KLEINCOMPUTER

KC compact

BASIC-Handbuch

KLEINCOMPUTER

KC compact

BASIC-Handbuch

veb mikroelektronik >wilhelm pieck<
mühlhausen
im veb kombinat mikroelektronik

Redaktionsschluß dieser Ausgabe: September 1989

digitalisiert von Ulrich Zander 06/2009 <zander@felix.sax.de>

G l i e d e r u n g

BASIC-Handbuch **Seite**

0. E i n l e i t u n g

BASIC (Beginners All-purpose Symbolic Instruction Code - Universelle Programmiersprache für Anfänger, wobei Anfänger für leichtes Erlernen steht) ist eine höhere Programmiersprache, die von den meisten Computern verarbeitet wird. Sie bietet einem Anfänger sehr gute Voraussetzungen zum schnellen Erlernen der Sprache, da die Sprachelemente (BASIC-Schlüsselwörter) weitgehend an die englische Sprache angelehnt sind. Das Schlüsselwort veranlaßt den Computer, eine, dem Schlüsselwort zugeordnete, Funktion auszuführen. Wie in jeder Sprache unterliegt auch BASIC den Regeln der Semantik (Wortbedeutungslehre) und der Syntax (Satzlehre). Werden diese Regeln nicht eingehalten, meldet sich der Computer mit der Ausschrift "Syntax error".
Der BASIC-Interpreter ist ein Programm, das die höhere Programmiersprache (BASIC) in Maschinensprache wandelt. Die eingegebenen BASIC-Schlüsselwörter werden vom Interpreter in die für den Computer verständliche Form aufbereitet.
Der BASIC-Interpreter ist komfortabel und bietet dem Anwender optimale Anwendungsmöglichkeiten, so daß z.B. grafische und musikalische Effekte erzielt werden können.

Nach dem Einschalten des Computers meldet sich der BASIC-Interpreter. BASIC arbeitet in zwei Betriebsarten, im

- Direkt-Mode oder im
- Programm-Mode.

Im Direkt-Mode werden Schlüsselwörter (Anweisungen) eingegeben und mit der Taste [RETURN] abgeschlossen. Diese Anweisungen werden vom Computer sofort ausgeführt.
Im Programm-Mode werden erst Programmzeilen geschrieben, die nach dem Programmaufruf zur Ausführung kommen.

Allgemeine Hinweise

Nachdem, wie in der Gerätebeschreibung erläutert, alle Verbindungen gesteckt und die Hinweise zur Inbetriebnahme berücksichtigt wurden, erscheint nach dem Einschalten des Computers folgende Ausschrift auf dem Bildschirm:

```
KC compact
Version a.b

BASIC 1.1
Ready
■
```

Dieses Anfangsbild erscheint immer beim Einschalten des Computers bzw. beim Zurücksetzen in den RESET-Zustand.
Statt "a.b" stehen Zahlen, die die Versionsnummer des Betriebssystems angeben.

RESET-Zustand

Der RESET-Zustand des Computers entspricht dem Betriebszustand nach dem Einschalten. Das heißt, der Speicher und alle vorher getroffenen Vereinbarungen, wie z.B. Tastenbelegungen und Zeichendefinitionen, sind gelöscht.
Durch folgende Tastenbetätigungen kann dieser Zustand hergestellt werden:

[CTRL]-[SHIFT] gleichzeitig unten halten und anschließend [ESC] drücken

Die Farbeinstellung im Anfangsbild ist stets leuchtend gelbe Schrift auf blauem Grund.
Nach dem Erscheinen des Anfangsbildes erwartet der Computer Eingaben. Die Eingaben erfolgen auf der Tastatur und die Anzeige auf dem Bildschirm.

Dem Nutzer steht in BASIC eine Speicherkapazität von ca. 42 KByte für Programme und Daten zur Verfügung.

Die Anschlußbedingungen für Peripheriegeräte wie:

- elektronische Schreibmaschine (z.B. S3004) bzw. Drucker
- Kassettengerät
- Diskettengerät
- Joystick

sind in der Gerätebeschreibung angeführt, die zum KC compact mitgeliefert wird.

1 . D i e T a s t a t u r

1.1. Aufbau der Tastatur

Die Tastatur besteht aus drei Tastenfeldern. Links befinden sich die Funktionstasten, in der Mitte die alphanumerischen Tasten und rechts die Cursortasten. Unterhalb der Cursortasten ist die [RETURN]-Taste angeordnet.
Nachfolgend werden die Funktionen der Editier- und Steuertasten beschrieben. Dazu wird folgende Schreibweise benutzt, die für das gesamte Buch beibehalten wird:

[Taste 1]-[Taste 2] Taste 1 und 2 werden gleichzeitig gedrückt
[Taste 1]+[Taste 2] Taste 1 und 2 werden nacheinander gedrückt

[RETURN] und [ENTER]

Die [RETURN]-Taste ist eine wichtige Taste beim Betrieb des BASIC-Interpreters. Mit dieser Taste werden eingegebene Kommandos ausgeführt und Programmzeilen im Computer gespeichert. Die gleiche Wirkung besitzt im RESET-Zustand auch die [ENTER]-Taste links unten im mittleren Tastenfeld.

[DEL]

Zeichen löschen (delete = ausradieren)
Mit dieser Taste wird das Zeichen gelöscht, das sich links vom Cursor befindet. Der Teil der Zeile, der sich rechts vom Cursor befindet, wird um ein Zeichen nach links verschoben. Bleibt die Taste länger gedrückt, erfolgt das Löschen bis zum Zeilenanfang (Autorepeatfunktion = wiederholtes Ausführen der Tastenfunktion bei längerem Tastendruck).

[CLR]

Zeichen löschen (clear = reinigen) Im Unterschied zur [DEL]-Taste wird hier das Zeichen gelöscht, auf dem sich der Cursor gerade befindet und die Zeile wird nach links verdichtet.

[SHIFT]

Umschaltung der Tastaturbelegung
Die Tastatur besitzt zwei [SHIFT]-Tasten (Umschalttasten). Wird eine dieser Tasten gedrückt, wird für die Dauer der Betätigung auf die Zweitbelegung (Großbuchstaben bzw. die auf den Tasten oben abgebildeten Zeichen) umgeschaltet.

[CAPS LOCK]

Umschaltfeststellung für die Buchstabentasten Wirkt wie die [SHIFT]-Taste mit dem Unterschied, daß diese Taste die Funktion bis zum erneuten Drücken feststellt und sich nur auf den Buchstabenbereich bezieht. Alle anderen Tasten bleiben in der Erstbelegung.

In Tabelle 1 sind die Tastenfunktionen aufgeführt, wobei in der Funktionserläuterung vom RESET-Zustand des Computers ausgegangen wird.

Eingabe	Funktion
[CAPS LOCK] einmal getastet	CAPS LOCK-Funktion eingeschaltet: Buchstabentasten auf Großbuchstaben umgeschaltet übrige Tasten davon unbeeinflußt
[CAPS LOCK] nochmal getastet	CAPS LOCK-Funktion ausgeschaltet: Buchstabentasten auf Kleinbuchstaben zurückgeschaltet
[CTRL]-[CAPS LOCK] einmal getastet	SHIFT LOCK- Funktion eingeschaltet: sämtliche Tasten haben jetzt ihre Zweitbelegung. Die Zweitbelegungen der Tasten sind (außer bei den Buchstabentasten) in der oberen Hälfte der Tastenaufdrucke abgebildet.
[CTRL]-[CAPS LOCK] nochmal getastet	SHIFT LOCK-Funktion ausgeschaltet: Tasten werden auf den Zustand, der vorher eingeschaltet war, zurückgestellt.

Tabelle 1: Tastenfunktionen der Taste [CAPS LOCK]

Sowohl bei eingeschalteter CAPS LOCK-Funktion als auch bei eingeschalteter SHIFT LOCK-Funktion hat die [SHIFT]-Taste keine Wirkung auf die umgeschalteten Tasten mehr.

[CTRL]

Control-Taste Mit dieser Taste wird auf die Drittbelegung der Tasten, wenn vorhanden, für die Dauer der Betätigung umgeschaltet. Die Drittbelegung der Tasten wird in vielen kommerziellen Programmen für die Realisierung bestimmter Steuerfunktionen genutzt. Im RESET-Zustand des Computers erzeugen die Drittbelegungen fast aller Buchstabentasten auf dem Bildschirm Grafikzeichen, die im Anhang B angegeben sind.

Die Tastenkombination [CTRL]-[ENTER] erzeugt die Ausgabe von

```
Run"
Press PLAY then any key: ■
```

auf dem Bildschirm, was beim Laden von Programmen hilfreich ist. Nach dem Betätigen von [RETURN] wird jetzt jedes beliebige Programm von Kassette geladen und gestartet. (siehe Abschnitt 2.2.)

[TAB]

Tabulatorsprung Die Taste [TAB] wird von vielen kommerziellen Programmen z.B. Textverarbeitungsprogrammen, dazu benutzt, den Cursor auf festgelegte Tabulatorpositionen springen zu lassen. Im RESET-Zustand des Computers ist diese Funktion allerdings nicht wirksam. Eine Betätigung der Taste [TAB] bringt dasselbe Grafikzeichen auf den Bildschirm wie eine Betätigung der Tasten [CTRL]-[I].

[ESC]

Unterbrechungstaste (escape = entkommen, ausströmen) Mit dieser Taste können bestimmte Funktionen oder Abläufe durch einmaliges Drücken unterbrochen und bei nochmaligem Drücken abgebrochen werden. Wurde nach einmaligem Drücken der [ESC]-Taste eine laufende Funktion unterbrochen, kann diese mit einer anderen Taste (außer [SHIFT], [CAPS LOCK], [CTRL] und [ESC] selbst) fortgesetzt werden.

[SHIFT]-[CTRL]-[ESC]

Das gleichzeitige Drücken dieser drei Tasten führt zum Zurücksetzen des Computers in seinen RESET-Zustand.

Achtung| Dieses Zurücksetzen hat das Löschen aller im Speicher befindlichen Programme und Daten zur Folge und ist deshalb mit Vorsicht bzw. erst dann anzuwenden, wenn wichtiger Speicherinhalt auf Kassette gerettet wurde (siehe Abschnitt 3.9.).

[F0] bis [F4]

Funktionstasten
Die Funktionen dieser Tasten können vom Anwender selbst festgelegt werden. Dazu dienen die Kommandos KEY und KEY DEF (siehe Abschnitt 3.12.). Im RESET-Zustand sind diese Tasten mit den Ziffern 0 bis 4 belegt.

[.]

Diese [Punkt]-Taste, die sich ganz links auf dem Tastenfeld befindet, kann wie die Funktionstasten vom Anwender mit einer neuen Funktion belegt werden. Außer den Tasten [F0] bis [F4] und der [.]-Taste kann weiterhin die [ENTER]-Taste wie eine Funktionstaste umdefiniert werden.

Cursortasten

Die Cursortasten sind die vier mittleren Tasten des rechten Tastenfeldes. Der auf jeder Taste aufgedruckte Pfeil zeigt in die Richtung, in die sich der Cursor (quadratisches Zeichen) auf dem Bildschirm bewegen läßt, wenn man auf die Taste drückt:

[↑] Cursor nach oben,
[←] Cursor nach links,
[↓] Cursor nach unten,
[→] Cursor nach rechts.

Die Bewegungsmöglichkeit des Cursors in alle vier Richtungen besteht direkt nach dem Einschalten und nach jeder [RETURN]-Betätigung. Sobald in eine Zeile ein alphanumerisches Zeichen eingegeben wurde, läßt sich der Cursor nur noch innerhalb der begonnenen BASIC-Zeile (siehe Einleitung zu Kapitel 3) vertikal bewegen.

[COPY]

Die [COPY]-Taste dient in der COPY-Cursor-Funktion dem Kopieren von Bildschirmzeilen. Ihre Anwendung und ihre Funktion sind im Abschnitt 1.2. beschrieben.

1.2. Die COPY-Cursor-Funktion

Die Zweitbelegung jeder Cursortaste ([SHIFT]-[Cursortaste]) bewegt den sogenannten COPY-Cursor in die aufgedruckte Richtung. Der COPY-Cursor sieht genauso aus wie der bisher betrachtete Haupt-Cursor, läßt sich aber im Gegensatz zum Haupt-Cursor immer in jeder Richtung und damit auf jede beliebige Zeichenposition bewegen.

Die COPY-Cursor-Funktion ermöglicht das Kopieren von Bildschirmzeilen ab der COPY-Cursor-Position in die Zeile, in der der Haupt-Cursor steht. Nach dem Positionieren des COPY-Cursors wird die COPY-Cursor-Funktion mit Hilfe der Taste [COPY] ausgeführt. Jedes Betätigen der [COPY]-Taste kopiert das Zeichen, auf dem der COPY-Cursor steht, an die Stelle, auf der der Haupt-Cursor steht. Auf diese Weise können alle auf dem Bildschirm darstellbaren Zeichen kopiert werden.

Die COPY-Cursor-Funktion wird eingeschaltet, sobald ein COPY-Cursor erzeugt wurde. Sie wird ausgeschaltet durch Betätigen der Taste [RETURN].
Bei eingeschalteter COPY-Cursor-Funktion kann man in die Zeile des Haupt-Cursors nicht nur Zeichen aus der Zeile des COPY-Cursors kopieren. Man kann die Zeile des Haupt-Cursors auch weiterhin über die alphanumerischen Tasten beschreiben. So ermöglicht die COPY-Cursor-Funktion z.B. das einfache Kopieren von Programmzeilen, die dann beliebig erweitert und/oder korrigiert werden können.

2 . D e r K a s s e t t e n b e t r i e b

2.1. Technische Details

2.1.1. Die Magnetbandkassette

Zur dauerhaften Speicherung von Programmen und Daten beim KC compact werden Magnetbandkassetten verwendet. Hierfür können alle Magnetbandkassetten eingesetzt werden, die im Handel für Tonaufzeichnung angeboten werden, das sind die Typen: "low noise" und "Chromdioxid HiFi". Für das eigene Abspeichern von Programmen empfiehlt es sich, Magnetbandkassetten möglichst kurzer Spieldauer einzusetzen, z.B. K20, 2x10 min Spielzeit, weil dort die Spulzeit zum Auffinden von Programmen nicht so lang ist.
Wird ein Tonbandgerät als externer Speicher verwendet, so können auch hier alle im Handel erhältlichen Magnetbandsorten für die Aufzeichnung von Programmen und Daten verwendet werden.

2.1.2. Der Kassettenrecorder

Als Aufzeichnungs- und Abspielgerät von Programmkassetten kann jeder handelsübliche Kassettenrecorder verwendet werden, der die elektrischen Bedingungen erfüllt, die in der Gerätebeschreibung (Abschnitt 1.2.) angegeben sind. Ergänzend zu den elektrischen Bedingungen sollte er noch über ein Bandzählwerk und über eine Motorsteuerung verfügen. Das Bandzählwerk erleichtert das Auffinden von Programmen auf der Kassette. Die Motorsteuerung wird von allen Anweisungen angesprochen, die BASIC zur Kassettenarbeit enthält. Der in der Gerätebeschreibung empfohlene Kassettenrecorder, LCR, erfüllt diese Bedingungen. Er ist speziell für den Betrieb am Computer ausgelegt.

Die hier genannten zusätzlichen Bedingungen sind zwar für die Kassettenarbeit sehr vorteilhaft, aber sie sind nicht so zwingend notwendig, wie das Einhalten der elektrischen Bedingungen. Beim Kassettenrecorder ohne Motorsteuerung muß der Anwender die Motorsteuerung über die Bedientasten des Gerätes selbst realisieren.

Für ein Tonbandgerät gelten die gleichen Empfehlungen wie für den Kassettenrecorder.

2.1.3. Verbindung Computer - Kassettenrecorder

Die Verbindung zwischen Computer und Kassettenrecorder erfolgt über ein handelsübliches Diodenkabel entsprechend der Gerätebeschreibung. Soll ein Kassettenrecorder mit Motorsteuerung am KC compact betrieben werden, so muß zur Verbindung ein Stereo-Diodenkabel verwendet werden, um die Schaltspannung übertragen zu können. Bei einem Kassettenrecorder ohne Motorsteuerung reicht auch Mono-Diodenkabel.

2.1.4. Motorsteuerung des Kassettenrecorders

Zur Motorsteuerung des Kassettenrecorders liefert der Computer eine Schaltspannung (Schaltsignal). Dieses Schaltsignal wird eingeschaltet, wenn Kassettenbefehle (siehe Abschnitt 2.4) abgearbeitet werden. Nach deren Abarbeitung bzw. nach deren Abbruch erfolgt das Abschalten des Signals.
Bei gesteuerten Kassettenrecordern ist der Bandtransport nur möglich, wenn diese Schaltspannung eingeschaltet ist. Das gilt für:
- Aufnahme (Abspeichern/Retten von Daten)
- Wiedergabe (Laden von Daten)
- Vor- und Zurückspulen (Suchen von Dateien).

2.2. Laden der Demo-Kassette

Zum Lieferumfang des KC compact gehört die Softwarekassette CC4001 DEMO KC compact. Das erste Programm dieser Kassette besteht aus drei Teilen und wird nach folgenden Schritten geladen:

- Kassette in den Kassettenrecorder einlegen
- Kassette an den Bandanfang zurückspulen
- Tastenbetätigung [CTRL]-[ENTER]
  auf dem Bildschirm erscheint:

```
Run"
Press PLAY then any key: ■
```

(sinngemäße Übersetzung: Abspieltaste am Kassettenrecorder und dann eine Computertaste drücken)

- Entsprechend der Aufforderung die Abspieltaste am Kassettenrecorder drücken und dann eine beliebige Taste am Computer (außer [ESC], [SHIFT], [CTRL] und [CAPS LOCK]).
- Danach beginnt das Abspulen der Kassette, über den Recorderlautsprecher kann man den Ladevorgang akustisch verfolgen.
- Auf dem Bildschirm erscheint die Meldung:

```
Loading DEMO1.BAS block 1
```

(sinngemäße Übersetzung: Laden des 1. Blockes des Programms mit dem Namen DEMO1.BAS)

- Nach dem Laden von Block 1 erscheint ein Titelbild (erster Teil des Programmes), das in der unteren Textzeile erneut die Aufforderung enthält

```
Press PLAY then any key: ■
```

- Da die Abspieltaste des Kassettenrecorders noch gedrückt ist, braucht jetzt nur nochmal eine Taste des Computers (außer [ESC], [SHIFT], [CTRL] und [CAPS LOCK]) gedrückt zu werden und der Ladevorgang geht weiter. Der zweite Teil des Programmes wird geladen und es erscheint:

```
Loading DEMOP1.BAS block 1
```

- Der zweite Teil hat mehrere Minuten Ladezeit. Auf dem Bildschirm kann man den Ladevorgang der einzelnen Blöcke verfolgen.

- Nach dem Laden des zweiten Teiles erscheint auf der untersten Bildschirmzeile erneut die Aufforderung:

  Press PLAY then any key: ■

- Diese ist wieder mit dem Druck auf eine Computertaste (außer auf einer der vier hier nicht erlaubten) zu beantworten. Der dritte Teil des Programmes wird geladen und es erscheint:

  Loading DEMOP1.BIN block 1

- Der dritte Teil hat noch ca. 1 Minute Ladezeit. Auf dem Bildschirm kann man weiter den Ladevorgang verfolgen. Nach dem Laden des dritten Teiles startet das Programm selbständig und beginnt nach einem Durchlauf immer wieder von neuem. Bei Kassettenrecordern mit Motorsteuerung hält der Spulvorgang an. Bei anderen Geräten kann der Spulvorgang über die [STOP]-Taste am Kassettenrecorder abgebrochen werden.

- Um das Programm zu verlassen, muß zweimal die Taste [ESC] gedrückt werden.

Soll das zweite Programm geladen werden, das auch wieder aus drei Teilen besteht, ist vor der Wiederholung der oben genannten Schritte das Band an die laut Zählerstand angegebene Stelle zu spulen. Bei ordnungsgemäßem Laden erscheinen auf dem Bildschirm

  Loading DEMO2.BAS block 1,

nach dem Titelbild, dem ersten Teil des Programmes, und entsprechender Tastenbetätigung

  Loading DEMOP2.BAS bolck 1

und nach dem zweiten Teil des Programmes,

  Loading DEMOP2.BIN block 1.

Der zweite Teil dieses Programmes hat auch mehrere Minuten Lade zeit. Man kann wieder auf dem Bildschirm den Ladevorgang verfolgen. Das Programm startet nach dem Laden selbständig und fordert dann die aktive Mitarbeit des Bedieners.

Beim Laden dieser Programme sind folgende Fehlermöglichkeiten denkbar:

Statt Loading DEMO1.BAS block 1 erscheint als erstes beim Laden des ersten Programms

Found DEMOP1.BAS block 2	(sinngemäße Übersetzung: Block 2 des Programmes mit Namen DEMOP1.BAS gefunden)

Hier stand das Band nicht am Programmanfang. Das Titelbild mit Namen DEMO1.BAS und der erste Block des Programmes DEMOP1.BAS wurden übersprungen. In diesem Fall wird auch der gefundene Block nicht geladen, sondern zur Orientierung nur Name und Nummer angezeigt. Das Band kann an den Anfang des Programms zurückgespult und der Ladevorgang wiederholt werden.

Weitere Fehlermöglichkeiten beim Laden von Programmen sind in Abschnitt 2.3. angegeben.

2.3. Das Inhaltsverzeichnis einer Kassette und Lesefehler

Das BASIC des KC compact enthält die Anweisung CAT (abgeleitet von catalouge), mit der man den Inhalt jeder Programmkassette auf dem Bildschirm ausgeben und dabei gleichzeitig die Kassette auf Lesbarkeit überprüfen kann. Nach dem Aufruf von CAT über [C] + [A] + [T] + [RETURN] erscheint

```
Press PLAY then any key: ■
```

Werden jetzt die Abspieltaste am Kassettenrecorder und eine Computertaste (außer [ESC], [SHIFT], [CTRL] und [CAPS LOCK]) gedrückt, so wird für jeden gefundenen Datenblock eine Meldung auf dem Bildschirm ausgegeben, die folgende Form hat:

```
Dateiname Blocknummer Dateikennzeichen OK
```

Die Dateikennzeichen bedeuten:

$ BASIC-Programm
% geschütztes BASIC-Programm
* ASCII-Textdatei
& Binäre Datei in Maschinensprache

Eine namenlose Datei erscheint bei CAT unter der Bezeichnung

```
Unnamed file .
```

Das "OK" am Ende der Meldung deutet an, daß der Block richtig gelesen wurde, und die Datei in den Computer geladen werden könnte.

Fehlt das "OK" nach dem Lesen eines Blockes bzw. werden die Mitteilungen

```
Read error a
```

oder

```
Read error b
```

auf dem Bildschirm ausgegeben, kann es dafür folgende Gründe geben:

- Der Kassettenrecorder wurde nicht richtig mit dem Computer verbunden (siehe Gerätebeschreibung/Abschnitt 1.2.).

- Der Kassettenrecorder erfüllt nicht die geforderten elektrischen Bedingungen (siehe Gerätebeschreibung).

- Die Tonkopfeinstellung des Kassettenrecorders stimmt nicht genau (von Werkstatt überprüfen und erforderlichenfalls einstellen lassen).

- Zufällige Bandaussetzer, Zurückspulen und Lesevorgang wiederholen.

Das Inhaltsverzeichnis der Kassette CC4001 DEMO KC compact hat bei fehlerfreiem Lesen folgendes Aussehen:

```
     DEMO1.BAS      block 1 $ OK
     DEMOP1.BAS     block 1 $ OK
     DEMOP1.BAS     block 2 $ OK
          .           .   . . .
          .           .   . . .
          .           .   . . .
     DEMOP1.BAS     block w $ OK
     DEMOP1.BIN     block 1 & OK
          .           .   . . .
          .           .   . . .
     DEMOP1.BIN     block x & OK
     DEMO2.BAS      block 1 $ OK
     DEMOP2.BAS     block 1 $ OK
          .           .   . . .
          .           .   . . .
     DEMOP2.BAS     block y $ OK
     DEMOP2.BIN     block 1 & OK
          .           .   . . .
          .           .   . . .
     DEMOP2.BIN     block z & OK
```

Die Buchstaben w, x, y und z stehen für die Blocknummern der letzten Datenblöcke der einzelnen Programmteile.

Man kann das Auslisten des Inhaltsverzeichnisses jederzeit mit der Taste [ESC] abbrechen. Nach Abbruch der Funktion CAT wird die Kassette in einem gesteuerten Kassettenrecorder automatisch gestoppt. Bei anderen Kassettenrecordern kann sie über die [STOP]-Taste am Gerät angehalten werden.

Bei gesteuerten Kassettenrecordern besteht durch die Eingabe des Kassettenbefehls CAT die Möglichkeit, da das Schaltsignal eingeschaltet ist, Spulvorgänge vorzunehmen (siehe Abschnitt 2.1.4.).

2.4. Übersicht der BASIC-Anweisungen zum Kassettenbetrieb

Das BASIC des KC compact enthält folgende Anweisungen zum Laden von BASIC-Programmen

CHAIN,	siehe Abschnitt 3.12.
CHAIN MERGE,	siehe Abschnitt 3.12.
LOAD,	siehe Abschnitte 3.1. und 3.12.
MERGE,	siehe Abschnitt 3.12.
RUN.	siehe Abschnitte 2.2., 3.1. und 3.12.

Dateieingaben von Kassette erfolgen mit Hilfe der Anweisungen

INPUT #9	siehe Abschnitt 3.12.
LINE INPUT #9	siehe Abschnitt 3.12.
OPENIN und CLOSEIN.	siehe Abschnitt 3.12.

Die Anweisungen zum Speichern von Programmen und Daten lauten

SAVE	siehe Abschnitte 3.9. und 3.12.
LIST #9	siehe Abschnitt 3.12.
OPENOUT und CLOSEOUT	siehe Abschnitt 3.12.
PRINT #9	siehe Abschnitt 3.12.
WRITE #9	siehe Abschnitt 3.12.

Das Inhaltsverzeichnis einer Kassette bekommt man mit

CAT	siehe Abschnitte 2.3. und 3.12.

Die Geschwindigkeit der Datenaufzeichnung beim Speichern von Programmen und Daten auf Kassette kann mit der Anweisung

SPEED WRITE	siehe Abschnitt 3.12.

eingestellt werden. Beim Einlesen von Aufzeichnungen erfolgt eine automatische Anpassung an die verwendete Aufzeichnungsgeschwindigkeit.

Während des Kassettenbetriebes können die Fehlermeldungen 7, 21, 24, 25 und 27 auftreten, die in Anhang A beschrieben sind.

3. Der BASIC-Interpreter

Um die Sprache BASIC zu erlernen, ist es notwendig, sich mit deren Semantik und Syntax bekanntzumachen. Die erste Voraussetzung dazu ist die ausschließliche Verwendung der zulässigen Schlüsselwörter. Ein Schlüsselwort ist eine Folge von Großbuchstaben, die die auszuführende Operation in symbolischer Form bezeichnen. Dabei werden englische Begriffe bzw. Abkürzungen für die Operationen verwendet.

Ein Programm ist eine Folge von Operationen zur Lösung eines Problems. Ein BASIC-Programm besteht aus numerierten Programmzeilen, die unabhängig von der Reihenfolge der Eingabe ihrer Zeilennummern abgearbeitet werden. Jede Programmzeile enthält mindestens eine BASIC-Anweisung. Mehrere Anweisungen in einer Zeile werden durch Doppelpunkt voneinander getrennt. Eine BASIC-Zeile kann maximal 255 Zeichen lang sein. Der Bereich der zulässigen Zeilennummern geht von 0 bis 65535.
Eine eingegebene BASIC-Zeile wird nach Drücken der Taste [RETURN] in den Speicher des Computers übernommen.

Bevor jedoch eigene Programme erstellt werden können, wird man zunächst fertige Programme von Kassette laden und damit arbeiten.

3.1. Programme laden und starten

Zum KC compact wird eine Reihe von Programmkassetten mit kommerziellen Programmen und Spielen im Handel angeboten. Das Laden und Starten eines Programmes von einer solchen Programmkassette kann in der gleichen Weise erfolgen, wie es für die Programme der DEMO-Kassette im Abschnitt 2.2. beschrieben wurde. Voraussetzung hierzu ist, daß das Magnetband genau am Programmanfang steht. Dann wird mit RUN" [RETURN]+[RETURN] bzw. [CTRL]-[ENTER]+[RETURN] jedes Programm geladen und gestartet.

Die Ladeanweisungen in den Kassettenbeschreibungen fordern jedoch eine andere Variante zum Laden und Starten der Programme von den Softwarekassetten.
Beim Laden ist einzugeben:

RUN"name.BAS" [RETURN] .

Die weitere Verfahrensweise ist identisch mit den Angaben in Abschnitt 2.2. Der Unterschied besteht hier darin, daß in der RUN-Anweisung der konkrete Programmname angegeben werden muß. Dies hat den Vorteil, daß der Computer von der Programmkassette nur das Programm lädt und startet, das den angegebenen Namen hat. Ein Beispiel, das mit der DEMO-Kassette nachvollzogen werden kann, erläutert diesen Vorgang und die Bedeutung der Bildschirmausschriften.

Das Magnetband der DEMO-Kassette ist an den Anfang zurückgespult, am Kassettenrecorder ist die Abspieltaste gedrückt und über die Computertastatur wurde eingegeben:

```
RUN"DEMO2.BAS" +[RETURN]+[RETURN]
```

Die Kassette läuft und auf dem Bildschirm erscheinen die Meldungen

```
Found   DEMO1.BAS block 1
Found   DEMOP1.BAS block w
Found   DEMOP1.BIN block x
Loading DEMO2.BAS block 1
```

Die ersten drei Meldungen zeigen an, daß die drei Teile des ersten Programmes gefunden wurden, wobei die Buchstaben w und x für Blocknummern der letzten Datenblöcke der Programmteile 2 und 3 stehen. Diese Programmteile werden nicht geladen.

Die letzte Zeile gibt an, daß der erste Teil des gewünschten Programmes gefunden wurde und geladen wird. Die dann folgenden Vorgänge sind identisch mit den in Abschnitt 2.2. beschriebenen.

Lesefehler werden durch die Ausschriften

```
Read error a
```

oder

```
Read error b
```

angezeigt. Ursachen für die Lesefehler und Möglichkeiten zu deren Beseitigung sind in Abschnitt 2.3. angegeben.

Nach den Anweisungen

```
LOAD" bzw. LOAD"name.BAS"
```

und den gleichen Tastenbetätigungen wie nach RUN" bzw. RUN"name.BAS" erfolgt das Laden entweder des nächsten gefundenen Programmes oder das Laden des Programmes mit dem angegebenen Namen. In diesem Fall wird das Programm nach Abschluß des Ladens jedoch nicht gestartet. Dies muß nach dem Laden durch Eingabe der Anweisung

```
RUN [RETURN]
```

erfolgen.

Die Bildschirmausschriften sind während des Ladens durch LOAD sowie durch die anderen in Abschnitt 2.4. aufgeführten Lade-Anweisungen identisch mit den Ausschriften während des Ladens durch RUN" . Gleiches gilt für die Fehlermeldungen.

3.2. Wichtige BASIC-Anweisungen

Hinweis: BASIC-Anweisungen können mit kleinen und großen Buchstaben eingegeben werden. Der Interpreter wandelt automatisch BASIC-Schlüsselwörter in Großbuchstaben um.

CLS

Löscht den Bildschirm (Clear screen = Bildschirm löschen) Der gesamte Bildschirm wird gelöscht und in der linken, oberen Ecke erscheinen die Ausschrift "Ready" und der Cursor.

PRINT

(print = drucken) Diese Anweisung ist für alle auf dem Bildschirm darstellbaren Zeichenfolgen (Wörter, Zahlen sowie Sonderzeichen) zuständig.

PRINT "Zeichenkette"
PRINT numerischer Ausdruck

z.B. Anzeigen des Wortes "Test"

Eingabe: PRINT "Test"
Anzeige: Test

z.B. Anzeigen der Ziffernfolge 98765

Eingabe: PRINT 98765
Anzeige: 98765

Die PRINT-Anweisung kann durch Betätigung der Taste [?] ersetzt werden, z.B. ? "Test".

Ist die Schreibweise einer eingegebenen BASIC-Anweisung fehlerbehaftet, erscheint die Ausschrift "Syntax error".

z.B.
Eingabe: PRRINT
Anzeige: Syntax error

Befindet sich der Fehler in einer Programmzeile

z.B. 10 PRRINT "Test"

wird der Fehler erst nach dem Start des Programms angezeigt.

```
Eingabe:  RUN
Anzeige:  Syntax error in 10
          10 PRRINT "Test"
```

Der Cursor steht auf der ersten Position der Zeile.

Nun kann der Fehler korrigiert werden. Der Cursor wird mit der Cursortaste [---^] (Cursor rechts) auf das erste "R" bewegt. Die Taste [CLR] löscht das erste "R" und [RETURN] gibt die verbesserte Zeile in den Computer ein. Nach dem Programmstart wird nun das Wort "Test" auf dem Bildschirm angezeigt.

LIST

Programm auflisten (list = auflisten)
Diese Anweisung listet ein Programm, beginnend mit der niedrigsten Zeilennummer, auf dem Bildschirm oder einem Druck gerät aus.

Weitere LIST-Anweisungen:
z.B.

Listen nur einer Zeile

```
Eingabe:  LIST 10
Anzeige:  10 CLS:PRINT
```

Listen eines Bereiches von bis

```
Eingabe:  LIST 10-30
Anzeige:  10 CLS:PRINT
          20 FOR i=1 TO 8
          30 READ i
```

Listen aller Zeilen ab Zeile -

```
Eingabe:  LIST 10-
Anzeige:  10 CLS:PRINT
          :
          :
          200 END
```

Listen aller Zeilen bis - Zeile

```
Eingabe:  LIST -20
Anzeige:  10 CLS:PRINT
          20 FOR i=1 TO 8
```

Beim Auslisten von Programmen werden die Programmzeilen nacheinander in aufsteigender Folge angezeigt. Ist die unterste Bildschirmzeile erreicht, werden die Zeilen nach oben gerollt. Beim Einblenden einer neuen BASIC-Zeile werden alle vorhandenen Zeilen nach oben verschoben, so daß die erste Zeile verschwindet (Scrollvorgang). Das Listen kann zur Kontrolle des Programms mit der Taste [ESC] angehalten und mit Betätigung einer beliebigen anderen Taste (außer [SHIFT], [ESC], [CAPS LOCK], [CTRL]) fortgesetzt werden. Der Abbruch des Scrollvorganges wird durch zweimaliges Drücken der Taste [ESC] erzielt.

INPUT

Eingabe von Daten
Die Eingabe INPUT bewirkt, daß der Computer auf eine Information wartet.
Die Informationseingabe ist mit [RETURN] abzuschließen.

z.B.

```
Eingabe:  10 CLS
          20 INPUT "In welcher Stadt wohnst Du";a$
          30 INPUT "Wie ist die Postleitzahl";b
          40 PRINT b;a$;"ist eine tolle Stadt."
          run
Anzeige:  In welcher Stadt wohnst Du ?
Eingabe:  Muehlhausen
Anzeige:  Wie ist die Postleitzahl ?
Eingabe:  5700
Anzeige:  5700 Muehlhausen ist eine tolle Stadt.
```

In diesem Beispiel werden mit INPUT zwei verschiedene Datenarten eingegeben. Es erfolgt eine Zeichenketteneingabe mit dem Wort "Muehlhausen" und eine numerische Eingabe mit der Zahl 5700. Durch die INPUT-Anweisung werden jeweils der Stringvariablen a$ (Zeichenkettenvariablen) und der numerischen Variablen b Werte zugeordnet. (Den Unterschied zwischen beiden Variablen erläutert Abschnitt 3.4.)

3.3. Interpunktion

Punkt

Auch in der Zahlendarstellung ist die englische Schreibweise (anstatt Dezimalkomma einen Dezimalpunkt) zu verwenden. Der Computer arbeitet mit ganzen reellen Zahlen. Dabei ist zu beachten, daß bei der Exponentendarstellung statt "mal zehn hoch" nur E geschrieben wird.

Einige Beispiele:

Mathematische Schreibweise	Computerschreibweise
4.8	4,8
0,02	.02
1,17*10	1.17E-15
7,58*10	7.58E+12

```
Eingabe:  PRINT 0.00000000000000117
Anzeige:  1.17E-15

Eingabe:  PRINT 7580000000000
Anzeige:  7.58E+12

Eingabe:  PRINT 6E3*1E3
Anzeige:  6000000
```

Komma

Das Komma in Verbindung mit PRINT gibt dem Computer an, daß eine formatierte Ausgabe (Abstand 13 Zeichen) bei der Anzeige auf dem Bildschirm erfolgt.

z.B.

```
Eingabe:  PRINT 7,8
Anzeige:  7 8

Eingabe:  PRINT 1,9
Anzeige:  1 9
```

Semikolon

Das Semikolon in Verbindung mit PRINT realisiert eine fortlaufende Anzeige; es verhindert eine Ausgabe auf eine neue Zeile.

z.B.

```
Eingabe:  PRINT 7;-8
Anzeige:  -7 -8
```

Doppelpunkt

Der Doppelpunkt ist das Trennzeichen zwischen zwei BASIC-Anweisungen innerhalb einer Zeile (maximal 255 Zeichen).

z.B.

```
  10 CLS:PRINT "1.Bild":PRINT 800
```

Konstanten

Konstanten sind festgelegte unveränderte Werte.
z.B. PI = 3.14159265

Ganze Zahlen stehen von -999999999 bis +999999999 zur Verfügung. Reelle Zahlen können zwischen -1.7E+38 und 1.7E+38 mit einer Genauigkeit von 9 Stellen liegen.

Klammerausdrücke

Runde Klammern ( ) müssen mit angegeben werden.

z.B. PRINT COS(45)

Eckige Klammern [ ] enthalten Angaben, die wahlweise hinzugefügt werden können.

z.B. RUN [*Zeilennummer*] kann verwendet werden als:

RUN oder RUN 50

Sonderzeichen

& oder H	Konstante ist hexadezimal
&x	Konstante ist binär
#	Kennzeichen für Stream (Bildschirmbereich, Ein-/Ausgabegerät)

3.4. Variablen

Variable:
Eine Variable ist eine veränderliche Größe. Sie besteht aus einem Variablennamen, dem Variablentyp und hat einen Wert.

Variablenname:
Der Name einer Variablen muß immer mit einem Buchstaben beginnen, z.B. WERT, A, B, C, X, PREIS, TYP. Variablennamen dürfen keine BASIC-Schlüsselwörter enthalten (siehe Anhang F). Für die Schreibweise von Variablen sind Groß- und Kleinbuchstaben zugelassen, jedoch werden diese nicht unterschieden.

Einer Variablen werden Werte zugeordnet. Sind diese Werte Zahlen, so spricht man von numerischen Variablen (z.B. a). Sind diese Werte Zeichenketten (Strings), so bezeichnet man diese als Stringvariablen (z.B. a$).
Die Kennung von Stringvariablen erfolgt durch das Anhängen eines $-Zeichens am Ende des Namens, z.B. A$, B$, X$, NAME$, TYP$.

Variablentyp:
Die Art der Variablen wird durch die Typangabe hinter dem Variablennamen gekennzeichnet.

Dabei heißt:

```
%    ganzzahlige Variablen
|    reell (Variablen werden als reelle Zahlen definiert)
$    Stringvariablen (Zeichenketten)
```

3.5. Programmänderung

Zur Programmänderung stehen die drei folgenden Möglichkeiten zur Verfügung:
1. Neuschreiben der fehlerhaften Zeile
2. Korrektur mit EDIT-Modus
3. Korrektur mit COPY-Cursor-Modus

3.5.1. Neuschreiben

Soll eine alte Zeile durch eine neue ersetzt werden, so ist der neuzuschreibende Text mit der alten Zeilennummer einzusetzen. Nach Betätigen der [RETURN]-Taste wird die alte Zeile durch die neue überschrieben.

3.5.2. EDIT-Modus

Zeile 10 weist z.B. einen Schreibfehler auf (statt der Anführungsstriche steht vor dem Wort "Test" die Ziffer 2):

```
10 PRINT 2Test"
```

```
Eingabe:   EDIT 10
Anzeige:   10 PRINT 2Test"
```

Der Cursor erscheint auf der ersten Ziffer der Zeilennummer. Mit der Taste [--->] wird der Cursor auf die 2 bewegt und durch Drücken von [CLR] gelöscht. Nun können die Anführungszeichen eingegeben werden.

3.5.3. COPY-Cursor-Modus

Mit dem Einstellen des COPY-Cursors kann eine BASIC-Zeile bis zur Fehlerstelle kopiert werden. Anschließend kann die Fehlerkorrektur erfolgen (siehe Abschnitt 1.2.).

3.6. Weitere BASIC-Anweisungen

GOTO

(go to = gehe zu)
Die Anweisung bewirkt einen Sprung zu einer anderen Zeilennummer im Programm. Der Grund hierfür kann das Überspringen eines Programmteils oder das Bilden einer Programmschleife sein.

Beispiel: Programmschleife

```
Eingabe:   10 CLS
           20 PRINT "Zeile wird immer wieder geschrieben."
           30 GOTO 20
```

Durch die Eingabe von RUN wird der Bildschirm mit "Zeile wird immer wieder geschrieben" beschrieben, so daß das Wort "Zeile" zu Beginn steht. Die Zeile 30 bewirkt dabei eine Endlosschleife. Das Programm kann nur mit der [ESC]-Taste unterbrochen und mit nochmaligem Drücken der Taste abgebrochen werden.

CONT

(continue = fortsetzen)
Nach einer Programmunterbrechung kann mit CONT das Programm weiterarbeiten.

IF

(if = wenn)
Der Computer wird nach einer bestimmten Bedingung abgefragt. Ist diese erfüllt, folgt mit THEN (dann) die Ausführung einer bestimmten Anweisung.

z.B.

```
Eingabe:   IF A=3 THEN PRINT "richtig"
Anzeige:   richtig
```

ELSE

ELSE wird in Verbindung mit der IF-THEN-Anweisung benutzt. Nach ELSE werden Ausführungshinweise angegeben, wenn in der IF-THEN-Kombination die Bedingung nicht erfüllt wurde.

z.B.

```
Eingabe:   IF A ^ 2 THEN PRINT "richtig" ELSE PRINT "falsch"
Anzeige:   richtig
```

Die im Beispiel aufgeführte Bedingung ist erfüllt, denn vier ist größer als zwei und es wird "richtig" auf dem Bildschirm ausgeschrieben.

END

END ist eine Anweisung, die ein Programm beendet.

z.B.
Eingabe:
```
10 CLS
20 INPUT "Eingabe der Ziffern 1-5";a
30 IF a 2 THEN PRINT "sehr gut":GOTO 80
40 IF a 3 THEN PRINT "gut":GOTO 80
50 IF a 4 THEN PRINT "befriedigend":GOTO 80
60 IF a 5 THEN PRINT "genuegend":GOTO 80
70 IF a 6 THEN PRINT "ungenuegend"
80 END
```

Dieses Beispiel zeigt, daß nach dem Eingeben der Ziffer 1 in Zeile 30 die Auswertung erfolgt. Ist a=1, so schreibt der Computer "sehr gut" aus. Das END in Zeile 30 verhindert, daß die Bedingung für a=2 auch ausgeführt wird.

FOR und **NEXT**

(for = für; to = bis; next = nächstes)
Soll eine Operation oder eine Anweisung mehrmals in einem Programm abgearbeitet werden, schließt man diese in eine FOR-NEXT-Schleife ein.

z.B.
Eingabe:
```
10 CLS
20 FOR i=0 TO 10
30 PRINT "Zahl";i
40 NEXT i
run
```

In Zeile 20 wird eine Variable i vereinbart, die als Laufvariable die Werte 0 bis 10 annimmt. Beim Durchlaufen wird der Wert der Variablen jedesmal um eins erhöht bis der Endwert erreicht ist.

STEP

(step = Schritt)
STEP legt den Wert fest, um den die Variable der FOR-NEXT-Schleife beim Durchlaufen erhöht werden soll. Erfolgen keine Angaben, wird 1 als Schrittweite angenommen.

Beispiel: Bei der Zahlenanzeige sollen auch die Zwischenwerte 0.5, 1.5 usw. angegeben werden.

Eingabe: `20 FOR i=0 TO 10 STEP .5`

Durch die Anweisung STEP .5 erfolgt jetzt die Zahlenanzeige in einer Schrittweite von 0.5.

Ebenso kann die FOR-NEXT-Schleife rückwärts zählen.

Eingabe: `20 FOR i=10 TO 0 STEP -.5`

REM

(remark = Bemerkung)
Beginnt eine BASIC-Zeile mit der Anweisung REM, wird eine Kommentarzeile definiert. Alle Anweisungen bzw. Zeichenketten, die hinter REM liegen (auf einer BASIC-Zeile), werden vom Computer beim Programmablauf ignoriert, aber beim Auslisten mit ausgegeben. Für REM kann auch das Zeichen "'" verwandt werden.

z.B.
Eingabe: 5 REM Zahlenanzeige 0 bis 10
oder
Eingabe 5 ' Zahlenanzeige 0 bis 10

GOSUB n

(go subroutine = gehe zum Unterprogramm)
Aufruf eines BASIC-Unterprogrammes (n = Zeilennummer)
Diese Anweisung ruft ein Unterprogramm auf, das auf der angegebenen Zeilennummer beginnt. Wird ein Programmabschnitt öfters im Programm benötigt, ist es günstig, diesen Abschnitt als Unterprogramm zu definieren. Das Hauptprogramm ruft dann mit GOSUB n das auf der Zeile n beginnende Unterprogramm auf und arbeitet dieses ab.

RETURN

Endeerkennung eines Unterprogrammes
Jedes Unterprogramm wird mit RETURN abgeschlossen. Nach dem Erreichen der RETURN-Anweisung kehrt das Programm an die Stelle des Hauptprogrammes zurück, an der es dieses nach der GOSUB-Anweisung verlassen hat.

3.7. Der Computer als Taschenrechner

Zur Verkürzung der Schreibweise kann für PRINT auch das Zeichen "?" benutzt werden. Dies bietet sich im "Taschenrechner-Modus" an. Durch das Betätigen der [RETURN]-Taste wird das Ergebnis sofort auf dem Bildschirm angezeigt.

Addition

```
Eingabe:   ? 4+5
Anzeige:   9

Eingabe:   ? 40+22
Anzeige:   62
```

Im Gegensatz zum Taschenrechner wird das Gleichheitszeichen "=" nicht eingegeben.

Subtraktion

```
Eingabe:   ? 10-4
Anzeige:   6

Eingabe:   ? 255-128
Anzeige:   127
```

Multiplikation

Als Multiplikationszeichen fungiert das Zeichen "*".

```
Eingabe:   ? 10*20
Anzeige:   200

Eingabe:   ? 2*6
Anzeige:   12
```

Division

Als Divisionszeichen dient das Zeichen "/".

```
Eingabe:   ? 20/4
Anzeige:   5

Eingabe:   9/3
Anzeige:   3
```

Ganzzahlige Division

Als Divisionszeichen fungiert der inverse Schrägstrich " ", der neben der [COPY]-Taste liegt.

```
Eingabe:   ? 40 6
Anzeige:   6

Eingabe:   ? 12 5
Anzeige:   2
```

Modulo

Bei der Modulo-Operation wird der Rest, der nach einer ganzzahligen Division bleibt (z.B. 10:8=1 Rest 2), angezeigt.

```
Eingabe:   ? 10 MOD 2
Anzeige:   0

Eingabe:   ? 10 MOD 8
Anzeige:   2
```

Quadratwurzel

Mit SQR(n) wird die Quadratwurzel der in Klammern stehenden Zahl n erhalten.

```
Eingabe:   ? SQR(4)
Anzeige:   2

Eingabe:   ? SQR(2*12.5)
Anzeige:   5
```

Potenzen

Für das Potenzieren wird das Zeichen " " links neben der Taste [CTRL] benutzt.
Eine Zahl wird so oft mit sich selbst multipliziert, wie es der Exponent angibt.

z.B. 2 = 2*2*2

Der Pfeil "↑" ([↑] -Taste) steht als Exponentenzeichen.

z.B. 2 = 2 3

```
Eingabe:   ? 4 2
Anzeige:   16

Eingabe:   ? 2 3
Anzeige:   8
```

Kubikwurzel

Ähnlich wie beim Potenzieren wird beim Ziehen der Kubikwurzel verfahren.

Die Kubikwurzel aus 64 wird so eingegeben:

```
Eingabe:   ? 64 (1/3)
Anzeige:   4

Eingabe:   ? 8 (1/3)
Anzeige:   2
```

Gemischte Rechenoperationen

Gemischte Rechenoperationen werden vom Computer auch ausgeführt. Dabei ist jedoch zu beachten, daß eine bestimmte Reihenfolge in der Abarbeitung besteht. An erster Stelle liegt die Multiplikation. Sie besitzt die höchste Priorität unter den vier Grundrechenarten.

Beispiel: `6 + 5 - 3 * 4 / 2`

In folgenden Schritten rechnet der Computer:

```
6 + 5 - 3 * 4 / 2

= 6 + 5 - 12 / 2
= 6 + 5 - 6
= 11 - 6
= 5
```

Werden nun Klammern verwendet, ändert sich die Reihenfolge der Berechnung.

Beispiel: `(6 + 5 - 3) * 4 / 2`

```
= (11 - 3) * 4 / 2
= 8 * 4 / 2
= 32 / 2
= 16
```

Prioritäten der Abarbeitung

Für alle mathematischen Operationen gelten folgende Prioritäten:

1. Klammer	( )
2. Potenzierung	↑
3. Modulo	MOD
4. Negatives Vorzeichen	-
5. Multiplikation und	*
Division	/
6. Ganzzahlige Division	\
7. Addition und	+
Subtraktion	-

3.8. Programm löschen

Ein Programm im Speicher des Computers kann durch

- Aus- und wieder Einschalten des Computers,
- durch Drücken der Tasten [CTRL]-[SHIFT]-[ESC] oder
- mit NEW

gelöscht werden.

NEW

Programm löschen

3.9. Programm abspeichern

Nach dem Erstellen eines Programms kann dieses auf Magnetbandkassette abgespeichert werden, wenn es später nochmals benötigt wird, da nach dem Ausschalten des Computers das Programm im Speicher gelöscht wird, und es somit nicht mehr zur Verfügung steht.

Zum Abspeichern (bzw. zum Retten) von Programmen dient das Kommando SAVE (siehe auch Abschnitt 3.12.).

Nach der Eingabe von

```
SAVE "Programmname" + [RETURN]
```

meldet sich der Computer mit

```
Press REC and PLAY then any key:■
```

(sinngemäße Übersetzung: Aufnahmetasten am Kassettenrecorder und dann eine Computertaste drücken)

Entsprechend der Aufforderung sind am Recorder die Tasten zur Aufnahme zu drücken und am Computer eine beliebige Taste (außer [ESC], [SHIFT], [CTRL] und [CAPS LOCK]). Das Band in der Kassette wird gespult und der Computer speichert das Programm ab. Auf dem Bildschirm erscheint die Meldung:

```
Saving Programmname block 1
```

(sinngemäße Übersetzung: Block 1 des Programmes mit Namen "Programmname" wird gerettet)

Danach folgen Anzeigen mit den übrigen Blocknummern, bis das Programm abgespeichert ist.

Der Programmname kann aus maximal 16 Zeichen bestehen. Innerhalb des Programmnamens sind Buchstaben, Ziffern und Satzzeichen (außer dem Ausrufungszeichen "!" und den Anführungsstrichen) erlaubt. Steht am Anfang des Dateinamens ein Ausrufungszeichen "!", so werden sowohl beim Retten als auch beim späteren Einladen die Bildschirmausschriften unterdrückt. Beim Retten braucht dann nach Eingabe von

```
SAVE "!Programmname" + [RETURN]
```

keine weitere Computertaste gedrückt zu werden, um den Abspeichervorgang zu starten. Hierbei ist darauf zu achten, daß die Aufnahmetasten am Kassettenrecorder gedrückt sind, bevor die [RETURN]-Taste gedrückt wird.

Wird der Speichervorgang durch Drücken der [ESC]-Taste abgebrochen, erscheint die Fehlermeldung:

```
Broken in
```

Das bis dahin auf Kassette gerettete Programmstück ist dann allerdings unbrauchbar.

3.10. Bildschirmmodus und Farben

3.10.1. Bildschirmmodus

Der KC compact kann in drei unterschiedlichen Darstellungsmodi arbeiten:

- Modus 0 = 20 Zeichen pro Zeile
- Modus 1 = 40 Zeichen pro Zeile
- Modus 2 = 80 Zeichen pro Zeile.

Nach dem Einschalten bzw. nach dem Zurücksetzen des Computers in den RESET-Zustand ist automatisch der Modus 1 eingestellt, das heißt, in einer Bildschirmzeile können 40 Zeichen geschrieben werden. Entsprechend stehen 20 Zeichen je Zeile im Modus 0 bzw. 80 Zeichen je Zeile im Modus 2 zur Verfügung. Die Zeilenanzahl auf dem Bildschirm ist immer 25, unabhängig vom eingestellten Modus.

Einstellen des Modus mit dem Schlüsselwort MODE
  z.B. MODE 2 + [RETURN]

3.10.2. Farben

Der Computer ermöglicht die Darstellung von 27 unterschiedlichen Farben, die auf einem Schwarz-/Weiß-Fernsehgerät in unterschiedlichen Grautönen erscheinen.

In Abhängigkeit des Modus stehen folgende Farben zur Auswahl:

Modus 0: 16 Farben der 27 Farben können maximal gleichzeitig benutzt werden.
Modus 1: 4 Farben der 27 Farben können maximal gleichzeitig benutzt werden.
Modus 2: 2 Farben der 27 Farben können maximal gleichzeitig benutzt werden.

Bildschirmaufbau

Der Bildschirm besitzt eine Schreibfläche, in der der Cursor (viereckiges Zeichen) über den Bildschirm bewegt werden kann. Diese Fläche kann auch als Bildschirmbereich = Stream #0 oder Fenster bezeichnet werden, wobei der Begriff Stream auch für ein Gerät, z.B. Bildschirmgerät, Drucker oder Diskettengerät stehen kann. Unterscheidungen werden hier nur in der angegebenen Stream-Zahl getroffen.

BORDER, PEN, PAPER

Der vom Cursor nicht erreichte Bereich des Bildschirmes ist der Bildschirmrand (BORDER). Ihm können alle 27 Farben, unabhängig vom eingestellten Modus, zugeordnet werden.
Soll die Farbe verändert werden, ist dies z.B. mit:

```
BORDER 3
```

möglich.

Beim Einschalten bzw. nach dem Zurücksetzen des Computers in den RESET-Zustand sind PAPER standardmäßig auf 0 und PEN auf 1 gestellt.
Es ist zu beachten, daß 0 und 1 PAPER- und PEN-Nummern sind und damit keine Farbnummern für die INK-Anweisung.

PEN ist der Schreibstift, mit dem geschrieben wird. Wie bei einem Kugelschreiber können hier auch dem Stift (PEN) Farben zugeordnet werden. Von den 16 Farbstiften (PEN), die im Computer zur Verfügung stehen, können jedem Stift eine von 27 Farben zugeordnet werden.

Farbtabelle

INK Nr.	Farbe	INK Nr.	Farbe	INK Nr.	Farbe
0	schwarz	9	grün	18	leuchtend grün
1	blau	10	blaugrün	19	seegrün
2	leuchtend blau	11	himmelblau	20	leuchtend blaugrün
3	rot	12	gelb	21	limonengrün
4	magenta	13	weiß	22	pastellgrün
5	mauve	14	pastellblau	23	pastell-blaugrün
6	leuchtend rot	15	orange	24	leuchtend gelb
7	purpur	16	rosa	25	pastellgelb
8	leuchtend magenta	17	pastellmagenta	26	leuchtend weiß

Tabelle 2: Farbtabelle des Computers

Nach der Auswahl des Farbstiftes PEN wird die Farbe mit der Anweisung INK vereinbart. Zum Beispiel können drei Stiften mit Nummer 0, 1, 2 eine der 27 Farben zugeordnet werden. Stift 0 kann die Farbe 3, Stift 1 die Farbe 1, Stift 2 die Farbe 9 erhalten. Diese drei Stifte besitzen nun die Kugelschreiberminen rot, blau und grün. Falls benötigt, können diese Farbminen gegen andere Farben (0 bis 26) ausgetauscht werden oder es können auch alle 3 die gleiche Farbe bekommen.

In Abhängigkeit vom Modus (es war MODE 1 eingestellt) kann nun die Auswahl der Farben erfolgen.

In Tabelle 3 sind die Farbzuordnungen von PAPER und PEN zu INK in Abhängigkeit vom eingestellten Bildschirmmodus angegeben, wie sie nach dem Einschalten des Computers eingestellt sind.

PAPER/ PEN	INK-Farbe Modus 0	INK-Farbe Modus 1	INK-Farbe Modus 2
0	1	1	1
1	24	24	24
2	20	20	1
3	6	6	24
4	26	1	1
5	0	24	24
6	2	20	1
7	8	6	24
8	10	1	1
9	12	24	24
10	14	20	1
11	16	6	24
12	18	1	1
13	22	24	24
14	blinkend 1,24	20	1
15	blinkend 16,11	6	24

Tabelle 3: Farbzuordnungstabelle für PAPER/PEN, MODE, INK

Der Farbstift PEN 1 (Spalte PAPER/PEN - Ziffer 1) besitzt standardmäßig im MODE 1 (Spalte INK-Farbe-Modus 1) die Farbe mit der Nummer 24.
Die Tabelle 2 gibt die der Nummer 24 zugeordnete Farbe an und das ist leuchtend gelb.
Diese Zuordnung zwischen PAPER, PEN und INK ist die Standardeinstellung des Computers beim Einschalten bzw. beim Zurücksetzen in den RESET-Zustand.
Die INK-Anweisung ermöglicht das Verändern des Schriftuntergrunds (PAPER) und der Schrift (PEN).
Die INK-Anweisung besteht aus zwei Parametern. Parameter eins gibt die Nummer des PENs (Farbstiftes) an bzw. die Nummer des PAPERs an, worauf geschrieben werden soll. Parameter zwei enthält die INK-Farbe des Stiftes.

INK ganze Zahl, ganze Zahl

Da zur Zeit PEN Nummer 1 eingestellt ist, wird mit

```
INK 1,6
```

die Schrift 6 = leuchtend rot gewählt (lt. Tabelle 2 und 3).

Die PAPER-Farbe kann ebenfalls mit INK verändert werden. PAPER besitzt nach dem Einschalten die Nummer 0. Deshalb muß mit INK folgende Eingabe getätigt werden:

```
INK 0,13
```

Der Hintergrund erscheint in weiß.

Soll ein anderer Farbstift (PEN) verwendet werden, ist z.B.

```
PEN 2
```

zu schreiben. Dies bewirkt, daß nur die Schrift nach dieser Anweisung ihre Farbe ändert.
Laut Tabelle 3 ist zu entnehmen, daß PEN 2 automatisch die INK-Farbe 20 (leuchtend blaugrün) besitzt.
PEN 2 kann eine andere Farbe zugewiesen bekommen, wenn

```
INK 2,15
```

eingegeben wird. Farbstift 2 erhält die Farbe orange.

Zusammenfassend bewirkt die INK-Anweisung:

1. Die Veränderung des Schrifthintergrundes PAPER
2. Die Veränderung der Farbe des Stiftes (PEN).

Es kann auch die Farbe eines PENs oder von PAPER verändert werden, die momentan nicht verwendet wird.

z.B. `INK 1,2`

gibt dem PEN 1 die Farbe blau. Mit der Eingabe von PEN 1 erscheint blaue Schrift. Mit dem gleichzeitigen Drücken der Tasten [CONTROL]-[SHIFT]-[ESC] wird der Computer in den RESET-Zustand versetzt.

Blinkende Farbeinstellung

Die auf dem Bildschirm angezeigte Schrift kann in 2 Farben ab wechselnd blinken, wenn in der INK-Anweisung zur Einstellung der PEN-Farbe 2 Farben angegeben werden.

```
INK 1,2,6
```

Dabei gilt:
1 = PEN-Nummer
2 = Farbe leuchtend blaugrün
6 = leuchtend rot

Ebenso kann der Schrifthintergrund in zwei Farben blinken. Dazu wird der INK-Anweisung, mit der die PAPER-Farbe eingestellt wird, eine zweite Farbe zugeordnet.
Mit

```
INK 0,19,9
```

werden für PAPER 0 (noch eingestellt) die Farben 19 (seegrün) und 9 (grün) vereinbart.

Der Computer stellt, wie in Tabelle 2 ersichtlich, im Modus 0 für PEN 14 und 15 bzw. PAPER 14 und 15 zwei blinkende Farben zur Verfügung.

Die Eingabe von MODE 0 und PEN 15 zeigt das Wort "Ready" abwechselnd himmelblau und rosa an. Das Gleiche gilt für PAPER 14 und CLS, hier erscheint der Hintergrund in gelb und in blau wechselnd.

Der Rand BORDER kann mit dem Eingeben zweier Farbwerte ebenfalls blinken.

```
BORDER 24,11
```

Der Rand leuchtet nun leuchtend gelb und himmelblau wechselnd auf.

3.11. Fenster

Wie schon erwähnt, ist standardmäßig der größte Bildschirmbereich Stream #0 = Fenster 0 eingestellt. In diesem Fenster erscheinen alle Meldungen von BASIC (z.B. Ready). Die Fensternummer wird immer mit einem # gekennzeichnet.
Mit der Anweisung WINDOW (engl. Fenster) kann ein Fenster eingerichtet werden.

WINDOW folgen 5 Werte, die folgende Bedeutung besitzen:

1. Wert = Fenster - Nummer mit #
2. Wert = links - Spaltennummer
3. Wert = rechts - Spaltennummer
4. Wert = oben - Zeilennummer
5. Wert = unten - Zeilennummer

Im eingestellten Modus 1 sind 40 Spalten und 25 Zeilen vorhanden. Mit

```
WINDOW #2,4,35,6,20
```

wird das Fenster 2 von Spalte 4 bis 35 und von Zeile 6 bis 20 definiert. Soll dieses nun farblich gestaltet werden, kann das durch:

```
INK 2,6 PAPER #2,2 CLS #2
```

erfolgen.
Es erscheint jetzt ein rotes Rechteck auf dem Bildschirm.

Insgesamt können 8 Fenster definiert werden, durch die Angabe des Streams #0, #1, #2, #3 usw. in der Anweisung. Ist keine Stream-Nummer vorhanden, wird automatisch Fenster 0 eingestellt.
Das Benutzen von Stream-Nummern in anderen Anweisungen bezieht sich auch immer auf Bildschirmbereiche für Stream #0 bis #7.

Stream #8 gilt für Druckgerät
Stream #9 gilt für Kassettengerät, Diskettengerät.

3.12. Liste der BASIC-Anweisungen

ABS (*numerischer Ausdruck*)

Beispiel: Eingabe: `PRINT ABS(-39.54)`
Anzeige: `39.54`

Funktion: Wandelt den in Klammern stehenden Ausdruck in den absoluten Wert (Betrag: Wert ohne Vorzeichen) um.

AFTER [*Intervall*] [,*Zeitgebernummer*] GOSUB *Zeilennummer*

Beispiel:
```
 10 AFTER 250,2 GOSUB 100
 20 PRINT "Programmaufruf nach 5 Sekunden "
 30 a$ = INKEY$
 40 IF a$ = "e" OR a$ = "E" THEN MODE1: END ELSE
    GOTO 30
100 CLS: PRINT "Hier ist das Unterprogramm"
110 FOR I = 1 TO 700: NEXT
120 MODE2
130 PRINT " M O D E 2"
140 RETURN
run
```

Funktion: Nach einem bestimmten Zeitabschnitt wird ein Unterprogramm aufgerufen. Die Einheiten des Zeitintervalls basieren auf 0,02 Sekunden (Fünfzigstelsekunden). Es stehen im Computer 4 unabhängige Zeitgeber zur Verfügung (0 bis 3), die eigenen Unterprogrammen zugeordnet sein können. Die Zeitgebernummer kann 0, 1, 2, 3 sein, wobei die 0 die niedrigste Rangfolge (Priorität) und 3 die höchste besitzt.
Von einer zentralen Uhr werden alle zeitabhängigen Vorgänge im Computer gesteuert, das heißt auch zeitabhängige Hardware (Geräte, Baugruppen)-Funktionen. Die Anweisung AFTER ermöglicht, daß im Beispiel nach einer Zeit von 250*0,02 Sekunden mit dem Zeitgeber 2 das Unterprogramm auf der Programmzeile 100 aufgerufen und abgearbeitet wird. Am Ende des Unterprogramms steht wie üblich RETURN. Das Programm kehrt zu der Stelle zurück, wo es vorher unterbrochen wurde. Mit der Taste [e] wird das Programm beendet.

Ausdruck **AND** *Ausdruck*

Beispiel: Die Zahlen 9 und 1000 werden miteinander AND verknüpft.

dezimale Darstellung	duale Darstellung
9	1001
1000	1111101000
9 AND 1000	0000001000

Hieraus geht hervor, daß das Ergebnis nur 1 ist, wenn beide zu verknüpfende Bits 1 sind.

Funktion: Logische UND-Funktion (Boolescher Verknüpfungsoperator) Bit für Bit wird die Verknüpfungsoperation mit ganzen Zahlen durchgeführt.
Dabei kann der Ausdruck sein:
- numerischer Ausdruck (ganze Zahl)
- logischer Operator (z.B AND, OR, NOT)
- Vergleichsoperator (z.B. >,=)

Von den logischen Operatoren besitzt AND die höchste Priorität.

ASC (*Stringausdruck*)

Beispiel: Eingabe: PRINT ASC("rs")
Anzeige: 114
Funktion: Es wird der numerische Wert des ersten Zeichens, des in Klammern stehenden Stringausdrucks, angegeben.

ATN (*numerischer Ausdruck*)

Beispiel: Eingabe: PRINT ATN(1)
Anzeige: .785398163
Funktion: Berechnung des ArcusTaNgens des in Klammern stehenden numerischen Ausdrucks. Das Ergebnis wird im Bogenmaß ausgegeben.

AUTO [*Zeilennummer*] [,*Schrittweite*]

Beispiel: Eingabe: AUTO 40,5
Anzeige: 40
Funktion: Automatisches Numerieren der Zeilen Ohne Angaben wird mit Zeile 10 und Schrittweite 10 als Standardwerte begonnen. Die Schrittweite kann die Werte 1 bis maximal 65535 annehmen. Der Abbruch dieser Funktion erfolgt durch Drücken der [ESC]-Taste.

BIN$ (*vorzeichenloser ganzzahliger Ausdruck* [,*Feldbreite*])

Beispiel: Eingabe: PRINT BIN$(16,8)
Anzeige: 00010000

Funktion: Wandelt den vorzeichenlosen ganzzahligen Ausdruck in BINäre Stringform um, mit der in der Feldbreite angegebenen Stellenzahl.
Wertebereich: von -32768 bis +32767
Wert der Feldbreite: von 0 bis 16
Auswertung Ergebnis:
- weniger Stellen als vereinbart - es werden Nullen vorangestellt
- mehr Stellen als vereinbart - das Ergebnis wird nicht gekürzt, also genau die benötigte Anzahl der Stellen wird ausgegeben.

BORDER *Farbe* [,*Farbe*]

Beispiel: Eingabe: BORDER 6 oder
BORDER 6,26

Funktion: Einstellen der Farbe der nichtbeschreibbaren Randfläche des Bildschirms. Farbwerte: von 0 bis 26 (siehe Tabelle 2) Bei Angabe der zweiten Farbe wechseln die Farben in der Geschwindigkeit, die in der SPEED INK-Anweisung angegeben ist.

BREAK (siehe ON BREAK CONT/GOSUB/STOP)

CALL *Adreßausdruck* [,*Liste von Parametern*]

Beispiel: Eingabe: CALL 0
setzt den Computer in den Einschaltzustand.

Funktion: Sprung zu einem Unterprogramm außerhalb von BASIC, das auf angegebener Adresse beginnt.

CAT

Funktion: (catalogue - Katalog)
Anzeige des Inhaltsverzeichnisses von Kassette oder Diskette.
Für Kassette gilt: Nach Aufruf von CAT erscheint

Press PLAY then any key: ■
(sinngemäße Übersetzung: Abspieltaste am Kassettengerät und dann eine Computertaste drücken)

Nach Drücken einer beliebigen Taste und Einstellen des Recorders auf Wiedergabe werden alle Programme, die sich auf der Kassette befinden, blockweise angezeigt mit

Programmname, Blockanzahl, Dateityp

Mögliche Dateitypen:
$ kennzeichnet ungeschützte BASIC-Datei
% kennzeichnet geschützte BASIC-Datei
* kennzeichnet ASCII-Datei
& kennzeichnet Binär-Datei

z.B.
TEST.BAS block 1 $ OK

Richtiges Einlesen des Blockes wird mit OK quittiert, ein Lesefehler wird mit
Read error a oder
Read error b angezeigt.

Fehlerursachen werden im Abschnitt 2.3. aufgeführt.

Am Ende wird die [ESC]-Taste zum Verlassen von CAT gedrückt.

CHAIN *Dateiname* [,*Zeilennummer*] (*Datei-/Programmname*)

Beispiel: Eingabe: `CHAIN "Beispiel.bas",500`

Funktion: Ein Programm wird von Kassette oder Diskette in den Speicher geladen und ab der angegebenen Zeilennummer oder von vorn gleich gestartet. Ein im Speicher befindliches Programm wird gelöscht. Geschützte Programme, durch P beim Abspeichern gekennzeichnet, können geladen und gestartet werden.

CHAIN MERGE *Dateiname* [,*Zeilennummer*][,DELETE *Zeilenbereich*]

Beispiel: Eingabe: `CHAIN MERGE "Testneu.bas",1000,DELETE 350-500`

Funktion: Mischen von Programmen; Ein Programm wird von Kassette oder Diskette in den Speicher geladen und einem bereits existierenden hinzugefügt. Dieses neue Programm wird sofort von vorn oder ab angegebener Zeilennummer gestartet. Wenn erforderlich, kann das vorherige Programm im Speicher mit DELETE Zeilenbereich vor dem Laden gelöscht werden.

Achtung: Programmzeilen, die im alten und im neuen Programm existieren, werden vom neuen Programm überschrieben.

CHR$ (*ganzzahliger Ausdruck*)

Beispiel: Eingabe: `PRINT CHR$(254)`
Funktion: Wandelt den ganzzahligen Ausdruck in einen String um entsprechend der Zuordnung in der Zeichentabelle (siehe Anhang B).
Werte des ganzzahligen Ausdrucks: 0 bis 255, dabei sind 0 bis 31 Kontrollzeichen.

CINT (*numerischer Ausdruck*)

Beispiel: Eingabe: `PRINT CINT(4.999)`
Anzeige: `5`
Funktion: Rundet den numerischen Ausdruck zu einer ganzen Zahl. Wertebereich: -32768 bis +32767 (Integerzahl)

CLEAR

Beispiel: Eingabe: `CLEAR`
Funktion: Alle Variablen werden Null. Offene Dateien, Felder und Anwenderfunktionen werden gelöscht. BASIC arbeitet in Bogenmaß.

CLEAR INPUT

Beispiel: Eingabe:
```
10 CLS
20 PRINT "Taste druecken"
30 FOR I=1 TO 200: NEXT
40 CLEAR INPUT
run
```

Funktion: Vergißt alle vorherigen INPUT-Informationen, die sich noch im Tastaturpuffer befinden. Wie das Beispiel zeigt, werden beim Tastendruck keine Zeichen angezeigt. Dies ändert sich, wird die Zeile 40 gelöscht.

CLG [*Farbstift*]

Beispiel: CLG 2

Funktion: Löschen des Grafik-Bildschirms und Ausfüllen mit der Farbe, die dem Farbstift mit INK zugeordnet wurde. Wird keine Angabe getätigt, bleibt der im letzten CLG-Befehl benutzte Farbstift eingestellt.

CLOSEIN

Beispiel: CLOSEIN

Funktion: Beendet die Eingabe einer Datei von Kassette oder Diskette.

CLOSEOUT

Beispiel: CLOSEOUT

Funktion:Beendet die Ausgabe einer Datei auf Kassette oder Diskette.

CLS [*# Stream-Nummer*]

Beispiel: Eingabe:
```
10 PAPER #3,6
20 CLS #3
run
```

Funktion: Der angegebene Bildschirmbereich wird gelöscht und mit der Farbe, die in PAPER vereinbart wurde, aufgefüllt.

Stream-Bildschirmbereich: Die Ziffern 0 bis 7 hinter dem Zeichen "#" kennzeichnen unterschiedliche Bildschirmbereiche (Fenster).

CONT

Funktion: Fortsetzen eines Programms, das mit zweimaligem Drücken der Taste [ESC] oder nach einer STOP-Anweisung unterbrochen wurde. Zwischen Abbruch und Fortsetzung dürfen Eingaben, die das Programm nicht verändern, getätigt werden.

COPYCHR$ (*# Stream-Nummer*)

Beispiel: Eingabe:
```
5 CLS 10 PRINT "1. Stelle"
20 LOCATE 1,1
30 A$=COPYCHR$(#0)
40 LOCATE 3,10:PRINT A$
run
```

Funktion: Ein Zeichen wird in den, mit Stream-Nummer angegebenen, Bildschirmbereich kopiert. Bei Nichterkennung des Zeichens erfolgt die Ausgabe eines Leerzeichens.

COS (*numerischer Ausdruck*)

Beispiel: Eingabe: `PRINT COS(pi/4)`
Anzeige: `0.707106781`

oder im Winkelmaß

Eingabe: `DEG PRINT COS(45)`
Anzeige: `0.707106781`

Funktion: Berechnet den COSinus des in Klammer stehenden numerischen Ausdrucks. Die Eingabe erfolgt standardmäßig im Bogenmaß, kann mit DEG im Winkelmaß vereinbart werden.

CREAL (*numerischer Ausdruck*)

Beispiel: Eingabe: `PRINT CREAL(PI)`
Anzeige: `3.14159265`

Funktion: Umwandlung des numerischen Ausdrucks in eine reelle Zahl.

CURSOR [*Systemschalter*] [,*Benutzerschalter*]

Beispiel: Eingabe:
```
10 CURSOR 1
20 A$=INKEY$:IF A$="" THEN 20
30 PRINT A$:CURSOR 0
run
```

Funktion: Ein-/Ausstellen des System-/Benutzerschalter für den Cursor.
System- und Benutzerschalter dienen beide zum Ein- und Ausschalten des Cursors. Dabei besitzt der Systemschalter die höhere Priorität, d.h., ist mit dem Systemschalter der Cursor eingeschaltet, kann er mit dem Benutzerschalter nicht mehr ausgeschaltet werden.
Werte für Systemschalter und Benutzerschalter:
Ein=1 Aus=0
Der Cursor ist sichtbar, wenn System- und Benutzerschalter auf 1 gestellt sind. Wird ein Parameter nicht angegeben, bleibt der Zustand des betroffenen Schalters erhalten; es muß mindestens ein Parameter eingegeben werden.
Bei der Anweisung INKEY$ ist im Normalfall der Cursor nicht eingeblendet, jedoch bei INPUT ist der Cursor zu sehen.

DATA *Liste von Konstanten*

Beispiel: Eingabe: `50 DATA 0,1,2,3...`

Funktion: Dem Programm werden Datenwerte zur Verfügung gestellt, die mit READ Variablen zugewiesen werden. Anschließend wird der Datenzeiger um eine Stelle auf der Datenliste verschoben. RESTORE setzt den Datenzeiger auf die erste Zeile der Datenliste (siehe READ, RESTORE).

DEC$ (*numerischer Ausdruck*, *Schablone*)

Beispiel: Eingabe: `PRINT DEC$(10 5,"$$########,.##")`
Anzeige: `$100,000.00`

Funktion: Der numerische Ausdruck wird als dezimale Zeichenfolge (String) wiedergegeben. Das Format ist durch die Schablone definiert.
Zulässige Zeichen für die Formatschablone:
+ - $ * # , .
Die Anwendung dieser Zeichen ist unter PRINT USING dargestellt.

DEF FN *Name* (*Formalparameter*) = *allgemeiner Ausdruck*

Beispiel: Eingabe:
```
10 DEF FNR(x) = 2*x*x-x/2
20 INPUT x
30 PRINT FNR(x)
run
```

Funktion: DEFinieren einer FuNktion, die nicht als Anweisung (Schlüsselwort) zur Verfügung steht. Das Benutzen erfolgt nur im Programm und in der Weise, wie z.B. die COS-Funktion in BASIC.

DEFINT *Liste von Buchstaben*

Beispiel: Eingabe: `DEFINT a,b,c,d` auch `DEFINT a-d`

Funktion: Alle Variablen, die mit aufgeführten Buchstaben beginnen, werden als ganzzahlige (INTeger) Werte festgelegt. Ohne Angabe des Variablentyps (|, %, $) erfolgt keine Vereinbarung, d.h., die Darstellung erfolgt mit reellen Zahlen. Dabei können auch einzelne Buchstaben oder -bereiche angegeben werden.

DEFREAL *Liste von Buchstaben*

Beispiel: Eingabe: `DEFREAL a,b,c,d` auch `DEFREAL a-d`

Funktion: Alle Variablen, die mit den aufgeführten Buchstaben beginnen, werden als reelle (englisch real) Zahlen vereinbart. Ohne Typkennzeichnung (|, %, $) wird wie bei DEFINT verfahren. Auch einzelne Buchstaben oder -bereiche können angegeben werden.

DEFSTR *Liste von Buchstaben*

Beispiel: Eingabe: `DEFSTR a-r`

Funktion: Alle Variablen, die mit den aufgeführten Buchstaben beginnen, werden als Zeichenketten (STRings) vereinbart. Ohne Kennzeichnung des Variablentyps (|, %, $) wird wie bei DEFINT verfahren.

DEG

Funktion: Umschalten auf Winkelmaß (englisch DEGree) Im Normalfall sind die Werte für die Funktionen SIN, COS, TAN, ATN auf Bogenmaß eingestellt. Mit DEG wird von Bogenmaß auf Winkelmaß umgeschaltet, so lange bis durch RAD, NEW, CLEAR, LOAD oder RUN die Umschaltung aufgehoben wird.

DELETE *Zeilenbereich*

Beispiel: mögliche Schreibweisen
Eingabe:
```
DELETE 80-400
DELETE -400
DELETE 80-
```

Funktion: (englisch delete - löschen) Es werden alle im Zeilenbereich liegenden Programmzeilen gelöscht.

Achtung: DELETE ohne Argument löscht das gesamte Programm!

DI

DI (englisch Disenable Interrupt - nicht erlaubte Unterbrechung) ermöglicht einen Programmablauf ohne Unterbrechung in den von DI und EI eingeschlossenen Programmzeilen.
Die Abarbeitung dieser Programmzeilen kann nur mit der [ESC]-Taste unterbrochen werden.

DIM *Liste von indizierten Variablen*

Beispiel: Variable Index

x	(0)	x(0)

```
DIM x(2,3)
```
zweidimensionales Array mit folgenden Variablen:
x(0,0), x(0,1), x(0,2), x(0,3)
x(1,0), x(1,1), x(1,2), x(1,3)
x(2,0), x(2,1), x(2,2), x(2,3)

Funktion: Die DIMension eines Arrays (Feld) wird festgelegt. Mit DIM wird ein Speicherbereich reserviert und der höchste Indexwert festgelegt. Ein Array enthält eine Variable, die für eine Gruppe von Elementen steht, die sich durch den Index unterscheiden. In BASIC ist die Indexoberegrenze festzulegen, ansonsten wird als Standardwert 10 angenommen. Der niedrigste Indexwert ist 0, demzufolge besitzt das erste Element in einem Array den Index 0. Das Variablenfeld x(I) ist eindimensional, da es ein Index enthält. Es können jedoch auch 2- oder n- dimensionale Variablenfelder vereinbart werden.

DRAW *x-Koordinate,y-Koordinate* [,*Farbstift*] [,*Farbstiftmodus*]

Beispiel: Eingabe: `DRAW x,y,f`

Funktion: Von der augenblicklichen Grafik-Cursorposition wird bis zur angegebenen absoluten x/y-Position auf dem Grafik-Bildschirm eine Linie gezeichnet. Die Farbstiftauswahl ist zwischen Farbstift 0 und 15 möglich. Die Auswahl des Farbmodus bestimmt, wie die Farbe des zu benutzenden Farbstiftes mit anderen Farben auf dem Bildschirm wirkt.

Farbstiftmodi:
0 = Normal (Punkt wird in eingestellter Farbe gesetzt) 1 = XOR (exklusiv ODER-Verknüpfung vorhandener Bildpunktfarbe mit neu festgelegter) 2 = AND (AND-Verknüpfung der Farben) 3 = OR (OR-Verknüpfung der Farben)

DRAWR *x-Versatz,y-Versatz* [,*Farbstift*] [,*Farbmodus*]

Beispiel: Eingabe: `DRAWR 100,0`

Funktion: Zeichnen einer Linie auf dem Grafik-Bildschirm von der augenblicklichen Position des Grafik-Cursors zur angegebenen relativen x/y-Position, d.h. versetzt um x und y Bildpunktpositionen. Die Farbstiftauswahl darf zwischen 0 und 15 erfolgen. Die Auswahl des Farbmodus ist wie unter DRAW.

EDIT *Zeilennummer*

Beispiel: Eingabe: `EDIT 10`

Funktion: EDIT dient zur Korrektur einer Programmzeile, die durch Angabe der Zeilennummer ausgewählt wird. Der Cursor erscheint auf der ersten Position der Programmzeile. Mit den Cursortasten kann er auf die Fehlerstelle bewegt und die entsprechende Korrektur vorgenommen werden.

EI

Funktion: Setzt die Anweisung DI zurück, wodurch Unterbrechungen des Programms wieder möglich sind (englisch Enable Interrupt - erlaubte Unterbrechung). Wurde in einer Unterbrechungssubroutine diese Unterbrechungsfunktion (EI) abgeschaltet, ist sie automatisch beim Erreichen des RETURNs am Ende des Unterprogramms wieder eingeschaltet.

ELSE (siehe IF)

END

Beispiel: Eingabe: `300 END`

Funktion: Die Anweisung END (Ende) beendet den Programmablauf. Anschließend erfolgt die Umschaltung auf Direkt-Mode. END steht in der letzten Programmzeile bzw. am Programmende und kann beliebig oft auftreten.

ENT *Hüllkurvennummer* [,*Hüllkurvenabschnitt*[],*Hüllkurvenabschnitt*] [,*Hüllkurvenabschnitt*][,*Hüllkurvenabschnitt*] [,*Hüllkurvenabschnitt*]

Beispiel: Eingabe:
```
10 ENT 2,8,-30,10,10,30,10
20 SOUND 1,400,200,10,1
run
```

Funktion: (Tonhüllkurve - englisch ENvelope Tone)
Die Tonhüllkurve für die SOUND-Anweisung wird definiert. Ihre Hüllkurvennummer (von 1 bis 15) ist in der SOUND-Anweisung anzugeben. Mit negativem Wert (-1 bis -15) der Hüllkurvennummer wiederholt sich die Hüllkurve bis ans Ende der SOUND-Anweisung.

Tonhüllkurve:
Zur Tonerzeugung sind im Computer drei Tongeneratoren für periodische Schwingungen und ein Tongenerator für rauschähnliche Signale enthalten. Der Ton umfaßt sinusförmige Schwingungen. Zu den Eigenschaften des Tons gehören Höhe, Stärke und sein Klangspektrum. Die Tonhöhe wird aus der Anzahl der Schwingungen pro Sekunde bestimmt.
Hohe Schwingungszahlen - hohe Töne, niedrige Schwingungszahlen - tiefe Töne.

Mit der ENT-Anweisung kann für die Dauer einer Note die Art der Schwingung festgelegt werden, d.h. die bestimmte Form der Schwingung in einer Periode. Die Tonhüllkurve beeinflußt nicht die Dauer einer Note. Ist die Tonhüllkurve vor der Note zu Ende, bleibt die zuletzt eingestellte Tonhöhe für den Rest der Zeit erhalten. Ist sie länger, so entfallen die letzten Abschnitte. Beim Neudefinieren einer Tonhüllkurve wird der alte Wert gelöscht.

Hüllkurvenabschnitt:
Es können maximal fünf Abschnitte definiert werden. Die Angabe eines Hüllkurvenabschnittes ohne Hüllkurvennummer bewirkt das Löschen der vorherigen Werte.
Jeder Hüllkurvenabschnitt kann durch zwei oder drei Parameter definiert werden. Für drei Parameter ist zu definieren:
Schrittanzahl, Schrittweite, Schrittzeit

Schrittanzahl:
Wertebereich von 0 bis 239
Die Schrittanzahl legt die Anzahl der Tonhöhenstufen fest. In 5 Sekunden sollen z.B. 5 Tonhöhenschritte durchlaufen werden, so ist die Schrittanzahl 5 anzugeben.

Schrittweite:
Die Schrittweite bestimmt den Abstand zwischen den einzelnen Tonhöhenschritten. Er kann zwischen -128 und 127 liegen. Die kürzeste Schrittweite beträgt 0.
Dabei nimmt die Tonhöhe mit negativen Schritten zu und mit positiven ab. Im Anhang E sind die Tonperioden aufgeführt.

Schrittzeit:
Die Schrittzeit legt den Zeitabstand zwischen den einzelnen Tonhöhenschritten fest. Als maximaler Wert ist 2,55 Sekunden möglich (0 wird als 256 angenommen).
Wertebereich: 0 bis 255
Einheit: 0,01 Sekunden

Achtung: Die Dauer aller Schritte (addiert) sollte gleich der Dauer des Tons sein (Summe aller Schritte=Tondauer). Die Tonausgabe wird sonst beendet, bevor alle Tonhöhen-Variationen durchlaufen sind.

Mit zwei Parametern gilt für ENT: Tonperiode, Schrittzeit

Tonperiode:
Es wird der neue absolute Wert für die Tonperiode angegeben.

Schrittzeit:
wie unter Hüllkurvenabschnitt

ENV *Hüllkurvennummer* [,*Hüllkurvenabschnitt*][,*Hüllkurvenabschnitt*]
[,*Hüllkurvenabschnitt*][,*Hüllkurvenabschnitt*]
[,*Hüllkurvenabschnitt*]

Beispiel: Eingabe:
```
10 ENV 2,20,-2,10,20,1,10
20 SOUND 2,150,400,20,1
run
```

Funktion: Die Lautstärkehüllkurve (englisch ENvelope Volume) für die SOUND-Anweisung wird definiert.
Wertebereich der Hüllkurvennummer: 0 bis 15, wird in die SOUND-Anweisung eingetragen.
Der Hüllkurvenabschnitt kann aus zwei oder drei Parametern bestehen. Das sind:

Schrittanzahl:
Bestimmt die Anzahl der Lautstärkestufen des Tones innerhalb eines Hüllkurvenabschnittes.
Wertebereich: 0 bis 127

Schrittweite:
Definieren von Lautstärkeschritten, die dann in der SOUND-Anweisung zu hören sind. Der Abstand zum jeweiligen Lautstärkeschritt (Stufe) kann zwischen 0 und 15 liegen (Schrittdifferenz).
Die Lautstärke geht nach jeweils 15 Schritten auf 0 zurück.
Wertebereich der Schrittweite: -128 bis 127

Schrittzeit:
Legt den Zeitabstand zwischen den einzelnen Lautstärkeschritten (Stufen) fest. Wertebereich: 0 bis 255, 0=256 Minimale Zeit: 0,01 s Maximale Zeit: 2,56 s

Für zwei Parameter gilt:

Registerwert:
Der angegebene Wert wird an das Lautstärkeregister übergeben (Ton-Chip-Bauteil).
Hüllkurvenperiode:
Der angegebene Wert wird an das Hüllkurvenregister des Ton-Chips übergeben.

Sollten die Hardware-Register direkt gesetzt werden, müssen ausreichend Kenntnisse in der Hardware (Gerätetechnik) vorhanden sein (vgl. System-Handbuch).

Hinweise: Der erste Schritt der Lautstärkehüllkurve wird sofort ausgeführt. Die Summe aller Schrittzeiten sollte nicht länger als die Angabe des Dauer-Parameters der SOUND-Anweisung sein (beim Nichteinhalten siehe ENT-Befehl). Wurde eine größere Zeitdauer in der SOUND-Anweisung eingetragen, bleibt die Lautstärke nach dem Abarbeiten aller Stufen auf der zuletzt eingestellten erhalten. Das Definieren einer Lautstärkehüllkurve ohne die dazugehörige Hüllkurvennummer bewirkt das Löschen der vorherigen Werte.

EOF

Beispiel: Eingabe:
```
10 OPENIN "Dat1.bas"
20 WHILE NOT EOF
30 LINE INPUT #9,A$
40 PRINT A$
50 WEND
60 CLOSEIN
70 END
```

Funktion: (englisch End Of File - Dateiende)
Überprüft, ob beim Einlesen einer Datei von Kassette oder Diskette das Dateiende erreicht ist. Wurde keine Datei eröffnet, oder die Datei ist zu Ende, wird der Wert -1 (richtig), sonst der Wert 0 (falsch) ausgegeben.

ERASE *Liste von Variablen*

Beispiel: Eingabe:
```
10 DIM x(50),A$(200)
20 ERASE x,A$
```

Funktion: (englisch erase - löschen)
Der Inhalt eines Arrays wird gelöscht. In Zeile 10 sind die Variablen x und A$ dimensioniert als eindimensionales Array (Feld). Der Speicher wird reserviert. Zeile 20 gibt den reservierten Speicher wieder frei.

ERL

Beispiel: Eingabe:
```
5 ON ERROR GOTO 20
10 GOTO 100
20 PRINT "Fehler in ";ERL
30 END
run
```

Funktion: Es wird die Zeilennummer (englisch line number) angegeben, in welcher der letzte Fehler (englisch error) aufgetreten war. Im Beispiel ist es die Zeile 10.

ERR

Beispiel: Eingabe: `GOTO 200`
Anzeige:
```
Line does not exist
Ready
```
Eingabe: `PRINT ERR`
Anzeige: `8`

Funktion: Angabe der Zahl, unter der die letzte Fehlermeldung in der Fehlerliste aufgeführt ist (siehe Fehlerliste). Dem Fehler 8 ist die Fehlermeldung 'Line does not exist' zugeordnet.

ERROR *ganzzahliger Ausdruck*

Beispiel: Eingabe: `IF A=1 THEN 80:ELSE ERROR 17`

Funktion: Aufruf des im Ausdruck angegebenen Fehlers (laut Fehlerliste). Der so aufgerufene Fehler aus einem Programm wird genauso behandelt, als wäre der Fehler vom BASIC-Interpreter aufgerufen, um anschließend die Fehlerbehandlungsroutine abzuarbeiten. Mit ERROR werden auch die zugehörigen Werte für ERR und ERL ausgegeben. Wird ERROR mit den Zahlen 33 bis 255 benutzt, können eigene Fehlermeldungen definiert werden.

EVERY *Zeitintervall* [,*Zeitgeber*] **GOSUB** *Zeilennummer*

Beispiel: Eingabe:

```
10 EVERY 100,1 GOSUB 50
20 A$=INKEY$
30 IF A$="" THEN 20
40 END
50 PRINT "Hallo"
60 RETURN
run
```

Funktion: Aufruf eines BASIC-Unterprogramms in regelmäßigen Zeitabständen. Die Einheiten des Zeitintervalls sind in Fünfzigstelsekunden (0,02 Sekunden) angegeben.

Zeitgeber:
Bereich von 0 bis 3; 0 ist Standardzeitgeber Zeitgeber 0 hat die niedrigste und Zeitgeber 3 die höchste Priorität. Jedem Zeitgeber kann ein Unterprogramm zugeordnet werden.

EXP (*numerischer Ausdruck*)

Beispiel: Eingabe: PRINT EXP(5.2)
Anzeige: 181.272242

Funktion: Die Eulersche Konstante e mit Näherungswert 2,7182818 wird mit dem in Klammern stehenden Ausdruck potenziert.

FILL *Farbstift*

Beispiel: Eingabe: FILL 2

Funktion: Ein beliebiger Bereich des Grafik-Bildschirms wird mit dem genannten Farbstift (Wertebereich 0 bis 15) ausgefüllt. Die Linien des zu begrenzenden Bereiches werden mit dem durch GRAPHICS PEN festgelegten Farbstift oder mit einem anderen Farbstift, der auch zum Ausfüllen verwendet wird, gezogen. Der Farbstift beginnt an der Grafik-Cursorposition. Befindet sich der Cursor am Rand, erfolgt keine Reaktion.

FIX (*numerischer Ausdruck*)

Beispiel: Eingabe: PRINT FIX(5.9999)
Anzeige: 5

Funktion: Abschneiden der Nachkommastellen des in Klammern stehenden Ausdrucks. Die vor dem Komma stehende Zahl bleibt als ganze Zahl erhalten.

FN (siehe DEF FN)

FOR *einfache Variable=Startwert* **TO** *Endwert* [**STEP** *num. Ausdruck*]

Beispiel: Eingabe:
```
10 MODE 0
20 FOR i=32 TO 255
30 FOR a=1 TO 3
40 PRINT CHR$(i);
50 PEN a
60 NEXT a,i
run
```

Funktion: Ein Programmteil, der zwischen FOR und NEXT steht, wird so oft wiederholt, bis der Endwert erreicht wurde, deshalb auch der Name FOR-NEXT-Schleife. In welcher Schrittweite die Schleife abgearbeitet wird, legt die Anweisung STEP fest. NEXT beendet die mit FOR begonnene Schleife. Bei einer Variablen kann NEXT allein stehen. Sind mehrere Schleifen ineinander verschachtelt, muß die innere Schleife, also diese Variable, zuerst genannt werden (Variable a). Anschließend wird die äußere Schleife (Variable i) geschlossen.

FRAME

Funktion: (englisch frame flayback - Strahlrücklauf)
Der Schreibvorgang auf dem Bildschirm wird mit dem Strahlenrücklauf synchronisiert. Damit wird eine ruhigere Bewegung von Zeichen und Grafik erreicht, also kein Flimmern und Flackern auf dem Bildschirm.

FRE (*numerischer Ausdruck*)
FRE (*String*)

Beispiel: Eingabe:
```
PRINT FRE(0)
PRINT FRE("")
```

Funktion: Angabe des freien Speicherbereiches, der vom BASIC-Interpreter nicht benötigt wird. Mit der Eingabe FRE("") wird ein Sammeln auch kleinster Speicherbereiche im Stringspeicher eingeleitet, bevor der Wert des freien Bereiches angezeigt wird.

GOSUB *Zeilennummer*

Beispiel: Eingabe: `GOSUB 900:CLS...`

Funktion: (englisch go subroutine - gehe zum Unterprogramm)
Sprung zum Unterprogramm, das ab der angegebenen Zeilennummer beginnt. Das Unterprogramm muß enden mit der Anweisung RETURN (kehre zurück), dadurch arbeitet das Programm die nachfolgende Anweisung von GOSUB ab.

GOTO *Zeilennummer*

Beispiel: Eingabe: `GOTO 20`

Funktion: Sprung zur angegebenen Zeilennummer.

GRAPHICS PAPER *Farbstift*

Beispiel: Eingabe:
```
10 MODE 0:MASK 10
20 GRAPHICS PAPER 7:DRAW 100,640
run
```

Funktion: Festlegen der Hintergrundfarbe (Farbstift 0 bis 15), auf der die Zeichnungen dargestellt werden sollen. Bei durchgehenden Linien ist die Hintergrundfarbe nicht sichtbar. Mit MASK wird im Beispiel eine durchbrochene Linie gezeichnet, um die Hintergrundfarbe 7 zu sehen.
Die eingestellte GRAPHICS PAPER-Farbe ist die Hintergrundfarbe von Zeichen, die durch die Anweisung TAG auf dem Grafikbildschirm ausgegeben werden. Sie ist auch die Standardfarbe, wenn der Grafikbildschirm mit CLG gelöscht wird.

GRAPHICS PEN [*Farbstift*] [,*Hintergrundmodus*]

Beispiel: Eingabe:
```
10 MODE 0
20 GRAPHICS PEN 7
30 MOVE 150,0
40 DRAW 150,300
50 MOVE 630,0
60 FILL 7
```

Funktion: Festlegen des Farbstiftes, mit dem Linien und Punkte gezeichnet werden.
Farbstiftbereich: von 0 bis 15
Hintergrundmodus kann
1= nicht transparenter Hintergrund
oder
0= transparenter (durchsichtiger) Hintergrund
sein.

Transparenter Modus: Der Hintergrund, der mit der TAG-Anweisung geschriebenen Zeichen bzw. Linien und Punkte auf dem Bildschirm, erscheint in der eingestellten GRAPHICS PAPER-Farbe.

Bei der Eingabe dieser Anweisung darf ein Parameter fehlen, aber nicht beide (siehe Fehler 22 Anhang A), denn dadurch ändert sich nichts an der Einstellung der Farben.

HEX$ (vorzeichenloser ganzzahliger Ausdruck [,Feldbreite])

Beispiel: Eingabe: `PRINT HEX$(127,6)`
Anzeige: `00007F`

Funktion: Umwandlung des in Klammern stehenden Ausdrucks in hexadezimalen String. Die Anzahl der Zeichen gibt die Feldbreite an, sie liegt zwischen 0 und 16. Ist die Anzahl der Stellen kleiner, so werden Vornullen eingetragen. Wird sie größer als vereinbart, so wird diese hexadezimale Zahl nicht gekürzt. Die Ausgabe erfolgt dann in voller Stelligkeit.
Wertebereich des vorzeichenlosen ganzzahligen Ausdrucks: von -32768 bis 65535
Z.B.: Eingabe: PRINT HEX$(-32768)
Anzeige: 8000

HIMEM

Beispiel: Eingabe: PRINT HIMEM
Anzeige: 42619

Funktion: Angabe der höchsten (englisch Highest) Speicheradresse, die vom BASIC-Interpreter belegt wird. Mit der Anweisung MEMORY kann sie neu festgelegt werden.

IF *logischer Ausdruck* **THEN** *Anweisung* [**ELSE** *Anweisung*]

Beispiel: Eingabe:
```
IF A=1 THEN 10
IF A 1 THEN PRINT "falsch"
IF A^12 THEN PRINT "richtig"
```

Funktion: Der Computer wird nach einer bestimmten Bedingung (nach IF) abgefragt. Ist diese erfüllt, erfolgt die Ausführungsanweisung nach THEN (dann). Bei Nichterfüllung wird die ELSE-Anweisung (ansonsten) abgearbeitet. Fehlt die ELSE-Anweisung, wird in der nächsten Programmzeile weitergearbeitet. IF-Anweisungen können ineinander verschachtelt und dürfen nicht länger als eine Zeile sein.

Folgende Vergleichsoperatoren sind möglich:

Symbol	Bedeutung
=	gleich
>	größer als
<	kleiner als
>=, =>	größer als oder gleich
<=, =<	kleiner als oder gleich
<>, ><	ungleich

INK Farbstift,Farbe [,Farbe]

Beispiel: Eingabe:
```
INK 1,6
INK 2,3,4
```

Funktion: Zuordnung der Farbe zu einem Farbstift. Einem Farbstift können eine oder zwei Farben zugeordnet werden. Sind zwei Farben angegeben, erfolgt ein Blinken durch Wechseln der beiden Farben. Als Farbstiftwerte sind ganze Zahlen von 0 bis 15 zulässig, diese werden auch in den PEN- und PAPER-Anweisungen eingesetzt. Das Zuordnen Farbstift-Farbe ist in Abhängigkeit vom eingestellten Bildschirmmodus zu berücksichtigen (siehe Abschnitt 3.10.).

INKEY (*ganzzahliger Ausdruck*)

Beispiel: Eingabe: 10 CLS 20 PRINT "Taste druecken" 30 IF INKEY(53)=0 THEN PRINT"f-richtig": ELSE 30

Funktion: Es erfolgt die Angabe, welche Taste der Tastatur betätigt wurde. Dabei wird die Tastatur im Abstand von 0,02 Sekunden abgefragt. Bei dieser Auswertung wird Groß- und Kleinschreibung nicht berücksichtigt. In Zeile 30 wird geprüft, ob die Taste [F] gedrückt wurde. Es wird auch berücksichtigt, wenn die [SHIFT]- oder [CTRL]-Taste betätigt wurden. An folgenden Zahlenwerten, die der Computer ausgibt, ist ersichtlich, in welcher Stellung sich die Tasten befinden:

Computer-Ausgabewert	Taste [SHIFT]	Taste [CTRL]	beliebige Taste
-1	ja/nein	Ja/nein	nicht gedrückt
0	nicht gedrückt	nicht gedrückt	gedrückt
32	gedrückt	nicht gedrückt	gedrückt
128	nicht gedrückt	gedrückt	gedrückt
160	gedrückt	gedrückt	gedrückt

Das Beispiel zeigt in Zeile 30 die Abfrage nach der [F]-Taste. Dabei ist der in Klammern stehende Wert die Tastennummer (siehe Anhang C). Für [F] ist sie 53. Die Bedingung für diese Taste ist aus der obigen Darstellung zu entnehmen.
Der Computerausgabewert 0 sagt aus:

[SHIFT]-Taste nicht gedrückt
[CTRL]-Taste nicht gedrückt
beliebige Taste, mit 53 definiert, ist gedrückt.

INKEY$

Beispiel: Eingabe:
```
10 a$=INKEY$:IF a$="" THEN 10
20 IF a$="f" OR a$="F" THEN 30:ELSE 10
30 PRINT "Taste richtig" run
```

Funktion: Es wird der Wert der gedrückten Taste, die für einen String steht, vom Computer zurückgegeben. Erfolgt keine Betätigung, wird ein Leerstring (Leerzeichen) angegeben. Im Unterschied zu INKEY wird hier Groß- und Kleinschreibung unterschieden. Im Beispiel wird dies in Zeile 20 berücksichtigt.

INP (*Schnittstellenadresse*)

Beispiel: Eingabe: `PRINT INP(&FF77)`
Anzeige: `255`

Funktion: Es wird der Eingabewert von der Eingabeschnittstelle (englisch INPut, I/O-Port (Kanal)) zurückgegeben.

INPUT [*#Stream*,] [;] [*Text Trennzeichen*] [*Variablenliste*]

Beispiel: Eingabe: `INPUT "Drei Zahlen eingeben";A,B,C`

Funktion: Daten werden vom angegebenen Stream (Eingabegerät) aufgenommen (ohne Angabe wird Stream #0 angenommen). Das erste Semikolon (;) ist wahlweise anzuwenden. Es verhindert die Eingabe auf einer neuen Zeile. Das Trennzeichen hinter Text kann ein Semikolon sein, dann erscheint nach dem auszugebenden Text ein Fragezeichen. Bei einem Komma hinter Text wird das Fragezeichen unterdrückt. Erfolgt eine falsche Eingabe des Variablentyps, z.B. Eingeben eines o statt einer 0, kommt in BASIC die Ausschrift:

```
? REDO from Start
(Eingabe wiederholen)
```

Die Textanzeige und Eingabe werden wiederholt. Die Eingaben werden im Unterschied zu INKEY mit [RETURN] abgeschlossen.

INSTR ([*ganzzahl.Ausdruck*,]*durchsuchter String*,*gesuchter String*)

Beispiel: Eingabe: `PRINT INSTR(1,"Apfelbaum","baum")`
Anzeige: `6`

Funktion: Es wird die Positionsnummer zurückgegeben, ab der der gesuchte String (Baum) im durchsuchten String (Apfelbaum) auftritt.
Der ganzzahlige Ausdruck kann zwischen 1 und 255 liegen. Er gibt die Position an, wo die Suche beginnen soll.
Bei erfolgloser Suche wird der Wert 0 ausgegeben.

INT (*numerischer Ausdruck*)

Beispiel: Eingabe: `PRINT INT(-15.897)`
Anzeige: `-16`

Funktion: Es erfolgt ein Abrunden der angegebenen Zahl auf die nächstkleinere ganze Zahl (englisch INTeger). Für positive Zahlen liegen die Werte wie bei der FIX-Funktion, für negative Zahlen sind die Werte um 1 kleiner als bei FIX.

JOY (*ganze Zahl*)

Beispiel: Eingabe: `JOY(0)=4`

Funktion: Die in Klammern stehende ganze Zahl (0 oder 1) bestimmt, welcher Joystick verwendet wird (Joystick 0 oder 1). Diese Anweisung liefert eine Zahlenkombination mit einer Stelligkeit von acht (acht Bit), die den Bewegungen des Joysticks entspricht. Diese acht Bit besitzen folgende Zuordnung:

Bewegungs-richtung	JOY-Anweisung		Tastennummern		
	Bit-Nr.	Wertzu-ordnung dezimal	Erster Joystick	Zweiter Joystick	Taste
aufwärts	0	1	72	48	'6'
abwärts	1	2	73	49	'5'
links	2	4	74	50	'R'
rechts	3	8	75	51	'T'
Feuer 1	4	16	76	52	'G'
Feuer 2	5	32	77	53	'F'

Die Bits 6 und 7 werden hier nicht ausgewertet.

Tabelle 4: Durch die JOY-Anweisung gelieferte Werte

Wie im Beispiel aufgeführt, gibt der erste Joystick (0) den Wert 4 an. Dieser Wert zeigt die letzte Bewegung des Joysticks zum Zeitpunkt der Abfrage an. Die "4" sagt aus, daß Joystick 0 nach links bewegt wurde. Die Abfrage der Tastatur erfolgt 50 Mal pro Sekunde.

Zu beachten ist, daß der zweite Joystick (JOY(1)) wie die entsprechenden Tasten behandelt wird. Der Computer unterscheidet nicht Joystick 1 oder Tastatur beim Abfragen, deshalb kann hier Joystick oder Tastatur benutzt werden.

KEY *ganze Zahl*, *String*

Beispiel: Eingabe: `KEY 141,"LIST"+CHR$(13)`

Funktion: Ein String wird einem Erweiterungszeichen, das durch die ganze Zahl definiert ist, zugeordnet. Die ganze Zahl hat einen Wertebereich von 128 bis 132 und 138 bis 159, d.h. es gibt 27 Erweiterungszeichen (sinngemäß Notizblätter) im Computer. Die folgende Tabelle gibt einen Überblick der Erweiterungszeichen und der Standardeinstellung:

ganze Zahl (Erweiterungszeichen) Nr.	Grundeinstellung	
	Zeichen	ASCII-Wert
(128)	0	&30
(129)	1	&31
(130)	2	&32
(131)	3	&33
(132)	4	&34
(138)	.	&2E
(139)	RETURN	&0D
(140)	RUN"[RETURN]	&52 &55 &4E &22 &0D

Tabelle 5: Erweiterungszeichen und Grundeinstellung

Die Erweiterungszeichen mit den Tastenwerten 141 bis 159 besitzen in der Grundeinstellung den Wert Null. Durch die Anweisung KEY wird ein Erweiterungszeichen gewählt, das mit KEY DEF einer Taste zugeordnet wird. Es können insgesamt 120 Zeichen den Tasten zugeordnet werden. Wird diese Anzahl überschritten, kommt es zur Fehlermeldung 5 "Improper Argument!" (Argument ist nicht gültig, siehe Anhang A).

KEY DEF *Tastennummer*,*Dauerfunktion* [,*Normalzeichen* [,SHIFT [,CONTROL]]]

Beispiel: Eingabe: `KEY 159,"BORDER6"+CHR$(13)`
`KEY DEF 6,1,159`

Funktion: Mit KEY wurde einem Erweiterungszeichen ein String zugeordnet. KEY DEF legt diesen String auf eine Taste, so daß nach dem Drücken dieser Taste die gewünschten Werte vom Computer ausgegeben werden. Soll für eine Taste eine Mehrfachbelegung (dreifach) erfolgen, müssen die Parameter Normal, SHIFT und CONTROL mit den verwendeten Erweiterungszeichen angegeben werden.

Dauerfunktion:
Die Dauerfunktion einer Taste wird mit 1 eingeschaltet und mit 0 abgeschaltet. Die Geschwindigkeit der Wiederholung eines Zeichens in der Funktion kann mit SPEED KEY verändert werden.

Im Beispiel wurde der [ENTER]-Taste (Tastennummer 6 siehe Anhang C) der auf dem Erweiterungszeichen 159 stehende String zugeordnet.

LEFT$ (*String*,*Länge*)

Beispiel:
```
10 FOR I=1 TO 5
20 PRINT LEFT$ ("MINNA",I)
30 NEXT
run
```

Funktion: Der String wird in der festgelegten Länge (Anzahl der Zeichen von 0 bis 255) wiedergegeben. Es wird links (englisch left) beim Ausgeben begonnen und jede Zeile wird um ein Zeichen verlängert. Ist der String kürzer als die Länge, so wird der String so oft wiederholt, bis die Länge erreicht ist.

LEN (*String*)

Beispiel: Eingabe: `a$="Taste" PRINT LEN(a$)`
Anzeige: `5`

Funktion: Die Anzahl der Zeichen in einem String (auch Leerzeichen) wird angegeben (englisch LENgth).

LET *Variable=Ausdruck*

Beispiel: Eingabe: `LET A=4.20 PRINT A`
Anzeige: `4.2`

Funktion: Der angegebenen Variablen wird ein Wert zugewiesen. LET kann auch weggelassen werden. So erfolgt die Schreibweise A=4.20.

LINE INPUT [*#Stream*,][;][*Ausgabetext Trennzeichen*] *Stringvariable*

Beispiel: Eingabe:
```
LINE INPUT "Text schreiben";a$
PRINT a$
```

Funktion: Einlesen eines Textes vom angegebenen Stream. Wird kein Stream angegeben, ist Stream #0 vereinbart. Das erste Semikolon kann wahlweise verwendet werden. Es verhindert den Zeilenvorschub nach erfolgter Eingabe. Als Trennzeichen sind Komma oder Semikolon einsetzbar. Das Komma bewirkt hinter dem Ausgabetext ein Fragezeichen und das Semikolon unterdrückt es. Mit dem Betätigen der Taste [RETURN] wird die Eingabe von der Tastatur beendet. Das Einlesen von Kassette oder Diskette (Stream #9) wird beendet, wenn der Bereich der String variablen (255 Zeichen) gefüllt oder das Dateiende erkannt ist.

LIST [*Zeilenbereich*] [,*#Stream*]

Beispiel: Eingabe - Möglichkeiten:
```
LIST 400-800,#1
LIST
LIST 10
LIST -10
LIST 10-
```

Funktion: Auslisten der angegebenen Programmzeilen im genannten Bildschirmbereich (Stream). Ohne Streamangabe wird Stream #0 angenommen (Stream #8 = Drucker). Das Auslisten kann durch einmaliges Drücken der Taste [ESC] angehalten, mit Drücken einer anderen Taste (Leertaste) fortgesetzt werden. Durch zweimaliges Betätigen der [ESC]-Taste bricht das Auslisten ab.

LOAD *Dateiname* [,*Adresse*]

Beispiel: Eingabe: `LOAD "Beispiel.bin",&2AF8`

Funktion: Ein BASIC-Programm wird in den Speicher geladen, dabei wird ein im Speicher befindliches Programm gelöscht. Mit der Angabe der Adresse kann eine binäre Datei (Datei besteht z.B. aus binären oder auch dualen Daten für Zeichendarstellung) auf eine andere Speicheradresse, als unter der sie abgespeichert wurde, geladen werden. Geschützte Programme können nur mit RUN oder CHAIN geladen werden.

LOCATE [*#Stream*,] *x-Koordinate*, *y-Koordinate*

Beispiel: Eingabe: LOCATE 1,1

Funktion: Der Text-Cursor wird auf die Stelle des angegebenen Bildschirmbereiches gesetzt, die durch die x/y-Koordinaten positioniert ist. Die Position 1,1 ist die obere, linke Ecke des angegebenen Bildschirmbereiches.

LOG (*numerischer Ausdruck*)

Beispiel: Eingabe: PRINT LOG(777)
Anzeige: 6.65544035

Funktion: Es wird der natürliche Logarithmus des in Klammern stehenden numerischen Ausdrucks (größer 0) berechnet.

LOG10 (*numerischer Ausdruck*)

Beispiel: Eingabe: PRINT LOG10(777)
Anzeige: 2.89042102

Funktion: Es wird der Logarithmus zur Basis 10 des in Klammern stehenden numerischen Ausdrucks (größer 0) berechnet.

LOWER$ (*String*)
Beispiel: Eingabe: PRINT LOWER$("KLEINE BUCHSTABEN")
Anzeige: kleine buchstaben

Funktion: Gibt alle im String angegebenen Buchstaben in Kleinbuchstaben wieder.

MASK [*ganzzahliger Ausdruck*] [,*erster Punkt*]

Beispiel: Eingabe: MASK 2 (5-x)

Funktion: Es wird festgelegt, mit welcher Maske (Schablone) das Zeichnen von Linien erfolgen soll. Der Wertebereich des ganzzahligen Ausdrucks liegt zwischen 0 und 255. Dieser Wert ist dual und stellt in der Zeichenmatrix (8x8 Zeichen) eine Zeile dar, d.h. 8 Punkte sind in einer Linie gesetzt. Als erster Punkt wird vereinbart, ob eine Linie den ersten Punkt zu zeichnen hat (1) oder nicht (0).
Beim Benutzen dieser Anweisung darf ein Parameter weggelassen werden, nicht beide, da es sonst zur Fehlerausschrift 22 kommmt (Operand missing, siehe Anhang A).

MAX (*Liste numerischer Ausdrücke*)

Beispiel: Eingabe: `PRINT MAX(1,2,3,4,5,900)`

Anzeige: `900`

Funktion: Der größte (maximale) Wert der in Klammern stehenden numerischen Ausdrücke wird zurückgegeben.

MEMORY *Adresse*

Beispiel: Eingabe: `MEMORY &30CC`

Funktion: Der Speicherbereich (englisch memory), der dem BASIC-Interpreter zur Verfügung steht, also HIMEM, wird verändert, indem die höchste Speicheradresse angegeben wird.

MERGE *Dateiname*

Beispiel: Eingabe: `MERGE "Neutest.bas"`

Funktion: Ein neues Programm wird zu einem bereits im Speicher existierenden dazugeladen, also gemischt (englisch merge - verschmelzen). Besitzen beide Programme die gleichen Zeilennummern, so wird das alte Programm vom neuen überschrieben.

MID$ (*String*,*Startstelle* [,*Länge des Teilstrings*])

Beispiel: Eingabe:
```
a$="Tannenbaum"
PRINT MID$(a$,7,4)
```
Anzeige: `baum`

Funktion: MID$(a$,x,y)
Es werden y Zeichen von a$, beginnend mit der x-ten Stelle wiedergegeben. Das heißt, aus einem vorhandenen String wird ein Teilstring (Anzahl der Zeichen) ab der angegebenen Stelle herausgelöst.
Wertebereich der Startstelle: 0 bis 255
Wertebereich des Teilstrings: 0 bis 255
Ist die Startstelle größer gewählt als der vorhandene String, so werden Leerzeichen ausgegeben.

MID$ (*Stringvariable*,*Einfügestelle* [,*neue Stringlänge*])
= *neuer String*

Beispiel: Eingabe:
```
10 a$="Laubbaum"
20 MID$(a$,1,4)="Birn"
30 PRINT a$
run
```
Anzeige: `Birnbaum`

Funktion: In einem vorhandenen String wird ein neuer, ab Einfügestelle, mit Angabe der Stelligkeit des neuen Strings hinzugefügt. Der neue String überschreibt vorhandenen Text ab der Einfügestelle mit der Länge(4), die im Parameter neue Stringlänge angegeben ist.

MIN (*Anzahl numerischer Ausdrücke*)

Beispiel: Eingabe: `PRINT MIN(1,2,3,9,12,20)`
Anzeige: `1`

Funktion: Es wird der kleinste numerische Wert aus der Anzahl numerischer Ausdrücke angezeigt.

Dividend **MOD** *Divisor*

Beispiel: Eingabe: `PRINT 10 MOD 8`
Anzeige: `2`

Funktion: Es wird der (gerundete) Rest (englisch modulo) vom Ergebnis nach der Division (Divident durch Divisor) ausgegeben. Im Beispiel heißt dies: 10 dividiert durch 8 ist gleich 1. Dabei wird der Rest 2 angezeigt.

MODE *ganzzahliger Ausdruck*

Beispiel: Eingabe: `MODE 1`

Funktion: Der Bildschirmmodus (0,1,2) und der Farbstift 0 werden eingestellt. Alle Fenstereinstellungen für Text und Grafik werden auf den Gesamtbildschirm gelegt. Der Text- und der Grafik-Cursor sind auf die Ausgangsposition gesetzt.

MOVE *x-Koordinate,y-Koordinate* [,*Farbstift*] [,*Farbmodus*]

Beispiel: Eingabe:

```
10 MODE 0:TAG
20 x=RND*400:y=RND*430
30 MOVE x,y
40 DRAW x+10,y
50 GOTO 20
run
```

Funktion: Es wird der Grafik-Cursor auf den absoluten, durch x/y definierten, Punkt bewegt (englisch move). Wahlweise können der Farbstift (Werte von 0 bis 15) und der Farbmodus angegeben werden, um die Farbe des Farbstiftes zu bestimmen und deren Zusammenwirken mit den anderen Farben auf dem Grafik-Bildschirm.

Farbmodi:

0 = Normal
1 = XOR (exklusives ODER)
2 = AND
3 = OR

MOVER *x-Versatz*,*y-Versatz* [,*Farbstift*] [,*Farbmodus*]

Beispiel: Eingabe: MOVER -16,16

Das vorherige Beispiel wird geändert in

Eingabe:
```
30 MOVER x+200,,y
40 DRAW x+10,y+10
run
```

Funktion: Der Grafik-Cursor wird von der augenblicklichen Cursorposition zu dem, durch den x/y-Versatz bestimmten, relativen Koordinatenpunkt bewegt. Der Farbstift kann wahlweise angegeben werden (Wertebereich 0 bis 15). Der Farbmodus, der auch wahlweise benutzt wird, bestimmt, wie der zu zeichnende Farbstift mit allen anderen Farben des Bildschirms reagiert.

Farbmodi:

0 = Normal
1 = XOR (exklusives ODER)
2 = AND
3 = OR

NEW

Funktion: Das im Speicher befindliche Programm wird gelöscht. Tastenzuordnungen sowie die Einstellung der Bildschirmanzeige (z.B. MODE, PEN, PAPER, INK) bleiben dabei erhalten. Es erfolgt kein Löschen des Bildschirms.

NEXT *Variablenliste*

siehe FOR

NOT *Ausdruck*

Beispiel: Eingabe:
```
IF NOT A=1 THEN PRINT "Fehler" ELSE PRINT "richtig"
```
Anzeige: Fehler
Eingabe: PRINT NOT 0
Anzeige: -1

Funktion: Der logische Operator NOT verknüpft bitweise dual dargestellte ganze Zahlen miteinander. War ein Bit 1, erhält es den Wert 0 und umgekehrt.

ON BREAK CONT

Beispiel: Eingabe:
```
10 ON BREAK CONT
20 PRINT "Keine Unterbrechung mit
   [ESC]-Taste möglich"
30 GOTO 20
run
```

Funktion: Diese Anweisung verhindert das Unterbrechen eines Programms, deshalb sollte diese Anweisung mit Vorsicht benutzt werden. Nach dem Starten des Programms ist ein Abspeichern nicht mehr möglich. Aus diesem Grund erst das Programm abspeichern und dann starten.

ON BREAK GOSUB *Zeilennummer*

Beispiel: Eingabe:
```
10 ON BREAK GOSUB 50
20 PRINT "Programm ist gestartet"
30 FOR i=1 TO 1000:NEXT
40 GOTO 20
50 PRINT "Hier beginnt das Unterprogramm"
60 FOR a=1 TO 700:NEXT
70 RETURN
run
```

Funktion: Das Programm geht zu der angegebenen Zeile, auf der das Unterprogramm steht, wenn die Taste [ESC] zweimal gedrückt wird. Das Programm kann nicht abgebrochen werden.

ON BREAK STOP

Beispiel: Eingabe:
```
10 ON BREAK GOSUB 50
20 PRINT "Programm ist gestartet"
30 FOR i=1 TO 100:NEXT
40 GOTO 20
50 PRINT "Hier ist das Unterprogramm"
60 FOR a=1 TO 1200:NEXT
65 ON BREAK STOP
70 RETURN
run
```

Funktion: Die Anweisungen ON BREAK CONT und ON BREAK GOSUB werden unwirksam, d.h. das Programm wird angehalten, wenn die Taste [ESC] zweimal gedrückt wird. Im Beispiel ist ON BREAK GOSUB nur einmal wirksam, da das Unterprogramm in Zeile 65 unterbrochen wird.

ON ERROR GOTO *Zeilennummer*

Beispiel: Eingabe:
```
10 ON ERROR GOTO 80
20 CLS:PRINT "Bei Fehler Programm auflisten"
30 FOR i=1 TO 2000:NEXT
40 GOTO 130
80 PRINT "Der Fehler ist in Zeile";ERL
90 PRINT: LIST
run
```

Funktion: Beim Auftreten eines Fehlers wird in die angegebene Zeile gesprungen. Durch ON ERROR GOTO 80 wird die Fehler-Falle beseitigt. BASIC gibt dann die Standard-fehlermeldungen aus.

ON *Selektor* **GOSUB** *Auswahl von Zeilennummern*

Beispiel: Eingabe:
```
10 CLS
20 PRINT "Funktionsauswahl"
30 PRINT "1 - Ende"
40 PRINT "2 - Randfarbe"
50 PRINT "3 - Modus einstellen"
60 INPUT "Eingabe der Ziffer:";a
70 ON a GOSUB 90,100,110
80 GOTO 10
90 END
100 INPUT "Border";b:BORDER b:RETURN
110 INPUT "Modus 0-2";c:MODE c:RETURN
run
```

Funktion: Das Unterprogramm, das in der angegebenen Zeilennummer steht und mit dem Selektor aufgerufen ist, wird abgearbeitet. Der Wertebereich des positiven ganzzahligen Selektors liegt zwischen 0 und 255.
Der vereinbarte Selektor a im Beispiel bestimmt mit seinen Werten, die wievielte Zeilennummer aufgerufen wird, z.B. ist Selektor a=1, dann wird die Zeile 90, bei a=2 die Zeile 100 ausgeführt usw.

ON *Selektor* **GOTO** *Anzahl Zeilennummern*

Beispiel: Eingabe:
```
10 CLS:PRINT "Auswahl"
20 PRINT "1 - Ende"
30 PRINT "2 - LIST"
40 PRINT "3 - MODE einstellen"
45 INPUT "Eingabe";x
50 ON x GOTO 60,70,80
60 END
70 LIST
80 INPUT "Eingabe MODE";a
90 MODE a
run
```

Funktion: Entsprechend dem Wert des Selektors wird zur angegebenen Zeilennummer gegangen. Der Wertebereich des positiven ganzzahligen Selektors liegt zwischen 0 und 255. Dabei bestimmt dieser Wert den Aufruf der Position der Zeilennummer in der Liste, d.h. wenn x=1, dann Zeilennummer 60, bei x=2 Zeilennummer 70 und bei x=3 Zeilennummer 80. Für Selektorwert 0 oder größere Werte als Zeilennummern vorhanden sind, erfolgt kein Zeilenaufruf.

ON SQ (*Kanal*) **GOSUB** *Zeilennummer*

Beispiel: Eingabe:

```
10 ENV 1,15,-1,10,15,1,10
20 ON SQ(1) GOSUB 60
30 MODE 0:ORIGIN 0,0,200,200,100,300
40 FOR x=1 TO 13:FRAME:MOVE 330,200,x
50 FILL x:NEXT:GOTO 40
60 READ s:IF s=0 THEN RESTORE:GOTO 60
70 SOUND 1,s,25,15,1
80 ON SQ(1) GOSUB 60:RETURN
90 DATA 50,60,90,100,35,200,24,500,0
run
```

Funktion: Das in der angegebenen Zeile stehende Unterprogramm wird aufgerufen, wenn in der Tonwarteschlange des angegebenen Kanals Platz ist. Die Angabe des Kanals ist eine ganze Zahl.

Parameter Kanal wird mit:
1 für Kanal A
2 für Kanal B
4 für Kanal C
verwendet.

OPENIN *Dateiname*

Beispiel: Eingabe:

```
10 REM Datei wird von Kassette eroeffnet
20 OPENIN "Datei":INPUT #9,x,x$
30 CLOSEIN:PRINT "Daten x,x$"
40 PRINT:PRINT x,x$
run
```

Funktion: Von Kassette/Diskette wird eine Eingabedatei eröffnet, die im arbeitenden Programm verwendet wird. Die Datei muß eine ASCII-Datei sein. Um dieses Beispiel zu benutzen, sollte erst das Beispiel von OPENOUT auf Kassette/Diskette gespeichert werden.

OPENOUT *Dateiname*

Beispiel: Eingabe:
```
10 REM Daten auf Kassette
20 INPUT "Eingabe von Zahlen";x
30 INPUT "Eingabe von Buchstaben";x$
40 OPENOUT "Datei"
50 WRITE #9,x,x$
60 CLOSEOUT:PRINT "Datenausgabe"
run
```

Funktion: Eine Ausgabedatei auf Kassette oder Diskette wird eröffnet.

Ausdruck **OR** *Ausdruck*

Beispiel:
```
Eingabe: PRINT 0 OR 1
Anzeige: 1
Eingabe: PRINT 1 OR 1
Anzeige: 1
Eingabe: PRINT 0 OR 0
Anzeige: 0
```

Funktion: Aus einem gegebenen Wertevorrat A und B werden bitweise Werte durch OR verknüpft. Das Ergebnis ist 1, wenn mindestens einer der beiden Werte 1 ist.

A	B	A OR B
0	0	0
0	1	1
1	0	1
1	1	1

Boolescher Verknüpfungsoperator

ORIGIN *x,y* [,*links,rechts,oben,unten*]

Beispiel: Eingabe:
```
10 BORDER 10:TAG
20 ORIGIN 0,0,150,440,200,100
30 GRAPHICS PAPER 2:CLG
40 FOR i=450 TO -200 STEP -5
50 MOVE i,200
60 PRINT "Ein Grafikfenster",
70 FRAME:NEXT:GOTO 40
run
```

Funktion: Der Standardkoordinatenpunkt (englisch origin) wird an den durch x/y bestimmten Punkt verlegt. Die vier wahlweise eingebbaren Parameter richten einen Grafikbildschirmbereich ein. Sind die Koordinaten außerhalb des Bildschirmbereiches, ist der Bildschirm die Begrenzung.

OUT *Interfaceadresse,ganzzahliger Ausdruck*

Beispiel: Eingabe:
```
OUT &F8F4,&FF
```

Funktion: Der Wert des ganzzahligen Ausdrucks wird an die Interfaceadresse (Schnittstellenadresse) ausgegeben. Der Wertebereich des ganzzahligen Ausdrucks liegt zwischen 0 und 255.

Achtung: Vorsicht beim Benutzen dieser Anweisung, da durch falsche Angaben unerwünschte Reaktionen des Computers folgen können.

PAPER [*#Stream,*] *Farbstift*

Beispiel: Eingabe:
```
10 MODE 0:PEN 0:INK 1,10
20 FOR i=1 TO 10
30 PAPER i:CLS
40 PRINT "Papierfarbe";i
50 FOR a=1 TO 1000:NEXT
60 NEXT
run
```

Funktion: Festlegen des Hintergrundes (Papier), auf dem die Schrift erscheint. Jedes auf dem Bildschirm abgebildete Zeichen wird mit einer Hintergrundfarbe, die dem Farbstift mit INK zugewiesen wurde, aufgefüllt, wenn nicht der Transparentmodus eingestellt wurde.
Ohne Streamangabe wird Stream #0 angenommen. Der eingestellte Bildschirmmodus ergibt die Anzahl der zu benutzenden Farbstifte.

PEEK (*Adresse*)

Beispiel: Eingabe:
```
10 MODE 2
20 REM
30 PRINT "A"
40 FOR adress=&C00 TO &F800 STEP &800
50 P$=BIN$(PEEK(adress),8))
60 FOR l=1 TO 8
70 IF MID$(p$,l,1)="1" THEN PRINT "x";
   ELSE PRINT ".";
80 NEXT l
90 PRINT
100 NEXT adress
run
```

Funktion: Von der in Klammern angegebenen Speicheradresse im Adreßbereich von 0 bis 65535 (hexadezimal 0 bis &FFFF) wird der Inhalt ausgegeben. Der angezeigte Wert (Inhalt) von PEEK liegt im Bereich von 0 bis 255 oder 0 bis &FF und bezieht sich auf RAM-Adressen (RAM - englisch Random Access Memory; Lese- und Schreibspeicher, d.h., in dieser Speichereinheit kann geschrieben und gelesen werden) und nicht auf ROM-Adressen (ROM - englisch Read only Memory; nur Lesespeicher, d.h. von dieser Speichereinheit kann nur gelesen werden, nicht geschrieben).

PEN [*#Stream*,] [*Farbstift*] [,*Hintergrundmodus*]

Beispiel: Eingabe:
```
10 MODE 0
20 INK 0,11
30 FOR n=1 TO 15
40 PEN n:PRINT TAB(20) "Farbstift"
50 NEXT
run
```

Funktion: Der angegebene Farbstift (Wertebereich von 0 bis 15), der in einem Bildschirmbereich (wenn kein anderer vereinbart wurde - gesamter Bildschirmbereich, Stream #0, braucht nicht extra angegeben zu werden) verwendet werden soll, wird zugeordnet. Der Hintergrundmodus kann transparent =1 oder nichttransparent =0 gewählt werden. Beim Weglassen eines Parameters verändert sich die Einstellung nicht (für zwei siehe MASK).

PI

Beispiel: Eingabe: `PRINT PI`
Anzeige: `3.14159265`

Funktion: Konstanter Wert, der das Verhältnis von Umfang und Durchmesser eines Kreises angibt.

PLOT *x-Koordinate,y-Koordinate* [,*Farbstift*] [,*Farbmodus*]

Beispiel: Eingabe:
```
10 MODE 0:INK 0,2:INK 1,26:DEG
20 FOR i=1 TO 300
30 PLOT RND*640,RND*400
40 NEXT
run
```

Funktion: Ein Punkt (englisch point) wird auf dem Grafikbildschirm dargestellt. Die Stelle wird durch die absoluten Werte von x/y-Koordinaten bestimmt. Ein Farbstift kann im Wertebereich von 0 bis 15 gewählt werden. Der Farbmodus ist wahlweise nutzbar und bestimmt das Zusammenwirken des Farbstiftes mit allen übrigen Farben des Bildschirms. Diese Farbmodi sind:

0 = Normal
1 = XOR
2 = AND
3 = OR

PLOTR *x-Versatz,y-Versatz* [,*Farbstift*] [,*Farbmodus*]

Beispiel: Eingabe:
```
10 MODE 1
20 FOR a=1 TO 640 STEP 2
30 PLOTR a,a*RND:DRAW 400,a
40 NEXT
```

Funktion: Ein Punkt (englisch point) wird auf dem Grafikbildschirm dargestellt. Dabei werden die x/y-Koordinaten relativ zur augenblicklichen Cursorposition angegeben.

0 = Normal
1 = XOR
2 = AND
3 = OR

POKE *Adresse,ganzzahliger Ausdruck*

Beispiel: Eingabe:
```
10 REM POKE
20 FOR adress=50000 TO 60000
30 POKE adress,100:NEXT
```

Funktion: Ein Wert, der im ganzzahligen Ausdruck angegeben ist, wird in die Speicheradresse des RAMs direkt eingetragen. Der Bereich des ganzzahligen Ausdrucks liegt zwischen 0 und 255.

Achtung: Vorsicht bei der Anwendung dieser Anweisung, da falsche Angaben unerwünschte Reaktionen des Computers verursachen können.

POS (*#Stream*)

Beispiel: Eingabe:
```
10 CLS
20 PRINT POS(#0) "Textcursorposition"
30 LOCATE 4,5: PRINT POS (#0) "Der Cursor
   ist hier"
run
```

Funktion: Die aktuelle Position des Text-Cursors auf der x-Achse (waagerecht) wird angegeben. Sie bezieht sich auf den linken Bildschirmrand. Der Stream-Parameter des Text-Cursors ist unbedingt anzugeben, da nicht automatisch Stream #0 vereinbart wird. Mit POS(#8) wird die Position des Schreibkopfes am Drucker angegeben, dabei bedeutet eins die erste Stelle in der Zeile. Mit POS(#9) wird die geschriebene Zeichenanzahl seit dem letzten RETURN auf der Kassette/Diskette angegeben.

PRINT [*#Stream*,] [*Ausgabeliste*]

Beispiel: Eingabe:
```
10 MODE 1:CLS
20 a$="kurzer Text"
30 a=120
40 b$="Hier ist der laengere Textteil"
50 PRINT a,a
60 PRINT a;a
70 PRINT a$,a$
80 PRINT a$;a$
90 PRINT b$,b$
100 PRINT b$;b$
110 PRINT a,a$,b$
120 PRINT a;a$;b$
run
```

Funktion: Die vereinbarte Ausgabeliste wird auf dem angegebenen Stream (Bildschirmbereich) angezeigt. Ist kein Stream benannt, wird auf Stream #0 ausgegeben. Steht ein Komma zwischen den einzelnen Ausdrücken, so wird in der nächsten Druckzone, d.h. ab der 13. Zeichenposition, ausgegeben (einfaches Formatieren). Steht ein Semikolon zwischen den einzelnen Ausdrücken, erfolgt Ausgabe für Ausgabe hintereinander.

PRINT SPC *#Stream*,] [*Liste von Ausdrücken*] [;] [SPC(*ganze Zahl*)] [*Liste von Ausdrücken*]

PRINT TAB [*#Stream*,] [*Liste von Ausdrücken*] [;] [TAB(*ganze Zahl*)] [*Liste von Ausdrücken*]

Beispiel: Eingabe: `PRINT SPC(6);"SPC-Funktion";SPC(20);"a"`

Eingabe: `PRINT TAB(6);"TAB-Funktion";TAB(20);"a"`

Funktion: Mit SPC wird eine Anzahl (ganze Zahl) von Leerstellen ausgegeben, und der folgende Ausdruck wird unmittelbar dahinter gesetzt, falls er noch in die Zeile paßt. Deshalb braucht kein Semikolon benutzt zu werden. Mit TAB wird eine Anzahl (ganze Zahl) von Leerstellen, beginnend vom linken Rand des Textfensters, ausgegeben und der folgende Ausdruck wird unmittelbar dahintergesetzt, falls er noch in die Zeile paßt. Deshalb braucht kein Semikolon gesetzt werden. Befindet sich der Cursor hinter der festgelegten Position, so er folgt ein Versetzen des nächsten Ausdrucks um eine Zeile.

PRINT [*#Stream*] [*Ausgabeliste*] [*;*] [USING *Formatschablone*] [*Trennzeichen Ausdruck*]

Beispiel: Eingabe:
```
10 CLS:FOR m=1 TO 1000000
20 PRINT "Menge";USING "#######,.##";m
30 NEXT
run
```

Funktion: Das Format des durch PRINT angezeigten Ausdrucks kann bestimmt werden. Dabei wird der Ausdruck in der bestimmten Formatschablone angezeigt. Als Trennzeichen sind Semikolon und Komma einsetzbar.

Folgende Formatzeichen können für die Formatschablone benutzt werden:

Formate numerischer Ausgaben
---

innerhalb einer Zahl:

(steht für eine Stelle)
z.B.
Eingabe: `PRINT USING "#######";123`
Anzeige: 123

Die Position des Dezimalpunktes wird festgelegt.
z.B.
Eingabe: `PRINT USING "###.##";12345`
Anzeige: 12345.00

(für eine Stelle stehend)
Ist nur vor dem Dezimalpunkt zu setzen. Es gibt an, daß die Ziffern vor dem Dezimalpunkt in Dreiergruppen (in Tausendern) einzuteilen sind.
z.B.
Eingabe: `PRINT USING "####,.###";1234.15`
Anzeige: 1,234.150

außerhalb einer Zahl:

££ (für zwei Stellen stehend)
Ein £-Zeichen soll vor der ersten Ziffer oder dem Dezimalpunkt bzw. nach entsprechenden Ziffern erscheinen. Dabei ist zu beachten, daß das £-Zeichen eine Stelle in der Ziffernfolge einnimmt.
z.B.
Eingabe: `PRINT USING " ££####,.#";1234.11`
Anzeige: `1,234.1`

** (für zwei Stellen stehend)
Die vorangehenden Leerstellen sollen durch ** dargestellt werden.
z.B.
Eingabe: `PRINT USING "*****###.##";123.45`
Anzeige: `***123.45`

**£ (für drei Stellen stehend)
Ist eine Kombination von Sternchen * und Pfundzeichen £, die der Zahl vorangestellt werden.
z.B. ** ####,.##;

$$ (für zwei Stellen stehend)
Ein Dollarzeichen $ soll vor der ersten Zahl oder dem Dezimalpunkt bzw. vor aber auch nach entsprechenden Zeichen stehen.
z.B. $$###,.##

*$$ (für drei Stellen stehend)
Ist eine Kombination des *- und des $-Zeichens in der Darstellung des Ausdrucks.
z.B. *$$###,.##

\+ Entsprechend dem Wert einer Zahl sollen ein + oder ein - vorangestellt werden. Stehen + oder - vor der Zahl oder eventuellem Währungszeichen, so wird das Zeichen (+/-) vor die Zahl gesetzt. Erscheinen das + oder - hinter der Zahl bzw. hinter einem eventuellen Währungszeichen, so werden + bzw. - hinter die Zahl und eventuellen Exponenten gesetzt.

\- Minus kann nur am Ende der Formatschablone stehen. Das Zeichen wird hinter jede negative Zahl bzw. Exponenten gesetzt. Für positive Zahlen wird eine Leerstelle ausgegeben. Enthält die Formatschablone kein Minuszeichen, wird vor jede negative Zahl automatisch ein "-" gesetzt.
z.B. ###.##-

↑↑↑↑ Zahlendarstellung in Exponentialschreibweise
Dieses Zeichen soll nach Ziffern, jedoch vor angehängten +/- Zeichen stehen.
z.B.
Eingabe: `PRINT USING "###.## ";678.90`
Anzeige: `67.90 E+01`

Hinweis:
20 Zeichen darf die Formatschablone für Zahlen besitzen!

Ist die Formatschablone für einen Ausdruck zu klein, so wird ein %-Zeichen vorangestellt. Dies bewirkt, daß die gesamte Länge des Ausdrucks ausgegeben wird und demzufolge wird das Format nicht eingehalten.

Formate für Strings

! Das erste Zeichen des Strings ist auszugeben.
z.B.
Eingabe: `PRINT USING "!";"Beispiel"`
Anzeige: `B`

\ \(Leerstellen)\
Die ersten in Leerstellen angegebene Anzahl von Zeichen soll ausgegeben werden. Dabei zählen die beiden Schrägstriche in der Länge der Schablone mit.

& Wie vorgegeben wird der String ausgedruckt.

Hinweis:
Formatschablonen können auch als Strings in String variablen dargestellt werden.

z.B.
Eingabe:
```
10 x$="#####,.##"
20 z$="!"
30 PRINT USING x$;1234.56;
40 PRINT USING z$;"Mark";
run
```

Eingabe:
```
10 a$="mnopqrstuvwxyz"
20 PRINT "Ausdruck:";a$
30 PRINT "| Formatzeichen = ";
40 PRINT USING "|";a$
50 PRINT " Leerzeichen Formatzeichen = ";
60 PRINT USING " = ";
70 PRINT "& Formatzeichen = ";a$
80 PRINT USING "&";a$
90 GOTO
90 run
```

RAD

Funktion: (englisch radians)
Die Umschaltung auf Bogenmaß (englisch radians) erfolgt. Von BASIC werden standardmäßig Bogenmaßwerte nach dem Einschalten bzw. nach dem Zurücksetzen in den RESET-Zustand erwartet bzw. ausgegeben.

RANDOMIZE [*numerischer Ausdruck*]

Beispiel: Eingabe:
```
10 RANDOMIZE 800.000
20 PRINT RND
run
```

Anzeige:
```
0.896940658
```

Funktion: Der Anfangswert des Zufallsgenerators wird eingestellt. Die Zufallszahlen sind dabei abhängig vom eingestellten Startwert. Der Anfangswert kann im numerischen Ausdruck festgelegt werden. Beim Fehlen des Parameters muß dann eine Angabe erfolgen, wenn der Computer fragt "RANDOM number seed?". RANDOMIZE TIME legt eine fast nicht wiederholbare Zahlenfolge fest.

READ *Variablenliste*

Beispiel: Eingabe:
```
10 MODE 0
20 FOR i=1 TO 8
30 READ a
40 BORDER i:FOR x=1 TO 1000:NEXT
50 NEXT
60 DATA 1,2,3,4,5,6,7,8
```

Funktion: Es werden aus der DATA-Anweisung Daten eingelesen und in Reihenfolge den angegebenen Variablen zugeordnet. Mit RESTORE wird der DATA-Zeiger auf den ersten DATA-Wert gesetzt. DATA-Werte können Zahlen und Strings (Textfolgen) sein. Variablen können demzufolge auch numerische oder Stringvariablen sein.

z.B.
DATA "LIST,RUN,EDIT" oder
DATA LIST,RUN,EDIT

RELEASE *Kanäle*

Beispiel:
```
RELEASE 1
RELEASE 2
RELEASE 3
```

Funktion: Der Wartezustand, in dem sich die Kanäle durch die SOUND-Anweisung befinden, wird aufgehoben. Wertebereich des Parameters Kanäle: ganze Zahlen von 1 bis 7

Bedeutung: 1 - Kanal A ist frei
2 - Kanal B ist frei
3 - Kanäle A und B frei
4 - Kanal C frei
5 - Kanäle A und C werden freigegeben
6 - Kanäle B und C frei
7 - Kanäle A, B und C frei

Weitere Hinweise sind dem Abschnitt 3.14. zu entnehmen.

REM *Zeilenrest*

Beispiel: Eingabe: 10 REM Kommentartext

Funktion: In das Programm wird eine Kommentarzeile eingefügt. Alle Zeichen nach REM werden von BASIC nicht beachtet, auch der Doppelpunkt und das Komma.
Statt REM kann auch das halbe Anführungszeichen (über der 7) ' benutzt werden.

REMAIN (*Zeitgeber*)

Beispiel: Eingabe:

```
10 AFTER 400, 3 GOSUB 40
20 AFTER 100, 0 GOSUB 50
30 PRINT "Programmstart": GOTO30
40 PRINT "Kein Aufruf"
50 PRINT "Zeitgeber 3 wird abgeschaltet durch REMAIN"
60 PRINT "Restzeit"
70 PRINT REMAIN(3)
run
```

Funktion: Mit dieser Funktion werden der Zeitgeber außer Kraft gesetzt und die verbleibende Restzeit (englisch REMAINing) des Zeitgebers ausgegeben.

RENUM [*neue Zeilennummer*][, [*alte Zeilennummer*] [,*Schrittweite*]]

Beispiel: Eingabe:

```
10 CLS:PRINT "Zeile 1"
20 PRINT "Zeile 2"
30 PRINT "Zeile 3"
RENUM 1, 10, 1
LIST
```

Funktion: Ein Programm erhält neue Zeilennummern. Dabei geben die Parameter an:

alte Zeilennummer:
Gibt an, ab welcher Zeilennummer neu zu numerieren ist. Wird der Parameter weggelassen, fängt die neue Numerierung ab der 1. Programmzeile an.

neue Zeilennummer
Dieser Parameter bestimmt, wie die erste neue Zeile heißen soll. Beim Weglassen dieses Parameters wird automatisch bei 10 begonnen.

Schrittweite
Die Schrittweite bestimmt die Abstände zwischen den einzelnen Zeilennummern. Ohne Angabe betragen die Abstände automatisch 10.

Bei dieser Funktion werden alle Sprunganweisungen mit Zeilenangaben, wie z. B. GOTO, GOSUB geändert. Innerhalb eines Textes (String z.B. bei der KEY-Anweisung) und in REM-Kommentaren erfolgt keine Änderung der Zeilennummer. Auch die Zeilennummern in CHAIN, CHAIN MERGE werden nicht verändert.

RESTORE [*Zeilennummer*]

Beispiel: Eingabe:
```
10 READ X$ : PRINT X$; " "
20 FOR P = 1 TO 200 : NEXT
30 RESTORE
40 GOTO 10
50 DATA Hier DATA-ZEILE
run
```

Funktion: Der DATA-Zeiger wird auf die angegebene Zeile gestellt. Fehlt die Zeilennummer, geht der Zeiger auf die 1. DATA-Anweisung im Programm zurück.

RESUME *Zeilenummer*

Beispiel: Eingabe:
```
 10 ON ERROR GOTO 100
 20 FOR I = 0 TO 300
 30 PRINT CHR$(I):END
100 PRINT " Fehler-Nr."; ERR; "Zeile"; ERL
110 RESUME 120
120 PRINT "Fehler war da"
run
```

Funktion: Das Programm wird in der nächsten Zeile bzw. in der angegebenen Zeilennummer fortgesetzt, wenn es durch ON ERROR GOTO unterbrochen wurde. Ohne Zeilenangabe arbeitet das Programm in der Zeile weiter, wo der Fehler auftrat.

RESUME NEXT

Beispiel: Eingabe:
```
10 ON ERROR GOTO 80
20 PRINT "Fehler bei PRINT"
30 PRINTT
40 INPUT "Eingabe RETURN"; a
50 INPUTT "2"; a
60 INPUUT "3"; a
70 END
80 PRINT "Fehler"; ERR; "in Zeile"; ERL
90 RESUME NEXT
run
```

Funktion: Wiederaufnahme (engl. resume) des Programms nach einem Fehler und der Ausführung der ON ERROR-Anweisung. Das Programm arbeitet ab der Zeile weiter, die der fehlerhaften Zeile folgt.

RETURN

Beispiel: Eingabe:
```
 10 PRINT "Pausenschleife als Unterprogramm"
 20 MODE 0
 30 INPUT "BORDER-Farbe"; a
 40 BORDER a
 50 GOSUB 100
 60 GOTO 30
100 FOR I = 1 TO 1000 : NEXT : RETURN
run
```

Funktion: Schließt ein Unterprogramm ab. Der BASIC-Interpreter kehrt zu der Stelle zurück (engl. return), die nach dem Aufruf (GOSUB) des Unterprogramms folgt.

RIGHT$ (*String*, *Länge*)

Beispiel: Eingabe:
```
RIGHT$ (A$, X)
10 A$ = "Lager"
20 PRINT LEFT$ (A$, 4)
30 PRINT RIGHT$ (A$, 4)
40 PRINT MID$ (A$, 5)
50 PRINT MID$ (A$, 3, 3)
run
```

Funktion: Der angegebene String wird in der festgelegten Länge (Anzahl der Zeichen von 0 bis 225) ausgegeben. Dabei wird der String von rechts (engl. right) beginnend gelesen. Bei kürzerem String als die Länge angibt, erfolgt ein entsprechendes Wiederholen des gesamten Strings.

RND [(*numerischer Ausdruck*)]

Beispiel: Eingabe:
```
10 FOR I = 1 TO 5
20 PRINT RND (1)
30 PRINT 4 * RND (1)
40 PRINT INT (4 * RND (1))
50 NEXT
run
```

Funktion: Eine neue Zufallszahl (engl. RaNDom number) in der entsprechenden Folge wird ausgegeben, falls der numerische Ausdruck positiv ist oder nicht angegeben wurde. Für eine negative Zahl, als Parameter eingesetzt, erfolgt die Ausgabe einer neuen Folge von Zufallszahlen, wobei die erste Zahl sich immer wiederholt.

ROUND (*numerischer Ausdruck* [,*Dezimalstellen*])

Beispiel: Eingabe: `PRINT ROUND (7891.9588, 3)`
Anzeige: `7891.959`
Eingabe: `PRINT ROUND (7891.234, -4)`
Anzeige: `10000`

Funktion: Der numerische Ausdruck wird zu einer Zahl gerundet (engl. round), die soviel Dezimalstellen bzw. Zehnerpotenzen (E) besitzt, wie der Parameter angibt.

RUN *String*

Beispiel: Eingabe: `RUN "Name"`

Funktion: Ein BASIC-Programm wird von der Kassette oder Diskette in den Computer geladen und anschließend gestartet (engl. run). Ein im Speicher befindliches Programm wird überschrieben. Auf diese Art können geschützte BASIC-Programme direkt gestartet werden.

Hinweis: Dabei ist zu beachten, daß der vereinbarte Name beim Laden von der Kassette vollständig angegeben werden muß, z. B. RUN "Test.bas".

RUN *Zeilennummer*

Beispiel: Eingabe: `RUN 1000`

Funktion: Ein Programm, das sich bereits im Speicher des Computers befindet, wird ab der angegebenen Zeilennummer gestartet. Der Start geschützter Programme ist auf diese Weise nicht möglich.

SAVE *Dateiname*. [,*Dateityp*] [,*Binärdateiparameter*]

Beispiel: Eingabe: `SAVE "Programm.bas"`
Das Programm wird als normales BASIC-Programm (Datei) ungeschützt gespeichert.

Eingabe: `SAVE "Programm.bas", p`
Abspeicherung als geschütztes BASIC-Programm

Eingabe: `SAVE "Datei.bas", A`
Speicherung als ASCII-Datei

Eingabe: `SAVE "Datei.bin", B, 4000, 2000, 4001`
Datei wird als Binärdatei gespeichert.
Das Programm benutzt im Computer den Speicherbereich ab Adresse 4000, mit einer Länge von 2000 Byte (Programmgröße) und der Startadresse von 4001.

Funktion: Das im Speicher befindliche Programm wird auf Kassette bzw. Diskette geschrieben. Eine Binärdatei stellt einen auf Kassette bzw. Diskette ausgegebenen Speicherbereich dar. Dabei sind folgende Binärdatei parameter aufzuführen:

Startadresse, Dateilänge [,Anfangspunkt]

Der Inhalt des Bildschirmes kann z. B. als Binärdatei abgespeichert werden mit:

```
SAVE "Anzeige", B, &C000, &4000
```

und durch:

```
LOAD "Anzeige"
```

wieder auf dem Bildschirm dargestellt werden.

SGN (*numerischer Ausdruck*)

Beispiel: Eingabe:
```
10 FOR I = - 10 TO 10
20 PRINT SGN (I)
30 NEXT
```

Funktion: Das Vorzeichen (engl. SIGN) des in Klammern stehenden numerischen Ausdrucks wird angegeben. Dabei erscheint in Abhängigkeit vom Ausdruck für:

Bedingung	Anzeige
numerischer Ausdruck 0 (ist kleiner 0)	eine -1
numerischer Ausdruck=0 (gleich 0)	eine 0
numerischer Ausdruck^0 (größer 0)	eine 1

SIN (*numerischer Ausdruck*)

Beispiel: Eingabe:
```
10 CLS : DEG
20 FOR I = 0 TO PI/2 STEP.01
30 Y = SIN (I)
40 PLOT i * 600, y * i : NEXT
50 FOR P = 1 TO 1500 : NEXT
run
```

Funktion: Der Sinus des in Klammern stehenden Ausdrucks wird berechnet. Die Eingabe wird im Bogenmaß erwartet, wenn nichts anderes vereinbart wurde. Dabei stellt DEG Winkelmaß und RAD Bogenmaß ein.

SOUND *Kanalstatus*,*Tonperiode* [,*Dauer* [,*Lautstärke*
[,*Lautstärkenhüllkurve* [,*Tonhüllkurve*[,*Geräuschperiode*]]]]]

Beispiel: Eingabe:
```
10 FOR T = 0 TO 3000
20 SOUND 1, t, 1, 15
30 NEXT
run
```

Funktion: Ein Ton wird ausgegeben. Folgende Parameter enthält diese Anweisung:

Parameter 1:
Kanalstatus Der Kanalstatus muß als ganze Zahl (Wertebereich von 1 bis 255) eingegeben werden. Dieser Parameter ist Bit-signifikant (bezogen), dabei besitzen die einzelnen Bits folgende Bedeutung:

Bit	dezimaler Wert	Kommando
0	1	Ton geht zum Kanal A
1	2	Ton geht zum Kanal B
2	4	Ton geht zum Kanal C
3	8	Rendezvous mit Kanal A
4	16	Rendezvous mit Kanal B
5	32	Rendezvous mit Kanal C
6	64	Tonkanal halten
7	128	Tonkanal leeren

z. B. der Kanalstatus besitzt den Wert 65 mit folgen der Bedeutung:
Der Ton aus dem Kanal A (dezimaler Wert 1) wird gehalten (dezimaler Wert 64).
Parameter 2:
Tonperiode
Mit diesem Parameter wird die Tonhöhe festgelegt, also wird die Note bestimmt, die zu spielen ist. Dabei ist jeder Note eine Zahl zugeordnet, die als Tonperiode benutzt wird. In der Tabelle Noten und Tonperioden (Anhang E) ist diese Zuordnung zu entnehmen.

Parameter 3:
Dauer
Mit diesem Parameter wird die Zeitdauer eines Tones festgelegt. Dabei gibt der Wert an, wieviele Hundertstel Sekunden der Ton erzeugt wird. Ist keine Dauer angegeben, wird automatisch der Standardwert 20 (= 1/5 Sekunden) vereinbart. Für den Wert 0 ist die Dauer des Tones bis zum Ende der Lautstärkenhüllkurve gegeben.

Parameter 4:
Lautstärke
Die Lautstärke zu Beginn eines Tones wird bestimmt, wobei der Wertebereich von 0 bis 15 (0 = keine Lautstärke, 15 maximale Lautstärke) liegt. Ohne Zahlenangabe wird standardmäßig Wert 12 vereinbart.

Parameter 5:
Lautstärkenhüllkurve
Während der Dauer eines Tones kann mit ENV eine Lautstärkenhüllkurve bestimmt werden, um die Lautstärke zu verändern. Es können 15 verschiedene Hüllkurven definiert werden. Der in der SOUND-Anweisung benutzte Lautstärkenhüllkurven - Parameter ist gleich der Lautstärkenhüllkurven-Nummer in der ENV-Anweisung. (siehe Anweisung ENV).

Parameter 6:
Tonhüllkurve
Während der Dauer eines Tones kann mit der Anweisung ENT die Tonhöhe verändert werden, indem die Tonhüllkurve definiert wird. Es können 15 verschiedene Tonhüllkurven bestimmt werden. Der in SOUND benutzte Tonhüllkurven-Parameter ist gleich der Tonhüllkurven-Nummer in der ENT-Anweisung (siehe Anweisung ENT).

Parameter 7:
Geräuschperiode
Den Ton kann in verschiedenen Varianten weißes Rauschen begleiten (bzw. kann das weiße Rauschen abgeschaltet sein).
Wertebereich des Parameters Geräuschperiode:
von 0 bis 31

SPACE$ (*ganzzahliger Ausdruck*)

Beispiel: Eingabe:
```
10 MODE 1
20 FOR I = 1 TO 10
30 PRINT SPACE$ (I) "Zeichenausgabe"
40 NEXT
run
```

Funktion: Leerzeichen (engl. Space) werden in der angegebenen Anzahl als String ausgegeben. Die Anzahl liegt im Wertebereich von 0 bis 255.

SPC siehe PRINT SPC

SPEED INK *Periode 1, Periode 2*

Beispiel: Eingabe:
```
10 BORDER 2, 18
20 FOR F = 20 TO 1 STEP -1
30 SPEED INK F, F
40 FOR i = 1 TO 1000: NEXT i, F
run
```

Funktion: Die Geschwindigkeit des Farbwechsels, der in INK oder BORDER eingestellten Farben wird festgelegt. Periode 1 legt die Farbdauer der ersten Farbe und Periode 2 die Farbdauer der zweiten Farbe fest. Die Zeiteinheit der Farbdauer ist 0,02 Sekunden oder Fünfzigstelsekunden.

SPEED KEY *Startverzögerung, Wiederholungsperiode*

Beispiel: Eingabe:
```
10 MODE 1
20 FOR I = 9 TO 1 STEP -2
30 INPUT "Eingabe des Alphabetes"; a$
40 SPEED KEY i,i
50 LINE INPUT a$ : NEXT
60 PRINT "Alle Buchstaben?"
```

Funktion: Es wird festgelegt, in welchen Zeitabständen sich ein Zeichen in der Dauerfunktion wiederholt. Der Startverzögerungswert bestimmt, wieviel Zeit (in 0,02 Sekunden) bis zum Start der Dauerfunktion vergehen soll. Im Parameter Wiederholungsperiode wird der Zeitabstand der Wiederholung der Dauerfunktion einer Taste festgelegt. Mit SPEED KEY können nur Tasten benutzt werden, die auch standardmäßig die Dauerfunktion besitzen oder durch KEY DEF auf Dauerfunktion geschaltet wurden.

Um das Standardtempo der Tastatur wieder einzustellen sollte beim Einstellen kleiner Startverzögerungswerte eine Zahlentaste mit dem Standardwert eingestellt werden.

z.B. `KEY 1, "SPEED KEY 30,2" + CHR$ (13)`

Die Dauerfunktion der Tastatur wird durch Drücken der Zahl 1 auf das Standardtempo eingestellt.

SPEED WRITE *ganzzahliger Ausdruck*

Beispiel: Eingabe: SPEED WRITE 1

Funktion: Legt die Geschwindigkeit fest, mit der Daten auf Kassette gespeichert bzw. geschrieben werden. Der Wert des ganzzahligen Ausdrucks beträgt:

0 = die Kassette wird mit 1000 Baud (Bit pro Sekunde) beschrieben, die auch die Standardgeschwindigkeit darstellt,

1 = die Kassette wird mit 2000 Baud (Bit pro Sekunde) beschrieben.

Der Computer wählt beim Laden von der Kassette automatisch die richtige Geschwindigkeit zum Lesen. Zur besseren Datensicherheit wird der Wert SPEED WRITE 0, also keine Änderung, empfohlen.

SQ (*Kanal*)

Beispiel: Eingabe:
```
10 SOUND 1, 400, 200, 10, 1
20 PRINT SQ (1)
run
```
Anzeige: 132

Funktion: Die Anzahl der freien Plätze in der Tonwarteschlange (englisch Sound Queue) des bezeichneten Kanals (1, 2 oder 4) wird angegeben. Dabei steht:

1 - für Kanal A
2 - für Kanal B
4 - für Kanal C.

Diese Funktion gibt eine Bit-bezogene Zahl aus, in der die Bits folgende Bedeutung besitzen:

Bit 0, 1, 2:	Anzahl der freien Plätze in der Warteschlange
Bit 3, 4, 5:	Rendezvous-Zustand am oberen Ende der Schlange
Bit 6:	oberes Ende der Schlange ist im Haltezustand
Bit 7:	Kanal ist z. Z. aktiv

Bit 0 hat die niedrigste und Bit 7 die höchste Priorität (Bedeutung). Ist Bit 6 gesetzt, kann Bit 7 nicht gesetzt werden und umgekehrt. Sind die Bits 3, 4, 5 gesetzt, können die Bits 6 und 7 nicht gesetzt werden.

SQR (*numerischer Ausdruck*)

Beispiel: Eingabe: `PRINT SQR (81)`
Anzeige: `9`

Funktion: Berechnet die Quadratwurzel (engl. SQuare Root) des in Klammern stehenden numerischen Ausdrucks.

STEP siehe FOR-NEXT-Schleife

STOP

Beispiel: Eingabe:
```
10 FOR I = 32 TO 100
20 PRINT CHR$ (I); : NEXT
30 STOP
40 FOR I = 100 TO 200
50 PRINT CHR$ (I); : NEXT
run
cont
```

Funktion: Ein BASIC-Programm wird gestoppt und mit der Anweisung CONT fortgesetzt. Damit können gewünschte Programmunterbrechungen, z.B. beim Testen, realisiert werden.

STR$ (*numerischer Ausdruck*)

Beispiel: Eingabe:
```
10 a$ = "++"
20 b = &X11111
30 PRINT a$ + STR$ (b)
run
```
Anzeige: `++ 31`

Funktion: Der in Klammern stehende numerische Ausdruck wird in einen Dezimalstring umgewandelt.

STRING$ (*Länge*, *Zeichenkette*)

Beispiel: Eingabe: `PRINT STRING$ (10, "#")`
Anzeige: `##########`

Funktion: Das in Anführungsstrichen stehende Zeichen wird sovielle Male, wie die Länge angibt, ausgegeben. Wertebereich der Länge: von 0 bis 255 Zeichen Das gleiche Ergebnis wird durch das Eingeben von:

```
PRINT STRING$ (10, 35)
```

erreicht, wobei die 35 dem ASCII-Wert des Zeichens "#" entspricht.

SWAP siehe WINDOW

SYMBOL *Zeichennummer, Anzahl von Zeilen*

Beispiel: Eingabe:

```
 10 MODE 1 : SYMBOL AFTER 121
 20 Zeile 1 = 19 : REM 00010011 duale Darstellung
 30 Zeile 2 = 51 : REM 00110011
 40 Zeile 3 =115 : REM 01110011
 50 Zeile 4 =255 : REM 11111111
 60 Zeile 5 =255 : REM 11111111
 70 Zeile 6 =112 : REM 01110000
 80 Zeile 7 = 48 : REM 00110000
 90 Zeile 8 = 16 : REM 00010000
100 SYMBOL 121, Zeile 1, Zeile 2, Zeile 3, Zeile 4,
    Zeile 5, Zeile 6, Zeile 7, Zeile 8
110 PRINT "Was zeigt nun beim Tastendruck das y?"
run
```

Funktion: Ein Zeichen wird in seiner Form auf dem Bildschirm verändert oder neu erstellt. Dabei muß jeder Parameter K eine Zahl zwischen 0 und 255 enthalten. Damit der Speicherplatz für das neu zu erstellende Zeichen geschaffen wird, ist zu Beginn die Anweisung

SYMBOL AFTER N

zu verwenden. Der Wert N (0 bis 255) legt fest, ab welchem ASCII-Code im Zeichenvorrat Veränderungen vorgenommen werden können. Standardmäßig ist 240 eingestellt, deshalb können die letzten 16 Zeichen des Zeichensatzes ohne Benutzung von SYMBOL AFTER verändert werden.

Jedes neu erstellte Zeichen kann unabhängig von der Tastatur durch PRINT CHR$ (K) angezeigt werden. Symbol K folgen acht Parameter, die jeweils eine waagerechte Bildpunktzeile darstellen. Ein Zeichen wird in einer 8 * 8 Matrix auf dem Bildschirm angezeigt. Beim Cursor sind z.B. alle 8 * 8 Bildpunkte gesetzt. SYMBOL verändert jedoch achtmal eine waagerechte Bildpunktezeile, deshalb sind acht Parameter anzugeben.

Für das neue Zeichen wird folgendes definiert:

Zeilen-Parameter	Muster-Bitwerte(dual)	Dezimalwert
1	00010011	= 19
2	00110011	= 51
3	01110011	= 115
4	11111111	= 255
5	11111111	= 255
6	01110000	= 112
7	00110000	= 48
8	00010000	= 16

Aus der Kombination der Bitwerte 1 kann das darzustellende Zeichen bereits erkannt werden. Diese erstellten Dezimalwerte sind die acht Parameter, die SYMBOL K folgen. Dabei ist K der ASCII-Code von y (Kleinbuchstabe). Die Angabe dieser Zahl bewirkt, daß das neue Zeichen auf dem Buchstaben y liegt und somit auf der Tastatur eingebbar ist. Die Angabe SYMBOL 255, 19, 51, 115, 255, 255, 112, 48, 16 legt auf den ASCII-Code 255 das veränderte Zeichen. Treten 0 am Schluß auf, können diese weggelassen werden.

Als binäre Parameter wird folgendes eingegeben: SYMBOL 255, &X00010011,...

Es braucht bei dieser Schreibweise keine Umrechnung in die Dezimalschreibweise erfolgen. Anzeige dieses Zeichens erfolgt mit: PRINT CHR$ (255)

SYMBOL AFTER *ganze Zahl*

Beispiel: Eingabe: SYMBOL AFTER 121

Funktion: Die Anzahl der selbst zu erstellenden Zeichen wird festgelegt, indem der ASCII-Code eines Zeichens aus dem Zeichensatz angegeben wird, ab welcher Stelle Änderungen vorgenommen werden sollen. Alle nachfolgen den Codes können dann veränderte Zeichen erhalten, wie z.B. bei SYMBOL 255, 19, 51, 115, 255, 255, 112, 48, 16 wird das vorherige Zeichen überschrieben (siehe Anhang B). Standardmäßig ist 240 eingestellt, d.h. 16 Zeichen können vom Benutzer selbst erstellt werden. Ist als ganze Zahl 40 angegeben, können alle Zeichen von ASCII-Code 40 bis 255 verändert werden.

SYMBOL AFTER 256 verhindert das Erstellen neuer Zeichen.
Erscheint die Anweisung SYMBOL AFTER 0, bekommen alle veränderten Zeichen ihre vorherige Bedeutung wieder.

Wurde der Wert von HIMEM (mit MEMORY) verändert bzw. ein Dateipuffer (mit OPENIN oder OPENOUT) eröffnet, kann diese Anweisung nicht benutzt werden. Sonst erscheint die Fehlermeldung 5 (Improper argument).

TAB siehe PRINT TAB

TAG [*# Stream*]

Beispiel: Eingabe:
```
10 MODE 2:INK 2,2:PEN 2
20 ORIGIN 360,200
30 TAG
40 PLOT-360,0:DRAW 360,0
50 MOVER-88,6:PRINT CHR$ (246);
60 MOVER-30,-9:PRINT "X";
70 PLOT 0,-200:DRAW 0, 200
80 MOVER-4,-4:PRINT CHR$ (244);
90 MOVER-17,10:PRINT "y";
100 TAGOFF
run
```

Funktion: Immer auf der Position des Grafik-Cursors erscheint der dem Stream zugeordnete Text. In dieser Form können Text und Symbole mit der Grafik gemischt werden oder auch Bildpunkt für Bildpunkt bewegt werden. Ohne Angabe des Parameters ist Standardwert Stream #0. Auf dem Grafik-Cursor liegt die linke, obere Ecke des Textzeichens. Beim Stream-Parameter 0 wird TAG abgeschaltet, wenn BASIC im Direktmodus zurückkehrt. (TAG heißt Text At Graphics)

TAGOFF [*# Stream*]

Beispiel: siehe TAG

Funktion: Setzt die TAG-Anweisung des angegebenen Streams (Stream #0 standardmäßig eingestellt ohne Parameterangabe) zurück. Der Text wird auf die Stelle ausgegeben, an der sich der Text-Cursor vor der TAG-Anweisung befand.

TAN (*numerischer Ausdruck*)

Beispiel: Eingabe: `PRINT TAN (PI/4)`
Anzeige: 1

Funktion: Der TANgens des numerischen Ausdrucks (Wertebereich zwischen -200000 und +200000) wird berechnet. Die Eingabe wird im Bogenmaß erwartet, wenn nichts anderes vereinbart wurde. Dabei stellt DEG Winkelmaß und RAD Bogenmaß ein.

TEST (*x-Koordinate*, *y-Koordinate*)

Beispiel: Eingabe:
```
10 CLS
20 PRINT "Textausgabe im Fenster 0"
40 PRINT "So ist die Nummer des Farbstiftes"
30 PRINT TEST (100, 310).
```

Funktion: Der Grafik-Cursor wird auf den angegebenen absoluten x/y-Koordinatenpunkt gesetzt, wobei die Farbstift-Nummer, die auf dieser Stelle benutzt wurde, ausgegeben wird.

TESTR (*x-Versatz*, *y-Versatz*)

Beispiel: Eingabe:
```
10 MODE 0
20 FOR I = 32 TO 55
30 FOR N = 1 TO 10:PEN n
40 PRINT CHR$ (i): NEXT n, i
50 MOVE 100,350:PEN 1
60 FOR a = 1 TO 11:LOCATE a, a+13
70 PRINT "Zeichen"; TESTR (2,-15):NEXT
```

Funktion: Unter der Angabe der Farbstift-Nummer wird der Grafik-Cursor von seiner augenblicklichen Position um die angegebenen x/y-Werte verschoben. Die Farbstift-Nummer enthält den Wert des an dieser Position benutzten Farbstiftes.

TIME

Beispiel: Eingabe:
```
 10 INPUT "Stunde Minute Sekunde"; h,m,s
 20 a = INT(TIME/300)
 30 WHILE h 24
 40 WHILE m 60
 50 WHILE b 60
 60 b = (INT(TIME/300) -a) +s
 70 LOCATE 1,1:PRINT h; m; b
 80 WEND
 90 b = 0 : s = 0: m = m+1
100 GOTO 20
110 WEND 120 m = 0:h = h+1
130 WEND 135 h=1
140 GOTO 40
run
```

Funktion: Es erfolgt die Zeitangabe, die nach dem Einschalten bzw. dem Zurücksetzen des Computers vergangen ist, dabei ist eine Einheit eine Dreihundertstelsekunde. Das Laden oder Abspeichern von/auf Kassette/Diskette wird zeitlich abgezogen, also nicht berücksichtigt.

TO siehe FOR
TROFF
TRON

Beispiel: Eingabe: TRON
run

Funktion: Wird vor dem Start eines Programms TRON (TRace ON) eingegeben, kann das Programm in den Ausführungen der Anweisungen genau verfolgt werden. Dabei erscheint vor jeder Zeile, in Klammern [ ] angegeben, die Zeilennummer. Durch TROFF (TRace OFF) erfolgt das Aufheben der Anweisung TRON (Anwendung bei Fehlersuche).

UNT (*Adreßausdruck*)

Beispiel: Eingabe: PRINT UNT (&FF00)
Anzeige: -256

Funktion: Eine ganze Zahl (englisch INTeger) zwischen -32768 und +32767 wird ausgegeben. Sie entspricht dem Zweierkomplement des in Klammern stehenden vorzeichenlosen (englisch UNsigned) Adreßwertes.

UPPER$

Beispiel: Eingabe:
```
10 A$ = "diese buchstaben werden alle als gr
   oßbuchstaben angezeigt"
20 PRINT UPPER$ (a$)
run
```

Funktion: Die im String angegebenen Buchstaben werden alle als Großbuchstaben ausgegeben.

USING siehe PRINT USING

VAL (*String*)

Beispiel: Eingabe:
```
10 a$ = "123"
20 B$ = "55"
30 PRINT a$; b$
40 PRINT VAL (a$) + VAL (b$)
run
```

Funktion: Der numerische Wert (engl. VALue) wird, einschließlich Vorzeichen und Dezimalpunkt, wiedergegeben.

VPOS (*# Stream*)

Beispiel: Eingabe: PRINT VPOS (#0)

Funktion: Gibt die aktuelle Position des Text-Cursors, bezogen auf den oberen Rand des Bildschirmbereiches, an.

WAIT *Interfaceadresse*, *Maske* [,*Inversion*]

Beispiel: Eingabe: WAIT & FF34, 20, 25

Funktion: WAIT hält den Programmablauf in einer Warteschleife, bis am angegebenen Interface (Ein-/Ausgabeschnittstelle) der erwartete Wert (0 bis 255) eingelesen wird. Von einer Portadresse wird ein bestimmter Wert (Bit-Kombination) eingelesen und analysiert. Alle Parameter sind numerische Ausdrücke, die evtl. gerundet und auf das niederwertigste Byte begrenzt werden.
Mit der WAIT-Anweisung wird folgendes realisiert:
- Bitmuster einlesen,
- Negieren der Bits des eingelesenen Musters, die in der Negierungsmaske (Inversion) angegeben sind,
- Herauslösen der in der Testmaske (Maske) angegebenen Bits,
- hat eins dieser Bits (laut Testmaske) den Wert 1, wird das Programm weiter abgearbeitet, wenn nicht,
- dann wird am Port ein neues Bitmuster eingelesen und der Ablauf beginnt von neuem.

Die Inversion darf bei der Angabe entfallen, BASIC nimmt dann den Wert 0 an.

Achtung: Vorsicht beim Benutzen dieser Anweisung, da durch falsche Angaben unerwünschte Reaktionen des Computers erfolgen können.

WEND

Beispiel: WEND

Funktion: Ist der Abschluß einer WHILE-Schleife. Die zugehörige WHILE-Anweisung wird im Programm automatisch auf gefunden.

WHILE *logischer Ausdruck*

Beispiel: WHILE h ^ 13 oder
WHILE TIME t + 3600

Funktion: In Abhängigkeit des logischen Ausdrucks wird der Programmteil so oft wiederholt, wie die Bedingung des logischen Ausdrucks zutrifft. Dabei steht WHILE am Anfang der Schleife und enthält die Bedingung.

WIDTH *ganzzahliger Ausdruck*

Beispiel: WIDTH 60

Funktion: Die Anzahl der Zeichen für eine Zeile wird für die Druckausgabe angegeben. Bei längeren Zeilen werden automatisch ein Wagenrücklauf und ein Zeilenvorschub eingefügt. Ohne Angabe des Parameters nimmt der Computer den Standardwert 132 an (WIDTH-dt Breite). WIDTH 255 bewirkt, daß der Drucker den Zeilenabschluß festlegt, hier kann ein Zeilenvorschub und Wagenrücklauf mit der PRINT-Anweisung durch Semikolon und Komma erfolgen.

WINDOW [*# Stream,*] *links, rechts, oben, unten*

Beispiel: Eingabe:

```
10 MODE 0
20 WINDOW #1,5, 16, 1, 18
30 INK 3, 6 : PAPER 3
40 CLS
run
```

Funktion: Stellt die Größe und Lage eines Fensters (englisch window) oder Bildschirmbereiches ein. Beim Bestimmen der Parameter links und rechts sollte der eingestellte Modus berücksichtigt werden. Ohne Angabe des Stream-Parameters ist automatisch Stream #0 (gesamter Bildschirm) eingestellt.

WINDOW SWAP *Stream*

Beispiel: Eingabe:

```
10 MODE 1 : INK 1,26 : INK 2,6 : PEN 1 : PAPER 0
20 WINDOW #1, 15, 35, 8, 18 : PAPER # 1,2
30 CLS : PRINT "Stream 0 - Fenster 0"
40 FOR i = 1 TO 1000 : NEXT
50 CLS # 1 : PRINT #1, "Stream 1 - Fenster 1"
60 FOR i = 1 TO 1000 : NEXT
70 FOR a = 1 TO 3
80 WINDOW SWAP 0, 1
90 PRINT "Fenster 0"
100 FOR i = 1 TO 1000 : NEXT
110 PRINT # 1, "Fenster 1"
120 FOR i = 1 TO 1000 : NEXT i, a
run
```

Funktion: Die beiden Streams (Bildschirmbereiche oder Fenster) werden gegeneinander ausgetauscht. Die Nummern der Streams müssen ohne # angegeben werden.

WRITE [*# Stream*,] [*Ausgabeliste*]

Beispiel: Eingabe:
```
10 a = 10 : a$ = "Beispiel"
20 OPENOUT "Datei"
30 WRITE # 9, a, a$
40 CLOSEOUT : PRINT "Datenausgabe"
run
```

Funktion: Die in der Ausgabeliste enthaltenen Werte werden auf den angegebenen Stream ausgegeben. Die Werte sind untereinander durch Komma zu trennen. Strings sind in Anführungszeichen einzuschließen, Stream #9 bedeutet Ausgabe auf Kassette bzw. Diskette.

Einlesen von Variablen:

```
10 OPENIN "Datei" : INPUT #9, a, a$
20 CLOSEIN : PRINT "Werte a, a$";
30 PRINT a, a$
run
```

Ausdruck **XOR** *Ausdruck*

Beispiel: Eingabe:
```
10 a = 10 : b = 20 : c = 40
20 IF a b XOR b ^ a THEN PRINT "richtig"
30 IF c ^ a XOR b ^ c THEN PRINT "richtig"
   ELSE PRINT "falsch"
run
```

```
Eingabe: PRINT 1 XOR 1
Anzeige: 0
Eingabe: PRINT 0 XOR 0
Anzeige: 0
Eingabe: PRINT 1 XOR 0
Anzeige: 1
```

Funktion: Mit ganzen Zahlen werden Boolesche Operationen durch geführt. Besitzen beide Argumente verschiedene Werte, ist demzufolge das Ergebnis 0. XOR, auch logisches exklusives ODER, verknüpft bitweise die Werte in den Argumenten. In folgender Tabelle wird ein Beispiel aufgeführt. Dabei besitzen Variable A und Variable B folgende Bit-Werte:

Binärwert	Bit-Nr.	Argument A	Argument B	A XOR B
2^0	0	0	1	1
2^1	1	1	0	1
2^2	2	1	1	0
2^3	3	0	0	0
2^4	4	0	0	0
2^5	5	0	0	0
2^6	6	0	0	0
2^7	7	0	0	0

Die Werte A und B sind bitweise dargestellt. Der Wert A stellt in den acht Bits eines Bytes die 6 dar, da
2 ↑ 1 (2 hoch 1) = 2 und
2 ↑ 2 (2 hoch 2) = 4, demzufolge 2 + 4 = 6 ist.

Für den Wert B ergibt sich analog:

2 ↑ 0 (2 hoch 0) = 1
2 ↑ 2 (2 hoch 2) = 4

Die Summe ergibt 5. Also heißt der Wert B = 5. In A XOR B ergibt sich durch die Operation der Wert 3.

XPOS

Beispiel: Eingabe:
```
10 MODE 2
20 MOVE 25, 25
30 DRAW 640, 25
40 PRINT "Position auf der x-Achse";
50 PRINT XPOS
run
```

Funktion: Die Position des Grafik-Cursors auf der x-Achse des Grafikbildschirms wird ausgegeben.

YPOS

Beispiel: Eingabe:
```
10 MODE 1
20 DRAW 400, 100
30 PRINT "Position auf der y-Achse",
40 PRINT YPOS
run
```

Funktion: Die Position des Grafik-Cursors auf der y-Achse des Grafikbildschirms wird angegeben.

ZONE *ganzzahliger Ausdruck*

Beispiel: Eingabe:
```
10 MODE 1
20 FOR i = 1 TO 20
30 PRINT "Test", "A"
40 ZONE i : NEXT
run
```

Funktion: Die Breite der Druckzone, der durch Kommata getrennten Ausdrücke in der PRINT-Anweisung (Standardeinstellung beträgt 13 Zeichenpositionen) wird verändert. Der ganzzahlige Ausdruck kann Werte von 1 bis 255 annehmen.

3.13. Grafik

3.13.1. Grafikzeichen

Im Computer sind verschiedene Zeichen gespeichert, die durch die Anweisung:

PRINT CHR$(Code des Zeichens)

auf dem Bildschirm angezeigt werden können. Der in Klammern stehende Ausdruck enthält den Code des Zeichens. Er sollte einen Wert zwischen 32 und 255 besitzen.
Das folgende kleine Programm zeigt die im Computer enthaltenen Zeichen auf dem Bildschirm an.

Beispiel:
```
10 FOR I = 32 TO 255
20 PRINT CHR$(I);I
30 NEXT
run
```

Das Drücken der Taste [ESC] stoppt die schnelle Anzeige der Zeichen und nochmaliges Drücken unterbricht sie. Im Anhang B ist der Zeichensatz des Computers aufgeführt.

3.13.2. Zeichen positionieren

Um den Cursor auf eine bestimmte Stelle des Bildschirms zu setzen, kann die Anweisung LOCATE verwendet werden. Ohne LOCATE wird der Cursor anfangs immer in der linken, oberen Ecke positioniert. Dieser Punkt ist der Koordinatenpunkt 1,1 , d.h. erste Stelle auf der x-Achse und erste Stelle auf der y-Achse. Hierbei ist der eingestellte Modus zu berücksichtigen.

Für den Modus 0 ist die letzte Zeichenposition in der Zeile 4 die Position 20,4. Demzufolge wird mit

Beispiel:
```
10 MODE 0
20 LOCATE 20,4
30 PRINT CHR$(248)
run
```

das Zeichen auf dem Bildschirm dargestellt. Im Modus 2 wäre die letzte Zeichenposition dann 80,4 , wie das Beispiel zeigt:

Beispiel:
```
10 MODE 2
20 LOCATE 80,4
30 PRINT CHR$(248)
run
```

Zur gleichmäßigen Bewegung von Symbolen auf dem Bildschirm kann die Anweisung FRAME benutzt werden. Dafür wird das erste Beispiel wie folgt verändert:

Beispiel:
```
10 MODE 0
20 FOR X = 20 TO 1 STEP -1
30 LOCATE X,X
40 FRAME
50 PRINT CHR$(248)
60 NEXT
run
```

3.13.3. Grafik-Cursor positionieren

Zur Bestimmung der Position des Grafik-Cursors auf dem Bildschirm wird z.B. die PLOT-Anweisung verwendet. Hierzu werden die Bild- punktkoordinaten benutzt (Bildpunkt ist ein Pixel, die kleinste Einheit der Bildschirmdarstellung).

Hinweis: Der Grafik-Cursor ist unsichtbar und nicht mit dem Text-Cursor identisch.

In waagerechter Richtung sind auf dem Bildschirm im:

Modus 0 160 Bildpunkte
Modus 1 320 Bildpunkte
Modus 2 640 Bildpunkte darstellbar.

In senkrechter Richtung besitzt der Bildschirm 400 Bildpunkte.

Der Koordinatenursprung (Punkt 0,0) liegt in der unteren, linken Ecke des Bildschirms. Die Pixel-Koordinaten verändern sich nicht. Sie bleiben in allen drei Modi gleich.

3.13.4. Liniendarstellung

Linien werden mit der Anweisung DRAW auf dem Bildschirm gezeichnet Zuvor wird mit PLOT der Grafik-Cursor positioniert und anschließend mit DRAW eine Linie gezogen.

Beispiel:
```
10 CLS
20 PLOT 30,30
30 DRAW 640,399
run
```

Das vorherige Beispiel zeigt eine Linie, die im Punkt 30,30 beginnt und diagonal in die obere, rechte Ecke läuft.

Mit der Anweisung MOVE wird der Grafik-Cursor wie mit PLOT auf einen bestimmten Koordinatenpunkt gesetzt. Jedoch wird hier auf der neuen Position kein Bildpunkt dargestellt.
Mit

```
CLS
MOVE 10,399
```

befindet sich der Grafik-Cursor in der linken, unteren Ecke, wobei er jedoch nicht zu sehen ist (siehe Hinweis in Abschnitt 3.13.3.). Durch das Ziehen einer Linie kann die Position gezeigt werden. Es ist nun

```
DRAW 10,300
```

einzugeben. Von der linken, oberen Ecke wird nach unten eine Linie gezeichnet.

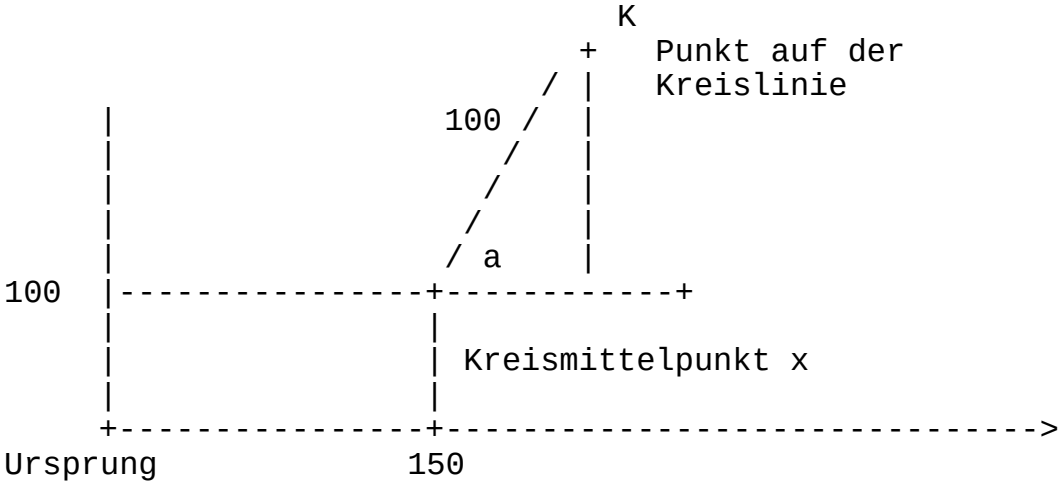
3.13.5. Kreisdarstellung

Für die Darstellung eines Kreises auf dem Bildschirm gibt es folgende Möglichkeiten:

- Koordinatenpunkte für jeden Bildpunkt der Kreislinie setzen (plotten).

Dabei werden dem Punkt K auf der Kreislinie folgende x- und y-Koordinaten zugeordnet:

```
x = 100 * COS(a)
y = 100 * SIN(a)
```



Koordinatensystem des Bildschirms

Der Mittelpunkt des Kreises liegt auf den Koordinaten 150,100.

Das folgende Beispiel zeichnet einen Kreis auf dem Bildschirm.

Beispiel:
```
10 CLS
30 FOR I = 0 TO 2 * PI STEP .01
40 PLOT 150+100*COS(I),100+100*SIN(I)
50 NEXT
run
```

Die Anweisung ORIGIN vereinfacht das Zeichnen eines Kreises, in dem der Koordinatenursprung in die Kreismitte verlegt wird. Das Programm sieht dann so aus:

```
10 CLS
20 ORIGIN 150,100
30 FOR I = 0 TO 2 * PI STEP .01
40 PLOT 100*COS(I),100*SIN(I)
50 NEXT
run
```

Die Anweisung FILL füllt einen Bildschirmbereich, der durch Linien bzw. durch den Rand begrenzt ist, aus.
Das vorherige Programm zeichnet einen Kreis auf den Bildschirm. Der Grafik-Cursor befindet sich nach dem Zeichnen des Kreises auf der Position 100,0 bezogen auf den durch ORIGIN 150,100 festgelegten neuen Koordinatenursprung in der Mitte des Kreises. Wird der Grafik-Cursor mit

```
MOVE 0,0
```

auf den Mittelpunkt des Kreises gesetzt und

```
FILL 3
```

eingegeben, färbt sich der Kreis rot.

Diese Änderung zeigt Farbeffekte.

```
60 MOVE 0,0
70 FOR i = 2 TO 15: FILL i
80 FOR a = 1 TO 600: NEXT a,i
run
```

3.13.6. Transparente Schrift

Auf dem Textbildschirm kann mit transparenten Zeichen gearbeitet werden.

Der Transparent-Modus wird mit CHR$(22) und CHR$(1) ein- und mit CHR$(22) und CHR$(0) abgeschaltet.

Beispiel:
```
10 PRINT#5, CHR$(22); CHR$(1)
20 LOCATE#5, 20, 10: PRINT#5 "?????#"
30 LOCATE#5, 20, 10: PRINT#5
40 PRINT#5, CHR$(22); CHR$(0)
run
```

Wie das Programm zeigt, können diese Fragezeichen gestrichen werden. Zeichen können also übereinander gelagert werden, auch in verschiedenen Farben.

3.13.7. Farbstift-Modi

In verschiedenen Anweisungen wird auf die Farbstift-Modi verwiesen. Es ist möglich, in einem Grafik-Farbstift-Modus zu zeichnen, der eine schon vorhandene Grafik auf dem Bildschirm mit neu zu zeichnender Grafik verbindet. D.h., daß die vorhandenen gefärbten Bildpunkte einer Grafik mit der neuen Grafik auf dem Bildschirm verknüpft werden. Die Farbe der Bildpunkte der neuen Grafik errechnet sich, indem der alte Farbstift für das Pixel mit dem Farbstift für GRAPHICS PEN oder PAPER logisch verknüpft wird. Die logischen Verknüpfungen erfolgen durch XOR, AND und OR. Als weitere Parameter kann dieser Farbstift-Modus in den Anweisungen DRAW/DRAWR, PLOT/PLOTR, MOVE/MOVER verwendet werden. Ebenso gilt die Steuercodefolge CHR$(Modus) in einer PRINT-Anweisung.

Dabei bewirken die Werte:

1 = Farbe der Bildpunkte gemäß XOR-Verknüpfung
2 = Farbe der Bildpunkte gemäß AND-Verknüpfung
3 = Farbe der Bildpunkte gemäß OR-Verknüpfung
0 = Zwangs-Modus ist der standardmäßig eingestellte Grafik-Farbstift.

3.14. Tonerzeugung

Die SOUND-Anweisung besitzt sieben Parameter:

- Kanalstatus
- Tonperiode
- Dauer der Note
- Lautstärke
- Lautstärkehüllkurve
- Tonhüllkurve
- Geräuschperiode

Kanalstatus

Die für diesen Parameter aufgeführten Werte sind bitorientiert (bitsignifikant), d.h. daß jedem Bit eine Wertung zugeordnet wird (acht Bit sind ein Byte und in der Kombination dieser Bits kann ein Byte Zahlen von 0 bis 255 darstellen).
Der Computer besitzt drei Kanäle. Über einen dieser Kanäle kann der Ton ausgegeben werden. Beim Anschluß eines Stereogerätes geht ein Kanal über den linken und der andere über den rechten Lautsprecher. Der dritte teilt sich auf beide Kanäle zu gleichen Teilen auf.

Zuordnung der Werte zu den Kanälen:

1 - Kanal A
2 - Kanal B
4 - Kanal C

Zur Benutzung von mehreren Kanälen können diese Werte addiert werden. Für Kanal C ergibt sich SOUND 4,239.

Der Kanal C besitzt deshalb den Wert 4, weil die Werte aus Zweierpotenzen (binär oder dual) gebildet sind. Da 2 hoch 0 (2 ↑ 0) = 1, 2 hoch 1 (2 ↑ 1) = 2 ist, ergibt sich für Kanal C der Wert 4.
In jedem Binärsystem werden die Zahlen nur durch 0 und 1 dargestellt. Die Zahlen 1 bis 5 werden wie folgt gebildet:

2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0	= 8 Bit
128	64	32	16	8	4	2	1	Werte
0	0	0	0	0	0	0	0	= 0
0	0	0	0	0	0	0	1	= 1
0	0	0	0	0	0	1	0	= 2
0	0	0	0	0	0	1	1	= 3
0	0	0	0	0	1	0	0	= 4
0	0	0	0	0	1	0	1	= 5

Werden nun für den Zustand eingeschaltet 1 und für ausgeschaltet 0 gewählt, ergibt sich für SOUND 4,239

C	B	A	
1	0	0	Kanal C eingeschaltet
0	0	1	Kanal A eingeschaltet
0	1	1	Kanäle A und B eingeschaltet

Für Kanal A und B eingeschaltet lautet SOUND 3,190.

Ebenso können die Kanalwerte addiert werden.

Tonperiode

Als Tonperiode kann eine Zahl zwischen 1 und 4095 angegeben werden. Jedoch nur wenige Tonperioden erzeugen Töne, die zur Tonleiter gehören. Im Anhang E "Noten und Tonperioden" ist aufgeführt, welche Tonperioden Töne der Tonleiter erzeugen. Aus dieser Tabelle ist zu entnehmen, daß die Tonperiode 478 die Note C oder 159 die Note G erzeugt. Die Tonperiode 0 ergibt keinen Ton.

Dauer der Note

Der angegebene Wert für die Dauer einer Note mißt die Länge in Hundertstelsekunden. Er kann zwischen 1 und 32767 liegen. Beträgt dieser Wert 0, so bestimmt die Lautstärkehüllkurve die Länge des Tones. Negative Werte geben die Anzahl der Wiederholungen der Hüllkurve an, z.B. -5 heißt, daß die Lautstärkehüllkurve fünf mal wiederholt wird.

Lautstärke

Der Wert für die Lautstärke eines Tones kann zwischen 0 und 15 liegen. Ohne Wertangabe wird automatisch der Wert 12 angenommen. Wird der Parameter Lautstärke mit einer Lautstärkehüllkurve definiert, so ist der Wert für die Lautstärke in der SOUND-Anweisung die Anfangslautstärke.

Lautstärkehüllkurve

Die Anweisung ENV definiert die Hüllkurve der Lautstärke der erzeugten Töne.

```
Beispiel:   ENV     1,7,2,6,7,-2,8
            SOUND   1,239,0,0,1
```

ENV muß immer vor SOUND stehen, da der Wert der Lautstärkehüllkurve der fünfte Parameter der SOUND-Anweisung ist, hier ist es die Ziffer 1.

In der ENV-Anweisung folgt als erstes die Hüllkurvennummer, dann die Dauer des Tones und Lautstärkeverlauf. Hierdurch können in der SOUND-Anweisung diese Parameter (Dauer und Lautstärke) 0 gesetzt werden.

Das oben genannte Beispiel sagt aus, daß die Lautstärke in sieben Stufen ansteigt, jede Stufe um Lautstärkegrad 2 zunimmt und 6 Hundertstelsekunden dauert. Anschließend fällt die Lautstärke in 7 Stufen um -2 mit der Dauer von 8 Hundertstelsekunden.

Nach der Angabe der Hüllkurvennummer (erster Parameter der ENV-Anweisung) folgt der erste Hüllkurvenabschnitt. Die Parameter des Hüllkurvenabschnitts sind:

1. Tonstufen beim An- bzw. Abschwellen des Tons
2. Lautstärkegrad, um den jede Stufe anwachsen bzw. fallen soll
3. Dauer des Tones auf jeder Lautstärkestufe

Die Gesamtzeit (Dauer) eines Hüllkurvenabschnitts wird berechnet, indem der erste Wert (Stufenanzahl) mit dem dritten Wert (Dauer einer Stufe) multipliziert wird.
Auch wird die gesamte Lautstärkedifferenz innerhalb eines Hüllkurvenabschnittes aus Stufenanzahl mal dem Anstieg/Abfall der einzelnen Stufen berechnet. Die gesamte Lautstärkedifferenz kann 15 nicht überschreiten.
Dabei ist die Gesamtlänge einer Hüllkurve aus mehreren Hüllkurvenabschnitten gleich der Summe der Länge aller einzelnen Abschnitte.

Die Anfangslautstärke in der SOUND-Anweisung muß nicht 0 sein. Folgendes Beispiel zeigt dies:

```
Beispiel: ENV   4,4,-2,1,10,0,1,3,1,1
          SOUND 1,200,0,,15,4
```

Die Hüllkurve vier besitzt drei Hüllkurvenabschnitte.

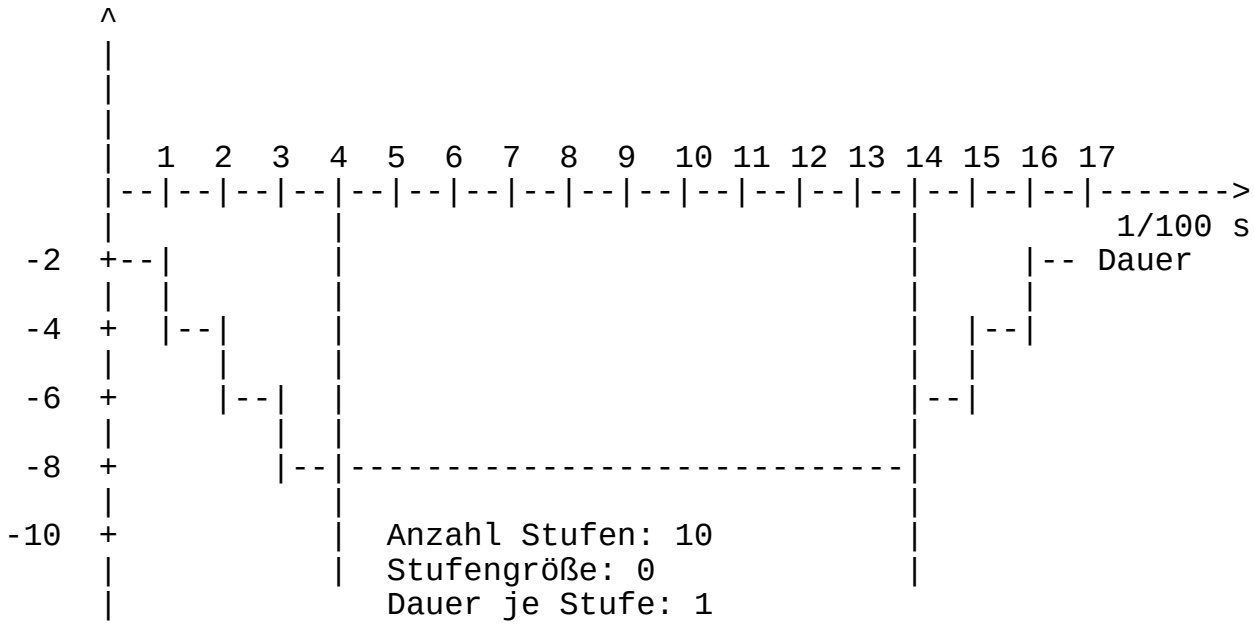
Einstellung:

Abschnitt 1:
Lautstärke ist in vier Stufen geteilt und nimmt um 2 Lautstärke grade je Stufe ab. Die Länge einer Stufe beträgt 1/100 Sekunde.

Abschnitt 2:
Dieser Abschnitt besitzt zehn Stufen. Jede Stufe hat einen Lautstärkean-/-abstieg von 0, d.h. die Lautstärke verändert sich nicht. Die Länge einer Stufe ist 1/100 Sekunde.

Abschnitt 3:
Der dritte Abschnitt besteht aus drei Stufen. In jeder Stufe nimmt die Lautstärke um einen Lautstärkegrad zu. Die Länge einer jeden Stufe beträgt 1/100 Sekunde.

Folgendes Bild zeigt die grafische Darstellung der Lautstärke-hüllkurve, die mit der eben erläuterten ENV-Anweisung definiert wird.



Anzahl Stufen: 4
Stufengröße: -2
Dauer je Stufe: 1

Bild 1: Darstellung von ENV 1,4,-2,1,10,0,1,3,1,1

ENV kann maximal 16 Werte besitzen (mit Nummer der Hüllkurve). Ist die Lautstärke beim Anstieg größer als 15 oder beim Absinken kleiner als 0, so ergibt sich das entgegengesetzte Lautstärke-Maximum, d.h.

15 wird 0 und
0 wird 15.

Die Angabe "Anzahl der Stufen" besitzt einen Wertebereich von 0 bis 127 und die "Stufengröße" einen Wertebereich zwischen -128 und +127. Negative Werte verursachen einen Lautstärkeabfall. "Dauer der Stufe" kann Werte zwischen 0 und 255 annehmen.

Tonhüllkurve

Die Anweisung ENT definiert eine Tonhüllkurve, die z.B. einen um seine Tonperiode schwingenden Ton erzeugt. Die Nummer dieser Hüllkurve steht an sechster Stelle der SOUND-Anweisung. Ebenso wie bei ENV muß auch ENT vor SOUND definiert werden.

Beispiel:
```
ENT 2,5,2,1,5,-2,1
SOUND 1,200,10,15,,2
```

Die Tonhüllkurve zwei wird definiert. Sie besteht aus zwei Hüllkurvenabschnitten.
Der erste Abschnitt besitzt fünf Stufen. Je Stufe wird die Ton um 2 vergrößert. Die Dauer einer Stufe beträgt 1/100 Sekunde.
Der zweite Abschnitt besitzt fünf Stufen. Je Stufe wird die Tonperiode um 2 verringert. Die Dauer einer Stufe beträgt 1/100 Sekunde.
Die Gesamtlänge ergibt sich aus 5+5=10 und dieser Wert muß in die SOUND-Anweisung eingetragen werden, da die Tonhüllkurve keinen Einfluß auf die Länge des Tones hat. Stimmt die Angabe nicht, ist sie z.B. zu kurz, wird der letzte Teil der Hüllkurve abgeschnitten. Bei größerer Dauer, als in der Tonhüllkurve definiert ist, bleibt der zuletzt eingestellte Ton für den Rest der Dauer erhalten. Gleich verhält es sich, wenn mit der Lautstärkehüllkurve die Länge einer Note bestimmt wird.
(In der SOUND-Anweisung wurde kein Wert für Lautstärke angegeben.)
Meist ist die Tonhüllkurve kürzer als die Tonlänge. Deshalb wird z.B. eine Tonhüllkurve wiederholt über die gesamte Tonlänge. Durch eine negative Hüllkurvennummer wird dies erreicht.

```
Beispiel: ENT -2,3,1,1,3,-1,1
          SOUND 1,200,100,15,,2
```

So wird ein schwingender Ton erzeugt. Zu beachten ist beim Definieren einer Tonhüllkurve, daß der Ton symmetrisch um die anfangs eingestellte Tonperiode schwingt.

Tonhüllkurven haben einen Wertebereich von 1 bis 15, dabei gibt ein negativer Wert die Wiederholung an.

Wertebereich "Anzahl der Stufen": 0 bis 239
Wertebereich "Stufengröße": -128 bis 127
Wertebereich "Dauer": 0 bis 255

Geräuschperiode

Der KC compact hält 31 verschiedene Arten von Rauschen zur Begleitung des Tones bereit, die durch den Parameter Geräuschperiode (von 1 bis 31) ausgewählt werden können.
Der siebente Wert bei SOUND gibt das Rauschen an, das den Ton begleitet. Für das Rauschen steht nur ein Kanal zur Verfügung, d.h. jede Wertänderung beendet auch das vorherige Rauschen.

```
Beispiel: ENV 4,15,1,2,15,-1,2

          FOR i=1 TO 8:SOUND 1,3600,0,0,4,15:NEXT
```

Das Beispiel erzeugt das Geräusch einer Lokomotive, ebenso kann es als Schlagzeugeffekt verwendet werden. Die Tondauer und die Lautstärke wurden in der SOUND-Anweisung 0 gesetzt, da die Hüllkurve diese Werte bestimmt.

Die oberen 5 Bits des Kanalstatus

Die Werte 8,16,32 für den Kanal bedeuten , daß sich der Ton mit einem anderen Kanal zum Rendezvous treffen soll.

Beispiel: SOUND 1,190,2000 SOUND 1,80,200

Bei der Eingabe der zweiten Anweisung ist festzustellen, daß die erste Anweisung noch nicht vollständig abgearbeitet ist. Grund für diesen Effekt ist, daß der Computer für jeden Kanal fünf SOUND-Anweisungen in einer "Warteschlange" speichern kann (Warteschlange steht für Speicherbereich oder Puffer).

Soll ein Ton auf Kanal B und danach ein Ton auf Kanal B und C gleichzeitig ausgegeben werden, ist dem Computer mitzuteilen, daß der Ton des Kanals C nicht eher beginnen darf, bis der erste Ton auf Kanal B beendet ist. Es muß also der Kanal C auf den Kanal B warten. Bezeichnet wird dies mit "Rendezvous" und kann folgendermaßen erzeugt werden:

Beispiel: SOUND 2,190,2000 SOUND 6,95,239

Der erste Ton geht zu Kanal B, der zweite zu Kanal B und C. Dabei wartet Ton C solange, bis Ton B fertig ist. Begrenzt wird diese Tonerzeugung darin, daß derselbe Ton an beide Kanäle geschickt wird und ein Ton sich im Wartezustand befindet. Im Beispiel wurde 95,239 an B und C ausgegeben.

Eine weitere Rendezvous-Möglichkeit besteht in:

```
SOUND 4,100,3000
SOUND 4+16,90,500
SOUND 2+32,159,500
```

Die erste Note des Kanals C wird ausgegeben. Die zweite Note von C trifft sich mit dem Ton von Kanal B und dieser trifft Note C. Dabei unterscheidet sich die zweite Note des Kanals C von der Note auf B. Sie sind miteinander verbunden und müssen deshalb warten, bis beide Kanäle für ein "Rendezvous" frei sind.

Binärzahl der Kanalnummer und deren Bedeutung:

	Rendezvouswerte Kanäle			Ausgabe der Kanäle		
Kanal	C	B	A	C	B	A
positiver Wert	32	16	8	4	2	1

Eine Note des Kanals B trifft sich mit C, so muß es heißen:

1	0	0	0	1	0

Binärwert = 100010, dezimal 2+32=34

Der Kanalstatus besitzt den Wert 34. Hieraus geht hervor, daß eine Note auf Kanal B zu spielen und auf eine Note zu warten ist, die zum Rendezvous mit Kanal C vorgesehen ist.
Wird zum Kanalstatus eine 64 (2 6) addiert, wird die Note gehalten, was heißt, der Ton wird erst nach der Anweisung RELEASE ausgegeben.

Wird zum Kanalstatus 128 (2 7) addiert, erfolgt das Löschen der Warteschlange des angegebenen Kanals. Für zu lange Töne kann somit ein schnelles Stoppen durchgeführt werden.

Ton stoppen mit: SOUND 1+128,0

Hier wird Kanal A gelöscht. Auch werden über Tastendruck mit [DEL] alle Tonkanäle im Direkt-Mode gelöscht.

Bei der Addition Kanalstatus plus 64 wird der Ton im Wartezustand gehalten. Dieser Zustand wird durch die Anweisung RELEASE aufgehoben. Die Zuordnung der Kanäle A=1, B=2, C=4 wird hier benutzt. Mit RELEASE 7 werden alle drei Kanäle freigegeben, denn 1+2+4=7.

Folgendes Beispiel zeigt die Verzögerung der Töne, die durch SOUND angegeben werden.

```
SOUND 1+64,80
SOUND 2+64,159
SOUND 4+64,200
RELEASE 3 FOR i=1 TO 600:NEXT:RELEASE 4
```

Wird einer Warteschlange (durch +64) im Haltezustand ein weiterer Ton zugeordnet, befinden sich alle Töne, die in der Warteschlange stehen, im Haltezustand. Ist die Anzahl der Töne einer Warteschlange größer als 4, hält das Programm an, bis ein Ton aus der Warteschlange abgearbeitet ist. Erst nach der Freigabe der Kanäle arbeitet es weiter.

Unterbrechungen

Unterbrechungen sind mit AFTER, EVERY oder nach ON BREAK GOSUB möglich. Diese Unterbrechungsmethode gibt in BASIC die Möglichkeit, ein Unterprogramm zur Tonerzeugung aufzurufen, wenn in der bestimmten Warteschlange ein Platz frei ist.
Hierzu ein Beispiel:

```
 10 t=10
 20 ON SQ(2) GOSUB 200
 30 LOCATE 1,1:PRINT "dezimal","hexadezimal"
 40 PRINT t,;HEX$(t,8)
 50 GOTO 30
200 t=t+10
210 SOUND 2,t,100
220 IF t 100 THEN ON SQ(2) GOSUB 200
230 RETURN
run
```

Das Programm läuft ununterbrochen. Hat die Warteschlange von Kanal B einen freien Platz (ON SQ(2) entdeckt dies), kommt der SOUND zur Wirkung. Zeile 20 zeigt an, wo das Unterprogramm steht, das zur Tonerzeugung nach ON SQ(2) aufgerufen wird.
Nach dem ersten Aufruf muß ON SQ(2) neu gestellt werden, was in Zeile 220 passiert. Solange der Wert unter 100 liegt, erfolgt das Stellen, ist er darüber, hört es auf. So kann zur Grafik auf dem Bildschirm eine Melodie im Hintergrund spielen.

Für ON SQ() GOSUB kann der Wert in Klammern 1, 2 oder 4 sein, je nach benötigter Warteschlange. Die Funktion SQ() gibt als einzelner Wert dagegen an, wie der momentane Zustand der Warteschlange des angegebenen Kanals ist. Mit PRINT SQ(2) wird ein bitorientierter Wert ausgegeben.

Folgende Bedeutung besitzen die einzelnen Bits:

Bit	dezimal	Aussage
0,1,2	1,2,4	Anzahl freier Plätze in der Warteschlange
3	8	Note am Anfang der Warteschlange ist zum Rendezvous mit A vorgesehen.
4	16	Note am Anfang der Warteschlange ist zum Rendezvous mit B vorgesehen.
5	32	Note am Anfang der Warteschlange ist zum Rendezvous mit C vorgesehen.
6	64	Für oberste Note der Schlange ist Halte- Bit gesetzt (Kanal blockiert).
7	128	Eine Note wird gespielt.

Tabelle 6: Bitzuordnung in der Funktion SQ()

Das folgende kleine Programm verdeutlicht die Aussage der Funktion SQ ( ):

```
10 SOUND 2,200,1000
20 ? SQ(2)
run
 132
```

Die ausgegebene Zahl 132 setzt sich zusammen aus 128 + 4. Das bedeutet entsprechend Tabelle 6, daß der abgefragte Tonkanal 2 gerade einen Ton ausgibt und daß noch 4 Plätze in der Warteschlange des Kanals 2 frei sind.

A n h a n g A

BASIC - Fehlermeldungen

Fehlernummer	Fehlermeldung
1	Unexpected NEXT - nicht erwartetes NEXT NEXT paßt nicht zur FOR-Anweisung, oder NEXT wurde außerhalb einer FOR-NEXT-Schleife gefunden.
2	Syntax Error - Syntaxfehler Von BASIC werden die eingegebenen Anweisungen nicht verstanden, sie enthalten unzulässige Zeichen.
3	Unexpected RETURN - nicht erwartetes RETURN Die Anweisung RETURN steht nicht im Unterprogramm.
4	Data exhaused - Ende der Daten Die DATA-Anweisung enthält keine Daten mehr, trotzdem wurde READ aufgerufen.
5	Improper argument - Das Argument ist nicht gültig Diese Mehrzweckmeldung gibt an, daß der Wert eines Argumentes in einer Funktion oder ein Parameter der Anweisung falsch sind.
6	Overflow - Überlauf Eine arithmetische Operation liefert ein Ergebnis, das größer als 1.7E+38 ist. Es besagt, das Ergebnis ist zu groß bzw. übergelaufen. Ursache dafür kann ein Gleitkomma-Überlauf sein, oder es wurde versucht, eine Gleitkommazahl in eine 16 Bit große Ganzzahl mit Vorzeichen zu wandeln.
7	Memory full - kein freier Speicher mehr (Speicher belegt) Das Programm benötigt einen größeren Speicherbereich als vorhanden ist, oder die Variablen sind zu umfangreich. Auch kann ein Programm durch GOSUB-, FOR-NEXT- oder WHILE-Anweisung zu sehr verschachtelt sein.
8	Line does not exist - Zeile fehlt Die angegebene Zeile wurde nicht gefunden.
9	Subscript out of range - Subscript (Index) liegt außerhalb des Bereiches In einer Matrix ist eine falsche Indexzahl aufgeführt, d.h. sie ist zu groß oder zu klein.

Fehlernummer	Fehlermeldung
10	Array already dimensioned - Matrix ist schon vorhanden Von einer DIM-Anweisung wurde eine der Matrizen schon definiert.
11	Division by zero - Erfolgte Division durch Null Mögliche Ursachen bei ganzzahliger Division: Modulo-Bestimmung (Divisionsrest) oder Potenzierung
12	Invalid direct command - Direktkommando ist falsch Die Eingabe einer Anweisung bzw. eines Kommandos im Direkt-Modus ist unzulässig.
13	Type mismach - Nichtübereinstimmung des Typs Es wurde ein falscher Stringwert eingegeben, z.B. numerischer Wert bei Stringtyp oder umgekehrt. Auch kann bei READ oder INPUT ein falscher Zahlentyp verwendet worden sein.
14	String space full - Stringbereich (Zeichenfolgenbereich) belegt Sogar kleinste Speicherbereiche sind von Zeichenfolgen belegt.
15	String to long - Zeichenkette zu lang Die Länge der Zeichenkette beträgt mehr als 255 Zeichen; kann durch Verketten hervorgerufen werden.
16	String expression to complex - Textausdruck zu umfassend Diese Meldung erscheint, wenn die Anzahl der Zeichenfolgen eine bestimmte Grenze überschritten hat.
17	Cannot CONTinue - CONT reagiert nicht Ein Programm kann aus unbestimmtem Grund mit CONT nicht fortgesetzt werden. CONT kann nach Drücken von zweimal [ESC] nach einem Fehler mit zwischen zeitiger Programmänderung den Wiederstart ermöglichen.
18	Unknow user function - Anwender-Funktion ist nicht bekannt Die aufgerufene Funktion wurde nicht mit DEF FN definiert.
19	RESUME missing - fehlen von RESUME Bei einer Fehlerbehandlungsroutine wurde das Programmende erreicht.

Fehlernummer	Fehlermeldung
20	Unexpected RESUME - RESUME wurde nicht erwartet RESUME darf nur bei einer Fehlerbehandlung verwendet werden, z.B. ON ERROR GOTO-Behandlungsroutine.
21	Direct command found - ein Direktkommando wurde gefunden
22	Operand missing - fehlender Operand Eine Anweisung ist unvollständig, wurde vom BASIC-Interpreter festgestellt.
23	Line to long - die Zeile ist zu lang Eine BASIC-Zeile hat mehr als 255 Zeichen, das interne BASIC-Format wurde nicht eingehalten.
24	EOF met - erreicht wurde EOF Eine Eingabedatei wurde versucht einzulesen, obwohl das Dateiende erreicht ist.
25	File type error - Dateityp ist falsch Die eingelesene Datei besitzt den falschen Typ. OPENIN eröffnet nur ASCII-Textdateien. LOAD, RUN usw. können nur auf Dateitypen verwendet werden, die mit SAVE abgespeichert wurden.
26	NEXT missing - fehlendes NEXT Das zugehörige NEXT kann für eine FOR-Anweisung nicht gefunden werden.
27	File already open - Datei wurde schon eröffnet Die Anweisungen OPENIN oder OPENOUT wurden benutzt, obwohl eine Datei schon eröffnet ist.
28	Unknow command - unbekanntes Kommando Externe Befehle, z.B. Befehle/Anweisungen mit einem Balken beginnend, sind für BASIC unbekannt.
29	WEND missing - fehlendes WEND Für eine WHILE-Anweisung kann ein WEND nicht gefunden werden.
30	Unexpected WEND - nicht erwartetes WEND WEND steht außerhalb einer WHILE-Schleife, oder es paßt nicht zu einer vorhandenen.

A n h a n g B1 ASCII-Steuerzeichen

ASCII Dez.	ASCII Hex.	Kurzzeichen/ Tastatur-betätigung	Grafik-zeichen	ASCII Dez.	ASCII Hex.	Kurzzeichen/ Tastatur-betätigung	Grafik-zeichen
0	00	NULL (1)		16	10	DLE/[CTRL]-[P] (1)(3)	
1	01	SOH/[CTRL]-[A]		17	11	DC1/[CTRL]-[Q]	
2	02	STX/[CTRL]-[B]		18	12	DC2/[CTRL]-[R]	
3	03	ETX/[CTRL]-[C]		19	13	DC3/[CTRL]-[S]	
4	04	EOT/[CTRL]-[D]		20	14	DC4/[CTRL]-[T]	
5	05	ENQ/[CTRL]-[E]		21	15	NAK/[CTRL]-[U]	
6	06	ACK/[CTRL]-[F]		22	16	SYN/[CTRL]-[V]	
7	07	BEL/[CTRL]-[G]		23	17	ETB/[CTRL]-[W]	
8	08	BS /[CTRL]-[H]		24	18	CAN/[CTRL]-[X]	
9	09	HT /[CTRL]-[I]		25	19	EM /[CTRL]-[Y]	
10	0A	LF /[CTRL]-[J]		26	1A	SUB/[CTRL]-[Z]	
11	0B	VT /[CTRL]-[K]		27	1B	ESC/[CTRL]-[[]	
12	0C	FF /[CTRL]-[L]		28	1C	FS /[CTRL]-[\]	
13	0D	CR /[CTRL]-[M] (1)(2)		29	1D	GS /[CTRL]-[]]	
14	0E	SO /[CTRL]-[N]		30	1E	RS /[CTRL]-[ ]	
15	0F	SI /[CTRL]-[O]		31	1F	KS /[CTRL]-[0]	

Erklärung zu (4) siehe Anhang B4.

A n h a n g B2 Alphanumerische ASCII-Steuerzeichen

ASCII Hex.	ASCII Dez.	Druckerausgabe	Bildschirmausgabe	ASCII Hex.	ASCII Dez.	Druckerausgabe	Bildschirmausgabe
20	32	SP (4)		30	48	0	
21	33	!		31	49	1	
22	34	"		32	50	2	
23	35	#		33	51	3	
24	36	$		34	52	4	
25	37	%		35	53	5	
26	38	&		36	54	6	
27	39	'		37	55	7	
28	40	(		38	56	8	
29	41	)		39	57	9	
2A	42	*		3A	58	:	
2B	43	+		3B	59	;	
2C	44	,		3C	60	<	
2D	45	-		3D	61	=	
2E	46	.		3E	62	>	
2F	47	/		3F	63	?	

Erklärung zu (4) siehe Anhang B4.

ASCII Hex.	Dez.	Druck-herausgabe	Bildschirmausgabe	ASCII Hex.	Dez.	Druck-herausgabe	Bildschirmausgabe
40	64	@		50	80	P	
41	65	A		51	81	Q	
42	66	B		52	82	R	
43	67	C		53	83	S	
44	68	D		54	84	T	
45	69	E		55	85	U	
46	70	F		56	86	V	
47	71	G		57	87	W	
48	72	H		58	88	X	
49	73	I		59	89	Y	
4A	74	J		5A	90	Z	
4B	75	K		5B	91	[Ä (5)(6)	
4C	76	L		5C	92	\ Ö (5)(6)	
4D	77	M		5D	93	] Ü (5)(6)	
4E	78	N		5E	94	↑	
4F	79	O		5F	95	_	

Erklärungen zu (5) und (6) siehe Anhang B4.

ASCII Hex.	Dez.	Druck-herausgabe	Bildschirmausgabe	ASCII Hex.	Dez.	Druck-herausgabe	Bildschirmausgabe
60	96	`		70	112	p	
61	97	a		71	113	q	
62	98	b		72	114	r	
63	99	c		73	115	s	
64	100	d		74	116	t	
65	101	e		75	117	u	
66	102	f		76	118	v	
67	103	g		77	119	w	
68	104	h		78	120	x	
69	105	i		79	121	y	
6A	106	j		7A	122	z	
6B	107	k		7B	123	{ ä (5)(6)	
6C	108	l		7C	124	| ö (5)(6)	
6D	109	m		7D	125	} ü (5)(6)	
6E	110	n		7E	126	~ ß (5)(6)(7)	
6F	111	o		7F	127	■ (1)	

Erklärungen zu (1), (5), (6) und (7) siehe Anhang B4.

A n h a n g B 3 ASCII-Grafikzeichen

ASCII Dez.	Hex.	Grafik-zeichen	ASCII Dez.	Hex.	Grafik-zeichen	ASCII Dez.	Hex.	Grafik-zeichen	ASCII Dez.	Hex.	Grafik-zeichen
128	80		144	90		160	A0		176	B0	
129	81		145	91		161	A1		177	B1	
130	82		146	92		162	A2		178	B2	
131	83		147	93		163	A3 (8)		179	B3	
132	84		148	94		164	A4		180	B4	
133	85		149	95		165	A5		181	B5	
134	86		150	96		166	A6		182	B6	
135	87		151	97		167	A7		183	B7	
136	88		152	98		168	A8		184	B8	
137	89		153	99		169	A9		185	B9	
138	8A		154	9A		170	AA		186	BA	
139	8B		155	9B		171	AB		187	BB	
140	8C		156	9C		172	AC		188	BC	
141	8D		157	9D		173	AD		189	BD	
142	8E		158	9E		174	AE		190	BE	
143	8F		159	9F		175	AF		191	BF	

Diese ASCII-Grafikzeichen sind im RESET-Zustand des Computers nicht über Tastatur erreichbar (vgl. Abschnitt 3.13.1., Ausnahme (8) siehe Anhang B4).

A n h a n g B 3 ASCII-Grafikzeichen

ASCII Dez.	Hex.	Grafik-zeichen	ASCII Dez.	Hex.	Grafik-zeichen	ASCII Dez.	Hex.	Grafik-zeichen	ASCII Dez.	Hex.	Grafik-zeichen
192	C0		208	D0		224	E0		240	F0	
193	C1		209	D1		225	E1		241	F1	
194	C2		210	D2		226	E2		242	F2	
195	C3		211	D3		227	E3		243	F3	
196	C4		212	D4		228	E4		244	F4	
197	C5		213	D5		229	E5		245	F5	
198	C6		214	D6		230	E6		246	F6	
199	C7		215	D7		231	E7		247	F7	
200	C8		216	D8		232	E8		248	F8	
201	C9		217	D9		233	E9		249	F9	
202	CA		218	DA		234	EA		250	FA	
203	CB		219	DB		235	EB		251	FB	
204	CC		220	DC		236	EC		252	FC	
205	CD		221	DD		237	ED		253	FD	
206	CE		222	DE		238	EE		254	FE	
207	CF		223	DF		239	EF		255	FF	

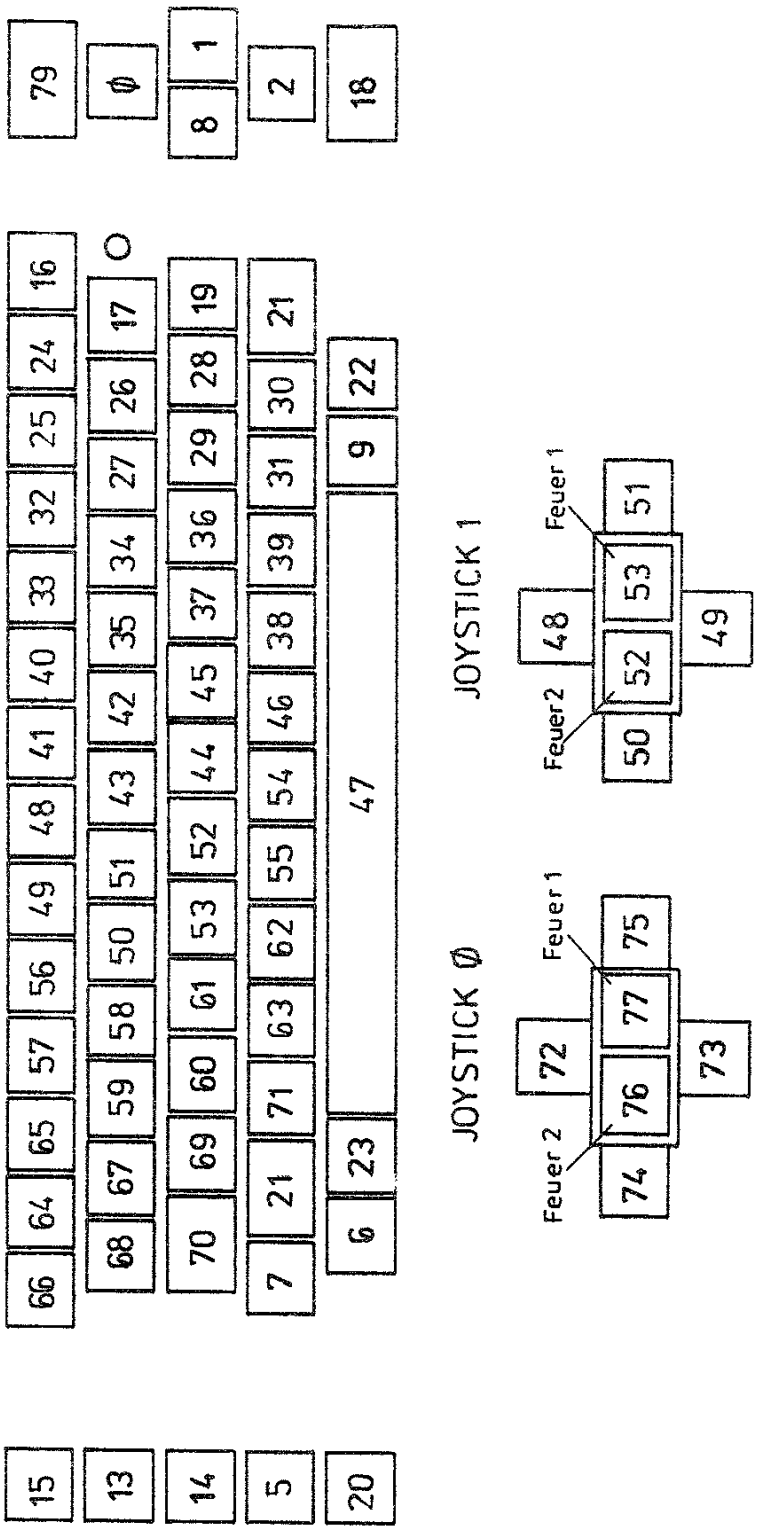
Diese ASCII-Grafikzeichen sind im RESET-Zustand des Computers nicht über Tastatur erreichbar.

A n h a n g B 4 Erklärungen

(1) Grafikzeichen über Tastatur nicht erreichbar
(2) Tastaturbetätigung wirkt wie [RETURN]
(3) Tastaturbetätigung wirkt wie [CLR]
(4) SP - SPace = Leerzeichen
(5) Drucker auf amerikanischen Zeichensatz eingestellt
(6) Drucker auf deutschen Zeichensatz eingestellt
(7) Grafikzeichen über Tastatur erreichbar: [CTRL]-[2]
(8) Grafikzeichen über Tastatur erreichbar: [SHIFT]-[£]

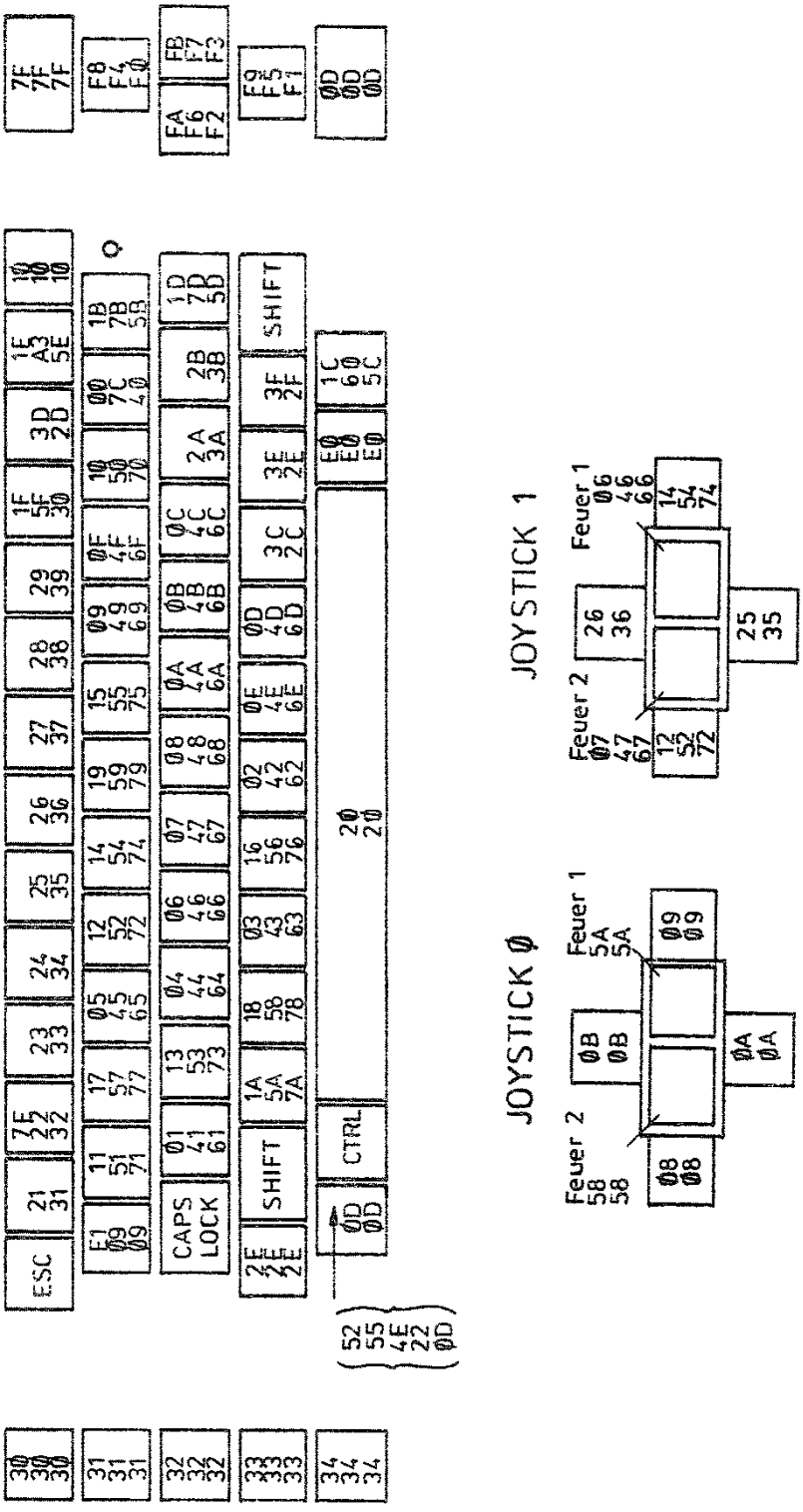
A n h a n g C

Zuordnung der Nummern zu Tasten und Joysticksignalen



A n h a n g D

Standardbelegung der Tasten hexadezimal



A n h a n g E

Noten und Tonperioden

Die vorgeschlagenen Tonperioden für Noten aller acht Oktaven enthält folgende Tabelle.
Dabei ist die erzeugte nicht die genau gewünschte Frequenz. Grund dafür ist die ganzzahlige Angabe der Perioden. Der relative Fehler gibt das Verhältnis der Differenz zwischen erzeugter und gewünschter Frequenz zur genauen Frequenz an.

Note	Frequenz	Periode	relativer Fehler	
C	16.352	3822	-0.007%	
C#	17.324	3608	+0.007%	
D	18.354	3405	-0.007%	
D#	19.445	3214	-0.004%	
E	20.602	3034	+0.009%	
F	21.827	2863	-0.016%	Oktave -4
F#	23.125	2703	+0.009%	
G	24.500	2551	-0.002%	
G#	25.957	2408	+0.005%	
A	27.500	2273	+0.012%	
A#	29.135	2145	-0.008%	
H	30.868	2025	+0.011%	

Note	Frequenz	Periode	relativer Fehler	
C	32.703	1911	-0.007%	
C#	34.648	1804	+0.007%	
D	36.708	1703	+0.022%	
D#	38.891	1607	-0.004%	
E	41.203	1517	+0.009%	
F	43.654	1432	+0.019%	Oktave -3
F#	46.249	1351	-0.028%	
G	48.999	1276	+0.037%	
G#	51.913	1204	+0.005%	
A	55.000	1136	-0.032%	
A#	58.270	1073	+0.039%	
H	61.735	1012	-0.038%	

Note	Frequenz	Periode	relativer Fehler	
C	65.406	956	+0.046%	
C#	69.296	902	+0.007%	
D	73.416	851	-0.037%	
D#	77.782	804	+0.085%	
E	82.407	758	-0.057%	
F	87.307	716	+0.019%	Oktave -2
F#	92.499	676	+0.046%	
G	97.999	638	+0.037%	
G#	103.826	602	+0.005%	
A	110.000	568	-0.032%	
A#	116.541	536	-0.055%	
H	123.471	506	-0.038%	

Note	Frequenz	Periode	relativer Fehler	
C	130.813	478	+0.046%	
C#	138.591	451	+0.007%	
D	146.832	426	+0.081%	
D#	155.564	402	+0.058%	
E	164.814	379	-0.057%	
F	174.614	358	+0.019%	Oktave -1
F#	184.997	338	+0.046%	
G	195.998	319	+0.037%	
G#	207.652	301	+0.005%	
A	220.000	284	-0.032%	
A#	233.082	268	-0.055%	
H	246.942	253	-0.038%	

Note	Frequenz	Periode	relativer Fehler	
C	261.626	239	+0.046%	Mittleres C
C#	277.183	225	-0.215%	
D	293.665	213	+0.081%	
D#	311.127	201	+0.058%	
E	329.628	190	+0.206%	
F	349.228	179	+0.019%	Oktave 0
F#	369.994	169	+0.046%	
G	391.995	159	-0.277%	
G#	415.305	150	-0.328%	
A	440.000	142	-0.032%	Kammerton A
A#	466.164	134	-0.055%	
H	493.883	127	+0.356%	

Note	Frequenz	Periode	relativer Fehler	
C	523.251	119	-0.374%	
C#	554.365	113	+0.229%	
D	587.330	106	-0.390%	
D#	622.254	100	-0.441%	
E	659.255	95	+0.206%	
F	698.457	89	-0.543%	Oktave 1
F#	739.989	84	-0.548%	
G	783.991	80	+0.350%	
G#	830.609	75	-0.328%	
A	880.000	71	-0.032%	
A#	932.328	67	-0.055%	
H	987.767	63	-0.435%	

Note	Frequenz	Periode	relativer Fehler	
C	1046.502	60	+0.462%	
C#	1108.731	56	-0.662%	
D	1174.659	53	-0.390%	
D#	1244.508	50	-0.441%	
E	1318.510	47	-0.855%	
F	1396.913	45	+0.574%	Oktave 2
F#	1479.978	42	-0.548%	
G	1567.982	40	+0.350%	
G#	1661.219	38	+0.992%	
A	1760.000	36	+1.357%	
A#	1864.655	34	+1.417%	
H	1975.533	32	+1.134%	

Note	Frequenz	Periode	relativer Fehler	
C	2093.004	30	+0.462%	
C#	2217.461	28	-0.662%	
D	2349.318	27	+1.469%	
D#	2489.016	25	-0.441%	
E	2637.021	24	+1.246%	
F	2793.826	22	-1.685%	Oktave 3
F#	2959.955	21	-0.548%	
G	3135.963	20	+0.350%	
G#	3322.438	19	+0.992%	
A	3520.000	18	+1.357%	
A#	3729.310	17	+1.417%	
H	3951.066	16	+1.134%	

Vom Kammerton A ausgehend, werden diese Werte wie folgt berechnet:

```
FREQUENZ  = 440 * (2 (OKTAVE + ((N-10)/12)))
PERIODE   = ROUND (62500/FREQUENZ)
```

Dabei gilt N=1 für C, N=2 für C#, N=3 für D usw.

A n h a n g F

BASIC-Schlüsselwörter

Folgende Schlüsselwörter sind reserviert und dürfen nicht als Variablennamen verwendet werden:

Die in den Klammern enthaltenen Angaben sind die zugehörigen Seitennummern.

A n h a n g E

mikroelektronik

veb mikroelektronik ›wilhelm pieck‹ mühlhausen
im veb kombinat mikroelektronik