei der Wahl des Programmnamens gelten die gleichen Bedingungen wie beim CSAVE-Kommando.

Das Betätigen der [LIST] -Taste hat die gleiche Wirkung wie die Eingabe des Kommandos LIST ohne Angabe einer Zeilennummer. Mit der Eingabe einer Zeilennummer kann ein Programm teilweise aufgelistet bzw. ausgelagert werden (siehe Format 1).

13.27. LOAD

Format: LOAD# Gerätenummer "Name"

Lädt ein durch ein LIST#-Kommando ausgegebenes Programm vom angegebenen Gerät.

Die Zeilen des eingelesenen Programms werden mit denen des schon im Speicher befindlichen entsprechend ihrer Numerierung gemischt. Die Gerätenummer ist die gleiche wie beim LIST#-Kommando. Der Name muß derselbe Programmname wie bei der entsprechenden LIST#-Anweisung sein.

13.28. NEW

Format: NEW

Löscht das im Speicher befindliche Programm und alle Variablen . Die NEW-Anweisung wird gewöhnlich im Kommandomodus eingegeben, um ein neues Programm einzugeben und aufzubereiten (zu editieren). Nach der Ausführung von NEW befindet sich der BASIC-Interpreter im Kommandomodus.

13.29. NULL

Format: NULL Ausdruck

Besonders beim Wiedereinlesen von LIST-gespeichertem Quellprogrammtext kann es infolge der durch den BASIC-Interpreter zu realisierenden zeilenweisen Übertragung in den Zwischencode zu Zeitproblemen kommen.

Das NULL-Kommando legt die Anzahl der am Ende einer Zeile auszugebenden bedeutungslosen Zusatzzeichen fest und kann auch zur Synchronisation von einfachen Ausgabegeräten benutzt werden (standardmäßig gilt NULL (10)).

13.30. ON...GOTO

Format: ON Ausdruck GOTO Zeilennummer [, Zeilennummer]...

Diese Anweisung verzweigt zu einer von mehreren angegebenen Zeilennummern in Abhängigkeit vom Wert des Ausdruckes.

Ausdruck: Der Wert des Ausdruckes wird vor der Verzweigung auf jeden Fall in eine ganze Zahl umgewandelt, d. h., evtl. vorhandene Dezimalstellen werden abgeschnitten.

Wenn der Wert des Ausdruckes z. B. 2 ist, stellt die angegebene zweite Zeilennummer das Sprungziel dar. Ist der Wert des Ausdruckes Null oder größer als die Anzahl der Zeilennummern, so wird das Programm in der nächsten Programmzeile fortgesetzt. Ist der Wert des Ausdruckes negativ, so erscheint die Fehlermeldung FC bei Ausführung des Programmes.

Beispiel: 10 INPUT S
20 ON S GOTO 50,60,70

In Abhängigkeit vom eingegebenen Wert der Variablen S wird bei der Programmausführung verzweigt.

13.31. ON... GOSUB

Format: ON Ausdruck GOSUB Zeilennummer [, Zeilennummer] ...

Wirkung wie ON...GOTO, jedoch werden Zeilennummern von BASIC-Unterprogrammen angegeben.

13.32. OUT

Format: OUT Port, Byte

Mit dieser Anweisung kann man eine 8-bit-Binärinformation über das angegebene Port (Kanaladresse) ausgeben. Damit wird praktisch der Assemblerbefehl OUT realisiert. Mit der OUT-Anweisung kann z. B. der Bildschirm auf das 20-Zeilen-Format umgeschaltet werden (siehe Anhang A4).

Es bedeuten:

Port: Ausdruck im Integerformat, d. h., der Wert des Ausdrucks ist eine ganze Zahl oder wird in eine ganze Zahl umgewandelt.

Byte: Ausdruck im Integerformat. Der Wert muß zwischen 0 und 255 liegen.

Beispiel: 100 OUT 128,32

Auf die Kanaladresse 128 wird der Wert 32 ausgegeben.

13.33. PAPER

Format: PAPER Farbcode

Diese Anweisung wird zur Einstellung der aktuellen Hintergrund-

farbe benutzt. Nach Ausführung dieser Anweisung erscheint der Hintergund der anschließend ausgegebenen Zeichen in der spezifizierten Farbe. Der Farbcode ist der gleiche wie bei der INK- und BORDER-Anweisung. Will man den gesamten Bildschirm mit einer Farbe einfärben, benutzt man zweckmäßigerweise die CLS-Anweisung.

Beispiel:

```
10 A=5
20 PAPER A
30 CLS
```

Nach Ausführung dieses Programms ist der Bildschirm blau, und der Kursor befindet sich auf der ersten Bildschirmposition des aktuellen Fensters. Der Bildschirmrand wird dadurch nicht verändert.

13.34. PAUSE

Format: PAUSE [Zeit]

Die PAUSE-Anweisung unterbricht die Ausführung eines Programms. Für die Unterbrechungszeit läßt sich eine ganze Zahl n angeben. Die Unterbrechung erfolgt in diesem Falle für n Zehntelsekunden. Ohne Zeitangabe erfolgt die Weiterarbeit nach Betätigen der Taste [CONT]. Die weitere Abarbeitung eines Programms ist auch vor Ablauf einer angegebenen Unterbrechungszeit mittels der Taste [CONT] möglich.

13.35. POKE

Format: POKE Adresse, Byte

Es wird eine 8-bit-Binärinformation in die spezifizierte Speicherzelle geschrieben.

Byte: ganzzahliger Ausdruck, dessen Wert zwischen 0 und 255 liegen muß.

Adresse: Adresse des Speicherplatzes

Beispiel:

```
10 POKE -4664,72
20 POKE -4663,65
30 POKE -4662,76 : POKE -4661,76
40 POKE -4660,79 : POKE -4659,33
```

Durch diese Anweisung wird mitten auf dem Bildschirm, in Zeile 11 beginnend, ab Spaltenposition 16 der Text HALLO! geschrieben. Zu beachten ist, daß der BASIC-Interpreter bei der Ermittlung der Bildschimposition immer seine Zählung mit 0 beginnt. Der Bildschirm hat demnach 0 bis 23 Zeilen und 0 bis 39 Zeichenpositionen pro Zeile. Der Zusammenhang zwischen Zeile und Spalte des Bildschirmes und der zugehörigen Adresse im Bildwiederholspeicher ist aus der Übersicht in der Tabellensammlung zu entnehmen.

13.36. PRINT

Format: PRINT [# Gerätenummer] [INK-Anweisung [,PAPER-Anweisung] ;] [Printelement] [,Printelement] ... [,]

#Gerätenummer — Gleiche Gerätenummern wie bei der LIST-Anweisung.

Printelement — Es sind numerische Ausdrücke, Zeichenketten-Ausdrücke, die TAB- und die SPC-Funktion erlaubt. Zeichenkettenkonstanten müssen in Anführungszeichen eingeschlossen werden.

INK
PAPER — Diese beiden Farbanweisungen sind innerhalb der PRINT-Anweisung zugelassen. Sie müssen mit einem Semikolon abgeschlossen werden. Sie haben hier nur lokale Wirkung. Nach Ausführung der PRINT-Anweisung gelten wieder die vorher eingestellten Farben.

Als Trennzeichen zwischen den Printelementen kann das Komma oder das Semikolon (;) stehen (mit Ausnahme der INK- und PAPER-Anweisung).

Bedeutung der Trennzeichen:

Der BASIC-Interpreter teilt eine Druckzeile in Druckzonen von je 13 Zeichen ein. Der Ausdruck nach dem Komma wird zu Beginn eines neuen 13-Zeichen-Feldes ausgegeben.

Es wird kein Zwischenraum zum vorhergehenden Ausdruck erzeugt. Das Semikolon kann als Trennzeichen nach einer Zeichenkettenvariablen, Zeichenkettenkonstanten, einer SPC-Funktion oder einer TAB-Funktion weggelassen werden. Vor einer Zeichenkettenkonstanten kann es auch entfallen.

Wird die PRINT-Anweisung mit einem Trennzeichen beendet, so beginnt die Ausgabe der nächsten PRINT-Anweisung auf der selben Zeile entsprechend dem Trennzeichen.

Numerische Werte werden rechts durch ein Leerzeichen begrenzt, während links entweder ein Leerzeichen oder Minuszeichen steht. Die Darstellung numerischer Werte erfolgt in dem unter 5.1 (numerische Konstanten) angegebenen Format. Zeichenketten werden exakt, d. h. ohne zusätzliche Leerzeichen, ausgegeben.

Für das reservierte Wort PRINT kann abkürzend auch das Fragezeichen stehen. Wird das Programm mit LIST aufgelistet, wird das Fragezeichen (?) durch PRINT ersetzt.

Beispiel:

```
10 INPUT "EINGABE ZAHL :";Z
20 PRINT Z "HOCH 2 IST" Z*Z "UND" ;
30 PRINT Z "HOCH 3 IST" Z^3
40 PRINT
50 GOTO 10
RUN
EINGABE ZAHL: 3
3 HOCH 2 IST 9 UND 3 HOCH 3 IST 27
EINGABE ZAHL:
```

Das Semikolon am Ende der Zeile 20 bewirkt, daß die Ausgaben der Zeilen 20 und 30 auf einer Zeile des Bildschirmes erscheinen.

Beispiel:

```
10 INK4:PAPER5
20 CLS
30 PRINT INK2, PAPER8; "Hervorhebung";:PRINT
40 PRINT "aktuelle Farben"
```

In der Zeile 10 wird die Vordergrundfarbe Gelb und die Hintergrundfarbe Blau eingestellt. Durch CLS in Zeile 20 wird der gesamte Bildschirm blau eingefärbt. Der Text "Hervorhebung" erscheint rot auf weißem Hintergrund. Die Ausschrift "aktuelle Farben" erscheint anschließend gelb auf blauem Hintergrund.

Weitere Beispiele sind in der folgenden Tabelle aufgezeigt. In diesen Beispielen wird auch die BASIC-Funktion TAB verwendet, die im Abschnitt 14.26 beschrieben wird.

Für die verwendeten Variablen wurden folgende Werte angenommen:

```
 X = 9.87654E+08
 Y = 1.23457E+08
X$ = "ABC"
Y$ = "DEF"
```

Die PRINT-Anweisung wirkt auf den aktuellen Ausgabebereich, der ggf. mittels WINDOW abweichend vom kompletten Bildschirm festgelegt sein kann.

Die Ausgabe erfolgt zunächst fortlaufend, bis der Ausgabebereich ausgeschöpft ist. Jede weitere ausgegebene Zeile erscheint dann als unterste Zeile im Ausgabebereich, während die davor befindlichen um eine Zeile höher rücken und die oberste verloren geht, d.h., das Bild "rollt".

13.37. PRINT AT

Format: PRINT [INK-Anweisung[,PAPER-Anweisung];]
AT (Zeile, Spalte); Ausdruck [,Ausdruck] ...

Die angegebenen Ausdrücke werden auf dem Bildschirm ab der durch Zeile und Spalte spezifizierten Position hintereinander ausgegeben. Die Parameter Zeile (0 bis 23) und Spalte (0 bis 39) sind ganzzahlige Ausdrücke (vgl. PRINT-Anweisung). In der "PRINT AT"-Anweisung sind die Funktionen SPC und TAB nicht erlaubt. Das Komma fungiert hier lediglich als Trennzeichen zwischen den Ausdrücken.

Die Anweisung WINDOW beeinflußt die Wirkung der "PRINT AT"-Anweisung nicht.

Beispiel:

```
10 INK4:PAPER5
20 CLS
30 PRINT INK2, PAPER8; AT (4,4);CHR$(34)+
   "PRINT AT"+CHR$(34)+"-Beispiel"
40 PRINT AT (6,4); "aktuelle Farben"
```

Auf der Zeile 4 wird ab der 4. Spaltenposition der Text
"PRINT AT"-Beispiel
mit lokaler Farbeinstellung (rot/weiß) ausgeschrieben. Gelb auf blauem Hintergrund erscheint ab Zeile 6, Spalte 4, der Text
aktuelle Farben

13.38. READ

Format: READ VARIABLE [, VARIABLE] ...

Die Daten von DATA-Anweisungen (siehe 13.11) werden den Variablen in der angegebenen Reihenfolge zugeordnet.

Jeder Variablen der READ-Anweisung wird dabei jeweils eine Konstante aus einer DATA-Anweisung zugewiesen. Die Daten- und Variablentypen müssen dabei übereinstimmen (Zeichenketten-Konstante und Zeichenketten-Variable bzw. numerische Konstante und

Tabelle: Beispiele zur PRINT-Anweisung

Position	Ø	1	2	3	4	5	6	7	8	9	1Ø	11	12	13	14	15	16	17	18	19	2Ø	21	22	23	24	25	26	27	28	29	3Ø	31	32	33	34	35	36	37	38	39
? "A", 4, "B"	A														4												B													
? "A"; 4; "B"	A		4		B																																			
?"ABC"5678"DEF"	A	B	C		5	6	7	8		D	E	F																												
? "ABC", 123	A	B	C												1	2	3																							
? TAB(16)1																		1																						
? TAB(1Ø)1												1																												
? TAB(1Ø),1															1																									
? 1SPC(5)2 SPC(5)3		1								2								3																						
? X,Y		9	.	8	7	6	5	4	E	+	Ø	8			1	.	2	3	4	5	7	E	+	Ø	8															
? X"+"Y		9	.	8	7	6	5	4	E	+	Ø	8		+		1	.	2	3	4	5	7	E	+	Ø	8														
? X$; Y$	A	B	C	D	E	F																																		
? X$ Y$	A	B	C	D	E	F																																		
? X$" "Y$	A	B	C		D	E	F																																	
? X$, Y$	A	B	C											D	E	F																								
? X$ + X$	A	B	C	D	E	F																																		
? TAB(1Ø)X$; TAB(5)Y$											A	B	C	D	E	F																								
? TAB(5)X$; TAB(1Ø)Y$						A	B	C			D	E	F																											
? ,1,2;3"ABC"															1												2				3		A	B	C				

numerische Variable).

Eine READ-Anweisung kann auf mehrere DATA-Anweisungen zugreifen. Ist die Anzahl der Variablen größer als die Anzahl der Datenelemente in den DATA-Anweisungen, erscheint die Fehlermeldung OD. Sind weniger Variable als Datenelemente vorhanden, lesen folgende READ-Anweisungen die noch nicht zugewiesenen Datenelemente. Folgt keine weitere READ-Anweisung, bleiben diese unberücksichtigt.

Beispiel:

```
10 DATA 10,"MAL",10,"IST",100
20 READ A,B$,C,D$,E
30 PRINT A;B$;C;D$;E
RUN
10 MAL 10 IST 100
```

Hier werden numerische und Zeichenketten-Konstanten aus der DATA-Anweisung in Zeile 10 gelesen und anschließend ausgegeben.

13.39. REM

Format: REM [Kommentar]

Kommentarzeile in einem BASIC-Programm. Statt REM kann auch ein Ausrufezeichen (!) stehen. REM-Anweisungen werden nicht ausgeführt, jedoch beim Listen mit der LIST-Anweisung wiedergegeben.

13.40. RENUMBER

Format: RENUMBER [Zeilennummer 1] [, Zeilennummer 2] [, Zeilennummer 3] [, Schrittweite]

Mit dem RENUMBER-Kommando kann ein Programm bzw. ein Programmabschnitt neu numeriert werden.

Dabei bedeuten:

Zeilennummer 1: Zeilennummer, ab welcher neu numeriert werden soll. Standardwert ist die niedrigste Zeilennummer.

Zeilennummer 2: Zeilennummer, bis zu welcher neu numeriert werden soll. Standardwert ist die aktuell höchste Zeilennummer des Programms.

Zeilennummer 3: Neue niedrigste Zeilennummer. Standardwert ist gleich der niedrigsten Zeilennummer vor der Ausführung von RENUMBER.

Schrittweite: Abstand der neunumerierten Zeilen. Standardwert ist 10.

Beachte: Wenn Parameter in diesem Kommando weggelassen werden, kann das hier nur von rechts nach links erfolgen.

Beispiel 1: RENUMBER

Numeriert die Zeilen eines Programms im Abstand von 10 neu.

Beispiel 2: Vor der Ausführung der RENUMBER-Anweisung hat das Programm folgendes Aussehen:

```
 5 PRINT "EINGABE STEUERWERT: 1=SIN,2=COS,3=ATN"
10 INPUT S
15 IF S=0 THEN 80
20 PRINT "EINGABE ARGUMENT IN GRAD:";
21 INPUT X
23 PRINT "ARGUMENT","FUNKTION","FUNKTIONSWERT"
24 PRINT "  GRAD  "
25 PRINT X,
30 X=X*PI/180
40 ON S GOTO 50,60,61
50 Y=SIN(X) : PRINT "SIN", : GOTO 73
60 Y=COS(X) : PRINT "COS", : GOTO 73
61 Y=ATN(X) : PRINT "ATN",
73 PRINT Y : GOTO 5
80 END
```

Das Programm wird durch die Anweisung

```
RENUMBER 21, 80, 25, 5
```

wie folgt numeriert:

```
 5 PRINT "EINGABE STEUERWERT: 1=SIN,2=COS,3=ATN"
10 INPUT S
15 IF S=0 THEN 75
20 PRINT "EINGABE ARGUMENT IN GRAD:";
25 INPUT X
30 PRINT "ARGUMENT","FUNKTION","FUNKTIONSWERT"
35 PRINT "  GRAD  "
40 PRINT X,
45 X=X*PI/180
50 ON S GOTO 55,60,65
55 Y=SIN(X) : PRINT "SIN", : GOTO 70
60 Y=COS(X) : PRINT "COS", : GOTO 70
65 Y=ATN(X) : PRINT "ATN",
70 PRINT Y : GOTO 5
75 END
```

Man beachte dabei, daß alle in Sprunganweisungen (GOTO, ON...GOTO usw.) enthaltenen Zeilennummern auch mit numeriert werden.

13.41. RESTORE

Format: RESTORE [Zeilennummer]

Nach der RESTORE-Anweisung können die mit den DATA-Anweisungen vereinbarten Werte ab der ersten oder der angegebenen Zeilennummer erneut gelesen werden. Die nächste READ-Anweisung greift dann auf das erste Datenelement der ersten bzw. der DATA-Anweisung mit der angegebenen Zeilennummer zu.

Beispiel:

```
10 READ E1,E2,E3
20 PRINT E1,E2,E3
30 RESTORE
40 READ E4,E5,E6
50 PRINT E4,E5,E6
60 DATA 100,200,300
RUN
100   200   300
100   200   300
```

13.42. RUN

Format: RUN [Zeilennummer]

Das RUN-Kommando startet die Ausführung eines Programms bei der niedrigsten bzw. der angegebenen Zeilennummer.

Die Ausführung eines Programms wird in den folgenden Fällen beendet bzw. abgebrochen:

- Es sind keine ausführbaren Zeilen mehr vorhanden.
- Der BASIC-Interpreter hat eine STOP- oder END-Anweisung ausgeführt.
- Es wurde die STOP-Taste des Rechners gedrückt.
- Es wurde die Taste PAUSE gedrückt.
- Während der Ausführung des Programms ist ein Fehler aufgetreten (Fehlermeldungen siehe A1).

Nach Betätigung der Taste PAUSE befindet sich der BASIC-Interpreter im Wartezustand (Programmmodus), in den anderen angeführten Fällen im Kommandomodus. Durch Betätigen der RUN-Taste kann ein Programm ebenfalls gestartet werden, jedoch ausschließlich zur Abarbeitung ab der niedrigsten BASIC-Zeilennummer.

13.43. STOP

Format: STOP

Mit dieser Anweisung erfolgt ein programmierter Programmabbruch, und der BASIC-Interpreter wird in den Kommandomodus gesetzt. STOP-Anweisungen dürfen an beliebiger Stelle im Programm stehen. Nach Ausführung der STOP-Anweisung meldet sich der Interpreter mit

BREAK IN n (n = Zeilennummer)

Durch Eingabe des Kommandos CONT bzw. durch Betätigung der Taste [CONT] kann das abgebrochene Programm fortgesetzt werden (siehe Abschnitt 13.9).

Beispiel:

```
100 INPUT X,Y,Z
150 SU = X + Y
200 STOP
250 SU = SU + Z:PRINT SU
RUN
? 10,20,30
BREAK IN 200
CONT
 60
```

Ein Programmabbruch durch den Bediener kann durch Betätigen der [STOP]-Taste erfolgen.

13.44. TROFF

Format: TROFF

Mit diesem Kommando wird das Kommando TRON (siehe 13.45) außer Kraft gesetzt. Die Programmausführung erfolgt wieder normal.

13.45. TRON

Format: TRON

Bewirkt, daß bei der Abarbeitung eines Programms die Nummern der Zeilen in der Reihenfolge ihrer Abarbeitung auf dem Bildschirm ausgeschrieben werden. Die Zeilennummern sind mit Dreiecksklammern (<>) eingeschlossen. Der logische Ablauf eines Programms kann auf diese Weise verfolgt werden.

13.46. WAIT

Format: WAIT Port, Ausdruck [,Ausdruck]

Diese Anweisung hält die Programmausführung an, bis eine angegebene Portadresse des HC ein spezifiziertes Bitmuster angenommen hat.

Dabei bedeuten:

Port:	Bitadresse (Kanaladresse) des HC.
Ausdruck:	Beide Ausdrücke sind ganzzahlige Ausdrücke.

Der Zustand des I/O-Ports (Kanaladresse) wird exklusiv-ODER mit dem 2. Ausdruck in der Anweisung verknüpft. Das Ergebnis wird durch ein logisches UND mit dem 1. Ausdruck verknüpft. Erst wenn das Ergebnis ungleich Null ist, wird das Programm fortgesetzt. Fehlt der 2. Ausdruck, so wird sein Wert mit Null angenommen.

13.47. WIDTH

Format: WIDTH Ausdruck

Durch dieses Kommando wird die Länge einer Ausgabezeile entsprechend dem ganzzahligen Wert des Ausdruckes festgelegt. Standard ist eine Länge von 40 Zeichen. Im einfachsten Fall ist der Ausdruck eine numerische Konstante. Die Anzahl der gemäß letztem Null-Kommando (bzw. Null-Standard) anschließend auszugebenden

bedeutungslosen Zusatzzeichen wird durch WIDTH nicht verändert.

13.48. WINDOW

Format: WINDOW[Zeile 1, Zeile 2, Spalte 1, Spalte 2]

Diese Anweisung gestattet einen Bildausschnitt auf dem Bildschirm zu definieren, innerhalb dessen sämtliche Ausgaben durch PRINT (nicht: PRINT AT) und Eingabeanforderungen erscheinen. Alle außerhalb des definierten Fensters liegenden Bildschirminhalte sind durch PRINT AT oder direkte Änderung im Bildwiederholspeicher (und ggf. Attributspeicher) mittels POKE bzw. DOKE veränderbar.

Alle 4 Parameter sind ganzzahlige Ausdrücke.

Zeile 1: erste Zeile des Fensters
Zeile 2: letzte Zeile des Fensters
Spalte 1: erste Spalte des Fensters
Spalte 2: letzte Spalte des Fensters

Die Parameter Zeile 1 und Zeile 2 können Werte zwischen 0 und 23 annehmen. Die Parameter Spalte 1 und Spalte 2 können zwischen 0 und 39 annehmen.

Standardmäßig, d. h. nach Neustart des BASIC-Interpreters, gilt der gesamte Bildschirm (24 Zeilen und 40 Spalten) als aktuelles Ausgabefenster. Das ist auch der Fall, wenn nur WINDOW eingegeben wird.

Beispiel 1: 100 WINDOW 1,20,0,39

Mit diesen Parametern werden die Zeilen 1 bis 20 als Fenster festgelegt.

Beispiel 2: 200 WINDOW

Mit dieser Anweisung wird der gesamte Bildschirm zum Fenster (Herstellung des Ausgangszustandes) erklärt.

Beispiel 3:

```
10   INK 1:PAPER 4:CLS
20   A$="BELIEBIGE ZEICHEN"
30   GOSUB 90
40   WINDOW 0,23, 18, 39
50   INK 1:PAPER 3:CLS
60   A$="NEUER TEXT"
70   GOSUB 90
80   END
90   FOR I=1 TO 20
100  PRINT A$
110  NEXT I
120  RETURN
```

Auf schwarzem Hintergrund mit Zeichenfarbe Gelb wird, beginnend auf der 1. Zeile, der Text

BELIEBIGE ZEICHEN

20mal ausgeschrieben. In Zeile 40 wird anschließend ein neues Fenster definiert (neue Bildbreite). Die folgenden Ausgaben (Schwarz auf Grün)

NEUER TEXT

zerstören dadurch nicht die vorhergehenden Ausgaben.

14. BASIC-Funktionen

In den folgenden Abschnitten sollen die BASIC-Funktionen detailliert beschrieben werden. Standardfunktionen des BASIC-Interpreters brauchen nicht definiert zu werden und werden durch ihren Namen im Programm aufgerufen. Die Argumente der Funktionen werden immer in runde Klammern eingeschlossen.

Zur Vereinfachung der Schreibweise werden folgende Vereinbarungen getroffen:

X und Y beliebige numerische Ausdrücke
I und J beliebige ganzzahlige Ausdrücke

X$ und Y$ beliebige Zeichenketten-Ausdrücke

Wird anstelle eines verlangten ganzzahligen Ausdruckes ein Gleitkommaausdruck angegeben, so wird dessen Wert vorher durch Abschneiden der Dezimalstellen in einen ganzzahligen Wert (Integer-Wert) umgewandelt. Die im Abschnitt 11 vereinbarte Beschreibungsform gilt auch hier.

14.1. ABS

Format: ABS(X)

Diese Funktion liefert den absoluten Betrag des Ausdrucks X.

Beispiel:
```
PRINT ABS(6*(-3))
18
```

14.2. ASC

Format: ASC(X$)

Mit dieser Funktion wird der ASCII-Code des ersten Zeichens des Zeichenketten-Ausdrucks X$ ermittelt.

Beispiel:
```
?ASC("ABC")
 65
A$="ABC"
?ASC(A$)
 65
```

In diesem Beispiel wurde anstelle von PRINT das Fragezeichen verwendet. In beiden Fällen ist das erste Zeichen des Zeichenketten-Ausdrucks das Zeichen A. Im Heimcomputer ist das Zeichen A in einem Byte mit der Codierung 100 0001 abgelegt, was einer Dezimalzahl 65 in binärer Codierung entspricht.

14.3. ATN

Format: ATN(X)

Es wird der Arkustangens des Ausdrucks X ermittelt. Das Resultat erscheint im Bogenmaß.

14.4. CHR$

Format: CHR$(I)

Mit Hilfe dieser Funktion kann das ASCII-Zeichen ermittelt werden, das die Codierung I besitzt. Der Wert des Ausdrucks I muß im Bereich zwischen 0 und 255 liegen.

Beispiel 1:
```
PRINT CHR$(65)
A
```

Beispiel 2:
```
I=65
PRINT CHR$(I)
A
```

In beiden Fällen wird auf der jeweils aktuellen Kursorposition des Bildschirms das Zeichen A ausgegeben.

14.5. COS

Format: COS(X)

Berechnung des Cosinus von X, wobei X im Bogenmaß angegeben werden muß.

Beispiel:
```
10 X=10*COS(.4)
20 PRINT X
RUN
9.21061
```

14.6. DEEK

Format: DEEK(I)

Es wird der Wert von 2 aufeinanderfolgenden Bytes gelesen. Der niederwertige Teil eines Speicherwortes (2 Bytes) wird aus der

Speicherzelle mit der Adresse I entnommen, während der höherwertige Teil aus der Speicherzelle I + 1 gelesen wird.

Beispiel:

```
100 A=30000
110 B=DEEK(A)
120 C=DEEK(21511)
```

Nach Ausführung des Programms hat die Variable B den Wert der beiden Speicherzellen 30000 und 30001, während C den Wert der beiden Zellen 21511 und 21512 hat.

14.7. EXP

Format: EXP(X)

Expotentialfunktion von X. Der Wert des Ausdrucks X muß ≦ 87.3365 sein.

14.8. FRE

Format: FRE(X)

Gibt die Anzahl der noch freien Speicherplätze im Arbeitsspeicher (RAM) an. Für das Argument X kann ein beliebiger Ausdruck eingesetzt werden.

Ist X eine Zeichenkettenvariable, so wird die Anzahl der noch freien Bytes des Speicherbereichs für Zeichenkettenvariable eingegeben (siehe CLEAR-Anweisung).

Beispiel 1: FRE(0)

Durch die Eingabe dieser Anweisung wird auf dem Bildschirm der noch freie Speicherplatz, der zur Ausführungszeit vorhanden ist, ausgeschrieben.

Beispiel 2:

```
100 A=FRE(ST$)
110 PRINT A
120 INPUT ST$
```

```
130 PRINT FRE(ST$)
RUN
 256
? ABC
 253
```

Geht man von der Standardlänge des Speicherbereichs für Zeichenkettenvariable von 256 Bytes aus, wird durch die Zeile 110 der Wert 256 auf dem Bildschirm ausgeschrieben. Nach erfolgter Eingabe, hier z. B. ABC, wird der Wert 253 durch die Anweisung in der Zeile 130 angezeigt.

14.9. INKEY$

Format: INKEY$

Die Funktion dient der Ermittlung einer Tastenbetätigung, die während der Programmabarbeitung und ohne deren Unterbrechung erfolgte.

Der gelieferte Funktionswert entspricht dem der zuletzt betätigten Taste äquivalenten Zeichen, d. h., er stellt eine Zeichenkette der Länge 1 dar, die auch nicht darstellbare Zeichen (z. B. für Kursortaste) enthalten kann. Wurde nach erfolgter Verarbeitung einer regulären Eingabe im Wartezustand des BASIC-Interpreters keine Taste oder eine der Tasten [STOP], [PAUSE] betätigt, liefert INKEY$ eine leere Zeichenkette (Länge 0).

Beispiel:

```
300 PRINT "*";
310 IF INKEY$ = "E" THEN 330
320 GOTO 300
330 PRINT "ENDE"
RUN
****....ENDE
```

Nach Betätigung der Taste [E]

14.10. INP

Format: INP(I)

Es wird ein Byte vom I/O-Port I (Kanaladresse) gelesen. Diese Anweisung entspricht dem Assemblerbefehl IN.

14.11. INSTR

Format: INSTR(X$,Y$)

Diese Anweisung liefert die Position des ersten Auftretens der Zeichenkette X$ in der Zeichenkette Y$. Ist der Wert von X$ in Y$ nicht enthalten, liefert die Funktion den Wert Null.

Beispiel:

```
10 X$="TEXT"
20 Y$="WO IST TEXT VERSTECKT ?"
30 PRINT INSTR(X$,Y$)
RUN
 8
```

Ab der 8. Position der Zeichenkette Y$ tritt die Zeichenkette X$ auf.

14.12. INT

Format: INT(X)

Durch diese Funktion wird der ganze Teil eines Ausdruckes X ermittelt.

Beispiel:

```
PRINT INT(34.78)
 34
PRINT INT(-34.78)
-35
```

Aus den beiden Beispielen ist ersichtlich, daß durch die INT-Funktion immer abgerundet wird.

14.13. JOYST

Format: JOYST(I)

Diese Funktion liefert den Wert eines Spielhebels entsprechend dessen aktueller Stellung. Der Wert des Ausdrucks I muß 1 oder 2 sein, da 2 Spielhebel zur Verfügung stehen.

14.14. LEFT$

Format: LEFT$(X$,I)

Liefert die ersten I Zeichen der Zeichenkettenvariablen X$, von links beginnend. Der Wert von I muß zwischen 0 und 255 liegen. Ist I größer als die Zeichenkettenlänge von X$, wird die gesamte Zeichenkette geliefert.

Im anderen Sonderfall, wenn I Null ist, liefert die Funktion als Ergebnis eine leere Zeichenkette (Länge 0).

Beispiel:

```
10 A$="ABCDEF"
20 PRINT LEFT$(A$,3)
RUN
ABC
```

14.15. LEN

Format: LEN(X$)

Diese Funktion liefert die Anzahl der Zeichen, die der Zeichenkettenvariablen X$ aktuell zugeordnet sind. Es werden alle Zeichen (z. B. auch nichtdruckbare und Leerzeichen) gezählt.

Beispiel:

```
100 X$="robotron COMPUTER"
200 PRINT LEN(X$)
RUN
17
```

14.16. LN

Format: LN(X)

Bildung des natürlichen Logarithmus von X, wobei X >0 sein muß.

Beispiel:
```
PRINT LN(43/6)
1.96944
```

14.17. MID$

Format: MID$(X$,I[,J])

Diese Funktion liefert die ersten J Zeichen der Zeichenkettenvariablen X$ ab der Position I. Ohne Angabe des Argumentes J werden alle Zeichen ab der Position I übernommen. Die Werte von I und J müssen zwischen 0 und 255 liegen.

Sonderfälle: Wenn I > Länge von X$ ist, wird eine Zeichenkette der Länge 0 geliefert.
Wenn J > Länge der Restzeichenkette ab I-ter Position ist, so werden sämtliche Zeichen ab I-ter Position übergeben.

Beispiel:
```
A$ = "ABCDEF"
PRINT MID$(A$,3,2)
CD
```

14.18. PEEK

Format: PEEK(I)

Diese Funktion liefert den Inhalt der Speicherzelle mit der Adresse I. Der Wert des ganzzahligen Ausdruckes I kann zwischen -32768 und +32767 liegen.

Achtung! Auf diese Weise lassen sich die Adressen des Bildwiederholspeichers nicht ohne weiteres einfach angeben. Es ist vielmehr diejenige negative Zahl einzugeben, deren Betrag im Komplement zu

65536 die gewünschte Adresse ergibt.

Die Bildschirmadressierung ist in Tabellenform in den Programmierhilfen enthalten.

Beispiel:

Die gewünschte Adresse im Bildwiederholspeicher sei 60720. Damit ist 65536 - 60720 = 4816 der Betrag der zu verwendenden negativen Zahl.

PEEK(-4816) oder PEEK(60720-65536)

liefert das auf dem Bildschirmplatz 60720 stehende Zeichen in ASCII-Codierung.

14.19. POS

Format: POS(X)

Diese Funktion liefert die aktuelle Spaltenposition des Kursors auf dem Bildschirm. Die äußerste linke Spalte hat die Position 0. Das Argument X ist ein beliebiges Argument, das nicht ausgewertet wird (Blindargument).

Beispiel:

```
10 PRINT"0123456";
20 PRINT POS(I)
RUN
0123456 7
```

Durch Zeile 10 werden 7 Zeichen ausgegeben. Durch das Semikolon bleibt der Kursor auf der nächsten freien Position stehen. Diese wird in Zeile 20 ermittelt und ausgedruckt.

14.20. RIGHT$

Format: RIGHT$(X$,I)

Liefert die ersten I Zeichen der Zeichenkettenvariablen, von

rechts beginnend. Der Wert von I muß zwischen 0 und 255 liegen. Wenn I = der Länge von X$ ist, wird die gesamte Zeichenkette von X$ geliefert.

Beispiel:

```
A$="ABCDEF"
PRINT RIGHT$(A$,3)
DEF
```

14.21. RND

Format: RND(X)

Diese Funktion liefert eine Zufallszahl zwischen 0 und 1 in Abhängigkeit vom Argument X. Drei verschiedene Arten der Erzeugung werden unterschieden:

X > 0	Es wird immer der nächste Wert einer Zufallszahlenreihe ausgegeben. Die Reihe ist vom Wert des Argumentes X abhängig.
X = 0	Der Wert der letzten Zufallszahl wird nochmals ausgegeben.
X < 0	Es wird eine neue Folge von Zufallszahlen begonnen.

Beispiel:

```
10 FOR I=1 TO 6
20 PRINT INT(RND(I)*100);
RUN
 27  85  64  84  73  57
```

14.22. SGN

Format: SGN(X)

Vorzeichenfunktion (Signumsfunktion) von X. Die Funktion liefert folgende Werte:

1	für	X > 0
0	für	X = 0
-1	für	X < 0

Beispiel:

```
10 X=0 : Y=-3.1 : Z=.5
20 PRINT SGN(X),SGN(Y),SGN(Z)
RUN
0            -1            1
```

14.23. SIN

Format: SIN(X)

Ermittlung von SIN(X), wobei X im Bogenmaß angegeben wird.

Beispiel:

```
PRINT SIN(1.4)
.98545
```

14.24. SPC

Format: SPC(I)

Diese Funktion ist nur in PRINT-Anweisungen anwendbar und liefert I Leerzeichen.

Beispiel:

```
PRINT"SPC-"SPC(10)"FUNKTION"
SPC-          FUNKTION
```

14.25. SQR

Format: SQR(X)

Quadratwurzel von X, mit X ≧ 0.

Beispiel:

```
20 FOR X=9 TO 25 STEP 4
30 PRINT X,SQR(X)
40 NEXT X
RUN
 9          3
 13         3.60555
 17         4.12311
 21         4.58258
 25         5
```

14.26. STRING$

Format: STRING$(Wiederholungszahl, X$)

Die Funktion liefert eine Zeichenkette, die durch Wiederholung des Wertes des Zeichenkettenausdrucks X$ entsteht. Wie oft X$ wiederholt werden soll, wird durch die angegebene Wiederholungszahl bestimmt. Die Wiederholungszahl kann Werte zwischen 1 und 255 annehmen.

Beispiel 1:

```
10 A$="*"
20 PRINT STRING$(10,A$)
RUN
**********
```

Beispiel 2:

```
10 A$="Ueberschrift"
20 PRINT A$
30 PRINT STRING$(LEN(A$),"=")
RUN
Ueberschrift
============
```

14.27. STR$

Format: STR$ (X)

Formt den numerischen Wert X in eine Zeichenkette um.

Beispiel:

```
100 INPUT"ZAHLENEINGABE:";I
110 PRINT I;LEN(STR$(I))
120 GOTO 100
RUN
ZAHLENEINGABE: -321
-321  4
ZAHLENEINGABE:  3
 3  2
ZAHLENEINGABE:  0001
 1  2
```

Im 2. Fall ist die Länge von STR$(3) deshalb 2, weil in der Zeichendarstellung von positiven Zahlen der Zahl immer eine Leerstelle vorangestellt wird.

Im 3. Fall beträgt der eingegebene numerische Wert 1. Die führenden Nullen sind bedeutungslos.

14.28. TAB

Format: TAB(I)

Tabulatorfunktion. Der Wert des numerischen Ausdrucks I stellt die Zeichenposition im Datensatz dar, ab der das nächste PRINT-Element einzutragen ist (siehe PRINT-Anweisung). Die Zählung der Zeichenposition beginnt am linken Rand (Position 0).

Steht der Kursor vor Ausführung einer TAB-Funktion bereits auf oder rechts der I-ten Spalte, wird diese TAB-Funktion nicht ausgeführt. Es ist u. a. nicht sinnvoll, innerhalb einer PRINT-Anweisung gleichzeitig Kommata und TAB-Funktionen zu verwenden.

Beispiel 1:

```
10 PRINT"GLEICHE"TAB(15)"ANFANGSPOSITION"
20 PRINT"A"TAB(15)"B"
30 PRINT"AB"TAB(15)"CD"
40 PRINT"ABC"TAB(15)"DEF"
RUN
GLEICHE         ANFANGSPOSITION
A               B
AB              CD
ABC             DEF
```

Beispiel 2:

```
10 INPUT P
20 IF P=0 THEN 70
30 PRINT "*"; TAB(P);"+"; TAB(20);"*"
40 PRINT "00000000001111111111222222222223"
50 PRINT "01234567890123456789012345678 90"
```

```
60 GOTO 10
70 END
RUN
? 7
*       +             *
0000000000111111111122222222223
0123456789012345678901234567890
? 24
*                        +*
0000000000111111111122222222223
0123456789012345678901234567890
? 0
```

Im Falle der Eingabe von 24 kann TAB(20) nicht ausgeführt werden, da die aktuelle Zeichenposition bereits 24 beträgt. Der zweite Stern wird deshalb auf der Zeichenposition 25 ausgegeben.

14.29. TAN

Format: TAN(X)

Tangens von X, wobei X im Bogenmaß anzugeben ist.

Beispiel:

```
PRINT TAN(4)/3
 .38594
```

14.30. USR

Format: USR(X)

Ruft ein Maschinencodeprogramm mit dem Argument X auf. Die Startadresse des Maschinencodeprogramms muß vorher in einem bestimmten Bereich des BASIC-Interpreters (WSP+4 und WSP+5) eingetragen werden. Im Unterschied zur Anweisung CALL kann ein Parameter X ins Maschinencodeprogramm übernommen bzw. von dort ins BASIC-Programm zurückgebracht werden.

14.31. VAL

Format: VAL(X$)

Liefert den numerischen Wert der Zeichenkettenvariablen X$. Wenn das erste Zeichen nicht eine Ziffer, ein Punkt, ein Plus- oder Minuszeichen ist, wird VAL (X$) = 0.

Beispiele:

```
PRINT VAL("-23ABC")
-23
PRINT VAL("456 333")
 456333
PRINT VAL("456-333")
 456
PRINT VAL(".0048")
 4.8E-03
```

Anhang A1:

Fehlermeldungen des BASIC-Interpreters

Nachdem ein Fehler aufgetreten ist, kehrt der BASIC-Interpreter in den Kommandomodus zurück. Ein fehlerhaftes Programm kann nicht mit CONT, sondern falls sinnvoll, nur mit GOTO Zeilennummer fortgesetzt werden. Der Zusammenhang aller GOSUB- und FOR/NEXT-Anweisungen wird erst bei einem erneuten Programmstart (RUN) wieder hergestellt. Der Fehler wird mit zwei Zeichen klassifiziert und erscheint in einer Zeile im folgenden Format:

?XX ERROR [IN n] (n = Zeilennummer)

Für XX stehen die beiden Zeichen, die den Fehler charakterisieren. Tritt der Fehler im Programmodus auf, erscheint zusätzlich die Zeilennummer, bei deren Abarbeitung der Fehler auftrat.

BS (Subscript out of range)
Es wurde eine indizierte Variable verwendet, deren Index größer ist als der maximale in der DIM-Anweisung angegebene. Falls nicht mit DIM dimensioniert worden ist, ist der Feldindex größer als 10.

CN (Can't continue)
Das Programm kann nicht mit CONT fortgesetzt werden (nach einer Fehlermeldung).

DD (Doubly defined array)
Es wurde versucht, ein Feld mit gleichem Namen ein zweites Mal zu dimensionieren. Der Fehler tritt auch auf, wenn das Feld nach seiner ersten Verwendung (automatische Dimensionierung) mit einer DIM-Anweisung dimensioniert werden soll.

FC (Illegal function call)
Es erfolgte ein unzulässiger Funktionsaufruf.

FD (Illegal direct)
Die Anweisungen INPUT und DEF FN dürfen nicht im Kommando-

modus verwendet werden.

IO (Input/Output)
Übertragungsfehler bei einer Eingabe vom Kassettengerät.

LS (String too long)
Durch eine Addition von Zeichenketten ist eine Zeichenkette entstanden, die länger als 255 Zeichen geworden ist.

MO (Missing operand)
Eine spezifizierte Anweisung ist unvollständig. Es fehlt ein Operand.

NF (NEXT without FOR)
Es wurde eine fehlerhafte FOR...NEXT-Anweisung gegeben (falsche Schachtelung oder zu NEXT fehlt das zugehörige FOR).

OD (Out of data)
Es wurden durch DATA-Anweisungen zuwenig Daten für READ-Anweisungen spezifiziert.

OM (Out of memory)
Der vorhandene Speicherplatz im Arbeitsspeicher (RAM) reicht für die Ablage bzw. Abarbeitung eines Programms nicht aus.

OS (Out of string space)
Der vereinbarte Speicherplatz für Zeichenkettenvariable reicht nicht aus.

OV (Numerical overflow)
Der Betrag des Ergebnisses einer Berechnung ist größer als 1.70141E38.

RG (RETURN without GOSUB)
Eine RETURN-Anweisung trat vor Ausführung einer GOSUB-Anweisung auf.

SN (Syntax error)
Syntaxfehler. Der BASIC-Interpreter fand eine Zeichenkombination, die er nicht versteht.

ST (Literal string pool table full)
Ein Zeichenkettenausdruck ist zu lang oder zu komplex.

TM (Type mismatch)
Es wurde versucht, einem Variablentyp einen falschen Datentyp zuzuweisen, bzw. eine Funktion erhielt nicht das richtige Argument (z. B. anstatt einer Zahl eine Zeichenkette).

UF (Undefined user function)
Es wurde eine Funktion aufgerufen, deren Definition durch DEF FN im Programm nicht erfolgte.

UL (Undefined line)
Es wurde eine nicht existierende Zeilennummer angegeben.

/0 (Division by zero)
Bei der Berechnung eines mathematischen Ausdrucks trat eine Division durch Null auf.

?REDO FROM START

Diese Meldung tritt bei einer INPUT-Anweisung auf, wenn eine fehlerhafte Eingabe erfolgte. Die Eingabe kann nach dieser Meldung wiederholt werden.

Anhang A2:

Speicherplatzbedarf von BASIC-Programmen

In der Eingabephase werden Quellzeilen, für den Benutzer nicht bemerkbar, vorübersetzt. Alle reservierten Worte werden z. B. durch interne 1 byte lange Abkürzungen ersetzt. Diese Maßnahme ist allgemein bei Interpretern für die Sprache BASIC üblich, um die Laufzeit von Programmen und den Speicherplatzbedarf zu verringern. Mit der Art und Weise, wie man Programme formuliert (eingibt), kann aber auch der Speicherplatzbedarf eines Programmes verringert werden.

- Leerzeichen in Programmzeilen werden in ihrem ASCII-Code (20H) mit gespeichert. Zur besseren Lesbarkeit von Programmen kann man sie eingeben, sie erhöhen aber den Speicherplatzbedarf.

- Da jede Programmzeile intern mit einer Koppeladresse und Zeilennummer, jeweils 2 byte lang, beginnt und mit einem Endekennzeichen abschließt, kann durch Eingabe von mehreren BASIC-Anweisungen in einer Zeile zu Lasten der Lesbarkeit Speicherplatz gespart werden.

- Lange Variablennamen verbessern zwar die Lesbarkeit von Programmen, erhöhen aber ebenfalls den Speicherplatzbedarf. Für Variablennamen sind nur 2 Zeichen eignifikant.

- Die Funktion SPC in PRINT-Anweisungen sollte aus Speicherplatzgründen erst benutzt werden, wenn mehr als 3 Leerzeichen erzeugt werden sollen.

Anhang A3:

Reservierte Worte

Der BASIC-Interpreter des Heimcomputers erkennt einige Worte im Programm als Aufruf für spezielle Operationen. Die Namen, die für solche Aufrufe verwendet werden, bezeichnet man als reservierte Worte. Sie dürfen nicht als Namen von Variablen benutzt werden; auch nicht als Teil an irgendeiner Stelle des Namens.

Tabelle: Reservierte Worte

ABS	ELSE	NEXT	STR$
AND	END	NOTD#	STRING$
ASC	EXP	NULL	TAB(
AT	FN	ON	TAN
ATN	FOR	OR	THEN
AUTO	FRE	OUT	TO
BEEP	GO	PAPER	TROFF
BORDER	GOSUB	PAUSE	TRON
BYE	GOTO	PEEK	USR
CALL	IF	PI	VAL
CALL*	INK	POKE	WAIT
CHR$	INKEY$	POS	WIDTH
CLEAR	INP	PRINT	WINDOW
CLOAD	INPUT	READ	
CLOAD*	INSTR	REM	
CLS	INT	RENUMBER	
CONT	JOYST	RESTORE	
COS	LEFT$	RETURN	
CSAVE	LEN	RIGHT$	
CSAVE*	LET	RND	
DATA	LINES	RUN	
DEF	LIST#	SGN	
DEEK	LIST	SIN	
DELETE	LN	SPC(	
DIM	LOAD#	SQR	
DOKE	MID$	STEP	
EDIT	NEW	STOP	

Anhang A4:

Spezielle Programmierhinweise

Der Heimcomputer robotron Z9001 erlaubt die Zeichendarstellung auf dem Bildschirm des angeschlossenen Fernsehgerätes wahlweise im Format 20 oder 24 Zeilen/Bild (vergleiche Bedienungsanleitung). Standardmäßig sind 24 Zeilen/Bild wirksam.

Die Kanaladresse 136 enthält die Information über das aktuelle Bildformat (20 oder 24 Zeilen) und die Farbe des Bildschirmrandes. Damit können durch die BASIC-Anweisung

```
OUT 136,I
```

sowohl Zeilenformat als auch die Bildschirmrandfarbe geändert werden. Für den Wert von I muß gelten:

20-Zeilen-Darstellung	I=4+8*(N-1)
24-Zeilen-Darstellung	I=0+8*(N-1)

wobei N dem Farbcode des Bildschirmrandes entspricht (siehe BORDER-Anweisung).

Da auch bei 20-Zeilen-Darstellung vom BASIC auf den vollen Bildspeicher zugegriffen wird, ist in diesem Fall die Begrenzung des Ausgabebereiches mittels WINDOW sinnvoll.

Beispiel:

```
50 CLS:WINDOW 0,19,0,39
60 OUT 136,12
```

In Anweisung 60 wird die 20-Zeilen-Darstellung bei rotem Bildschirmrand eingestellt.

Es ist zu beachten, daß nach einer BORDER-Anweisung stets die standardmäßige 24-Zeilen-Darstellung wirksam wird.