robotron

Programmierhandbuch

Teil 1

VEB Robotron-Meßelektronik
»Otto Schön« Dresden

Inhaltsverzeichnis

1. Einleitung

Das Programmierhandbuch des Heimcomputers robotron Z 9001 enthält eine ausführliche Beschreibung der Programmiersprache BASIC, sowie Hinweise und Informationen zum Betriebsystem und weiteren Programmiermöglichkeiten.
Wie Sie sicher bereits wissen, ist Ihr Computer sehr schnell und zuverlässig bei der Ausführung Ihrer Instruktionen, kann aber ohne diese gar nichts für Sie tun. Ihr Wunsch besteht also darin, sich auf möglichst einfache und effektive Weise mit Ihrem Heimcomputer über dessen Aufgaben zu verständigen. Das betrifft sowohl sofort ausführbare Anweisungen und Kommandos zur Beeinflussung der Programmabarbeitung als auch die Formulierung einer wiederholt nutzbaren Problemlösung, also eines Programms.
Beides gestattet die dialogorientierte Programmiersprache BASIC auf komfortable Weise, deshalb ist sie im internationalen Maßstab die Programmiersprache für Heimcomputer.
Mit Hilfe der Programmiersprache BASIC des Heimcomputers robotron Z 900 können sowohl einfache mathematische Operationen sofort ausgeführt, als auch Programme eingegeben, abgearbeitet und auf Magnetbandkassette gespeichert werden. Auch die Ausgabe von Programmen, Rechenergebnissen und anderen Informationen auf einen Drucker ist möglich. Ebenso können die heimcomputer-spezifischen Ergänzungsmodule (Farbmodul, Spielhebel u. a.) sowie der für die grafische Gestaltung bedeutsam Grafik-Zeichensatz (128 spezielle Grafikzeichen) vom BASIC des „robotron Z 9001“ einfach angesprochen werden. Außerdem ist es möglich, Informationen von angeschlossenen Meßgeräten abzurufen und weiter zu verarbeiten und auch Steuersignale an entsprechende Geräte zu übertragen.
Das BASIC des „robotron Z 9001“ ist in seinem Aufbau (Anweisungs- und Kommandovorrat, syntaktische Struktur) dem international weitverbreiteten Standard angeglichen, so daß eine Übertragung von BASIC-Programmen vieler anderer Rechner auf den „robotron Z 9001“ in der Regel leichtfallen wird.
Für Ihren Heimcomputer wird der BASIC-Interpreter in zwei Varianten angeboten. In der Grundvariante ist er als **RAM-BASIC** von einer Magnetbandkassette, der „Grundkassette“, in den RAM-Speicher (Arbeitsspeicher) des Rechners zu laden. Außerdem wird er als **ROM-BASIC** im sogenannten BASIC-Modul angeboten. Die Nutzung dieser Version bringt den Vorteil, daß das BASIC nach dem Einschalten des Rechners sofort

verfügbar ist und dann außerdem wesentlich mehr Speicherplatz (insgesamt etwa 15 kbytes) für Anwenderprogramme zur Verfügung steht.
Sie können mit dem „robotron Z 9001“ zahlreiche käuflich erwerbbare BASIC-Anwenderprogramme unterschiedlichsten Inhalts abarbeiten. Diese Programme sind auf Magnetbandkassetten gespeichert und müssen von Ihnen mit Hilfe eines Kassettenrecorders in den Heimcomputer geladen (eingelesen) werden. Wollen Sie ein **Programm** selbst schreiben (programmieren) und dann abarbeiten, so gehört dazu einerseits, daß Sie die für die Realisierung des Programms erforderlichen **Anweisungen** in der richtigen Reihenfolge in den Rechner eingeben (laden). Andererseits sind zum Test, zur Abarbeitung und auch zur Modifizierung eines Programms eine ganze Anzahl von **Kommandos** notwendig.

In der Dialogsprache BASIC sind, im Gegensatz zu anderen Programmiersprachen, wie FORTRAN, PL/1 oder PASCAL, neben den notwendigen Anweisungen auch solche (Steuer-) Kommandos enthalten.

Dementsprechend kennt der BASIC-Interpreter auch zwei typische Arbeitszustände:

- den **Kommando-Modus**, in dem die Tastatureingabe von Kommandos durch den Nutzer erfolgt. Die Ausführung dieser Kommandos ist im wesentlichen Voraussetzung für die Programmabarbeitung bzw. beeinflußt diese (Programm starten, unterbrechen, fortsetzen, speichern, laden, anzeigen, ändern, ...).
- den **Programm-Modus**, innerhalb dessen die Abarbeitung von Anweisungen des gestarteten BASIC-Programms stattfindet. Diese Anweisungen dienen der eigentlichen Problemlösung und werden interpretierend abgearbeitet, d. h. in vorgegebener Reihenfolge einzeln analysiert und unmittelbar ausgeführt.

Wie Sie bereits wissen, ist BASIC in seinen Grundelementen sehr einfach gestaltet, so daß auch diejenigen, die mit der Rechentechnik noch nicht vertraut sind, schon nach kurzer Zeit eigene Programme schreiben können. Trotzdem werden Sie besonders in der ersten Zeit noch viele Fehler machen. Das sollte Sie jedoch nicht entmutigen. Außerdem kann der Rechner durch Fehlbedienung nicht beschädigt werden. Er weist falsche Eingaben in der Regel durch Fehlermeldungen selbständig zurück.

Der in das BASIC des „robotron Z 9001“ einführende Abschnitt 3 vermittelt Grundkenntnisse, die insbesondere für die Nutzung käuflich erworbener Anwenderprogramme erforderlich sind. im Abschnitt 4 werden Sie dann systematisch durch das BASIC Ihres Heimcomputers geführt, so daß Sie nach und nach in der Lage sein werden, auch kompliziertere BASIC-Programme selbst zu erarbeiten.

Sollten Sie bereits über weitergehende Kenntnisse in der Mikrorechentechnik und in der Programmierung verfügen, so können Sie Ihren Heimcomputer auch in **Assembler** oder in der **K-1520-Maschinensprache** programmieren. Dazu müssen Sie jedoch die entsprechenden Assembler-Kassetten bzw. -Module erwerben.

Reicht Ihnen der freie Speicherplatz Ihres Heimcomputers für größere Programme nicht mehr aus, so können Sie durch einen oder zwei **RAM-Erweiterungsmodule** jeweils 16 kbyte Speicherplatz zusätzlich stecken (siehe auch Abschnitt 5 der Bedienungsanleitung).

Falls Sie ihren Heimcomputer mit Hilfe der Bedienungsanleitung bereits ordnungsgemäß angeschlossen und eingeschaltet haben, so können Sie nun beginnen, ihre ersten Handlungen am Computer auszuführen.

2. Tastatur

Nachdem Sie Ihren Heimcomputer gemäß Abschnitt 3.6 der Bedienungsanleitung in Betrieb genommen haben, befindet sich sein Betriebssystem zunächst im Grundzustand, und auf dem Bildschirm erscheint:

In diesem Zustand des „robotron Z 9001“ sind nur wenige Bedienhandlungen möglich, die komplett im Abschnitt 7 beschrieben werden. insbesondere können an dieser Stelle noch keine Rechenoperationen ausgeführt werden. Dazu müssen Sie erst den BASIC-Interpreter starten. Das dafür erforderliche Kommando übermitteln Sie dem Rechner über die Tastatur, die zunächst gesondert betrachtet werden soll.
Die prinzipielle Funktion der Tasten, insbesondere der Umschalttasten [SHIFT], [SHIFT LOCK] sowie der unbeschrifteten Leertasten, wurde bereits in der Bedienungsanleitung (Abschnitt 4.2 und Bild 5) erläutert.
Zur besseren Übersicht ist nachstehend die Tastatur nochmals abgebildet:

Am besten, Sie probieren die Grundfunktion wichtiger Tasten gleich aus. Das ist auch im Grundzustand des Rechners möglich. Drücken Sie also, nacheinander einige Buchstabentasten, und lassen Sie dann den Finger so lange auf einer Taste, z. B. dem Buchstaben "Z", bis etwa eineinhalb Zeilen am Bildschirm beschrieben sind. Danach brechen Sie die Eingabe mit der [STOP]-Taste ab. Nun könnte etwa folgendes Bild sichtbar sein:

```
robotron Z 9001
OS
>ABCD  robotron Z 9001  ABCD  1234567890
ZZZZZZZZZZ
OS
>
```

Der Rechner geht im Betriebssystem (OS) wieder in die Eingabebereitschaft, die durch das Aufforderungszeichen ">" angezeigt wird. Zur Gewöhnung sollten Sie diese Eingaben mit verschiedenen Tasten, auch unter Nutzung der Tasten [SHIFT] und [SHIFTLOCK] wiederholen.

Weitere Zeichen, die Grafikzeichen, können Sie mit Hilfe der Taste [GRAPHIC] darstellen.

[GRAPHIC]
schaltet den Rechner in den Grafikmodus (GRAPHIC-Anzeige leuchtet). jetzt können Sie mit Hilfe der Tastatur die im Anhang B dargestellten Grafikzeichen eingeben. Durch nochmalige Betätigung von GRAPHIC wird auf Normaleingabe zurückgeschaltet (GRAPHIC-Anzeige verlischt).

Es ist zweckmäßig, wenn Sie sich vor der Arbeit mit dem BASIC-Interpreter noch über die prinzipielle Wirkung einiger wichtiger Funktionstasten informieren.

[RESET]
Durch diese Taste wird der Computer wieder in den Grundzustand des Betriebssystems gebracht. Dabei werden einige interne Informationen gelöscht, und auf dem Bildschirm erscheint wieder

Der Computer ist erneut eingabebereit. Probieren Sie es aus!

[ENTER]
Der Computer wird angewiesen, die vorher eingetippte Information zu übernehmen und zu verarbeiten. Mit dieser Taste werden im Normalfall alle Eingaben abgeschlossen.

[STOP]
Die eingegebene Information wird nicht übernommen und abgearbeitet. Ein laufendes Programm wird unterbrochen bzw. abgebrochen.

[←] [→]
Diese Kursortasten dienen der Bewegung des Kursors auf einer Zeile. Im Betriebssystemmodus wird durch ← gleichzeitig das Zeichen vor dem Kursor gelöscht, im BASIC ist das nicht der Fall.

[CONTR]
Diese Taste bewirkt in Verbindung mit anderen Tasten die Ausführung bestimmter Sonderfunktionen, von denen viele nur im BASIC bzw. bei vorhandenem Farbmodul sinnvoll sind. Die vollständige Liste dieser Sonderfunktionen (Steuerzeichen) finden Sie im Anhang A.

Wenn Sie Interesse haben, können Sie z. B. die Funktionen

[CONTRL] [L] - Löschen des Bildschirmes (auch "■" verschwindet!)

[CONTRL] [Q] - Tonausgabe bei Tastenbetätigung bzw. Bildschirmanzeige (wird durch nochmaliges Drücken von [CONTRL] [Q] wieder abgestellt) sofort, d. h. auch im Betriebssystemmodus, ausprobieren. Beachten Sie, daß [CONTRL] **gleichzeitig** mit der entsprechenden Taste gedrückt werden muß.

[COLOR]
Diese Taste wirkt nur nach Erweiterung des Grundgerätes "robotron Z 9001.10" auf Farbwiedergabe bzw. bei der Variante "robotron Z 9001.11". Nach COLOR ist stets eine der Tasten 1 bis 8 zu drücken (Näheres siehe Abschnitt 4.16).

[ESC]
Diese Taste kann in Programmen zur Ausführung besonderer Funktionen genutzt werden. Sie muß dabei über ihre Codierung (vgl. Abschnitt 4.14 und Anhang A) ausgewertet werden.

[↑] [↓]
Diese Tasten können in Programmen über ihre Codierungen genutzt werden, um z. B. Positionierungen in vertikaler Richtung vorzunehmen. Im Normalfall haben die Tasten, außer ↓ im EDIT-Modus des BASIC, keine Funktion.

[LIST] [RUN] [PAUSE] [CONT] [INS] [DEL] [CL LN] [I←] [→I]
Diese Tasten sind nur wirksam, wenn Sie im BASIC arbeiten. Eine genaue Funktionsbeschreibung, finden Sie in den Abschnitten 3 und 4.

3. Was Sie über BASIC wissen müssen

Der BASIC-Interpreter ist für den "robotron Z 9001' in zwei Varianten verfügbar, als RAM-BASIC und als ROM-BASIC. Die Nutzung beider Varianten unterscheidet sich nur beim Starten des BASIC-Interpreters nach dem Einschalten des Gerätes sowie im Speicherplatz, der anschließend für den Anwender verfügbar ist.

3.1. Starten des BASIC-Interpreters

RAM-BASIC
Wollen Sie den BASIC-Interpreter von der Grundkassette laden, so bringen Sie den Computer zunächst in den Grundzustand des Betriebssystems. Dieser wird unmittelbar nach dem Einschalten des Gerätes oder nach [RESET] erreicht. Nun kann das RAM-BASIC durch folgende Handlungen von der Grundkassette in den Heimcomputer geladen werden:

1. Legen Sie die Grundkassette in das Kassettenmagnetbandgerät ein, positionieren Sie das Band vor den Anfang des BASIC-Interpreters (5-Sekunden-Vorton beachten!) und halten Sie es dann an.
   Empfehlung:
   Vor dem ersten Ladevorgang sollten Sie sich ruhig den Anfang des BASIC-Interpreters über den Lautsprecher des Kassettengerätes "anhören", damit Sie den Vorton und die kurzen Zwischentöne deutlich unterscheiden lernen.
2. Geben Sie über Tastatur "BASIC" ein, und drücken Sie anschließend die [ENTER]-Taste:

   ```
   BASIC [ENTER]
   ```

   Der Bildschirm hat dann folgendes Aussehen:

   ```
   robotron Z 9001

   OS
   >BASIC

   start tape
   ■
   ```

3. Starten Sie jetzt die Magnetbandkassette durch Drücken der **Wiedergabetaste**.

4. Betätigen Sie die [ENTER]-Taste am Heimcomputer, spätestens beim Ertönen des Vortones.
   Der Heimcomputer beginnt nun, den BASIC-Interpreter einzulesen (zu „laden“). Diesen Vorgang können Sie akustisch und optisch verfolgen. Der Kursor am Bildschirm muß nach jedem kurzen Zwischenton um eine Position nach rechts springen. In dieser Zeit werden 128 Zeichen (1 Record) in den Rechner eingelesen und geprüft. Auf dem Bildschirm haben Sie etwa folgendes Bild:

   ```
   robotron Z 9001

   OS
   >BASIC

   start tape
                  ■
                  ↑ Cursor bewegt sich nach rechts
   ```

Das Einlesen des BASIC-Interpreters dauert nicht ganz 2 Minuten. Nach erfolgreichem Einiesen meldet sich der BASIC-Interpreter mit:

```
HC-BASIC
MEMORY SIZE? :
```

Sie können nun das Kassettengerät ausschalten.

Achtung!
Sollte der Einlesevorgang durch Fehlermeldungen der Art
BOS-error: ...
unterbrochen werden, so drücken Sie zunächst die [STOP]-Taste und informieren sich dann im Anhang H des Programmierhandbuches bzw. im Anhang 3 der Bedienungsanleitung über die möglichen Ursachen bzw. Korrekturhandlungen.

Auf die Frage nach der Größe des Speicherplatzes

```
MEMORY SIZE? :■
```

reagieren Sie im Normalfall nur mit [ENTER]. (Falls Sie auch mit Maschinenprogrammen arbeiten wollen, sind hier andere Eingaben erforderlich. Näheres dazu finden Sie im Abschnitt 5.3)
Nach [ENTER] bekommen Sie den aktuellen, für ihre Programme und Daten verfügbaren, Speicherplatz in Bytes (in einem Byte kann jeweils ein Zeichen gespeichert werden) angezeigt.

Der BASIC-Interpreter geht danach in den Kommandomodus über und wartet auf Ihre Eingaben.

```
HC-BASIC

MEMORY SIZE? :
4846 BYTES FREE

OK
>■
```

Achtung!
Die vollständige Abarbeitung von Kommandos und Programmen wird im Kommandomodus des BASIC-Interpreters stets durch "OK" angezeigt. Nach dem Aufforderungszeichen ">" ist dann die Eingabe neuer Kommandos oder Anweisungen möglich.

ROM-BASIC
Wenn Sie den BASIC-Modul gesteckt haben, so müssen Sie nach dem Einschalten des Heimcomputers nur

```
BASIC [ENTER]
```

eingeben. Der Interpreter meldet sich dann sofort mit

```
HC-BASIC

MEMORY SIZE? :■
```

Drücken Sie nun wieder [ENTER], so ist der BASIC-Interpreter arbeitsbereit.

Verlassen, Abbruch und Neustart des BASIC
Die Arbeit im BASIC kann auf zweierlei Arten abgebrochen werden. Durch Eingabe des Kommandos

```
BYE [ENTER]
```

gelangen Sie wieder in den Betriebssystemmodus (OS-Modus).

```
BYE

OS
>■
```


Andererseits können Sie durch [RESET] den Grundzustand des Betriebssystems wiederherstellen. Diese Möglichkeit sollte aber nur genutzt werden, wenn sich der Rechner durch andere Tasten (auch [STOP] bzw. [ENTER]) nicht mehr bedienen läßt bzw. auf keinerlei Eingaben reagiert. Wollen Sie nach dem Abbruch der Arbeit im BASIC wieder neu beginnen, so können Sie das für RAM- und ROM-BASIC in gleicher Weise durch

```
WBASIC [ENTER]
```

In diesem Fall bleiben vorher vorhandene BASIC-Programme erhalten. Der BASIC-Interpreter meldet sich nur kurz durch

```
OK
>■
```

Natürlich können Sie auch wieder durch

```
BASIC [ENTER]
```

starten, wobei dann die vorher vorhandenen Programme gelöscht werden und das BASIC-Anfangsbild wieder erscheint. Probieren Sie diese Möglichkeiten ruhig aus, damit Sie mit dem Rechner und dessen Reaktionen vertraut werden.

3.2. Sofort ausführbare Anweisungen

Nach dem Starten des BASIC-Interpreters können Sie bereits viele einfache Anweisungen (Rechenoperationen) ausführen, unter anderem auch die mathematischen Grundoperationen, etwa im Bereich eines wissenschaftlichen Taschenrechners. Man spricht deshalb gelegentlich auch vom „Taschenrechnermodus“ des Heimcomputers.

Numerische Operationen
Wollen Sie z. B. die Aufgabe

235 + 117

lösen, so geben Sie nach dem Aufforderungszeichen zeichenweise über die Tastatur ein

```
PRINT 235+117
```

und drücken danach die [ENTER]-Taste. Auf dem Bildschirm sind dann Aufgabe und Ergebnis in folgender Weise dargestellt:

```
>PRINT 235+117
 352
OK
>■
```

Durch „OK“ wird dabei wieder die vollständige Abarbeitung der Rechenoperation quittiert. Das Aufforderungszeichen „>“ signalisiert die erneute Eingabebereitschaft des Computers. Ab der Position des Kursors ■ können Sie eine neue Aufgabe eingeben.
Damit das Ergebnis auch angezeigt wird, muß jeweils das Schlüsselwort **PRINT** („Drucke“) vorangestellt werden. Zur Vereinfachung der Eingabe kann das häufig benötigte **PRINT** auch durch ein Fragezeichen („?“) abgekürzt werden. Geben Sie z. B. ein:

```
?768-374 [ENTER]
```

Zur Ausführung solcher einfachen Rechenoperationen stehen Ihnen die Operationszeichen

+	für Addition
-	für Subtraktion
*	für Multiplikation
/	für Division
^	für Potenzierung

zur Verfügung. Weiterhin können Sie einige mathematische Standardfunktionen (siehe auch Abschnitt 4.2) wie z. B.

SQR(X)	- Quadratwurzel
SIN(X)	- Sinus (X im Bogenmaß)
LN(X)	- Natürlicher Logarithmus (X > 0)
PI	- Konstante π

nutzen. Wenn Sie zur Übung einige Aufgaben lösen wollen, so beachten Sie bitte, daß jede Anweisung durch [ENTER] abgeschlossen werden muß. Auf dem Bildschirm ergibt sich z. B.

```
>?SIN(PI*45/180)
 .707107
OK
>?3^9.5
 34092
OK
>?0.5*LN(10)/(5-3*5)
-.115129
OK
```



```
>?7*7*PI
 153.938
OK
>■
```

Achtung!

1. Bei nichtganzzahligen Werten muß stets ein Dezimalpunkt eingegeben werden, kein Komma. Das entspricht dem internationalen Standard der Rechentechnik.
2. Zur Unterscheidung von dem Buchstaben 0 wird die Ziffer 0 (Null) auf dem Bildschirm und auf der Tastatur durchgestrichen dargestellt.

Zeichenketten

Neben Zahlen kann Ihr Heimcomputer auch Zeichenketten verarbeiten. Zeichenketten sind Folgen von Buchstaben, Ziffern, Sonderzeichen oder Grafikzeichen, die im allgemeinen durch Anführungszeichen ("...") begrenzt werden. Sie werden zur Textspeicherung und -darstellung benötigt. Geben sie z. B. ein:

```
PRINT "robotron" [ENTER]
```

Der Rechner verarbeitet dann die Zeichenfolge **robotron** als Zeichenkette und gibt diese wieder am Bildschirm aus.

Eingabekorrekturen

Wenn Sie sich bei der Eingabe der Anweisungen und Kommandos in den Rechner gelegentlich vertippen, können Sie mit Hilfe der Kursortasten [←] und [→] den Kursor auf der Eingabezeile bewegen und die falschen Eingaben durch richtige überschreiben (ersetzen). Probieren Sie diese Möglichkeiten anhand selbstgewählter Beispiele aus, und vergessen Sie nicht, daß der Rechner Ihre Aufgaben erst dann abarbeitet, wenn Sie [ENTER] gedrückt haben.

Achtung!

1. Wenn Sie eine Anweisung fehlerhaft eingeben und [ENTER] drücken, reagiert der Rechner mit der Meldung „?SN ERROR“ (Syntaxfehler). Sie können dann die Eingabe wiederholen.
2. Nach [STOP] wird die Anweisung nicht ausgeführt, Sie können erneut eingeben.

Noch wesentlich einfacher können Sie korrigieren, wenn Sie die Tasten [INS], [DEL], [CL LN], [I←] und [→I] nutzen. Diese sind besonders dann

nützlich, wenn mehrere Zeichen innerhalb der aktuellen Eingabezeile gestrichen oder hizugefügt werden sollen.

[←] [→]
Diese Tasten bewegen den Kursor auf das erste bzw. das letzte Zeichen der Eingabezeile.

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

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

[CL LN]
(CLEAR-LINE-Taste). Durch [CL LN] wird die aktuelle Eingabezeile gelöscht. Der Kursor rückt auf den Zeilenanfang. Wenn Sie sich auch mit diesen Korrekturmöglichkeiten vertraut gemacht haben, so können Sie sich nun der Nutzung vorhandener (z. B. käuflich erworbener) BASIC-Programme zuwenden oder auch versuchen, erste kleine Programme selbst zu schreiben.

3.3. Nutzung vorhandener Programme

Für den Heimcomputer robotron Z 9001 werden zahlreiche BASIC-Anwenderprogramme auf Magnetbandkassetten angeboten. Inhaltlich lassen sich diese Programme (Kassetten) in die Rubriken

Lehre und Lernen
Datenverarbeitung
Spiele
Wissenschaft und Technik
Heim und Hobby

einordnen. Alle Programme sind farbig gestaltet, können aber auch in der Schwarzweißausführung des „robotron Z 9001“ gut verwendet werden. Vor der Nutzung solcher robotron-Anwenderprogramme können Sie sich anhand der zur Kassette gelieferten Programmbeschreibung über den wesentlichen Inhalt der Programme und Besonderheiten bei ihrer Abarbeitung (Spielregeln u. ä.) informieren. Weitere Bedienerhinweise erhalten Sie vom Programm auf dem Bildschirm. Ist Ihr Kassettengerät mit einem Bandzählwerk ausgestattet, können Sie sich vor dem ersten Laden der Programme die Zählerstände der Programmanfänge notieren, die jeweils an einem 5-Sekunden-Vorton erkennbar sind.
Wollen Sie ein Programm in den Rechner einlesen, müssen sie darauf achten, daß der im Rechner noch freie Speicherplatz für das Programm ausreicht. Gegebenenfalls können Sie sich durch die Anweisung

```
PRINT FRE(0) [ENTER]
```

diesen Speicherplatz (in Bytes) anzeigen lassen.

Achtung!
Einige Programme werden vom Hersteller geschützt. Bei diesen Programmen können die Kommandos CSAVE, EDIT und LIST (vgl. Abschnitt 4) nicht abgearbeitet werden.

Laden von BASIC-Programmen
Die auf Magnetbandkassetten gespeicherten BASIC-Anwenderprogramme können durch das Kommando

```
CLOAD "progname"
```

in den Rechner geladen werden. Dabei muß für *progname* der konkrete Name des gewünschten Programmes eingegeben werden, z. B. also

```
CLOAD "R+HANOI"
```

für das Programm R+HANOI auf der Grundkassette R 0111 des "robotron Z 9001". Die für das Laden der BASIC-Programme erforderlichen Handlungen sind im folgenden aufgeführt:
1. Löschen Sie die noch im Rechner gespeicherten Programme durch

```
NEU [ENTER]
```

2. Legen Sie die Kassette in das Magnetbandkassettengerät ein, und Positionieren Sie das Magnetband vor das zu ladende Programm. (Der Programmanfang ist an einem etwa 5 Sekunden andauernden Vorton zu erkennen.)

3. Geben Sie am Heimcomputer das Kommando

```
CLOAD "progname"
```

z.B. also

```
CLOAD "R+HANOI"
```

ein. Drücken Sie aber noch nicht [ENTER]

4. Starten Sie nun bitte das Magnetband (Wiedergabetaste!).

5. Beim Ertönen des Vortones drücken Sie [ENTER].
Nach dem Einiesen des ersten Datensatzes vom Magnetband zeigt der Computer den Namen des gefundenen Programms am Bildschirm an:

```
>CLOAD "R+HANOI"
  ***R+HANOI      ■
```

Danach rückt der Kursor bei jedem richtig gelesenen Datensatz (Record, je 128 Bytes) um eine Position nach rechts (etwa nach jeweils einer Sekunde). Nach erfolgreich abgeschlossenem Einlesen des gesamten Programms erscheint die Meldung

```
FILE FOUND
```

(d. h. „Programm gefunden“) auf dem Bildschirm, und der BASIC-Interpreter befindet sich wieder im Kommandomodus. Sie können Ihr Kassettengerät abschalten.

Der Bildschirm hat nun etwa folgendes Aussehen:

```
>CLOAD "R+HANOI"
  ***R+HANOI                              FILE FO
UND
OK
>
>■
```


Achtung!
Sollte das Einlesen eines Programms anders als beschrieben verlaufen, z. B. durch eine der Fehlermeldungen

```
BOS-error: . . .
```

unterbrochen werden, so drücken Sie bitte [STOP] und informieren sich dann im Anhang H des Programmierhandbuches bzw. im Anhang 3 der Bedienungsanleitung über die möglichen Fehler bzw. Korrektur-handlungen.

Programmstart
Nach dem Laden des Programms möchten Sie es sicher starten und abarbeiten. Der Start erfolgt am einfachsten durch Drücken der Taste [RUN]. Nach kurzer Zeit sehen Sie auf dem Bildschirm das erste Bild des Programms, welches in der Regel über Namen und Inhalt dieses Programms informiert.

Programmabarbeitung
Bei der Abarbeitung eines Programms sollten Sie die am Bildschirm angezeigten Bedienerhinweise beachten. Die Programme befinden sich dann in einem Wartezustand und fordern zu Eingaben bzw. Bedienhandlungen auf. Im einfachsten Fall muß dabei nur [ENTER] betätigt werden. Alle Eingaben von Zahlen, Buchstaben und Texten sind durch [ENTER] abzuschließen, sofern in der Programmbeschreibung keine anderen Hinweise gegeben werden. Bei der Eingabe von Zahlen ist entsprechend der bei Rechnern üblichen Schreibweise ein Dezimalpunkt statt eines Kommas zu schreiben, z. B.

```
123.45 [ENTER]
```

Sind mehrere Zahlen einzugeben, so sind diese jeweils durch ein Komma zu trennen. Wird eine Zahl fehlerhaft eingegeben bzw. statt einer Zahl eine Zeichenkette (Buchstabenfolge), so kommt es zur Fehlermeldung

```
?REDO FROM START
```

Die Eingabe kann dann wiederholt werden. Normale Texteingaben werden ebenfalls durch [ENTER] abgeschlossen. Sollte man sich bei den geforderten Eingaben vertippen, so ist eine Korrektur möglich, falls noch nicht [ENTER] gedrückt wurde. Im Abschnitt 3.2 wurde bereits beschrieben, wie Sie solche Eingabekorrekturen ausführen können.

Beachten Sie bitte:

1. Bei einigen Programmen werden bestimmte Tasten für spezielle Bedienhandlungen genutzt. Diese müssen dann nicht durch [ENTER] abgeschlossen werden. Genaue Angaben dazu erhalten Sie in der Programmbeschreibung bzw. über den Bildschirm.
2. Zur Vermeidung von Irrtümern und Fehlbedienungen sollten Sie sich alle Informationen und Fragestellungen am Bildschirm genau durchlesen.
3. Bei Alternativfragen wird im allgemeinen eine der möglichen Antworten wesentlich häufiger zu erwarten sein. Diese Entscheidungsmöglichkeit ist eingeklammert, da sie nur die Betätigung von [ENTER] erfordert, z.B.

   ```
   NEUES SPIEL: (J)/N ■
   ```

   In diesem Fall ist
   [J] [ENTER]
   oder nur [ENTER] für JA bzw.
   [N] [ENTER] für NEIN einzugeben.

4. In einigen Spiel-Programmen ist der Einsatz von Spielhebeln zur Steuerung der Bewegungsabläufe zweckmäßig. Die Spielerel sind an der entsprechenden Buchse (vgl. Bedienungsanleitung, Abschnitt 3.1) anzuschließen und mit der Aktionstaste in Richtung Bildschirm zu halten.

Programmende bzw. Programmabbruch

Wird ein Programm normal beendet, so wird nach einem kurzen Abschlußbild der Kommandomodus wieder erreicht. Sie können das an der Ausgabe

```
OK
>■
```

erkennen und dann im BASIC weiterarbeiten. Wird ein Programm jedoch vorzeitig mittels der STOP-Taste abgebrochen, so erscheint die Ausschrift

```
BREAK IN nnnn
OK
>■
```

wobei für nnnn die Zeilennummer steht, bei deren Abarbeitung das Programm abgebrochen wurde. Es ist dabei möglich, daß vom Programm

nicht der gesamte Bildschirm als Ausgabebereich freigegeben wurde oder die angegebene Ausschrift sogar nicht vollständig sichtbar wird.

Die Grundeinstellung des Bildschirms wird dann durch Eingab von

```
WINDOW [ENTER]
CLS [ENTER]
```

erreicht (vgl. auch Abschnitte 4.4 und 4.15). Ein erneuter Start des Programms durch [RUN] ist natürlich möglich.
Vor dem Einlesen eines neuen Programms muß durch

```
NEU [ENTER]
```

das alte Programm gelöscht werden. Im Ausnahmefall ist auch der Zeichenkettenspeicherbereich durch

```
CLEAR 256 [ENTER]
```

zurückzusetzen (vgl. Abschnitt 4.17).

3.4. Erste Schritte zur BASIC-Programmierung

Sicher wissen Sie bereits, daß ein Rechnerprogramm aus ein Folge von Anweisungen besteht, die dem Computer mitteilen, welche Operationen mit Zahlen oder Zeichen er in welcher Reihenfolge ausführen soll.
Im BASIC des „robotron Z 9001“ beginnt jede Programmzeile mit einer Zeilennummer und enthält eine oder mehrere Anweisungen, die durch einen Doppelpunkt getrennt werden. Die Zeilennummern legen gleichzeitig die Abarbeitungsreihenfolge der Anweisungen fest, wenn sie nicht durch Sprunganweisung geändert wird.
Bei der Erarbeitung eines Programms wird jede Programmzeile über die Tastatur eingegeben. Zur Erleichterung späterer Programmergänzungen sollte dabei von Beginn an im Zehnerabstand numeriert werden.

Programmeingabe

Als erstes Übungsbeispiel soll ein Programm zur Berechnung des Mittelwertes von drei Zahlen aufgestellt werden. Im einfachsten Fall müssen Sie dazu folgendes eingeben:

```
10  A=5 [ENTER]
```

```
20  B=7  [ENTER]
30  C=12 [ENTER]
40  M=(A+B+C)/3 [ENTER]
50  PRINT"MITTELWERT   :";M [ENTER]
```

Beachten Sie bitte:

1. Die Eingabe jeder Programmzeile muß mit [ENTER] abgeschlossen werden. Erst dann wird die Zeile in das Programm. übernommen.
2. Fehler bei der Eingabe einer Programmzeile (z. B. Tippfehler) können, wie im Abschnitt 3.2 beschrieben, korrigiert werden, falls noch nicht [ENTER] betätigt wurde.
3. Bereits mit [ENTER] übernommene Zeilen können nach Neueingabe mit der gleichen Zeilennummer geändert bzw. überschrieben werden. Eine Programmzeile kann gelöscht werden, indem nur die Zeilennummer, und [ENTER] eingegeben werden. Ausführlicher wird auf Korrekturprobleme im Abschnitt 4.7 (EDIT-Kommando) eingegangen.

Programmabarbeitung

Nach beendeter Eingabe können Sie Ihr Programm starten, indem Sie die Taste [RUN] („Start“) betätigen. Danach arbeitet der Computer die Programmzeilen der Reihe nach ab. Beim angegebenen Mittelwertprogramm passiert dabei folgendes:

Durch die Anweisungen 10 bis 30 werden die Variablen A, B und C mit den Werten 5, 7 und 12 belegt. In Zeile 40 wird der Mittelwert M berechnet und in Zeile 50 auf dem Bildschirm ausgegeben.

```
>RUN
Mittelwert    : 8
OK
>■
```

Durch „OK“ wird das Ende der Programmabarbeitung angezeigt. Der Rechner wartet nun wieder auf eine Anweisung bzw. ein Kommando.

Programmänderung

Will man die Mittelwerte verschiedener Zahlengruppen berechnen, so ist es unzweckmäßig, wenn man die Zeilen 10 bis 30 immer wieder neu schreiben muß. Deshalb gibt es eine Eingabeanweisung, die durch das Schlüsselwort INPUT charakterisiert wird. Mit Hilfe der INPUT-Anweisung können Sie während des Programmlaufs Zahlen oder Zeichen eingeben, d. h. Variable mit Eingabedaten belegen. In unserem Beispiel kann die Zeile 10 einfach überschrieben (ersetzt) werden, indem Sie eingeben:

```
10 INPUT "3 Zahlen eingeben :";A,B,C [ENTER]
```

Das Löschen der jetzt überflüssigen Zeilen 20 und 30 erfolgt einfach durch

```
20 [ENTER]
30 [ENTER]
```

Wenn Sie nun wieder [RUN] betätigen, erscheint durch die INPUT-Anweisung in Zeile 10 die Aufforderung

```
3 Zahlen eingeben : ■
```

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

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

Danach wird gemäß Zeile 40 der Mittelwert M berechnet und mit Hilfe der Anweisung 50 auf dem Bildschirm ausgegeben. Insgesamt entsteht folgendes Bild:

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

Diese kleine Rechnung läßt sich sicher auch auf einem Taschenrechner sehr schnell ausführen. Wollen wir aber zum arithmetischen Mittelwert gleichzeitig noch das geometrische und quadratische Mittel bestimmen, so ist das im BASIC wesentlich einfacher möglich.
Zur Realisierung dieser Programmerweiterung wird zum bereits gespeicherten Programm folgendes eingegeben:

a) Zur Berechnung der gesuchten Werte wird folgendes eingefügt:

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

b) Zur Ausgabe der Werte wird ergänzt:

```
60 PRINT"Geometr. Mittel:";MG [ENTER]
70 PRINT"Quadrat. Mittel:";MQ [ENTER]
```

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

```
3 Zahlen eingeben: 5,7,12
Mittelwert         : 8
Geometr. Mittel: 7.4887
Quadrat. Mittel: 8.52448
OK
>■
```

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

Darstellung von Programmen auf dem Bildschirm

Nach einigen Modifikationen des Programms muß man sich in der Regel wieder einen genauen Überblick über das aktuell gespeicherte Programm verschaffen. Dazu wird der Programmtext durch Drücken der [LIST]-Taste am Bildschirm ausgegeben („gelistet“). Wurde das Programm zur Mittelwertberechnung wie oben beschrieben eingegeben und ergänzt, so wird es nach [LIST] Wie folgt angezeigt:

```
>LIST

10 INPUT"3 Zahlen eingeben:";A,B,C
20 M=(A+B+C)/3
30 MG=(A*B*C)^(1/3)
40 MQ=SQR((A*A+B*B+C*C)/3)
50 PRINT "MITTELWERT     : ";M
60 PRINT "Geometr. Mittel: ";MG
70 PRINT "Quadrat. Mittel: ";MQ
OK
>■
```


Zeichenketten

Neben den im ersten Beispiel verwendeten numerischen Variablen zur Speicherung von Zahlen kann sich der BASIC-Interpreter auch Zeichenketten und Zeichenkettenvariable merken. An die Bezeichnung dieser Variablen wird jeweils das Zusatzzeichen $ (Dollarzeichen) angefügt, z. B.

B$, A$, Z$, T$.

Eine Wertzuweisung für solche Variablen erfolgt, wie auch für numerische Variable, durch das Gleichheitszeichen oder auch eine INPUT-Anweisung.

Beispiele:

```
20 A$="ZEICHENFOLGE"
21 T$="robotron"
22 Z$="beliebiger Text :"
40 INPUT"Texteingabe :";W$
```

Bevor Sie ein kleines Programm mit einer Zeichenkettenvariablen ausprobieren, müssen Sie das "alte" Programm durch

```
NEW [ENTER]
```

löschen. Kontrollieren Sie die Wirkung von NEW, indem Sie anschließend [LIST] drücken!
Geben Sie nun ein:

```
10 INPUT"IHR GEBURTSTAG:";T$ [ENTER]
20 PRINT"Sie wurden am ";T$;" geboren." [ENTER]
```

und starten Sie durch [RUN]. Nach der entsprechenden Aufforderung geben Sie Ihren Geburtstag ein, z. B.

```
28. Februar 1950 [ENTER]
```

Nachdem Sie sich die Antwort des Rechners angesehen haben, können Sie das Programm erneut starten. Geben Sie Ihren Geburtstag nun anders ein, z. B. also

```
28.2.1950 [ENTER]
```

Was macht Ihr Computer? Er belegt die „Zeichenkettenvariable“ T$ in Zeile 10 mit der von Ihnen eingegebenen Zeichenfolge und gibt Sie in Zeile 20 zwischen zwei festen Texten („Zeichenkettenkonstanten“) wieder aus.
Im Abschnitt 4.13 wird die Arbeit mit solchen Zeichenketten eingehend erläutert.

Löschen des Bildschirmes
Sie werden unterdessen festgestellt haben, daß die Inhalte der Bildschirmzeilen nach oben rücken, sobald der untere Bildrand erreicht ist. Gelegentlich möchten Sie jedoch einen „sauberen“ (leeren) Bildschirm zu Beginn eines Programms haben. Dazu dient die CLS-Anweisung, die sowohl sofort, d. h. im Kommandomodus, als auch im Programm ausgeführt werden kann.
Probieren Sie Z. B.

```
CLS [ENTER]
```

aus, und ergänzen Sie anschließend das vorhandene Programm durch die Zeile 5:

```
5 CLS [ENTER]
```

Die Wirkung dieser Anweisung erkennen Sie durch mehrmaligen Programmstart. Versuchen Sie nun, Ihr erworbenes Wissen zu festigen, indem Sie eigene kleine Programme in den Rechner eingeben und abarbeiten. Dabei können Sie analog zu den oben angeführten Beispielen vorgehen. Wenn Sie jedoch mehr über BASIC erfahren wollen, so wenden Sie sich den systematischen und ausführlichen Darstellungen im Abschnitt 4 zu.

4. Programmierung in BASIC

Aus den Abschnitten 1 und 3 wissen Sie jetzt schon eine ganze Menge über BASIC. Bevor Sie sich nun intensiver mit dieser leicht erlernbaren Programmiersprache befassen, hier noch einige Ratschläge:
Die folgende BASIC-Beschreibung ist so angelegt, daß Sie sowohl in der Reihenfolge der Abschnitte die Sprache und ihre Anwendung auf einfache

Weise erlernen können als auch, was hin und wieder nötig sein wird, durch Nachschlagen schnell die wesentlichen Informationen finden können.
Die bei der Darstellung der einzelnen Anweisungen und Kommandos gegebenen, teilweise sehr detaillierten Hinweise können Sie bei einem ersten Durcharbeiten ohne Nachteil übergehen. Wichtig ist, daß Sie praktische Erfahrungen sammeln, indem Sie möglichst viele Beispiele nachvollziehen. Denn Sie wissen ja - „Probieren geht über Studieren“!
Trotzdem: Der Heimcomputer robotron Z 9001 will es, wie jeder andere Computer, von Ihnen stets genau wissen. Damit Sie die „Grammatik“ der Sprache, in der Sie sich mit ihm verständigen, möglichst schnell und sicher überblicken können, werden im nächsten Abschnitt einige wenige Darstellungsregeln eingeführt, die später durchgängig benutzt werden.

4.1. Schreibweise und Darstellungsregeln

Was ist ein BASIC-Programm, und wie schreibt man es? Das BASIC-Programm ist eine Folge von numerierten BASIC-Programmzeilen (kurz: BASIC-Zeilen), die Anweisungen enthalten und die in aufsteigender Folge ihrer Zeilennummern nacheinander ausgeführt werden. Die BASIC-Programmzeile enthält eine oder mehrere BASIC-Anweisungen, die durch Doppelpunkt : voneinander getrennt werden. Sie kann bis zu 72 Zeichen (also fast zwei Bildschirmzeilen) lang sein. Leerzeichen zählen dabei mit, die vorangestellte Zeilennummer nicht. Beispiel:

```
10 CLS:PRINT "BASIC IST EINFACH!"
```

Die BASIC-Anweisung
dient der Formulierung einer bestimmten Aufgabe, die der Heimcomputer zum Zeitpunkt ihrer Ausführung erfüllen soll. Sie kann natürlich nicht länger als eine BASIC-Programmzeile sein.

Die BASIC-Zeilennummer
muß im Bereich von 1 bis 65527 liegen, ganzzahlig sein und darf nur einmal auftreten.

Die BASIC-Programmeingabe
kann in Groß- oder Kleinbuchstaben erfolgen, wobei letztere automatisch in Großbuchstaben gewandelt werden, wenn sie nicht in Anführungszeichen eingeschlossen (als Zeichenkette), als Eingabedaten für Zeichenkettenvariable oder in Kommentaranweisungen auftreten. Leerzeichen

werden in der BASIC-Programmzeile mitgespeichert. Sie sind ohne Bedeutung, falls nicht, wenn man sie wegläßt, ein BASIC-Schlüsselwort (siehe Anhang G) an falscher Stelle entsteht.

Das BASIC-Kommando
und die sofort ausführbare BASIC-Anweisung unterscheiden sich von der BASIC-Programmanweisung äußerlich durch das Fehlen einer BASIC-Zeilennummer. Zur Darstellung der Anweisungen und Kommandos werden in diesem Handbuch folgende Vereinfachungen benutzt:

Darstellung	Bedeutung
Worte in Großbuchstaben	Schlüsselworte von BASIC, die exakt so geschrieben werden müssen (z. B. **AUTO**).
Worte in Kleinbuchstaben	Parameter (Ersatzworte), die vom Programmierer durch aktuelle Zahlen, Zeichen, Variablenbezeichnungen, Ausdrücke u. ä. ersetzt werden müssen (z. B. *zeilennummer*).
Sonderzeichen	Mit Ausnahme von [ ] Sonderzeichen von BASIC, die exakt so geschrieben werden müssen.
[text]	Der *text* (ohne []) kann wahlweise auftreten oder entfallen.
[text] ...	Der *text* (ohne []) kann wahlweise mehrfach hintereinander auftreten (bis zur Maximallänge von 72 Zeichen/BASIC-Zeile).
0	Null (zur Unterscheidung vom Buchstaben O); wird nur im BASIC-Text verwendet.
[TASTE]	Taste der Tastatur Ihres Heimcomputers

In den angeführten Beispielen wird in der Regel auf die Darstellung des Eingabeabschlusses mit [ENTER] und auch der Vollzugsmeldung OK sowie des Aufforderungszeichens > verzichtet.

4.2. Elemente von BASIC

A, 8, d, *, I, ...	Zeichensatz
123, 4.75, "HAUS", ...	Konstanten
X, AB, Z$, TX$, ...	Variable
ABS, ATN, COS, EXP. INT, LN	Standard
SGN, SIN, SQR, TAN, PI	-funktionen
A-EXP(X-3), "NAME: "+NA$(I)	Ausdrücke

Alle Anweisungen und Kommandos in BASIC bestehen aus Schlüsselwörtern, die der englischen Sprache entstammen (z. B. READ für „Lies“ oder GOTO für „Setze fort bei“), und Elementen, die in ähnlicher Weise im Mathematikunterricht jeder Schule Verwendung finden. Außer mit Zahlen kann jedoch Ihr Heimcomputer auch noch mit Text, sogenannten Zeichenketten, operieren. Er kann diese ähnlich wie Zahlen miteinander verknüpfen, vergleichen oder ein- und ausgeben. Allerdings muß der Programmierer beachten, ob er in einer Anweisung mit Zahlen oder Zeichenketten arbeitet, denn nur in wenigen Anweisungen dürfen diese Datentypen auch gemischt auftreten.
Bevor Sie mit der Formulierung einzelner Anweisungen oder Kommandos im BASIC beginnen, informieren Sie sich zunächst über die „Bausteine“, die Sie dabei immer wieder verwenden werden.
Für den „Einsteiger“ in die BASIC-Programmierung genügt zunächst ein Überblick. Einzelheiten sollten bei Bedarf später nachgeschlagen werden.

Zeichensatz
Der Zeichensatz besteht aus alphanumerischen und Sonderzeichen sowie 128 speziellen Grafikzeichen. Außerdem enthält er eine Reihe von Steuerzeichen, die nicht auf dem Bildschirm dargestellt werden können. Jedes Zeichen wird intern in einem Byte (8 Bits) verschlüsselt. Mit Ausnahme der Grafik- und einiger Steuerzeichen geschieht das nach dem international gebräuchlichen ASCII-Code. Anhang A zeigt den vollständigen Zeichensatz des „robotron Z 9001“.

Konstanten
Konstanten sind feste, d. h. unveränderliche, Werte, die im BASIC-Programm benutzt werden. Nach dem Datentyp ist zwischen numerischen Konstanten (Zahlen) und Zeichenkettenkonstanten (Text) zu unterscheiden:

Numerische Konstanten

a) Ganze Zahlen und Festkommazahlen
Sie werden wie üblich benutzt. Zu beachten ist lediglich, daß
- statt des Dezimalkommas unbedingt ein Dezimalpunkt gesetzt werden muß (1/2 darf also nicht 0,5, sondern muß 0.5 geschrieben werden),
- bei Eingabe von mehr als 6 gültigen Ziffernstellen auf 6 Stellen gerundet wird (1.23456789 wird intern zu 1.23457)

- positives Vorzeichen und auch die 0 vor dem Dezimalpunkt entfallen können (statt +5.5 genügt 5.5, statt 0.33 genügt .33)

Beispiele: 100, -1.345, -3.001, -3, 0

b) Gleitkommazahlen
Zur Unterstützung der Lesbarkeit wandelt Ihr Heimcomputer alle Zahlen außerhalb der Bereiche von ± 0.01 bis ± 999999 bei der Bildschirmausgabe in Gleitkommazahlen (wissenschaftliche Darstellung) um. Sie können jedoch auch jede Zahl des Zahlenbereiches in dieser Form eingeben. Die Darstellung hat die Form

mantisse* E *exponent

Z. B. -1.03E+7
Der Wert der Zahl berechnet sich zu

zahlenwert = *mantisse* * $10^{exponent}$

d. h., für das obige Beispiel ergibt sich der Wert -10300000. Zu beachten ist, daß
- die Mantisse eine ganze oder Festkommazahl sein muß,
- der Exponent ganzzahlig sein muß, wobei ein positives Vorzeichen auch entfallen kann.

Beispiele:

ganze Zahl, Festkommazahl	Gleitkommazahl
30000	3E4
-20500	-20.5E+3
12300000	1.23E+7
0.000000123	1.23E-7

c) Zahlenbereich
Der Bereich zulässiger Zahlenwerte wird durch die interne 4-Byte-Gleitkommadarstellung sämtlicher Zahlen bestimmt. Der zulässige Zahlenbereich wird gebildet durch ± 9.40396E-39 bis ± 1.70141E+38 und die 0.

Zahlen, deren Absolutwert größer ist, führen zu einem Überlauffehler. Bei Unterschreitung des zulässigen Zahlenbereiches wird automatisch der Wert 0 angenommen.

Zeichenkettenkonstanten
Zeichenkettenkonstanten können aus alphanumerischen, grafischen oder Sonderzeichen des Zeichensatzes gebildet werden. Sie werden mit Anführungszeichen vom übrigen BASIC-Text abgegrenzt und dürfen 0 bis maximal 255 Zeichen lang sein. Im Sonderfall einer Zeichenkette der Länge 0 Zeichen spricht man von einer „leeren“ Zeichenkette. Diese darf nicht verwechselt werden mit einer Zeichenkette, die ein oder mehrere Leerzeichen enthält! Leerzeichen in Zeichenketten werden wie jedes andere Zeichen behandelt.

Beispiele:
"HEIMCOMPUTER robotron Z 9001"
"" (leere Zeichenkette!)
" " (Zeichenkette, die 2 Leerzeichen enthält)
"1234" (numerische Zeichenkette)
"/*" (Zeichenkette mit Sonderzeichen)

Variable
Unter einer Variablen versteht man eine Größe, deren Wert (im Gegensatz zur Konstanten) verändert werden kann. Im BASIC- Programm erhält jede Variable, wie in der Mathematik auch, einen Namen. Der Computer reserviert außerdem für jede Variable Speicherplatz, auf dem er sich den aktuellen Wert dieser Variablen merkt. Somit kann er mit einer Variablen rechnen. Er kann ihr Werte zuweisen oder auch Operationen mit ihr ausführen.

Beispiel:
Statt der Konstanten 3 und 4 in der Anweisung

```
PRINT 3+4
```

könnten Sie ebensogut mit 2 Variablen arbeiten, denen Sie vor der Addition Ihre gewünschten Werte zuweisen, also

```
A=3
B=4
PRINT A+B
```

Die Verwendung dieser Variablen A und B hat den Vorteil, daß unsere Zahlenwerte unter ihrem Namen erhalten bleiben, solange keine neue

Wertzuweisung für sie vorgenommen wird. Wir könnten im obigen Beispiel weitere Anweisungen ergänzen, z. B.

```
PRINT B-A
```

oder auch (mit Änderung des Wertes der Variablen A auf 21)

```
A=14+A+B
```

Erst die Verwendung von Variablen ermöglicht in den meisten Fällen die Lösung von praktischen Aufgaben. Sie gestatten es, Eingabedaten, Zwischenergebnisse usw. zu speichern und wieder aufzufinden. Außerdem gelingt mit ihrer Hilfe die von den Daten des speziellen Anwendungsfalles unabhängige und deshalb wiederverwendungsfähige Lösung.

Ebenso wie bei den Konstanten sind zwei Arten (Typen) von Variablen zu unterscheiden:

Numerische Variable und
Zeichenkettenvariable

Der Typ wird jeweils bei der Namensvergabe festgelegt: Eine Zeichenkettenvariable wird durch das Zeichen $ am Ende ihres Namens gekennzeichnet. Sonst können die Variablennamen bei Beachtung folgender Einschränkungen frei gewählt werden:

- Sie müssen mit einem Buchstaben beginnen und dürfen beliebig lang sein. Für den Heimcomputer sind jedoch nur die ersten beiden Zeichen bedeutsam, die übrigen werden zwar mitgespeichert, dienen aber nicht zur internen Unterscheidung verschiedener Variablen.

- Im Namen dürfen an keiner Stelle BASIC-Schlüsselworte enthalten sein (siehe Anhang G).

Beispiele:

Variablenname	Bemerkung
NAME	richtig, aber nur die ersten zwei Buchstaben sind von Bedeutung
X	richtig
A1	richtig
1Y	falsch, Variablenname muß mit einem Buchstaben beginnen
B!	falsch, Sonderzeichen außer $ sind unzulässig
C X	richtig, Leerzeichen sind bedeutungslos
TOR	falsch, enthält das reservierte Wort OR (logischer Operator)
NA	richtig, aber identisch mit der Variablen NAME
TEXT$	richtig, Zeichenkettenvariable
na	richtig, der BASIC-Interpreter wandelt die Kleinbuchstaben in Variablenbezeichnungen bei der Eingabe in Großbuchstaben um, identisch mit (NAME und NA)
TE$	richtig, identisch mit TEXT$

Die bislang verwendeten Variablen heißen auch „einfache Variable“, da jede einzelne Variable ihren eigenen Namen besitzt. Es können auch Speicherplätze für mehrere Werte unter einem gemeinsamen Namen zusammengefaßt werden. Man spricht dann von einer Feldvariablen oder kurz, einem Feld (siehe Abschnitt 4.8). Hierbei verkörpern die Feldelemente Einzelwerte und sind durch Angabe von Indizes (in runden Klammern hinter dem Feldnamen) eindeutig bestimmt. Ein solches Feldelement heißt deshalb auch „indizierte Variable“ und ist inhaltlich einer einfachen Variablen gleichzusetzen. Die Regeln zur Namensvergabe gelten gleichermaßen für Feldvariable wie für einfache Variable.

Beispiele:

```
A (5)=13
```

Dem Feldelement des Feldes A mit dem Index 5 wird der Wert 13 zugewiesen. (Die Zählung der Feldelemente beginnt stets mit Null!)

```
PRINT T$(3,5)
```

Eine Zeichenkette des Zeichenkettenfeldes T$ wird ausgegeben.

Standardfunktionen
Anstelle von Konstanten oder Variablen können Sie in Anweisungen ebensogut eine der im BASIC enthaltenen mathematischen Standard-funktionen einsetzen, z. B.

```
A=128/2
PRINT SQR(A)
```

Dann wird die Quadratwurzel des vorher bestimmten Wertes der Variablen A berechnet und ausgegeben. (Beachten Sie auch, daß für die Division / statt : geschrieben werden muß!)

Folgende Standardfunktionen stehen Ihnen zur Verfügung:

ABS(X) - Absoluter Betrag
ATN(X) - Arcustangens
COS(X) - Cosinus (X im Bogenmaß)
EXP(X) - Exponentialfunktion ex (x < 87. 3366)
INT(X) - Nächstkleinere ganze Zahl zu X
LN(X) - Natürlicher Logarithmus (X > 0)
SGN(X) - Vorzeichenfunktion : SGN(X)
1 fürX>0
0 fürX=0
1 fürX<0
SIN(X) - Sinus (X im Bogenmaß)
SQR(X) - Quadratwurzei (X - 0)
TAN(X) - Tangens (X im Bogenmaß)
PI - Konstante π = 3.14159

Ausdrücke
Ein Ausdruck besteht im einfachsten Fall aus einer einzelnen Konstanten, im allgemeinen aus einer Verknüpfung von Operanden (Konstanten, Variablen, Funktionsaufrufen, Ausdrücken) mittels Operationszeichen (kurz: Operatoren).

Er stellt wiederum genau einen aktuellen Wert dar, der numerisch oder eine Zeichenkette sein kann. Beispielsweise hat der Ausdruck PI/2 den (numerischen) Wert 1.5708. Demzufolge gibt es numerische und Zeichenkettenausdrücke, deren Wert, falls er einer Variablen zugewiesen werden soll, vom selben Typ wie die Variable sein muß. Wichtig ist, daß die Wertbestimmung eines Ausdrucks nach festen Regeln (Reihenfolge!) erfolgt, die mit den in der Elementarmathematik üblichen Rechenregeln übereinstimmen.

Operationen mit Zahlen
Die nachfolgende Tabelle zeigt Rang und Schreibweise der Operatoren:

Rang	Operator	Bedeutung		
1	^	Potenzierung	\	
2	-	negatives Vorzeichen	I	
3	*	Multiplikation	I	arithmetische
	/	Division	I	Operatoren
4	+	Addition	I	
	-	Subtraktion	/	
5	<	kleiner als	\	
	<=	kleiner als oder gleich	I	
	=	gleich	I	
	Vergleichs-			
	<>	ungleich	I	
	Operatoren			
	>	größer als	I	
	>=	größer als oder gleich	/	
6	NOT	Negation (Verneinung)	\	logische
	AND	Konjunktion (UND)	I	
	Operatoren			
	OR	Disjunktion (ODER)	/	.

Die Berechnung eines Ausdrucks erfolgt unter Berücksichtigung des Rangs der Operatoren. Dem Rang 1 kommt die höchste Priorität zu, diese Operation wird zuerst ausgeführt. Besitzen zwei Operatoren gleichen Rang, erfolgt ihre Abarbeitung von links nach rechts. Klammerungen und Funktionsaufrufe werden noch vor den in der Tabelle mit Rang ange-gebenen Operationen ausgewertet.
Trifft der BASIC-Interpreter während der Auswertung eines Ausdrucks auf eine Division durch Null, so wird die Fehlermeldung /0 angezeigt.
Wird der Betrag des Ergebnisses einer Berechnung größer als der maximal erlaubte Wert (1.70141E+38), reagiert der BASIC- lnterpreter mit der Fehlermeldung OV.
Der logische Operator NOT ist, ebenso wie das negative Vorzeichen, ein einseitiger Operator, d. h., er bezieht sich nur auf einen Operanden, vor dem er steht. Die übrigen Operatoren sind zweiseitig und verknüpfen zwei logische bzw. arithmetische Operanden.

Ausdrücke, die logische Operatoren und Vergleichsoperatoren enthalten, bezeichnet man als logische Ausdrücke. Sie liefern einen logischen Wert, der „wahr“ oder „falsch“ sein kann. Das Ergebnis kann in Verbindung mit der Anweisung IF ... THEN ... ELSE zur Steuerung des Programmablaufes genutzt werden.
Wenn in einem Ausdruck auch arithmetische Operatoren auftreten, werden nach der oben angegebenen Reihenfolge diese zuerst abgearbeitet.

Beispiele:

```
A+B<(X-1)/Y
```

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

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

Das sind zwei Beispiele, wie das Ergebnis von logischen Ausdrücken zur Steuerung von Programmabläufen genutzt werden kann.
Beispiele zur Bildung von numerischen Ausdrücken:

Nr.	BASIC-Darstellung	mathematische Schreibweise	Bemerkungen
1	A/B/C	$\frac{A}{B - C}$	richtig
2	A/(B*C)	$\frac{A}{B * C}$	richtig, Beispiele 1 und 2 liefern unter Umständen verschiedene Werte und sind nicht identisch (wegen unterschiedlicher Abarbeitungsreihenfolge, Rundung!)
3	K*-52	-52 * K	richtig
4	K*(-52)	-52 * K	richtig
5	-52*K	-52 * K	richtig
6	4*PI*R^3/3	$4/3\pi * R^3$	richtig
7	SQR(A^2+B^2)	$\sqrt{A^2 + B^2}$	richtig, Verwendg. einer Funktion
8	A+FELD(-3)	-	falsch, -3 ist ein unzulässig. Index

Beispiele für syntaktisch richtige und falsche Vergleichs- und logische Ausdrücke:

Nr.	BASIC-Darstellung	mathematische Schreibweise	Bemerkungen zur BASIC-Schreibweise
1	NOT(A>B)	$\neg(A > B)$	richtig
2	A <= B	$A \leq B$	richtig, logisch gleichwertig zu Beispiel 1
3	A<X AND X<=B	$A < X < B$	richtig
4	A OR B*(I$J)	-	falsch
5	X< =Y>Z	-	falsch

Operationen mit Zeichenketten
Zeichenketten können durch das Operationszeichen + miteinander verkettet werden.

Beispiel:

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

Außerdem können Zeichenketten mit Hilfe der Vergleichsoperatoren verglichen werden. Der Vergleich erfolgt zeichenweise von links nach rechts. Dabei wird die interne Darstellung der jeweiligen Zeichen miteinander verglichen. Sind alle Zeichencodes der beiden Zeichenketten gleich, so sind auch die Zeichenketten identisch. Andernfalls gilt die Zeichenkette als die kleinere, in welcher zuerst ein Zeichen mit niedrigerem ASCII-Code auftritt.
Wird während eines Vergleichs das Ende einer Zeichenkette erreicht, so gilt die kürzere als die kleinere Zeichenkette.
Die folgenden logischen Ausdrücke liefern den Wert „wahr“:

```
"AA"<"AB"
"X%">"X$" AND "a"<"b"
"SPACE ">"SPACE"
D$<"84."+M$++"11"
```

(mit D$="84.11.08" und M$="11. ")

Den Wert „falsch“ liefern dagegen die logischen Ausdrücke

```
"B"<"A" AND "C"<"D"
A$>B$
```

mit A$="TEXT1" und B$="TEXT2".
Auch bei der Bestimmung des Wertes von Zeichenkettenausdrücken gilt, daß Klammerungen und Funktionsaufrufe zuerst ausgewertet werden. Ebenso hat der Operator + Vorrang vor den Vergleichs- bzw. logischen Operatoren.

4.3. Programmeingabe, -anzeige und -start

NEW		Löschen von Programmen
programmzeile		direkte Programmeingabe
AUTO		automatische Zeilennumerierung
LIST,	[LIST]	Programmanzeige
LINES		Zeilenanzahl je Anzeigeschritt
RUN,	[RUN]	Programmstart

Bevor Sie mit der praktischen Programmierung beginnen können, benötigen Sie noch einige „Werkzeuge“. Der BASIC-Interpreter des „robotron Z 9001“ stellt sie Ihnen mit den oben angegebenen Kommandofunktionen zur Verfügung. Mit ihrer Hilfe können Sie BASIC-Programmzeilen über die Tastatur eingeben und in den Arbeitsspeicher übernehmen, die gespeicherten BASIC-Zeilen auf dem Bildschirm anzeigen und schließlich auch die Ausführung des Programms starten.

Beachten Sie bitte.
- Voraussetzung für die Eingabe eines Kommandos bzw. für die direkte Programmeingabe ist, daß sich der BASIC-Interpreter im Kommandomodus befindet.
- Ihre Eingaben sind mit der [ENTER]-Taste abzuschließen. Dies entfällt, wenn Sie von einer BASIC-Funktionstaste Gebrauch machen ([LIST], [RUN]).
- Benutzen Sie zur Eingabe- bzw. Korrekturerleichterung die Tasten [←], [→], [I←], [→I], [CL LN], [DEL], [INS], die im Abschnitt 3.2 beschrieben wurden.

- Nach Ausführung des jeweiligen Kommandos kehrt der BASIC-Interpreter in den Kommandomodus zurück.
- Erkennt der BASIC-Interpreter Fehler bei der Kommandoausführung, erscheint eine Fehlermitteilung (siehe Anhang H), und es wird ebenfalls der Kommandomodus erreicht.

Löschen von Programmen

Format[1]: **NEW**
Funktion:
Das im Arbeitsspeicher befindliche BASIC-Programm wird gelöscht.

Hinweise:
1. Dieses Kommando sollte vor der Neueingabe eines BASIC-Programmes gegeben werden, um eine ungewollte Überlagerung mit einem vorhandenen Programm zu vermeiden.
2. Durch NEW wird eine abweichend vom Standard festgelegte Größe des Speicherbereiches für Zeichenkettenvariable (siehe Abschnitt 4.17) nicht verändert.
3. Nach NEW haben numerische Variable den Wert Null, und Zeichenkettenvariable bekommen die leere Zeichenkette zugewiesen.

Direkte Programmeingabe
Format[2]: *zeilennummer anweisung[:anweisung]...*
Funktion:
Die BASIC-Programmzeile wird nach Betätigung der [ENTER]-Taste entsprechend ihrer Zeilennummer in das im Arbeitsspeicher befindliche BASIC-Programm eingeordnet.

Hinweise:
1. Eine unter der angegebenen Zeilennummer bereits vorhandene BASIC-Zeile wird durch die zuletzt eingegebene ersetzt.

---

1) Als Format wird im folgenden eine formale Beschreibung der Anweisungen, Kommandos und BASIC-Funktionen bezeichnet.
2) Eckige Klammern kennzeichnen wahlfreie Angaben und dienen zur Unterscheidung von Pflichtangaben. Beachten Sie, daß die eckigen Klammern selbst nicht geschrieben werden.

2. Die Eingabe einer Zeilennummer ohne nachfolgende BASIC-Anweisung (*zeilennummer* [ENTER]) bewirkt das Streichen einer vorhandenen BASIC-Zeile mit dieser Nummer.

Beispiel:
Geben Sie folgende BASIC-Zeilen ein:

```
10 PRINT "1.ZEILE" [ENTER]
20 PRINT "2.ZEILE" [ENTER]
```

Betätigen Sie nun die [LIST]-Taste und überzeugen Sie sich, daß der BASIC-Interpreter Ihre Programmzeilen gespeichert hat.

Geben Sie jetzt ein:

```
15 PRINT "2.ZEILE" [ENTER]
20 PRINT "3.ZEILE" [ENTER]
5 PRINT "UEBERSCHRIFT" [ENTER]
```

Wenn Sie anschließend erneut die [LIST]-Taste betätigen, finden Sie die gewünschte BASIC-Zeilenfolge in Ihrem aktuellen „Programm vor:

```
5 PRINT "UEBERSCHRIFT" [ENTER]
10 PRINT "1.ZEILE" [ENTER]
15 PRINT "2.ZEILE" [ENTER]
20 PRINT "3.ZEILE" [ENTER]
```

Besonders bei der Eingabe von BASIC-Zeilen in geordneter Reihenfolge ersparen Sie sich Eingabeaufwand, wenn Sie die Zeilennummernvergabe dem Heimcomputer übertragen.

Programmeingabe mit automatischer Zellennumerierung

Format:
AUTO [*zeilennummer* [,*schrittvveite*]]
zeilennummer - erste vom Heimcomputer zu vergebende Zeilennummer (Standardwert: 10)
schrittweite - Differenz zweier aufeinanderfolgender Zeilennummern (Standardwert: 10)

Funktion:
Erzeugung aufsteigender BASIC-Zeilennummern, die vom Nutzer zu vollständigen BASIC-Programmzeilen ergänzt werden können.
Hinweise:
1. Erzeugt der Heimcomputer eine Zeilennummer, unter der im Arbeitsspeicher bereits eine BASIC-Zeile existiert, erscheint ein Stern * hinter der Zeilennummer auf dem Bildschirm. Sie können dann
   - den * ignorieren, die BASIC-Zeile vervollständigen und dadurch die bereits vorhandene ersetzen oder
   - durch eine Leereingabe (*zeilennummer* [ENTER]) die bereits vorhandene BASIC-Zeile streichen oder
   - mittels der [STOP]-Taste die Programmeingabe mit automatischer Zeilennumerierung ohne Veränderung der vorhandenen BASIC-Zelle beenden.
2. Ebenso wie bei der direkten Programmeingabe führt der Versuch, eine **nicht** vorhandene BASIC-Zeile mittels Leereingabe zu streichen, zu einer Fehlermitteilung (UL ERROR).
3. Sie beenden den AUTO-Modus durch [STOP].
4. Soll eine bereits eingegebene BASIC-Zeile geändert werden, so ist das durch Überschreiben möglich. Wesentlich bessere Korrekturmöglichkeiten für umfangreiche Änderungen bestehen jedoch bei Verwendung des EDIT-Kommandos (siehe Abschnitt 4.7).

Beispiel:
Die erneute Eingabe der vier BASIC-Zeilen des vorangegangenen Beispiels bereiten wir mit den Kommandos

```
NEW  [ENTER]
AUTO [ENTER]
```

vor.
Sie erhalten nun die erste BASIC-Zeilennummer auf dem Bildschirm und vervollständigen die BASIC-Zeile:

```
>AUTO
10 PRINT "UEBERSCHRIFT" [ENTER]
```

Mit den Folgezeilen verfahren Sie ebenso und beenden die Programmeingabe nach der Zeile 40 mit

```
50 [STOP]
```

Danach können Sie sich mittels der [LIST]-Taste das aktuelle BASIC-Programm aus dem Arbeitsspeicher auf dem Bildschirm anzeigen lassen. Um bei der Programmeingabe mit automatischer Zeilennumerierung dieselbe Zeilennummernfolge wie im vorangegangenen Beispiel zu erzielen, hätten Sie das Kommando

```
AUTO 5,5 [ENTER]
```

verwenden können.

Programmanzeige

Format:
LIST [*zeilennummer*] oder [LIST]
zeilennummer - bestimmt die BASIC-Zelle, ab der das Programm angezeigt werden soll
(Standard: niedrigste vorhandene Zeilennummer)

Funktion:
Das im Arbeitsspeicher befindliche BASIC-Programm wird abschnittsweise auf dem Bildschirm angezeigt („gelistet").

Hinweise:
1. Die Anzeige erfolgt in aufeinanderfolgenden Abschnitten mit einer festen Anzahl von BASIC-Zeilen, die durch LINES (siehe unten) festgelegt werden kann. Standardmäßig werden jeweils 10 Zeilen angezeigt. Reicht die freie Zeilenzahl auf dem Bildschirm für die Anzeige der BASIC-Zeilen nicht aus, „rollt" das Bild um die entsprechende Zeilenzahl nach oben.
2. Nach der Anzeige jedes Abschnittes wartet der Heimcomputer darauf, daß Sie die Anzeige des nächsten Abschnittes verlangen. Sie tun dies durch [ENTER]. Der Vorgang wiederholt sich, bis entweder die höchste Zeilennummer Ihres BASIC-Programms erreicht wurde oder Sie die Programmanzeige mit [STOP] beenden.
3. Die Verwendung der [LIST]-Taste hat dieselbe Wirkung wie die Eingabe des LIST-Kommandos ohne Zeilennummer, d. h. das BASIC-Programm wird ab der niedrigsten Zeilennummer angezeigt.
4. Die [LIST]-Tastenbenutzung erfordert die Gültigkeit der unteren Tastenbelegung, d. h. [SHIFT] bzw. [SHIFT LOCK] dürfen nicht wirken.

Beispiel:
Probieren Sie die gezielte Anzeige ab einer vorgegebenen Zeilennummer
aus! Bezüglich des letzten Beispieles erhalten Sie nach

```
LIST 30 [ENTER]
```

die Bildschirmanzeige

```
30 PRINT "2.ZEILE"
40 PRINT "3.ZEILE"
```

Anschließend befinden Sie sich wieder im Kommandomodus.

Auf einfache Weise können Sie jedoch auch die Anzahl der in einem Abschnitt anzuzeigenden BASIC-Zeilen selbst festlegen:

Format:
LINES *zeilenanzahl*
zeilenanzahl - ganzzahliger Wert im Bereich von 1 bis 65535
(Standard: 10)

Funktion:
Festlegung der maximalen Anzahl von BASIC-Zeilen, die bei Ausführung des LIST-Kommandos ohne Unterbrechung nacheinander auf dem Bildschirm angezeigt werden.

Hinweis:
Die mit dem LINES-Kommando getroffene Festlegung gilt bis zur erneuten Eingabe eines LINES-Kommandos.

Beispiel:
Lassen Sie sich Ihr aktuelles BASIC-Programm noch einmal Zeilenweise anzeigen:

```
LINES 1 [ENTER]
[LIST]
[ENTER]
[ENTER]
   .
   .
   .
[STOP]
```

Mit dem Kommando

```
LINES 10 [ENTER]
```

können Sie anschließend die beim Einschalten des Heimcomputers automatisch wirksame Einstellung wieder herstellen.

Programmstart

Format:
RUN [*zeilennummer*] oder [RUN]
zeilennummer - BASIC-Zeilennummer, ab der die Programmabarbeitung beginnt (Standard: niedrigste vorhandene BASIC-Zeilennummer)

Funktion: Start der Programmabarbeitung

Hinweise:
Vor der Abarbeitung der ersten BASIC-Anweisung werden sämtliche Variablen gelöscht. Numerische Variable erhalten den Wert Null, Zeichenkettenvariable die leere Zeichenkette (" ") zugewiesen. Außerdem erfolgt die Organisation der Arbeit mit internen Daten (DATA, siehe 4.9) noch vor Ausführung der ersten Anweisung.

Beispiel:
Starten Sie das zuletzt eingegebene Beispielprogramm zum Vergleich mit der Taste [RUN] und mit dem Kommando

```
RUN20 [ENTER]
```

4.4. Einfache Anweisungen

[**LET**]	VVertzuweisung
PRINT	Ausgabeanweisung
CLS	Bildschirm löschen
INPUT	Eingabeanweisung
REM oder **!**	Kommentaranweisung
BEEP	akustisches Signal

In diesem Abschnitt werden Sie die einfachsten BASIC-Anweisungen kennenlernen, so daß Sie dann in der Lage sind, eigene kleine Programme zu formulieren und auszuprobieren. Zu den ständig benötigten Anweisungen eines dialogorientierten BASIC-Programms gehören Wertzuweisungen für Variable ebenso wie Anweisungen zur Ein- und Ausgabe von Daten über Tastatur bzw. Bildschirm. Demgegenüber sind erläuternde Kommentare im Programm sowie die Möglichkeit der Ausgabe eines Summertones („Pieps“) als hilfreiche Ergänzungen anzusehen.

Noch ein wichtiger Hinweis:
In diesem und den folgenden Abschnitten wird auf den Eingabeabschluß mittels [ENTER] in der Regel nicht mehr hingewiesen!

Wertzuweisung

Format:
[**LET**] *variable* = *ausdruck*
variable - Bezeichnung einer einfachen oder indizierten Variablen (numerische oder Zeichenkettenvariable)
ausdruck - Ausdruck, **vom selben Typ** wie die Variable

Funktion:
Der aktuelle Wert des rechts stehenden Ausdrucks wird bestimmt und der Variablen zugewiesen.

Hinweise:
1. Es ist darauf zu achten, daß sämtliche im rechtsstehenden Ausdruck benutzten Variablen zum Zeitpunkt der Anweisungsausführung mit den beabsichtigten Werten belegt sind. (Zum Zeitpunkt des Programmstarts (RUN) werden alle Variablen gelöscht.)
2. Das Zeichen = fungiert in der Anweisung nicht als Gleichheitszeichen sondern als Zuweisungsoperator. Ein Vertauschen von Variabler und Ausdruck ist nicht zulässig.
3. Das BASIC-Schlüsselwort LET wird in der Praxis meist weggelassen.
4. Die Anweisung kann (ohne BASIC-Zeilennummer) ebenso wie die meisten anderen BASIC-Anweisungen im Kommandomodus sofort ausgeführt werden.

Beispiele:
Berechnen Sie die Länge der Hypotenuse c = $\sqrt{a^2+b^2}$ eines rechtwinkligen Dreiecks mit bekannten Seitenlängen a = 10 und b = 20 !

Hier das BASIC-Programm:

```
10 LET A=10
20 LET B=10
30 LET C=SQR(A*A+B*B)
40 PRINT C
```

Starten Sie es mit [RUN], ergibt sich

```
22.3607
```

für den gewünschten Wert C.

Ändern Sie jetzt eine Seitenlänge durch Ersetzen der entsprechenden BASIC-Zeile, z. 8. (unter Verzicht auf LET)

```
20 B=15
```

und starten Sie erneut, so erhalten Sie nun

```
18.0278
```

als Hypotenusenlänge.

Und nun noch ein Beispiel mit Zeichenkettenvariablen:

```
100 X$=" Z9001"
110 Y$="Heimcomputer"
120 X$=Y$+"? Glueckwunsch zum "+X$+"!"
130 PRINT X$
```

Nach

```
RUN 100
```

erscheint dann auf dem Bildschirm

```
Heimcomputer? Glueckwunsch zum Z9001!
```

Achten Sie bitte auf die doppelte Verwendung von X$ in Zeile 120! Wenn die BASIC-Zeilen 10 bis 40 noch nicht gelöscht wurden und Sie die Programmabarbeitung mittels [RUN] wiederholen, werden beide Beispiele nacheinander abgearbeitet. Überlegen Sie: warum?

Ausgabeanweisung[1])

Format:
PRINT [*ausdruck* [*trennzeichen ausdruck*] ...]
ausdruck - numerischer oder Zeichenkettenausdruck
trennzeichen - kann sein
1. Komma (,) für (Standard-) Tabellenausgabe
2. Semikolon (;) für fortlaufende Ausgabe
Funktion:
Die aktuellen Werte der in der PRINT-Anweisung angegebenen Ausdrücke werden bestimmt und entsprechend den Darstellungsregeln sowie den verwendeten Trennzeichen auf dem Bildschirm angezeigt.
Werden nach PRINT keine Ausdrücke angegeben, wird eine Leerzeile ausgegeben.

Hinweise:
1. Darstellungsregeln
   a) Darstellung von Zahlen
      Vor und nach jeder Zahl erscheint stets ein Zeichen: vor der Zahl ihr Vorzeichen (Minuszeichen oder Leerzeichen, falls positiv) und nach der Zahl ein Leerzeichen zur Trennung von eventuell anschließenden Ausgaben. Je Zahl werden bis zu 6 aufeinanderfolgende Dezimal-ziffern (außer Exponent) entsprechend der Rechengenauigkeit ausgegeben. Nullen am Ende einer Zahl (nach dem Dezimalpunkt) werden unterdrückt.
      Zahlen die betragsmäßig kleiner als 0.01 oder größer als 999999 sind, werden in Gleitkommadarstellung ausgegeben, alle übrigen Zahlen in ihrer natürlichen Darstellung, d. h. als ganze Zahl oder Festkommazahl ohne Exponent.

[1]) Erweiterte Ausgabemöglichkeiten sind in den Abschnitten 4.15 und 4.16 dargestellt

Beispiele:

PRINT-Anweisung	Bildschirmdarstellung
? 0.00123456789	1.23457E-03
? 0.0123456789	0.0123457
? 0.123456789	.123457
? 1.23456789	1.23457
? 1234.56789	1234.57
? 123456.789	1.23457
? 1234567.89	1.234567E+06
? 10000000	1E+07

b) Darstellung von Zeichenketten
Zeichenketten werden unverändert und ohne zusätzliche Zeichen ausgegeben.

2. Wirkung der Trennzeichen
   a) Trennzeichen Komma (,)
   Die 40 Zeichen fassende Bildschirmzeile wird gedanklich in 3 Zonen zu je 13 Zeichen unterteilt (am rechten Rand bleibt eine Zeichenposition frei). Die Werte der durch Komma getrennten Ausdrücke werden dann jeweils linksbündig in aufeinanderfolgenden 13-Zeichen-Zonen ausgegeben. Dabei werden die Darstellungsregeln selbstverständlich eingehalten. Sind mehr als 3 Werte auszugeben, wird die Ausgabe automatisch in der ersten Ausgabezone der nächsten Zeile fortgesetzt usw.
   b) Trennzeichen Semikolon (;)
   Die Werte werden fortlaufend unmittelbar nacheinander ausgegeben. Bei vollständig gefüllter Zeile erfolgt ebenfalls automatischer Zeilenwechsel, bis die Ausgabe aller Ausdrücke abgeschlossen ist.

3. Jede PRINT-Anweisung beginnt mit ihren Ausgaben auf einer neuen Zeile (Ausnahmen sind im Abschnitt 4.15 beschrieben). Welche Zeile des Bildschirms die nächste Zeile ist, wird durch vorangegangene PRINT-, INPUT-, CLS- und WINDOW-Anweisungen sowie eventuelle Fehlermeldungen des Betriebssystems bestimmt. Ist der Bildschirm vollständig gefüllt, und es erfolgt eine weitere Ausgabe, so „rollt“ das Bild um jeweils eine Zeile nach oben.

4. Zur Vereinfachung der Eingabe können Sie statt PRINT ohne weiteres auch einfach nur ? eingeben. Bei einer Programmanzeige mittels LIST wird das Zeichen ? wieder durch PRINT ersetzt.

Beispiele:
Die nachfolgende Tabelle zeigt ihnen einige Beispiele, die Sie durch eigene schnell ergänzen können. Überzeugen Sie sich von der Wirkung der Trennzeichen!

BASIC-Anweisung	Bildschirmausgabe
? X	123
? X,Y,Z,0.123456789	123 1E-03 -2.22222E-04 .123457
? A$	ABC
? " X"," X^2"," X^3"	X X^2 X^3
? X,X^2,X^3	123 15129 1.86087E+06
?	
? "A";"B";"C"	ABC
? "PRINT-"+A$,X*Y	PRINT-ABC .123
? "SIN(PI/4)= ";SIN(PI/4)	SIN(PI/4)= .707107
? 2^10;B$;	1024 Byte
? C$;1;B$;" PRO ZEICHEN":?	1 Byte PRO ZEICHEN
? "ZEICHENKETTE SEI","9999999", "ZAHL WIRD",9999999; "OK?"	ZEICHENKETTE SEI 9999999 ZAHL WIRD 1E+07 OK?

```
|          |          |          |
0         12|13        25|26...39
```

```
|          |          |          |
0         12|13        25|26      39|
```

^ Spaltennummer der Bildschirmanzeige

***) Anmerkungen:** **1. ? steht abkürzend für PRINT**
2. Vorausgesetzte Variablenwerte:

```
X=123              A$="ABC"
Y=0.001            B$="Bytes"
Z=-2.22222E-04     C$=""
```

Bildschirm löschen

Format **CLS**

Funktion: Die Bildschirmanzeige wird gelöscht.

Hinweise:
1. Der Löschvorgang wirkt auf den aktuellen Ausgabebereich auf dem Bildschirm, der (mittels WINDOW, siehe Abschnitt 4.15) auch abweichend vom gesamten Bildschirm festgelegt werden kann.
2. Nach Ausführung von CLS wird die Ausgabe am linken oberen Rand des aktuellen Ausgabebereiches fortgesetzt.
3. CLS kann auch im Kommandomodus vorteilhaft eingesetzt werden.

Eingabeanweisung

Format:
INPUT ["*hinweis*"] *variable* [,*variable*] ...
variable - numerische oder Zeichenkettenvariable, für die ein Wert eingegeben werden soll
hinweis - Zeichenkette, die Informationen für den Programmanwender enthält

Funktion:
Nach Ausgabe des *hinweis*-Textes wird die laufende Programmabarbeitung unterbrochen, und über Tastatur einzugebende Werte werden den in der INPUT-Anweisung aufgeführten Variablen zugewiesen. Ist dies vollständig geschehen, wird der Programmlauf fortgesetzt.

Hinweise:
1. Die INPUT-Anweisung ist nur im BASIC-Programm zulässig, nicht im Kommandomodus!
2. Auf dem Bildschirm wird der *hinweis*-Text an der Stelle im Programm ausgegeben, an der die nächste PRINT-Ausgabe erfolgen würde. Ist auf den *hinweis* verzichtet worden, erscheint statt dessen ein Fragezeichen (?). Anschließend geben Sie Daten ein und beenden die Eingabe mit [ENTER].
3. Reihenfolge, Typ und Anzahl der einzugebenden Werte müssen mit der Liste der Variablen in der INPUT-Anweisung übereinstimmen. Zeichenkettenkonstanten können, falls sie keine Leerzeichen oder Kommas enthalten, ohne die begrenzenden Anführungszeichen ("…") eingegeben werden. Das Komma dient als Trennzeichen zwischen den einzugebenden Werten und darf nicht mit dem Dezimalpunkt verwechselt werden.
4. Sind zu wenige Werte eingegeben worden, erfolgt mittels ?? die Aufforderung zur Eingabe der restlichen Werte. Sind zu viele Werte eingegeben worden, werden die überflüssigen ignoriert. Die Mitteilung EXTRA IGNORED erscheint, und die Programmabarbeitung wird fortgesetzt.
5. Bei Eingabe einer Zeichenkette statt einer numerischen Variablen erhalten Sie die Ausschrift ? REDO FROM START und müssen dann die Dateneingabe für diese INPUT-Anweisung von Anfang an wiederholen.
6. Bei einer Leereingabe für eine Variable wird deren aktueller Wert nicht verändert. Die Leereingabe erzeugen Sie durch sofortige [ENTER]-Betätigung.

Beispiel:
Berechnen Sie noch einmal die längste Seite eines rechtwinkligen Dreiecks. Im Gegensatz zum Beispiel bei der Erläuterung der LET-Anweisung sollen jetzt die Seitenlängen a und b erst während der Programmabarbeitung eingegeben werden.
Nach dem Kommando NEW geben Sie ein:

```
10 INPUT "Werte fuer a und b? ";A,B
20 C=SQR(A*A+B*B)
30 PRINT "c=";C
```

Starten Sie nun mit [RUN], erscheint zunächst

```
Werte fuer a und b?
```

Sie geben nun Ihre Zahlenwerte ein, z. B.

```
10,20 [ENTER]
```

und erhalten anschließend die Lösung:

```
Werte fuer a und b? 10,20
c= 22.3607
```

Natürlich hätten Sie die Werte für a und b auch einzeln abfragen können, z. B. durch

```
10 INPUT "Wert fuer a? ";A
20 INPUT "Wert fuer b? ";B
```

Probieren Sie auch die Fehlermöglichkeiten einmal aus!

Kommentaranweisung

Format:
REM [*kommentar*]
kommentar - beliebiger Text

Funktion:
Kommentare dienen der besseren Lesbarkeit eines BASIC-Programms. Auf die Programmabarbeitung haben sie keinen Einfluß.

Hinweise:
1. Statt des Schlüsselwortes REM kann abkürzend ein Ausrufezeichen (!) geschrieben werden.
2. Tritt in einer BASIC-Zeile eine Kommentaranweisung auf, kann dahinter keine weitere BASIC-Anweisung in derselben Zeile stehen, da der gesamte Text bis zum Zeilenende als Kommentar betrachtet wird.

Beispiel:
Zur Illustration nachfolgend Ausschnitte aus einem BASIC-Programm.

```
10 REM  PROGRAMM XYZ
20 REM STAND: 11.11.1984
30 A1=999  :  A2=1E+06  :! Anfangswerte
```

Akustisches Signal

Format: BEEP

Funktion:
Der Summer ihres Heimcomputers wird veranlaßt, einen kurzzeitigen Ton zu erzeugen.

Hinweis:
Die Anweisung ist nützlich, um die Aufmerksamkeit des Programmnutzers zu lenken, z. B. auf das Auftreten von Fehlern oder den Abschluß länger andauernder Rechnungen.

Beispiel:
Eine Zahleneingabe wird geprüft. Ist die Zahl kleiner als Null, erfolgt eine Fehlermeldung und erneute Eingabe.

```
200 INPUT"ZAHL?";Z
210 IF Z>=0 THEN 250 :! Fallunterscheidung
220 BEEP
230 PRINT "NEGATIVE ZAHL UNZULÄSSIG! "
240 GOTO 200
250 PRINT "Eingegeben wurde: ";Z
```

4.5. Anweisungen zur Steuerung des Programmablaufes

GOTO	unbedingte Verzweigung
ON ... GOTO	berechnete Verzweigung
IF ... THEN ... :ELSE...	bedingte Verzweigung
FOR ... NEXT	Wiederholungsanweisung
PAUSE	Programmunterbrechung
STOP	Programmabbruch
END	Programmende

Mit der Kenntnis der im vorangegangenen Abschnitt behandelten Anweisungen können Sie ohne weiteres bereits eigene kleine BASIC-Programme formulieren. Sie werden dabei jedoch schnell bemerken, daß die starre Abarbeitungsreihenfolge der BASIC-Anweisungen gemäß ihrer Zeilennumerierung oft stark einschränkend ist. Die Lösung praktischer Aufgaben erfordert häufig z. 8. die Unterscheidung verschiedener Fälle, verbunden mit der Abarbeitung unterschiedlicher Programmteile, die mehrfache Ausführung von Anweisungen oder auch die Unterbrechung des Programmlaufs an einer vorgegebenen Stelle.
Sie finden anschließend BASIC-Anweisungen erläutert, mit deren Hilfe Sie solche Probleme auf einfache Weise lösen können.

Unbedingte Verzweigung (Sprung)

Format:
GOTO *zeilennummer*
zeilennummer - BASIC-Zeilennummer, ab der die Ausführung des Programms fortgesetzt werden soll

Funktion:
Die Programmabarbeitung wird in der BASIC-Programmzeile fortgesetzt, deren Zeilennummer angegeben wurde.

Beispiel:
Fortlaufende Berechnung der Quadratwurzeln einzugebender Zahlen.

```
10 INPUT"Eingabezahl ? ";N
20 PRINT „SQR(X)= ";SQR(X)
30 GOTO 10:! Sprung nach Zeile 10
```

Dieses Programm muß mit [STOP] beendet werden.

Berechnete Verzweigung

Format:
ON *ausdruck* **GOTO** *zeilennummer* [,*zeilennummer*) ...
ausdruck - numerischer Ausdruck mit einem Wert größer als oder gleich Null, dessen ganzer Anteil benutzt wird
zeilennummer - BASIC-Zeilennummer, ab der die Ausführung des Programms fortgesetzt werden soll

Funktion:
Der ganzzahlige VVert des Ausdrucks wird bestimmt und sei gleich I. Das Programm wird dann mit der Zeilennummer fortgesetzt, die an I-ter Stelle (von links) in der angegebenen Liste von Zeilennummern steht.

Hinweise:
1. ist der ganzzahlige Wert des Ausdrucks gleich Null oder größer als die Anzahl der angegebenen Zeilennummern, wird das Programm mit der auf die ON ... GOTO-Anweisung folgenden BASIC-Anweisung fortgesetzt.
2. Ein negativer Wert des Ausdrucks führt zu einer Fehlermeldung.

Beispiel:
Im folgenden Demonstrationsbeispiel soll ein "Menü" auf dem Bildschirm erscheinen und der Nutzer zur Wahl einer Programmfunktion aufgefordert werden. Der angegebene Programmausschnitt zeigt eine Möglichkeit zur Organisation der Programmfortsetzung nach der Funktionsauswahl.

```
100 CLS   :! LOESCHEN DES BILDSCHIRMES
110 PRINT "AUSWAHL PROGRAMMFUNKTION"
120 PRINT : PRINT : PRINT
130 PRINT "1 – EINGABE" : PRINT
140 PRINT "2 – BERECHNUNG" : PRINT
150 PRINT "3 – AUSGABE" : PRINT
160 PRINT "4 – PROGRAMMENDE" : PRINT
170 PRINT : PRINT : PRINT
180 !EINGABE KENNZAHL
190 KZ=0 : INPUT "KENNZAHL FUNKTION?";KZ
200 ON KZ GOTO 500,800,1000,1500
210 BEEP : GOTO 190
500 CLS   :! Programmteil EINGABE
510 PRINT "Aufruf EINGABE"
520 PAUSE 30 :! 3 Sekunden Pause
  .
  .
730 GOTO 100
800 CLS   :! Programmteil BERECHNUNG
810 PRINT "Aufruf BERECHNUNG"
820 PAUSE 30 :! 3 Sekunden Pause
  .
  .
980 GOTO 100

1000 CLS   :! Programmteil AUSGABE
1010 PRINT "Aufruf AUSGABE"
1020 PAUSE 30 :! 3 Sekunden Pause
  .
  .
1410 GOTO 100

1500 CLS   :! PROGRAMMENDE
1510 PRINT "PROGRAMMENDE"
  .
  .
1600 END
```

Bedingte Verzweigung

Format 1:
IF *bedingung* THEN *zeilennummer* [:ELSE *zeilennummer*]

Format 2:
IF *bedingung* THEN *anweisung* [:*anweisung*] ...
[:ELSE *anweisung* [:*anweisung*] ...]

bedingung - Vergleichs- oder logischer Ausdruck, Wert gleich Null entspricht "falsch" bzw. "Bedingung nicht erfüllt", Wert ungleich Null entspricht "wahr" bzw. "Bedingung erfüllt"

zeilennummer - BASIC-Zeilennummer, ab der das Programm fortgesetzt werden soll

anweisung - beliebige BASIC-Anweisung mit Ausnahme erneuter IF-Anweisung.

Funktion:
Ist die angegebene Bedingung erfüllt, werden die Anweisungen nach THEN ausgeführt, andernfalls die Anweisungen nach ELSE. Wird nicht aus dem betreffenden Anweisungszweig zu einer anderen BASIC-Zeile verzweigt, erfolgt die Programmfortsetzung mit der auf die IF-Anweisung folgenden BASIC-Zeile. Das trifft auch zu, wenn kein ELSE-Zweig existiert und die angegebene Bedingung nicht erfüllt ist.

Hinweise:
1. Fehlt der ELSE-Zweig, gehören sämtliche hinter THEN stehenden Anweisungen dieser BASIC-Zeile zum THEN-Zweig.
2. Beachten Sie bei der Formulierung von Bedingungen (Vergleichsausdrücken), daß die interne Zahlendarstellung Ihres Heimcomputers für „praktisch gleich große“ Zahlen, die auf verschiedene Weise gewonnen wurden, infolge Rundungsunterschieden geringfügig unterschiedlich ausfallen kann. Ein Test auf Gleichheit würde in solchen Fällen mit dem Ergebnis „falsch“ enden und Ihr Programm möglicherweise an ungewollter Stelle fortgesetzt werden.
3. Erinnern Sie sich, daß Schlüsselwörter des BASIC (siehe Anhang G) nicht in gewählten Bezeichnungen oder Zusammenziehungen solcher mit Schlüsselwörtern (ohne trennendes Leerzeichen) auftreten dürfen! Beispielsweise würde

```
999 IFB<ATHEN200
```

wegen des darin enthaltenen Schlüsselwortes AT zum Syntaxfehler [1]) führen, nicht aber

```
999 IF B<A THEN 200
```

Beispiel:
Probieren Sie das folgende Zahlenratespiel aus!

```
10 ! Spieler A gibt eine ganze Zahl vor
20 INPUT "Zu erratende Zahl? ";A : CLS
30 ! Spieler B darf nun raten
40 INPUT "Gesuchte Zahl ist? ";B
50 IF B=A THEN 80
60 IF B<A THEN PRINT B; "IST ZU KLEIN"
     :ELSE PRINT B; "IST ZU GROSS"

70 GOTO 40
80 PRINT "RICHTIG! ";B; "IST DIE GESUCHTE ZAHL"
90 A$="J":INPUT"NOCH EINMAL: (J)/N";A$
100 IF A$="J" THEN 20
110 IF A$<>N THEN BEEP:GOTO90
120 PRINT"ENDE"
```

Wiederholungsanweisung (Schleife)

Format:
FOR *laufvariable=anfwert* **TO** *endwert* **STEP** *schrittweite*
.
.
.
NEXT [*laufvariable*]

laufvariable - numerische Variable, deren Wert nach jedem Schleifendurchlauf um die Schrittweite verändert wird.

[1]) Ein Fehler, der sich durch einen Verstoß gegen die in den Formaten beschriebenen Regeln ergibt, wird als Syntaxfehler bezeichnet.

anfwert - numerischer Ausdruck, der den Wert der Laufvariablen beim ersten Durchlauf bestimmt.

endwert - numerischer Ausdruck, dessen Wert nach jedem Schleifendurchlauf mit dem veränderten Wert der Laufvariablen verglichen wird.

schrittweite - numerischer Ausdruck, dessen Wert nach jedem Durchlauf zum Wert der Laufvariablen addiert wird (Standardwert: 1)

Funktion:

Die zwischen FOR... und NEXT befindlichen BASIC-Anweisungen werden wiederholt ausgeführt. Die Zahl der Wiederholungen wird dadurch bestimmt, daß der Wert der Laufvariablen beim nächsten Schleifendurchlauf den vorgegebenen Endwert nicht überschreiten (bei positiver Schrittweite) bzw. nicht unterschreiten (bei negativer Schrittweite) darf. Jede Schleife wird mindestens einmal durchlaufen. Nach dem letzten Schleifendurchlauf wird das Programm mit der auf NEXT folgenden BASIC-Anweisung fortgesetzt (siehe Beispiel).

Hinweise:

1. Innerhalb der Anweisungsfolge einer Wiederholungsanweisung ist das Auftreten weiterer FOR ... NEXT-Anweisungen zulässig. Man spricht dann von „geschachtelten Schleifen“. Dabei ist zu beachten, daß jede durch FOR und NEXT begrenzte „innere“ Schleife vollständig innerhalb der „äußeren“, d. h. zeitlich zuvor begonnenen, Schleife liegt und daß in jedem Fall unterschiedliche Laufvariable verwendet werden.
2. Enden mehrere geschachtelte Schleifen an derselben Programmstelle, kann ein gemeinsamer Schleifenabschluß mit NEXT, gefolgt von den Laufvariablen in der richtigen Reihenfolge, geschrieben werden.
3. Wird auf die Angabe einer Laufvariablen hinter NEXT verzichtet, bezieht sich NEXT stets auf die zuletzt eröffnete Schleife. Wiederholungsanweisung (Schleife)
4. Der Eintritt in eine Wiederholungsanweisung darf nur über FOR... erfolgen. Ein Hineinspringen (z. B. mittels GOTO) in die Anweisungsfolge innerhalb einer Schleife führt bei Erreichen von NEXT zum Fehler.
5. Eine Wiederholungsanweisung kann vorzeitig verlassen werden (z. B. mittels einer IF- oder GOTO-Anweisung). Der Heimcomputer gibt aber dann den für die Schleifenorganisation benötigten Speicherplatz (16 Bytes) nicht wieder frei, so daß es bei häufigem vorzeitigen Verlassen von Schleifen zu Speicherüberlauf kommen kann. Besser ist es in solchen Fällen, innerhalb der Schleife den Wert der Laufvariablen auf den Endwert zu setzen und die Schleife über NEXT zu verlassen.

Beispiele:

Berechnen Sie doch zunächst einmal, wie Ihr Sparguthaben wächst, wenn Sie es eine bestimmte (ganze) Zahl von Jahren bei einem Zinssatz von 3,25 % unangetastet lassen:

```
10 INPUT "GUTHABEN (MARK)? ";G
20 INPUT "WIEVIELE JAHRE KEINE ABHEBUNG?";AJ
30 IF AJ<1 THEN PRINT "BEDAURE!":GOTO 90
40 FOR J=1 TO AJ
50 G=G+G*0,0325
60 NEXT J
70 G=INT(G*100+0.5)/100: Rundung auf Pfennige
80 PRINT:PRINT"DANACH WAEREN ES";GMARK! ":PRINT
90 PRINT "NOCH EINMAL? RUN! "
```

Wenn Sie sich z. B. für den Guthabenzuwachs nach Monaten oder sogar Tagen (oder auch nur für BASIC) interessieren, ändern Sie das Programm nach Ihren Vorstellungen ab!

Hier noch ein Beispiel für „geschachtelte Schleifen“:

```
10 FOR I=3 TO 1 STEP -1
20 FOR J=1 TO 4 STEP2
30 PRINT I,J
40 NEXT J
50 NEXT I
```

Starten Sie dieses Programm mit RUN, erhalten Sie

```
3             1
3             3
2             1
2             3
1             1
1             3
```

Für die Zeilen 40 und 50 hätten Sie auch schreiben können:

```
40 NEXT J,I
```

Programmunterbrechung

Format:
PAUSE [*dauer*]

dauer - numerischer Ausdruck, dessen Wert (positiv und ganzzahlig) die Dauer der Programmunterbrechung in Zehntelsekunden bestimmt.

Funktion:
Das Programm wird, ohne den Programmodus zu verlassen, in seiner Ausführung unterbrochen. Nach Betätigung der Taste [CONT] oder nach Ablauf einer angegebenen Unterbrechungsdauer wird es mit der nachfolgenden BASIC-Anweisung fortgesetzt.

Hinweis:
Während der Programmunterbrechung können Sie, da Sie sich nicht im Kommandomodus befinden, keine Anweisungen oder Kommandos abarbeiten lassen.

Beispiel:
Ein Text soll wiederholt (zeilenweise) ausgegeben werden. Die Zeit zwischen zwei Ausgaben wird mit einer PAUSE-Anweisung festgelegt.

```
10 FOR I=1 TO 10
20 PRINT "*** robotron Z 9001 ***"
30 PAUSE 30
40 NEXT I
```

Bitte variieren Sie durch Neueingabe der Zeile 30 die Ausgabegeschwindigkeit. Probieren Sie auch PAUSE ohne Angabe der Unterbrechungsdauer und PAUSE I*2 aus.

Programmabbruch

Format: **STOP**

Funktion:
Das laufende BASIC-Programm wird bei Erreichen der STOP-Anweisung abgebrochen, die aktuelle BASIC-Zeilennummer angezeigt, und der BASIC-Interpreter kehrt in den Kommandomodus zurück.

Hinweise:
1. Die Abbruchmitteilung des BASIC-Interpreters lautet

   ```
   BREAK IN zeilennummer
   OK
   >■
   ```

   wobei *zeilennummer* die BASIC-Zeile der den Abbruch aulösenden STOP-Anweisung kennzeichnet.
2. Nach STOP bleiben alle aktuellen Variablenwerte und Systemzustände erhalten
3. Im Gegensatz zur PAUSE-Anweisung können Sie nach STOP-Anweisungen Kommandos ausführen lassen. lnsbesondere können Sie das Programm mittels RUN neu starten, wobei aktuellen Variablenwerte wieder zurückgesetzt werden. Wollen Sie das Programm mit den aktuellen Variablenwerten fortsetzen, so können Sie das mit der Anweisung

   GOTO *zeilennummer*.

   Dabei ist die zur Programmfortsetzung vorgesehene BASIC-Zeilennummer anzugeben. Gegebenenfalls können Sie Variablen vor Ausführung der GOTO-Anweisung durch sofort ausführbare Anweisungen modifizieren.
4. Mit dem Kommando CLEAR (siehe Abschnitt 4.17) können Sie alle Variablen vor der Programmfortsetzung löschen. Bei Benutzung der Kommandos AUTO, EDIT, RENUMBER und DELETE gehen die aktuellen Variablenwerte ebenfalls verloren.
5. Nach STOP ist auch eine Programmfortsetzung mit der Taste [CONT] möglich, falls Sie zwischenzeitlich keine anderen Kommandofunktionen (AUTO, EDIT, LIST, ...) benutzt haben.
6. STOP-Anweisungen können an beliebigen Stellen im Programm stehen. Sie lassen sich u. a. zum Programmtest vorteilhaft einsetzen.

Beispiel:
Fügen Sie in das vorhergehende Beispiel zur PAUSE-Anweisung ein:

```
15 IF I>5 THEN STOP
```

und überzeugen Sie sich Mittels [RUN] von der Wirkung Kontrollieren Sie nach Abbruch des Programms den aktuellen Wert der Laufvariablen mittels

```
? I
```

und modifizieren Sie z. B.

```
I=9
```

Danach setzen Sie die Ausgabe durch [CONT] oder

```
GOTO 20
```

fort.

Programmende

Format **END**

Funktion:
Das laufende BASIC-Programm wird beendet. Der BASIC-Interpreter kehrt ohne eine Mitteilung in den Kommandomodus zurück.

Hinweis:
Die END-Anweisung ist nicht zwingend erforderlich. Ein Programm wird im Normalfall nach der letzten Anweisung beendet. Trotzdem ist diese Anweisung empfehlenswert, da nach dem logischen Programmende weitere BASIC-Anweisungen im Arbeitsspeicher stehen können (z. B. Unterprogramme, andere Programme). Fehlt die END-Anweisung, würden diese anschließend ausgeführt.

Beispiel:
Sie sollten künftig Programme stets mit END abschließen, Fügen Sie an das letzte Beispiel an

```
50 END
60 PRINT "NAECHSTER PROGRAMMBEGINN"
```


und starten Sie erneut. Wollen Sie die Anweisung in Zelle 60 erreichen, erinnern Sie sich an das Kommando RUN 60.

4.6. Steuerung des Programmablaufes durch den Nutzer

[RUN]	Programmstart
[STOP]	Programmabbruch
[PAUSE]	Programmunterbrechung
[CONT]	Programmfortsetzung

Haben Sie schon diese sehr wichtige Eigenschaft Ihres Heimcomputers bemerkt? Er läßt sich von Ihnen selbst dann noch steuern, wenn er gerade Ihr BASIC-Programm abarbeitet!
Während die im vorherigen Abschnitt angegebenen Anweisungen zur Steuerung des Programmablaufes unter denselben Bedingungen natürlich stets an denselben Programmstellen wirken, haben Sie es mit den oben angegebenen BASIC-Funktionstasten selbst in der Hand, den Zeitpunkt Ihres Eingreifens zu bestimmen.
Es gibt vielfältige Gründe, die Sie zu einem solchen Eingriff veranlassen können, z. B. wenn Sie

- ein nicht mehr benötigtes oder fehlerhaft ablaufendes BASIC-Programm vorzeitig abbrechen wollen,
- Zwischenergebnisse einer Berechnung ansehen möchten, für die programmseitig keine Ausgabe vorgesehen war,
- Variablenwerte verändern möchten,
- ein bestimmtes Bildschirmbild länger als im Programm vorgesehen betrachten wollen.

Im zuletzt genannten Fall empfiehlt sich die Anwendung von [PAUSE]/[CONT], in den anderen Fällen [STOP] mit entsprechender Fortsetzung. Die Wirkung der Tasten [PAUSE] und [STOP] entspricht der gleichnamigen Anweisungen (siehe Abschnitt 4.5). Bezüglich der Fortsetzungsmöglichkeiten sind Sie an dieser Stelle ebenfalls informiert worden.

Beachten Sie bitte noch, daß

- die Taste [STOP] auch zur Beendigung von Kommandofunktionen des BASIC-Interpreters benutzt wird (AUTO, LIST, EDIT), die [PAUSE]-Taste in diesen Fällen jedoch unwirksam ist,

- nach [STOP] (oder der STOP-Anweisung) die zuletzt, z. B. im abgebrochenen Programm, erzeugten Systemzustände (einschl. der durch WINDOW, CLEAR, LINES und NULL festgelegten) erhalten bleiben.

Über Abbruch und Unterbrechung hinaus gibt es weitere Möglichkeiten, auf den Ablauf Ihres Programms Einfluß zu nehmen. Diese müssen Sie dann allerdings programmseitig vorbereiten. Naheliegend ist die mit INPUT zu realisierende Aufforderung zur Eingabe von Steuerdaten, die im Programm ausgewertet werden und seinen weiteren Ablauf bestimmen. Dieser Möglichkeit bedient man sich z. B. bei der sogenannten „Menü-Technik“, die Sie schon kennengelernt haben (Beispiel zu ON ... GOTO im Abschnitt 4.5).
Steuernden Einfluß auf Ihr Programm können Sie auch mit Hilfe der im Abschnitt 4.14 beschriebenen speziellen Eingabefunktionen ausüben. Beispielsweise gestattet Ihnen die Funktion INKEY$, Funktionstasten zu vereinbaren, deren Betätigung den weiteren Programmablauf bestimmt. Der wesentliche Unterschied zur INPUT-Anweisung besteht darin, daß das Programm auch ohne ihr Eingreifen „weiterläuft“. Der Zeitpunkt, zu dem steuernd eingegriffen wird, kann somit vom Nutzer bestimmt werden.

4.7. Programmänderung

EDIT	Änderung von Programmzeilen
DELETE	Streichen von Programmzeilen
RENUMBER	Neunumerieren von Programmzeilen

Jetzt sind Sie sicher bereits in der Lage, eine Reihe von Aufgaben mit einem selbst geschriebenen kleinen BASIC-Programm zu lösen. Sie werden dabei festgestellt haben, daß die Programmänderung durch Neueingabe kompletter BASIC-Zellen manchmal recht umständlich ist. Mit den Kommandos EDIT und DELETE stehen Ihnen komfortablere Korrekturmöglichkeiten zur Verfügung.
Nach umfangreichen Programmänderungen ist oftmals eine neue, durchgängige Zeilennumerierung nützlich. Durch das Kommando RENUMBER können Sie auch das erreichen.

Ändern von Programmzeilen

Format:
EDIT *zeilennummer*
zeilennummer - Nummer der BASIC-Zeile, die als erste zur Änderung bereitgestellt werden soll.

Funktion:
Beginnend mit der angegebenen Zeilennummer, werden in aufsteigender Numerierungsfolge BASIC-Zeilen einzeln angezeigt und zur Änderung bereitgestellt. Sie können die betreffende Zeile direkt ändern und sich dazu auch der Tasten [←[, [→], [I←], [→I], [CL LN], [DEL], [INS] (siehe Abschnitt 3.2) bedienen. Die BASIC-Zeilennummer läßt sich nicht ändern.
Jede einzelne BASIC-Zeile wird mit der Taste [ENTER] oder der [↓]-Taste beendet, anschließend wird, falls vorhanden, diejenige mit der nächsthöheren Zeilennummer bereitgestellt. Wurde die Zeile Mit [ENTER] abgeschlossen, wird die entsprechende Zeile im Arbeitsspeicher durch die geänderte ersetzt. Im Sonderfall der Leereingabe (*zeilennummer* [ENTER]) erfolgt die vollständige Streichung der betreffenden Zelle. Wird die Zeile mit der [↓]-Taste abgeschlossen, bleibt die ursprüngliche Zeile im Arbeitsspeicher unverändert erhalten, unabhängig davon, ob ihr angezeigtes Abbild geändert wurde oder nicht.

Hinweise:
1. Nutzen Sie beim Ausführen von Änderungen in einer Zeile die im Abschnitt 3.2. beschriebenen Tasten zur Kursorbewegung sowie zum Streichen bzw. Einfügen von Zeichen.
2. Der EDIT-Modus kann vor Erreichen der letzten Programmzeile mit [STOP] beendet werden. Dabei bleibt die zuletzt angezeigte BASIC-Zeile unverändert.
3. Tritt ein Stern (*) hinter der Zeilennummer der bereitgestellten Zeile auf, bedeutet das, daß die Zeile die Länge von 72 Zeichen überschreitet. Die Ursache liegt dann darin, daß Sie bei der Eingabe dieser Zeile abkürzend ? statt PRINT geschrieben haben. Der BASIC-Interpreter bietet Ihnen im EDIT-Modus (ebenso wie bei LIST) jedoch die „rückübersetzte“ Form der Zeile an, d. h. PRINT (5 Zeichen) statt ? (1 Zeichen). Soll die Zeile unverändert erhalten bleiben, dürfen Sie sie

jetzt nicht mit [ENTER] abschließen (da Sie in diesem Fall durch die angezeigten 72 Zeichen ersetzt wird) sondern mit [↓] bzw. [STOP].
4. Falls Sie eine nicht existierende Zeilennummer im EDIT-Kommando angeben, erhalten Sie eine Fehlermitteilung.

Beispiel:
Wiederholen Sie die eine oder andere Programmänderung in Verbindung mit den zuvor gezeigten Beispielen. Nutzen Sie dazu jetzt das EDIT-Kommando! Wollen Sie zum Beispiel mit der Zeile 40 beginnend ändern, geben Sie ein:

```
EDIT 40
```

Streichen von Programmzeilen

Format:
DELETE *zeilennummer1* [,*zeilennummer2*]
zeilennummer1,2 - kennzeichnet niedrigste bzw. höchste zu streichende BASIC-Zeile

Funktion:
Der Programmabschnitt, der durch die angegebenen BASIC-Zeilennummern begrenzt ist, wird gestrichen.

Hinweise:
1. Soll nur eine Zeile mit DELETE gestrichen werden, darf nur *zeilennummer1* angegeben werden.
2. Die mit *zeilennummerl,2* spezifizierten BASIC-Zeilen müssen im Arbeitsspeicher vorhanden sein, andernfalls führt die Ausführung des Kommandos zu einer Fehlermitteilung.

Beispiel:
In einem BASIC-Programm, das sich im Arbeitsspeicher befindet und in Zehnerschritten von 10 bis 5660 numeriert ist, sollen die Zeilen 2840 bis 3620 (einschließlich!) gestrichen werden. Das Kommando lautet

```
DELETE 2840,3620
```


Neunumerieren von Programmzeilen

Format:
RENUMBER [*zlnralt1* [,*zlnralt2* [,*zlnrneu1* [,*schrittweite*]]]]
zlnralt1,2 -kennzeichnet niedrigste bzw. höchste alte Zeilennummer des neu zu numerierenden Programmabschnittes (Standardwerte: *zlnralt1*: niedrigste vorhandene Zeilennummer *zlnralt2*: höchste vorhandene Zeilennummer)
zlnrneu1 - kennzeichnet niedrigste Zeilennummer des neu numerierten Programmabschnittes (Standardwert: *zlnralt1*)
schrittweite - Differenz zweier aufeinanderfolgender Zeilennummern (Standardwert: 10)

Funktion:
Das im Arbeitsspeicher befindliche BASIC-Programm, bzw. ein angegebener Abschnitt daraus, wird gemäß Vorgabe oder standardmäßig neu numeriert. Dabei werden auch alle Bezugnahmen auf Zeilennummern (GOTO .... IF ... THEN ... usw.) entsprechend verändert.

Hinweise:
1. Mit dem RENUMBER-Kommando ist es nicht möglich, die Reihenfolge der Programmzeilen zu verändern! Deshalb müssen Sie streng darauf achten, daß durch abschnittsweises Neunumerieren die aufsteigende Zeilennumerierung nicht verletzt wird und keine doppelten Zeilennummern entstehen. Andernfalls kommt der BASIC-Interpreter in Schwierigkeiten, die möglicherweise nur durch [RESET] behoben werden können.
2. Lassen Sie im RENUMBER-Kommando Parameter weg, weil Sie die Standardwerte nutzen wollen, so ist dies nur von rechts nach links möglich.

Beispiel:
Ein von 5 bis 250 in 5er-Schritten numeriertes Programm soll in 10er-Schritten numeriert werden. Es soll außerdem mit der Zeilennummer 10 beginnen. Das RENUMBER-Kommando lautet:

```
RENUMBER 5,250,10,10
```

Der letzte Parameter (10) hätte auch entfallen können (Standardwert). Nach Ausführung des Kommandos hat das Programm die Numerierung von 10 bis 500 in 10er-Schritten. Überzeugen Sie sich an einem

selbstgewählten Beispiel, indem Sie das neunumerierte Programm anschließend mit [LIST] anzeigen!

4.8. Felder

DIM Feldvereinbarung

Unter einem Feld versteht man die Zusammenfassung mehrerer Variablen desselben Typs (numerisch oder Zeichenkette) unter einer gemeinsamen Bezeichnung, dem Feldnamen.
Der Zugriff auf die einzelnen Variablen, die Feldelemente, erfolgt durch Angabe eines Index (oder mehrerer Indizes) nach dem Feldnamen. Wird nur ein Index verwendet, spricht man von einem eindimensionalen Feld, d. h., die Feldelemente oder indizierten Variablen sind in einer Reihe angeordnet. Bei zwei Indizes ist es üblich, von Zeilen und Spalten zu sprechen, in denen die Feldelemente stehen.
Ihr BASIC-Interpreter kann Felder verarbeiten, die theoretisch bis zu 255 Dimensionen haben könnten. Für die meisten praktischen Probleme sind jedoch Felder mit bis zu 3 Dimensionen ausreichend. Außerdem begrenzen die maximale BASIC-Zeilenlänge und der verfügbare Arbeitsspeicher die Zahl der verwendbaren Dimensionen auf einen wesentlich, niedrigeren Wert. Damit Sie Ihre Felder „nach Maß“ vereinbaren können, steht Ihnen die folgende Anweisung zur Verfügung:

Format:
DIM *feldname* (*index* [,*index*] ...) [,*feldname*(*index*[,*index*] ...)] ...

feldname - nach den für Variable gültigen Regeln gewählte Bezeichnung (siehe Abschnitt 4.2)
index - numerischer Ausdruck, der die höchste Ordnungsnummer der Feldelemente einer Dimension festlegt und dessen ganzzahliger Wert zwischen 0 und 32766 liegen muß

Funktion:
Es wird Speicherplatz für die in der DIM-Anweisung festgelegten Feldelemente reserviert. Gleichzeitig bekommen alle Elemente eines numerischen Feldes den Wert 0 und die eines Zeichenkettenfeldes die leere Zeichenkette ("") zugewiesen.

Hinweise:
1. Die Zählung der Feldelemente einer Dimension beginnt stets mit 0. Es werden also je Dimension *index*+1 Feldelemente vereinbart.
2. Standardmäßig gelten 11 Feldelemente (eindimensional) als vereinbart, wenn ein Feldname im BASIC-Programm benutzt wird, ohne daß dieses Feld mit einer DIM-Anweisung vereinbart wurde.
3. Eine DIM-Anweisung muß in der Reihenfolge der Programmabarbeitung vor der ersten Benutzung des Feldnamens in einer BASIC-Anweisung stehen. Andernfalls erfolgt die Dimensionierung gemäß 2, und bei Erreichen der DIM-Anweisung für das Feld wird es neu vereinbart. Dabei werden alle Feldelemente gelöscht.
4. Bei Zeichenkettenfeldern ist darauf zu achten, daß der Speicherbereich für Zeichenketten zur Zeit des Programmlaufs ausreichend groß ist (siehe Abschnitt 4.17/CLEAR).

Beispiel:
Zunächst vereinbaren Sie ein eindimensionales Feld, weisen einigen Feldelementen Werte zu und geben aus:

```
10 DIM W(5)
20 FOR I=! TO 3 : W(I)=10+I : NEXT I
30 FOR I=0 TO 5 : PRINT "W(";I; ")=";W(I)
40 NEXT I
```

Nach dem Programmstart erhalten Sie auf dem Bildschirm:

```
W( 0 )= 10
W( 1 )= 11
W( 2 )= 12
W( 3 )= 13
W( 4 )= 0
W( 5 )= 0
```

Hätten Sie die Zeile 10 weggelassen (oder vergessen), hätte alles genauso funktioniert, da das Feld weniger als 11 Elemente besitzt. Überzeugen Sie sich, indem Sie Zeile 10 streichen und erneut starten! Nachteilig ist in diesem Fall, daß Sie zusätzlichen Speicherplatz für die nichtbenötigten Feldelemente W(6) bis W(l0) reservieren.

Benutzen Sie jetzt Felder, deren Größe Sie erst zum Zeitpunkt der Programmabarbeitung festlegen:

```
10 INPUT "WIEVIELE PERSONEN?"; N
20 IF N<1 THEN BEEP: GOTO 10
30 DIM NAME$(N-1,1), ALTER(N-1)
40 FOR I=0 TO N-1
50 INPUT "NAME, VORNAME, ALTER?";
   NAME$(I,0),NAME$(I,1),ALTER(I)
60 NEXT I
70 PRINT "AELTER ALS 17 JAHRE SIND:":
   PRINT: KP=0
80 FOR I=0 TO N-1
90 IF ALTER(I)>17 ZHEN PRINT NAME$(I,1);
   " ";NAME$(I,0): KP=1
100 NEXT I
110 IF KP=0 THEN PRINT "KEINE PERSONEN"
120 PRINT: END
```

Bevor Sie [RUN] drücken, überlegen Sie, ob die Gesamtzahl der einzugebenden Zeichen für das Feld NAME$ größer als 256 Zeichen werden wird. Wenn ja, vergrößern Sie den Speicherbereich für Zeichenketten, mit CLEAR 1000 z. B. auf maximal 1000 Zeichen (siehe Abschnitt 4.17). Haben Sie nun das Programm gestartet und die Daten der von Ihnen festgelegten Anzahl von Personen eingegeben, erhalten Sie anschließend die Liste der Personen über 17 Jahre. Wie wäre es, wenn Sie das Mini-„Recherche-Programm“ noch etwas ausbauen? Zum Beispiel könnten sie weitere Daten der Personen erfassen und auch das „Recherchekriterium“ wählen lassen.

4.9. Interne Daten

DATA	Vereinbarung von Daten
READ	Lesen von Daten
RESTORE	Setzen des Datenzeigers

Wie Sie wissen, kann ihr „robotron Z 9001“ beachtliche Datenmengen speichern. Ebenso wie Ihr BASIC-Programm wollen Sie die „Problemdaten“ jedoch oft nicht bei jeder Benutzung erneut eingeben. Mit den im folgenden erläuterten Anweisungen gelingt es ihnen ohne weiteres, auch größere Datenmengen direkt in das Programm aufzunehmen und diese gemeinsam mit dem Programm auf Magnetbandkassette zu speichern.

Format:
DATA *konstante* [,*konstante*] ...
konstante - numerische oder Zeichenkettenkonstante

Funktion:
Die angegebenen Daten werden gespeichert. Der Zugriff auf diese Daten erfolgt ausschließlich mittels der READ-Anweisung.

Hinweise:
1. Die DATA-Anweisung ist nur im Programmodus erlaubt, d. h. nicht als sofort ausführbare Anweisung im Kommandomodus.
2. Da alle DATA-Anweisungen bei Programmstart (RUN) ausgewertet werden, ist ihre Stellung (Anordnung) im Programm beliebig. Sie können also auch nach den entsprechenden READ-Anweisungen auftreten. Während des Programmlaufs werden DATA-Anweisungen übergangen.
3. Anführungszeichen ("...") zur Begrenzung von Zeichenketten-konstanten können entfallen, wenn keine Schlüsselwörter, Kommas bzw. führende oder nachfolgende Leerzeichen in ihnen enthalten sind.

Format:
READ *variable* [, *variable*] ...
variable - Name einer numerischen oder Zeichenkettenvariablen, die den Wert einer Konstanten aus einer DATA-Anweisung erhält

Funktion:
Den Variablen der READ-Anweisung(en) werden Konstanten aus DATA-Anweisungen als aktueller Wert zugewiesen. Die Zuweisung erfolgt fortlaufend in der Reihenfolge des Auftretens der READ-Anweisungen und der Variablennamen in ihnen. Dabei werden die Konstanten der DATA-Anweisungen in ihrer Folge gemäß aufsteigender Zeilennumerierung zugewiesen.

Hinweis:
Der Typ der Variablen (numerisch oder Zeichenkette) muß mit dem Typ der ihr zugewiesenen Konstanten aus der DATA-Anweisung übereinstimmen.

Beispiel:

```
10 DATA 1065,75, "MEYER, FRITZ"
20 DATA 960,35, "ENDER, SYLVIA"
30 READ M,N$
40 PRINT: PRINT N$;" verdient";M; "MARK": PRINT
```

Bei Abarbeitung dieses Programms erhalten Sie zunächst auf dem Bildschirm

```
MEYER, FRITZ verdient 1065,75 MARK
```

Geben Sie nun GOTO 30 ein und überprüfen Sie das Ergebnis!

Format:
RESTORE [*zeilennummer*]
zeilennummer - BASIC-Zeilennummer einer DATA-Anweisung (Standard: niedrigste Zeilennummer einer DATA-Anweisung)

Funktion:
Die Wertzuweisung für die nächsten READ-Anweisungen erfolgt, beginnend mit dem ersten Datenelement der DATA-Anweisung, in der angegebenen BASIC-Zeile bzw. der ersten DATA-Anweisung im Programm.

Beispiel:
Vielleicht haben Sie beim vorigen Beispiel ein zweites Mal versucht, das Programm mit GOTO 30 fortzusetzen und einen OD-Fehler erhalten, da keine weiteren Daten verfügbar waren.

In diesem Falle hätte

```
RESTORE 10
```

oder einfach

```
RESTORE
```

genügt, um mit

```
GOTO 20
```

erneut beginnen zu können.

Was passiert, wenn sie

```
RESTORE 20
```

eingeben, bevor Sie mit

```
GOTO 30
```

oder [RUN] fortsetzen?
Wenden Sie sich nun noch einmal dem Beispiel aus Abschnitt 4.8 zu. Angenommen, Sie wollen die „Personaldaten“ gemeinsam mit Ihrem Programm speichern und nicht vor jeder „Re- cherche“ neu eingeben. Dann ersetzen Sie, wie im folgenden Programm gezeigt, die INPUT-Dateneingabe durch DATA-Anweisungen, deren Daten Sie mittels READ einlesen.

```
10 DATA 8 :!ANZAHL PERSONEN
20 READ N :! EINLESEN PERSONENANZAHL
30 DIM NAME$(N-1,1),ALTER(N-1)
40 ! EINLESEN DATEN
50 FOR I=0 TO N-1
60 READ NAME$(I,0),NAME$(I,1),ALTER(I)
70 NEXT I
80 ! AUSWERTUNG DATEN
90 INPUT "ALTERSGRENZE?";AG
100 IF AG<1 THEN BEEP: GOTO 90
110  PRINT  "AELTER  ALS";AG;  "JAHRE  SIND:
";PRINT:
    KP=0
120 FOR I=0 TO N-1
130 IF ALTER(I)>AG THEN PRINT NAME$(I,1);" ";
    NAME$(I,0):KP=1
140 NEXT I
150 IF KP=0 THEN PRINT "KEINE PERSONEN"
```

```
200 !DATEN
210 DATA HINZ,MARIO,16,KUNZ,INES,13
220 DATA FENDER,TOM,23,COHN,SARAH,19
230 DATA TILLE,HANS-JUERGEN,34,SPETH,ROBERT,20
240 DATA SCHNEIDER,OLGA,67,SACHSE,CAESAR,12
300 END
```

Mit Hilfe des EDIT-Kommandos können Sie Ihre Daten „pflegen“. Der Umfang läßt sich erweitern, indem Sie weitere DATA-Anweisungen hinzufügen.

4.10. Zufallszahlen

RND Erzeugen einer Zufallszahl

Zufällig erzeugte Zahlen, die der Nutzer eines Programms nicht vorhersagen kann oder will, haben große praktische Bedeutung unter anderem für die

- Programmierung von Spielen (z. 8. wenn sie elektronisch „würfeln“ wollen),
- Zufällige Auswahl von Fragen in Lehr- und Lernprogrammen (der Computer zieht eine Frage),
- Erzeugung von Testdaten (oder würden Sie lieber 300 Zahlen von Hand eintippen, um ein Sortierprogramm auf Richtigkeit und Geschwindigkeit zu testen?).

In diesen und vielen anderen Fällen verwenden Sie die BASIC-Funktion RND.

Format:

RND (*ausdruck*)

ausdruck - numerischer Ausdruck, dessen Wert die zu erzeugende Zufallszahl beeinflußt.

Typ: BASIC-Funktion[1])

Funktion:

Es wird eine Zufallszahl zwischen 0 und 1 erzeugt. Abhängig vom Wert des als Argument angegebenen Ausdrucks wird folgendes ausgeführt:

Argument (X = Ausdruck)	Funktionswert: 0<RND(X)
X > 0	nächste Zahl einer Folge von Zufallszahlen, die ihrerseits vom Wert des Arguments abhängt
X > 0	nächste Zahl einer Folge von Zufallszahlen, die ihrerseits vom Wert des Arguments abhängt .
X=0	identisch mit Funktionswert beim vorangegangenen Funktionsaufruf, d. h. Wiederholung der letzten Zufallszahl
X < 0	wie im Fall X > 0, jedoch wird danach eine neue Folge von Zufallszahlen begonnen.

Hinweise:

1. Die praktisch auftretenden Funktionswerte (Zufallszahlen) liegen etwa zwischen 10^{-6} und $1 - 10^{-6}$.
2. Auch für unverändertes Programm und ein- und denselben Vorgang bei Programmstart, z. B. nach Einschalten des Rechners, kann ein jeweils unterschiedlicher Beginn der Zufallszahlenfolge erreicht werden. Der Absolutwert des RND-Funktionsarguments muß dann seinerseits zufällig sein. Das läßt sich z. B. durch Abfrage des aktuellen Sekundenstandes der eingebauten Systemuhr (siehe Abschnitt 5.1) erreichen.

Beispiel:

Programmieren Sie sich einen elektronischen „Würfel“, der jeweils nach [ENTER]-Betätigung eine Zahl zwischen 1 und 6 anzeigt!

```
10 PRINT "Z9001 WUERFELT FUER SIE! ENTER! "
20 PRINT : N=0
30 INPUT"";X$
40 N=N+1 : Z=1+INT(6*RND(1))
50 PRINT,N; "-TER WURF: ";Z
60 GOTO 30
```

[1]) Eine BASIC-Funktion ist keine selbständige Anweisung. Sie wird, wie bereits für die mathematischen Standardfunktionen erläutert, benutzt, d. h. sie besitzt stets einen aktuellen Funktionswert und kann in Ausdrücken bzw. wie ein solcher verwendet werden.

Dieses Programm können Sie nur durch [STOP] beenden. Beachten Sie weiterhin, daß die BASIC-Funktion INT den ganzen Anteil ihres Argumentwertes liefert. Zur Übung ersetzen Sie vielleicht noch die Vorgabe der zu ratenden Zahl im Beispiel des Abschnitts 4.2 (IF-Anweisung) durch eine im Programm erzeugte Zufallszahl!

4.11. Nutzerfunktionen

DEF Fn*name*	Definition einer Nutzerfunktion
Fn*name*	Aufruf einer Nutzerfunktion

Die Benutzung von Funktionsaufrufen in Ausdrücken (Formeln) ist nicht auf die fest im BASIC enthaltenen Funktionen beschränkt. Genauso wie Sie numerische Standardfunktionen einsetzen können (siehe Abschnitt 4.2), können Sie auch Funktionen benutzen, für die Sie zuvor selbst festlegen, wie der Funktionswert bestimmt wird.

Format:
DEF FN*name* [(*parameter*)] = *ausdruck*

name - zweiter Teil des Funktionsnamens, gebildet wie der Name einer numerischen Variablen

parameter - Name einer numerischen Variablen, die im rechtsstehenden Ausdruck verwendet werden kann

ausdruck - numerischer Ausdruck, der die Berechnungsvorschrift für den Funktionswert bildet und unter anderem numerische Variable des Programms sowie weitere Funktionsaufrufe enthalten kann.

Funktion:
Eine vom Nutzer vorzugebende numerische Funktion wird vereinbart. Wahlweise kann ein formaler Parameter zur Bestimmung des Funktionswertes benutzt werden, der zum Zeitpunkt des Funktionsaufrufes durch den aktuellen Argumentwert ersetzt wird.

Hinweise:
1. Die Anweisung DEF FN ist nur im Programmodus erlaubt, d. h. nicht als sofort ausführbare Anweisung im Kommandomodus.
2. Die Länge der Anweisung darf insgesamt eine BASIC-Zeile nicht überschreiten.
3. Die Anweisung DEF FN muß vor der ersten Verwendung der Nutzerfunktion durchlaufen werden. Der Aufruf zur Berechnung einer Nutzerfunktion erfolgt durch Angabe ihres Namens einschließlich eines eventuell vereinbarten Arguments (aktueller Parameter).

Beispiel:
Mit Hilfe der BASIC-Standardfunktionen ATN(X) und SQR(X) soll die Berechnung von

$x \arcsin(x) = \arctan(x/\sqrt{1-x^2})$

über eine Nutzerfunktion erfolgen.

```
10 DEF FNAS(X)=ATN(X/SQR(1-X*X))
   :
   :
   :
100 A=.707
110 B=FNAS(A)
120 PRINT "arc sin(";A; ")=";B
```

Nach [RUN] erscheint auf dem Bildschirm

```
arc sin( .707 )= .785247
```

Besonders für häufig im Programm zu berechnende Ausdrücke bringt die Verwendung einer Nutzerfunktion erhebliche Vorteile.

4.12. Unterprogramme

GOSUB	Aufruf eines Unterprogramms
RETURN	Rückkehr aus Unterprogrammen
ON ... GOSUB	Berechneter Aufruf von Unterprogrammen

Der Einsatz von Unterprogrammen (auch „Subroutinen“ genannt) dient vorrangig einer effektiven Programmgestaltung:

- klare, übersichtliche Programmstruktur und damit verbunden gute Testbarkeit und Änderungsfreundlichkeit,
- kürzerer, speicherplatzsparender Programmtext durch mehrfache Verwendung nur einmal formulierter Anweisungsblöcke.

Außerdem gelingt damit auf einfache Weise die Wiederverwendung vorhandener Programmlösungen beim Erarbeiten neuer Programme, einschließlich des Einsatzes von käuflich erhältlichen Unterprogrammen für Standardaufgaben.

Ein Unterprogramm ist für sich allein nicht abarbeitungsfähig. Es bedarf eines übergeordneten Programms, von dem aus es aufgerufen wird. Aufruf eines Unterprogramms

Format:
GOSUB *zeilennummer*
zeilennummer - BASIC-Zeilennummer, ab der die Programmabarbeitung fortgesetzt wird

Funktion:
Die Programmabarbeitung wird mit der angegebenem BASIC-Zelle fortgesetzt. Nach Auftreten einer RETURN-Anweisung (siehe unten) erfolgt die Rückkehr und Programmfortsetzung mit der Anweisung, die auf die GOSUB-Anweisung folgt.

Hinweise:
1. Eine Parameterübermittlung (etwa wie bei Nutzerfunktionen) ist nicht möglich. Alle im Programm verwendeten Variablen sind uneingeschränkt gültig. Es ist darauf zu achten, daß keine Fehler durch ungewollte Mehrfachnutzung von Variablennamen entstehen!
2. Das mit der GOSUB-Anweisung aufgerufene Unterprogramm wird wegen 1. auch nicht gesondert vereinbart. Es empfiehlt sich, eine kennzeichnende Kommentaranweisung voranzustellen.
3. Endet das übergeordnete Programm bezüglich der Zellennumerierung vor einem Unterprogramm, so muß es durch END abgeschlossen werden, da sonst das nachfolgende (Unterprogramm ungewollt erreicht wird.

Rückkehr aus Unterprogrammen

Format: **RETURN**

Funktion:
Das Ende des Unterprogramms ist erreicht. Die Programmabarbeitung wird im übergeordneten Programmteil fortgesetzt. Die Fortsetzung erfolgt unmittelbar nach der GOSUB-Anweisung, die den Eintritt in das Unterprogramm bewirkte.

Hinweis:
Ein Unterprogramm kann an mehreren Stellen (logisch) enden, die jeweils durch RETURN gekennzeichnet werden müssen.

Beispiel:
Angenommen, Sie sollen für eine nach Punkten bewertete schriftliche Leistungskontrolle einer Schulklasse die entsprechenden Noten sowie den Klassendurchschnitt nach Noten und Punkten bestimmen. Sie können dafür z. B. folgendes kleine BASIC-Programm benutzen:

```
10 CLS:PRINT"AUSWERTUNG LEISTUNGSKONTROLLE"
20 PRINT:PRINT
30 INPUT"ANZAHL DER SCHUELER? ";A$
40 INPUT"HOECHSTPUNKTZAHL LEISTUNGSKONTROLLE?"
   ;HP
50 PRINT:PRINT
60 DIM ERG(A$-1,1) :! Feld fuer Punkte und Noten
70 FOR I=1 TO A$-1 :! Einzelergebnisse
80 PRINT"PUNKTE DES";I+1; "-TEN SCHUELERS";
90 INPUT P: ERG(I,0)=P :!Punktzahlspeicherung
100 GOSUB 300
110 ERG(I,1)=N     :! Notenspeicherung
120 PRINT:PRINT"NOTE";N:PRINT
130 NEXT I
140 PRINT:PRINT: "KLASSENDURCHSCHNITT":PRINT
150 KD=0:GOSUB 500:! Punktzahldurchschnitt
160 PRINT"NACH PUNKTEN: ";D; "(VON";HP; ")":
    PRINT
170 KD=1:GOSUB 500:! Notendurchschnitt
180 PRINT"NACH NOTEN: ";D;:PRINT:PRINT
190 END       :! Ende Hauptprogramm
300 ! Unterprogramm Notenermittlung
310 PP=100*P/HP   :! P zu HP in Prozent
```

```
320 PP=INT(PP+.5) :! Rundung zu Prozent
330 IF PP>=96 THEN N=1 : RETURN
340 IF PP>=80 THEN N=2 : RETURN
350 IF PP>=60 THEN N=3 : RETURN
360 IF PP>=37 THEN N=4 : RETURN
370 N=5 : RETURN
500 ! Unterprogramm Durchschnittsberechnung
510 D=0
520 FOR I=0 TO AS-1
530 D=D+ERG(I,KD) :! Punkte
540 NEXT I
550 D=D/AS
560 D=INT(D*100+.5)/100:! Rundung
570 RETURN
```

Sicher hätten Sie bei diesem einfachen Beispiel auch ohne Unterprogramme auskommen können. Berücksichtigen Sie jedoch, daß eine ausgebaute Problemlösung zum Beispiel auch noch Korrekturmöglichkeiten, eine Datenspeicherung auf Magnetbandkassette sowie weitere Auswertemöglichkeiten umfassen sollte! Sie werden dann die Vorteile einer modularen Programmgestaltung, d. h. die Realisierung von fest umrissenen Teilfunktionen in getrennten Unterprogrammen, schnell zu schätzen wissen.

Berechneter Aufruf von Unterprogrammen

Format:
ON *ausdruck* **GOSUB** *zeilennummer* [,*zeilennummer*] ...
ausdruck - numerischer Ausdruck mit einem Wert größer als oder gleich Null, dessen ganzer Anteil benutzt wird
zeilennummer - BASIC-Zeilennummer des aufzurufenden Unterprogrammes

Funktion:
Der ganzzahlige Wert des Ausdrucks wird bestimmt und sei gleich 1. Das Programm wird dann durch Eintritt in ein Unterprogramm bei der Zeilennummer fortgesetzt, die an 1-ter Stelle in der angegebenen Zeilennummernliste steht.

Nach Auftreten einer RETURN-Anweisung erfolgt die Rückkehr und Programmfortsetzung mit der Anweisung, die auf die ON ... GOSUB-Anweisung folgt.

Hinweis:
Ist der ganzzahlige Wert des Ausdrucks gleich Null oder größer als die Anzahl der angegebenen Zeilennummern, wird das Programm mit der auf die ON ... GOSUB-Anweisung folgenden Anweisung fortgesetzt.

Beispiel:
Zur Übung verwenden Sie das Beispiel zu ON ... GOTO aus Abschnitt 4.5. Ersetzen Sie die ON ... GOTO-Anweisung durch die

```
200 ON KZ GOSUB 500,800,1000,1500
205 IF KZ>0 AND KZ<4 THEN 100
```

und fügen Sie an den entsprechenden Stellen RETURN ein.

4.13. Zeichenkettenfunktionen

ASC	ASCII-Code zu gegebenem Zeichen
CHR$	Zeichen zu gegebenem ASCII-Code
VAL	Zahl aus gegebener Zeichenkette
STR$	Zeichenkette aus gegebener Zahl
LEN	Länge einer Zeichenkette
LEFT$	
RIGHT$	Übernahme einer Teilzeichenkette
MID$	
INSTR	Suche einer Zeichenkette in einer anderen Zeichenkette
STRING$	Wiederholung einer Zeichenkette

Diese Zeichenkettenfunktionen sind im BASIC Ihres „robotron Z 9001“ ständig verfügbar. Sie bieten Ihnen wirkungsvolle Unterstützung beim Umgang mit Zeichenketten, d. h. bei der „nichtnumerischen Datenverarbeitung“. Ihre Benutzung erfolgt in der gleichen Weise wie die der numerischen Funktionen.

Bitte beachten Sie:
- Ebenso wie Zeichenkettenvariable enden die Namen derjenigen Zeichenkettenfunktionen, die eine Zeichenkette als Funktionswert liefern, mit dem Sonderzeichen Dollar ($).
- Zeichenketten können mit dem Operationszeichen + verkettet, d. h. aneinandergehängt, werden (z. B. liefert "AB"+"BA" die Zeichenkette "ABBA")
- Die „leere Zeichenkette“ enthält kein Zeichen und hat die Länge Null.

Wandlung Zeichen ⟷ ASCII-Code

Format:
ASC(*zeichenkette*)
zeichenkette - beliebiger Zeichenkettenausdruck, nur erstes Zeichen wird ausgewertet

Typ: BASIC-Funktion

Funktion:
Als Funktionswert wird der (dezimale) Wert des ASCII-Codes des ersten Zeichens der angegebenen Zeichenkette angenommen (zur Codierung vgl. Anhang A).

Hinweis:
Der Aufruf von ASC mit der leeren Zeichenkette als Funktionsargument führt zu einem Programmabbruch (FC ERROR).

Beispiel:

```
>A$="AB" : PRINT ASC(A$)
```

- Diese sofort ausführbare Anweisung liefert den Wert 65, ebenso wie auch

```
>PRINT ASC("AB")
```

oder

```
>PRINT ASC("A")
```

(Das Zeichen A hat die interne Codierung 01000001. Berechnen Sie den Dezimalwert dieser Binärzahl, so erhalten Sie 1*2^0+1*2^6=1+64=65.)

Format:
CHR$(*ausdruck*)
ausdruck - numerischer Ausdruck, dessen ganzzahliger Wert zwischen 0 und 255 liegen muß

Typ: BASIC-Funktion

Funktion:
Als Funktionswert wird das Zeichen geliefert, das den als Funktionsargument angegebenen (dezimalen) Wert des ASCII-Codes hat.

Beispiel:

```
10 I=32
20 A$=CHR$(I+33)
30 PRINT A$
```

Dieses Programmstück bewirkt die Ausgabe des Zeichens A auf dem Bildschirm, da das Zeichen A die (dezimale) ASCII-Codierung 65 hat.
Schauen Sie sich nun einmal die darstellbaren Zeichen des Zeichensatzes auf dem Bildschirm an:

```
10 CLS
20 FOR I=32 TO 127
30 PRINT CHR$(I);
40 NEXT I
50 PRINT:PRINT
60 PAUSE
70 FOR I=128 TO 255
80 PRINT CHR$(I);
90 NEXT I
100 PRINT:PRINT:END
```

Nach der Anzeige der alphanumerischen Zeichen betätigen Sie [CONT] und erhalten die Grafikzeichen.

Wandlung Zeichenkette ⟷ Zahl

Format:
VAL(*zeichenkette*)
zeichenkette - Zeichenkettenausdruck, der die Zeichenkettendarstellung einer Zahl enthält

Typ: BASIC-Funktion

Funktion:
Als Funktionswert wird die Zahl geliefert, die der im Argument übergebenen Zeichenkette entspricht. Beginnt diese nicht mit einer Ziffer, einem Dezimalpunkt oder einem Vorzeichen, wird der Funktionswert Null angenommen.

Hinweis:
In der Argumentzeichenkette nach einer Ziffer auftretende Sonderzeichen oder Buchstaben werden, wie alle nachfolgenden Zeichen, nicht weiter ausgewertet. Davon ausgenommen sind Zahlen in Gleitkomma-darstellung.

Beispiele:
Vollziehen Sie die nachfolgenden Anweisungen im Kommandomodus nach!

```
> A$="1234": PRINT VAL(A$)+1000
2234
> PRINT VAL(A$+".5")
1234.5
> PRINT VAL("5152-45-56047")
5152
```


Format:
STR$(ausdruck)
ausdruck - numerischer Ausdruck

Typ: BASIC-Funktion

Funktion:
Als Funktionswert wird die Zeichenkette geliefert, die d des im Argument angegebenen numerischen Ausdruck stellt.

Hinweis:
Wenn der Wert des numerischen Ausdrucks nicht negativ ist, bekommt die Zeichenkette ein Leerzeichen (auf der Position des Vorzeichens) vorangestellt.

Beispiele:

```
>PPINT STR$(1234+1000)
2234
>A=-11.77:PRINT A,STR$(A)
-11.77   -11.77
```

Länge einer Zeichenkette

Format:
LEN(*zeichenkette*)
zeichenkette - zeichenkettenausdruck

Typ: BASIC-Funktion

Funktion:
Die Anzahl der in der Zeichenkette enthaltenen Zeichen (einschließlich nicht darstellbarer und Leerzeichen) wird als Funktionswert geliefert.

Beispiel:

```
>T$="robotron Z9001":PRINT LEN(T$)
  14
>T$="":PRINT LEN(T$)
  0
```

Übernahme einer Teilzeichenkette

Format:
LEFT$(*zeichenkette*, *zeichenanzahl*)
zeichenkette - Zeichenkettenausdruck
zeichenanzahl - Anzahl der aus der Zeichenkette (von links beginnend) zu übernehmenden Zeichen; Wert zwischen 0 und 255

Typ: BASIC-Funktion

Funktion:
Der Funktionswert entspricht der Zeichenkette, die, von links beginnend, aus der angegebenen Anzahl Zeichen der Zeichenkette im Argument gebildet wird.

Hinweis:
Ist die geforderte Zeichenanzahl größer als die vorhandene, wird die gesamte Zeichenkette übernommen. Falls die gewünschte Zeichenanzahl Null ist, wird die leere Zeichenkette zugewiesen.

Beispiel:

```
>A$="ABCDEF":PRINT LEFT$(A$,3)
ABC
```

Format:
RIGHT$(*zeichenkette*, *zeichenanzahl*)
zeichenkette - Zeichenkettenausdruck
zeichenanzahl - Anzahl der aus der Zeichenkette (von rechts beginnend) zu übernehmenden Zeichen; zwischen 0 und 255

Typ: BASIC-Funktion

Funktion:
Der Funktionswert entspricht der Zeichenkette, die, von rechts beginnend, aus der angegebenen Anzahl Zeichen der Zeichenkette im Argument gebildet wird.

Hinweis:
Siehe Hinweis zu LEFT$

Beispiel:

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

Format:
MID$(*zeichenkette*, *ab_position* [,*zeichenanzah*l])
zeichenkette - Zeichenkettenausdruck
ab_position - Position des ersten zu übernehmenden Zeichens in *zeichenkette*; Wert zwischen 1und 255
zeichenanzahl - Anzahl der aus *zeichenkette* (von *ab_position* nach rechts fortschreitend) zu übernehmenden Zeichen; Wert zwischen 0 und 255

Typ: BASIC-Funktion

Funktion:
Der Funktionswert entspricht der Zeichenkette, die, von der angegebenen Position nach rechts fortschreitend, aus der Zeichenkette im Argument gebildet wird. Ist eine Zeichenanzahl angegeben, werden maximal so viele Zeichen übernommen.

Hinweise:
1. Liegt die *ab_position* hinter dem letzten vorhanden Zeichen, wird die leere Zeichenkette geliefert.
2. Ist die geforderte Zeichenanzahl ab vorgegebener Position größer als die vorhandene, wird die gesamte restliche Zeichenkette zugewiesen.

Beispiel:

```
>A$="ABCDEF":PRINT MID$(A$,3,2)
DEF
```

Suche einer Zeichenkette in einer Zeichenkette

Format:
INSTR(*zeichenkette1*, *zeichenkette2*)
zeichenkette1,2 - Zeichenkettenausdrücke

Typ: BASIC-Funktion

Funktion:
Es wird ermittelt, ob die *zeichenkette1* vollständig in der *zeichenkette2* enthalten ist. Als Funktionswert wird die Position (des ersten Zeichens) der *zeichenkette1* bezüglich ihres ersten Auftretens in der *zeichenkette2* angenommen. Wird die *zeichenkette1* nicht gefunden, ergibt sich der Funktionswert Null.

Beispiel:

```
T1$="01":PRINT INSTR(T1$,"robotron Z9001")

13
```

Wiederholung einer Zeichenkette

Format:
STRING$(*wiederholungen*, *zeichenkette*)
wiederholungen - numerischer Ausdruck, dessen ganzzahliger Wert festlegt, wie oft die Zeichenkette zu wiederholen ist; Wert zwischen 1 und 255
zeichenkette - Zeichenkettenausdruck

Typ: BASIC-Funktion

Funktion:
Der Funktionswert entspricht der Zeichenkette, die durch die angegebene Anzahl von Wiederholungen der Zeichenkette im Argument entsteht.

Hinweis:
Die Länge der erzeugten Zeichenkette darf 255 Zeichen nicht überschreiten.

Beispiele:

```
>PRINT STRING$(5, "Z9001 ")
Z9001 Z9001 Z9001 Z9001 Z9001
```

Unterstreichen Sie nun noch eine Überschrift durch Wiederholung des Zeichens mit dem ASCII-Code 160. Die LEN-Funktion nimmt Ihnen die Mühe ab, die Zahl der erforderlichen Wiederholungen zu ermitteln.

```
10 NE$="*** robotron Z9001 ***"
20 PRINT NE$
30 PRINT STRING$(LEN(NE$),CHR$(160))
```

Informieren Sie sich bitte auch im Abschnitt 6.2. über weitere Anwendungsmöglichkeiten der Zeichenkettenfunktionen.

Dieses Programmierhandbuch wurde verfaßt
von einem Autorenkollektiv
des VEB Robotron-Meßelektronik »Otto Schön« Dresden
Dr.-Ing. Hans-Jürgen Busch
Dr.-Ing. Joachim Haase
Dr. rer. nat. Gert Keller
Dr.-Ing. Hans-Jörg Nowottne

DEWAG Dresden . ATN 33442 031/4 . Regie: Mros; Gestaltung: Bucher
6/85a . Jt 2234/85 und Jt 2235/85 . II-13-1

Inhaltsverzeichnis

4.14. Spezielle Eingabefunktionen

INKEY$ Tastaturabfrage
JOYST Spielhebelabfrage

Diese beiden BASIC-Funktionen liefern Informationen über die Betätigung der Tastatur bzw. der Spielhebel.

Tastaturabfrage

Format: **INKEY$**

Typ: BASIC-Funktion

Funktion:
Als Funktionswert erhält man ein Zeichen (Zeichenkette der Länge 1), wenn vor Aufruf der Funktion eine Taste gedrückt worden ist. Das Zeichen ist der zuletzt gedrückten Taste, deren Betätigung noch nicht ausgewertet wurde, äquivalent. Es kann auch ein nicht darstellbares Zeichen sein (z. B. bei gedrückter Kursortaste). Ist keine Taste gedrückt worden, ergibt sich als Funktionswert die leere Zeichenkette ("").

Hinweise:

1. Im Gegensatz zur INPUT-Anweisung erlaubt die INKEY$-Funktion die Abfrage der Tastaturbetätigung, ohne die Programmabarbeitung zu unterbrechen. Wird in einem Programm die Tastaturbetätigung zyklisch abgefragt und ausgewertet, kann der Zeitpunkt, zu dem das Programm an einer anderen Stelle fortgesetzt werden soll, von ihnen festgelegt werden. Im ersten Beispiel wird dieser Weg demonstriert.
2. Mit der INKEY$-Funktion können Eingaben übernommen werden, die nicht durch Betätigung der [ENTER]-Taste abgeschlossen sein müssen. Das kann für den Nutzer eines Programms mit häufigen Eingabeaufforderungen, z. B. eines Spielprogramms, angenehm sein.

Beispiele:

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

Nach dem Programmstart sehen Sie das folgende Bild:

```
****************************************
****ENDE
```

Da die PRINT-Anweisung in Zeile 300 mit einem Semikolon abgeschlossen ist, erfolgt die Ausgabe der Sterne unmittelbar nacheinander, solange die Taste [E] nicht gedrückt ist. Nach dem Betätigen von [E] wird ENDE ausgegeben. Unter Verwendung der CHR$-Funktion ist es auch möglich, eine Taste abzufragen, bei deren Betätigung ein nicht darstellbares Zeichen als Funktionswert der INKEY$-Funktion geliefert wird. Soll beispielsweise anstelle der Taste [E] die Taste [ESC] (Code: 27, vgl. Anhang A) abgefragt werden, ist im Beispiel Zeile 310 durch

```
310 IF INKEY$=CHR$(27) THEN 330
```

zu ersetzen. Mit dem folgenden Programm kann festgestellt werden, welches Zeichen vom Heimcomputer übernommen wird, wenn Sie eine spezielle Taste drücken. Außerdem wird der dazugehörige Code des Zeichens ausgegeben (vgl. Anhang A).

```
10 ZK$=INKEY$:IF ZK$="" THEN GOTO 10
20 Z=ASC(ZK$)
30 IF Z>31 THEN PRINT "ZEICHEN: ";ZK$,
   "CODE:";Z:ELSE  PRINT; "CODE: ";Z
40 GOTO 10
```

Durch [STOP] kann die Programmabarbeitung beendet werden.

Spielhebelabfrage

Sie können für Ihren Heimcomputer zwei Spielhebel erwerben. In Ihrem BASIC-Programm kann abgefragt werden, wie Sie und Ihr Mitspieler diese Spielhebel betätigen. In Abhängigkeit von dem Ergebnis dieser Abfrage können Sie dann den weiteren Ablauf Ihres BASIC-Programms steuern. Damit die Spielhebelbetätigung richtig ausgewertet werden kann, muß der Spielhebel so gehalten werden, daß die flache schmale Taste, die Aktionstaste, vom Körper wegzeigt. Der Anschluß der Spielhebel an Ihren Heimcomputer ist in der Bedienungsanleitung (vgl. Abschnitt 3.1) beschrieben. Verfügen Sie über keine Spielhebel, können Sie auch die entsprechenden Kursortasten betätigen.

Format:
JOYST(*spielhebel*)
spielhebel - numerischer Ausdruck (ganzzahliger Wert: 1 oder 2)

Typ: BASIC-Funktion

Funktion:
In Abhängigkeit vom Wert des numerischen Ausdrucks *spielhebel* wird der Spielhebel 1 oder der Spielhebel 2 abgefragt. Als Funktionswert erhält man einen numerischen Wert, der durch die Stellung des Spielhebels bestimmt wird.

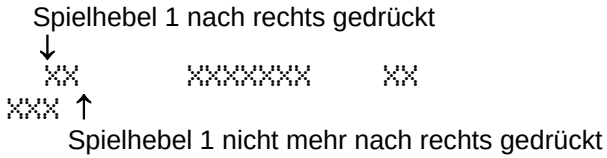
Spielhebeldruckpunkt	Funktionswert von JOYST	äquivalente Tastaturbetätigung
Ruhestellung	0	keine Taste gedrückt
	1	[←]
	2	[→]
	4	[↓]
	8	[↑]
	5	[←][↓]
	6	[→][↓]
	9	[←][↑]
	10	[→][↑]
Aktionstaste	16	[ESC]

Hinweise:
1. Während die Spielhebel im Programm abgefragt werden, sollte keine Tastaturbetätigung erfolgen. Ausgenommen ist der Fall, daß anstelle eines Spielhebels die Taste [ESC] oder die Kursortasten betätigt werden.
2. Es ist zu beachten, daß durch die Spielhebelbetätigung der Arbeitszustand der Tastatur verändert werden kann. Ist der Grafikmodus (GRAPHIC-Anzeige leuchtet) durch Spielhebelbetätigung eingeschaltet worden, kann er durch Drücken der Taste [GRAPHIC] wieder ausgeschaltet werden. Ist nach Spielhebelbetätigung keine Eingabe über die Tastatur mehr möglich, drücken Sie bitte eine der Tasten [1] bis [8].

Beispiel:

```
10 IF JOYST(1)=2 THEN PRINT "X";:ELSE PRINT " "
20 GOTO 10
```

Solange der Spielhebel 1 nach rechts gedrückt wird, erscheint eine Folge von Zeichen X auf dem Bildschirm. Sonst werden Leerzeichen ausgegeben. Durch das Abschließen der PRINT-Anweisungen mit einem Semikolon erfolgt die Ausgabe der Zeichen unmittelbar nacheinander.



Mit [STOP] kann die Programmabarbeitung beendet werden.

4.15. Weitere Ausgabemöglichkeiten

WINDOW	Festlegung des Ausgabebereiches
PRINT	Ausgabeanweisung
TAB	Tabulatorfunktion
SPC	Leerzeichenausgabe
POS	Abfrage der Kursorstellung
PRINT AT	Ausgabe ab vorgegebener Bildschirmposition
OUT 136, *code*	20/24-Zeilen-Modus

Die angegebenen Anweisungen und Funktionen sollen Ihnen helfen, die von Ihren Programmen ermittelten Ergebnisse in übersichtlicher Form auf den Bildschirm auszugeben. Für die PRINT-Anweisung werden die Erläuterungen aus Abschnitt 4.4 ergänzt.

Festlegung des Ausgabebereiches
Standardmäßig stehen für die Ausgabe alle 24 Bildschirmzeilen mit je 40 Spalten zur Verfügung. Oft ist es aber wünschenswert, als Ausgabebereich für die PRINT-Anweisungen und die Eingabeaufforderungen der INPUT-Anweisungen nur einen Teil des Bildschirms zuzulassen und den Rest der Bildschirmdarstellung nicht zu verändern.

Format:
WINDOW [*zeile1*, *zeile2*, *spalte1*, *spalte2*]
zeile1 - erste Bildschirmzeile des Ausgabebereiches
zeile2 - letzte Bildschirmzeile des Ausgabebereiches
spalte1 - erste Bildschirmspalte des Ausgabebereiches
spalte2 - letzte Bildschirmspalte des Ausgabebereiches

Funktion:
Diese Anweisung gestattet, einen rechteckigen Abschnitt des Bildschirms als Ausgabebereich zu definieren. Innerhalb des Ausgabebereiches erscheinen sämtliche Ausgaben von PRINT-Anweisungen, Eingabeaufforderungen von INPUT-Anweisungen, Fehlermeldungen des BASIC-Interpreters sowie Ein- und Ausgaben im Kommandomodus.
Für die in der Anweisung stehenden Zeilen und Spalten sind numerische Ausdrücke mit ganzzahligen Werten zugelassen, die den Bedingungen

$0 \le \text{zeile1} \le \text{zeile2} \le 23$
$0 \le \text{spalte1} \le \text{spalte2} \le 39$

genügen müssen. Standardmäßig, d. h. nach Neustart des BASIC-Interpreters, gilt der gesamte Bildschirm (Zeile 0 bis 23, Spalte 0 bis 39) als Ausgabebereich. Der gesamte Bildschirm wird auch als Ausgabebereich festgelegt, wenn in der Anweisung die Zeilen- und Spaltenangaben weggelassen werden.

Hinweise:
1. Außerhalb des Ausgabebereiches liegende Teile des Bildschirms bleiben bei Ausgaben mit der PRINT-Anweisung „eingefroren“. Diese Teile können mit der Anweisung PRINT AT oder durch POKE (vgl. Abschnitt 5.1) verändert werden.
2. Mit der CLS-Anweisung wird nur der durch eine WINDOW- Anweisung definierte Ausgabebereich gelöscht.
3. Nach Ausführung einer WINDOW-Anweisung steht der Kursor in der linken oberen Ecke des Ausgabebereiches. Weitere Ausgaben erfolgen ab dieser Position.

Beispiel:
Zunächst wird der gesamte Bildschirm als Ausgabebereich definiert und gelöscht. Danach wird die Überschrift für die folgende Tabelle in die 0. Bildschirmzeile ausgegeben. Durch die folgende Definition der 2. bis 23. Bildschirmzeile als Ausgabebereich erreicht man, daß bei Ausführung der folgenden PRINT- Anweisungen die Überschrift nicht gelöscht wird.

```
10 WINDOW:CLS
20 PRINT VZAHL","QUADRATZAHL"
30 WINDOW 2,23,0,39
40 FOR I=1 TO 50
50 PRINT I,I*I:PAUSE 3
60 NEXT I
```

Legen Sie bitte nach Ausführung dieses Beispiels durch Eingabe von WINDOW im Kommandomodus wieder den gesamten Bildschirm als Ausgabebereich fest.

Ergänzungen zur PRINT-Anweisung
Wie Sie mit der PRINT-Anweisung Werte ausgeben können, haben Sie bereits im Abschnitt 4.4 kennengelernt. Die in folgenden erläuterten Möglichkeiten sollen Ihnen helfen, die Ausgaben noch übersichtlicher zu gestalten.

Format:
PRINT [*ausgabeliste* [*endezeichen*]]
ausgabeliste - Folge von Ausgabeelementen
endezeichen - Komma oder Semikolon

Funktion:
Die Ausgabeliste ist eine Folge von Ausgabeelementen, zwischen denen Trennzeichen stehen. Es gibt drei Arten von Ausgabeelementen:
- Numerische Konstanten, Variable und Ausdrücke bewirken die Ausgabe eines numerischen Wertes, gefolgt von einem Leerzeichen.
- Zeichenkettenkonstanten, -variable und -ausdrücke ergeben die Ausgabe einer Folge von Zeichen.
- Funktionen zur Steuerung der Ausgabe (TAB, SPC), die dazu dienen, den Beginn der Ausgabe des nächsten numerischen Wertes oder der nächsten Zeichenkette festzulegen.

Die Ausgabe des ersten Elements der Ausgabeliste beginnt, wenn die Ausgabeliste der letzten davor ausgeführten PRINT-Anweisung nicht durch ein Endezeichen abgeschlossen wurde, an der linken Begrenzung des Ausgabebereiches auf der Position Null einer neuen Zeile. Die weitere Stellung der Ausgabeelemente auf dem Bildschirm ergibt sich durch die vorangestellten Trennzeichen. Als Trennzeichen sind Komma und Semikolon zugelassen (vgl. Abschnitt 4.4)
Steht nach der Ausgabeliste kein Endezeichen, beginnt die Ausgabe der Elemente der Ausgabeliste der nächsten PRINT-Anweisung in einer neuen Zeile des Ausgabebereiches. Durch die Beendigung der Ausgabeliste mit einem Endezeichen (Komma, Semikolon) wird die Ausgabeliste nicht abgeschlossen. Das Endezeichen wirkt wie ein Trennzeichen zwischen dem letzten Element der nicht abgeschlossenen Ausgabeliste und dem ersten Element der Ausgabeliste der nächsten PRINT-Anweisung.

Hinweis:
1. Durch PRINT wird die aktuelle Kursorposition verändert. Bei Ausführung einer PRINT-Anweisung steht der Kursor jeweils hinter dem gerade zuletzt ausgegebenen Zeichen.
2. Ist die Ausgabeliste einer PRINT-Anweisung nichf abgeschlossen und folgt danach eine INPUT-Anweisung, so wird der *hinweis*-Text ab der Position ausgegeben, in der das erste Element einer neuen Ausgabeliste beginnen würde.

Beispiel:

```
10 DIM A(6)
20 FOR I=0 TO 6
30 INPUT A(I)
40 NEXT I
50 FOR I=0 TO 6
60 PRINT A(I),
70 NEXT I
```

Das Feld A wird dimensioniert, und dann werden Sie zur Eingabe der 7 Werte der Elemente von A aufgefordert. Anschließend werden diese Werte wieder ausgegeben. Da die PRINT- Anweisung in Zeile 60 ein Komma als Endezeichen hat, erfolgt die Ausgabe der Feldelemente in Standardtabellenform. Fehlt dieses Komma, wird jedes Element von A in eine neue Zeile ausgegeben.
Das folgende kleine Beispielprogramm, das Sie zur Eingabe von Wörtern mit verschiedenen Anfangsbuchstaben auffordert, demonstriert die Verknüpfung von PRINT- und INPUT-Anweisung.

```
10 FOR I=65 TO 90 STEP 3
20 PRINT "WORT MIT ";CHR$(I);
30 INPUT W$
40 NEXT I
```

Nach Programmstart erscheint folgendes Bild:

```
WORT MIT A?
```

Das Semikolon, mit dem die Ausgabeliste der PRINT-Anweisung in Zeile 20 beendet wird, ermöglicht die Anzeige von Frage und Antwort in eine, Bildschirmzeile. Das Fragezeichen ergibt sich durch den fehlenden *hinweis*-Text bei der INPUT-Anweisung. Ist es nicht erwünscht, muß die Zeile 30 durch

```
30 INPUT"";U$
```

ersetzt werden.

Funktionen zur Steuerung der Ausgabe

Ähnlich wie bei der Schreibmaschine kann mit der Tabulatorfunktion eine Position festgelegt werden, ab der das nächste Element der Ausgabeliste einer PRINT-Anweisung auszugeben ist.

Format:
TAB(position)
position - numerischer Ausdruck
(ganzzahliger Wert von 0 bis 255)

Funktion:
Die Tabulatorfunktion kann nur als Ausgabeelement in der Ausgabeliste einer PRINT-Anweisung verwendet werden. Der Wert des Ausdrucks gibt die Position an, auf der die Ausgabe des folgenden Elements der Ausgabeliste beginnt. Die Tabulatorfunktion hat keine Wirkung, wenn bereits auf eine Position rechts von der als Argument der Tabulator-funktion angegebenen Position ausgegeben wurde, d. h., wenn vor ihrer Ausführung die aktuelle Kursorposition bereits rechts von der durch *position* festgelegten Stelle des Ausgabebereiches steht.

Hinweise:
1. Als Trennzeichen vor und nach der Tabulatorfunktion sollte nur das Semikolon verwendet werden.
2. Alle Zeichen zwischen den Kursorpositionen vor und nach Ausführung der Tabulatorfunktion werden gelöscht.

Beispiel.:
Das erste Beispiel soll ihnen die Zahlung der Positionen demonstrieren.

```
10 PRINT "A"
20 PRINT TAB(0);"B"
30 PRINT TAB(1);"C"
```

Nach der Programmausführung ergibt sich das folgende Bild:

```
A
B
 C
```

Bei der Ausgabe von Tabellen wird die TAB-Funktion häufig angewendet.

```
10 PRINT "GLEICHE";TAB(10);"ANFANGSPOSITION"
20 PRINT "AB";TAB(10);"CD"
30 PRINT "RST";TAB(10);"UVW"
```

Dieses Programm liefert das folgende Bild:

```
GLEICHE  ANFANGSPOSITION
AB       CD
RST      UVW
```

Eine definierte Anzahl von Leerzeichen wird durch die folgende Funktion ausgegeben.

Format:
SPC(*anzahl*)
anzahl - numerischer Ausdruck (ganzzahliger Wert von 0 bis 255)

Funktion:
Die SPC-Funktion kann nur als Ausgabeelement in der Ausgabeliste einer PRINT-Anweisung verwendet werden. Der Wert des ganzzahligen numerischen Ausdrucks gibt an, wie viele Leerzeichen auszugeben sind.

Hinweis:
SPC(I) hat als Ausgabeelement dieselbe Wirkung wie STRING$(I," ").

Beispiel:

```
OK
>PRINT "SPC-";SPC(7)";FUNKTION"
SPC-       FUNKTION
```

Abfrage der Kursorstellung

Format: **POS(0)**

Typ: - BASIC-Funktion

Funktion:
Diese Funktion liefert als Funktionswert die Position, auf die das nächste Ausgabeelement einer PRINT-Anweisung ausgegeben wird. Der Funktionswert liegt zwischen 0 und der durch das WIDTH-Kommando festgelegten Zeilenlänge (Standardwert: 255).

Beispiel:
Mit dem folgenden Programm können alle Kleinbuchstaben ausgegeben werden. Dabei erscheinen jeweils nur fünf in einer Zeile.

```
10 FOR I=97 TO 122
20 PRINT CHR$(I);
30 IF POS(0)=5 THEN PRINT
40 NEXT I
```

In Zeile 30 wird überprüft, ob das nächste Zeichen auf die 5. Position (die Zählung der Positionen beginnt mit 0) ausgegeben werden soll. Ist das der Fall, wird die Ausgabeliste durch die Anweisung PRINT abgeschlossen und das nächste Zeichen in einer neuen Zeile ausgegeben.

Ausgabe ab vorgegebener Bildschirmposition

Ausgaben mit der PRINT-Anweisung beginnen stets hinter der aktuellen Kursorposition. Oft ist es aber wünschenswert, auch davor die Darstellung auf dem Bildschirm zu verändern. Die "PRINT AT"-Anweisung erlaubt es Ihnen, in beliebiger Reihenfolge auf beliebige Stellen des Bildschirms auszugeben.

Format:
PRINT AT (*zeile*,*spalte*); *ausgabeliste*
zeile - numerischer Ausdruck (ganzzahliger Wert von 0 bis 23)
spalte - numerischer Ausdruck (ganzzahliger Wert von 0 bis 39)
ausgabeliste - Folge von Ausdrücken, die durch Kommas getrennt sind.

Funktion:
Die angegebenen Ausdrücke der Ausgabeliste werden, beginnend auf der durch *zeile* und *spalte* festgelegten Bildschirmposition, ausgegeben. Es sind zugelassen:
- numerische Konstanten, Variable und Ausdrücke
- Zeichenkettenkonstanten, -variable und -ausdrücke.

Die Werte dieser Ausdrücke werden **fortlaufend** ausgegeben. Das Komma fungiert lediglich als Trennzeichen zwischen den Ausdrücken der Liste. Es hat hier dieselbe Wirkung wie das Semikolon in der Ausgabeliste einer PRINT-Anweisung.

Hinweise:
1. Durch PRINT AT wird die aktuelle Position des Kursors nicht beeinflußt. Ein Überschreiben des Zeichens auf der Kursorposition kann aber dazu führen, daß nach Ausführung einer folgenden PRINT- oder WINDOW-Anweisung dieses Zeichen wieder gelöscht ist.
2. Zwischen PRINT und AT braucht bei Eingabe der Anweisung kein Leerzeichen zu stehen. Anstelle von PRINT kann auch ein Fragezeichen ? eingegeben werden.

Beispiele:

```
10 WINDOW:CLS
20 PRINT AT (10,5);-3, "GRAD"
```

Auf dem gelöschten Bildschirm wird in Zeile 10, in Spalte 5 beginnend, der Text

```
-3 GRAD
```

angezeigt. Dieselbe Wirkung hat das folgende Programm, das Ihnen demonstriert, daß Sie für *zeile* und *spalte* auch numerische Ausdrücke und Variable setzen können.

```
10 WINDOW:CLS
20 Z=3:SP=5
30 W=-5
40 PRINT AT (Z+7,SP);W, "GRAD"
```

20/24-Zeilen-Modus

Standardmäßig können Sie bis zu 24 Zeilen auf dem Bildschirm anzeigen (24-Zeilen-Modus). Bei Bedarf ist es aber auch möglich, die angezeigten Zeilen auseinanderzurücken und die Ausgabe von nur 20 Zeilen pro Bild anzuweisen (20-Zeilen-Modus). Damit ist z. B. die übersichtliche Ausgabe längerer Texte möglich.

Format:
OUT 136, *code*
code - numerischer Ausdruck mit ganzzahligem Wert (siehe Tabelle)

Funktion:
Mit Hilfe des Wertes des numerischen Ausdrucks *code* können Sie den Zeilenmodus und die Bildschirmrandfarbe (wenn Ihr Heimcomputer eine Farbwiedergabe ermöglicht) beeinflussen.

Bildschirmrandfarbe	20-Zeilen-Modus	24-Zeilen-Modus
schwarz	4	0
rot	12	8
grün	20	16
gelb	28	24
blau	36	32
purpur	44	40
zyan	52	48
weiß	60	56

Andere Werte als die in der Tabelle angegebenen sind für den Wert von *code* nicht zulässig. Verfügt Ihr Heimcomputer über keine Möglichkeit zur Farbwiedergabe, können Sie zur Einstellung des gewünschten Zeilenmodus einen beliebigen *code*-Wert aus der entsprechenden Spalte verwenden.

Hinweise:
1. Nach einer BORDER-Anweisung (vgl. Abschnitt 4.16) ist stets der standardmäßige 24-Zeilen-Modus wirksam.
2. Bei Einstellung des 20-Zeilen-Modus können alle Speicherplätze des Bildspeichers beschrieben werden. Dabei werden aber nur die Zeilen 0 bis 19 angezeigt. Es ist daher sinnvoll, den Ausgabebereich vor Umschalten in den 20-Zeilen-Modus mit einer Anweisung

   WINDOW *zeile1*,*zeile2*,*spalte1*,*spalte2*
   mit $0 \le zeile1 \le zeile2 \le 19$
   $0 \le spalte1 \le spalte2 \le 39$
   einzuschränken. Wird in den 24-Zeilen-Modus zurückgeschaltet, ist, falls notwendig, der Ausgabebereich mittels WINDOW-Anweisung wieder zu vergrößern.

Beispiele:

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

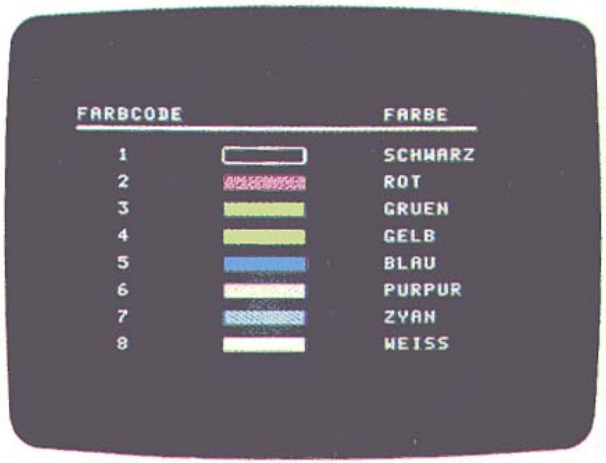
Mit der Anweisung in Zeile 60 wird der 20-Zeilen-Modus bei rotem Bildschirmrand eingestellt. In den 24-Zeilen-Modus bei schwarzem Bildschirmrand kann mit den folgenden Anweisungen zurückgeschaltet werden:

```
80 WINDOW
90 OUT 136,0
```

4.16. Farbige Ausgabe

PAPER		Hintergrundfarbe
INK		Vordergrundfarbe
BORDER		Bildschirmrandfarbe
PRINT	mit Farbe	lokale Farbeinstellung
PRINT AT	mit Farbe	lokale Farbeinstellung

Ist Ihr Heimcomputer nicht farbtüchtig, erscheinen die darzustellenden Zeichen stets weiß auf dunklem Hintergrund auf dem Bildschirm. Ermöglicht Ihr Gerät eine Farbwiedergabe, können Sie mit einfachen BASIC-Anweisungen die Farbe, in der die Zeichen dargestellt werden sollen, die Farbe des Hintergrundes und die Farbe des Bildschirmrandes beeinflussen.
Acht Farben, die Sie in beliebiger Weise kombinieren können, stehen Ihnen zur Verfügung. Den Farben sind die Zahlen von 1 bis 8 als Farbcodes zugeordnet.



Einstellung der Bildschirmfarben

Mit den nächsten Anweisungen können Sie die Farbdarstellung aller folgenden Ausgaben beeinflussen.

Format:
PAPER *farbcode*
farbcode - numerischer Ausdruck mit ganzzahligem Wert entsprechend Farbcodetabelle (1 bis 8)

Funktion:
Die Hintergrundfarbe (paper - Papier) für alle folgenden Ausgaben wird durch den Wert des Ausdrucks *farbcode* festgelegt.

Hinweis:
Im Kommandomodus wird dieselbe Wirkung auch durch gleichzeitiges Drücken der Tasten [SHIFT] [COLOR] (oder [CONTR] [U]) und anschließende Betätigung einer Zifferntaste (außer 3 und 4), die dem gewünschten Farbcode entspricht, erreicht. Wird keine der Ziffern von 1 bis 8 eingegeben, kann die Arbeit mit dem Heimcomputer nicht fortgesetzt werden.

Beispiel:

```
10 PAPER 5
20 CLS
```

Nach Ausführung dieser Anweisungen ist der gesamte Bildschirm blau. Die Farbe des Bildschirmrandes wird nicht verändert.
Mit den folgenden Anweisungen können Sie wieder Schwarz als Hintergrundfarbe einstellen.

```
30 PAPER 1
40 CLS
```

Format:
INK *farbcode*
farbcode - numerischer Ausdruck mit ganzzahligem Wert entsprechend Farbcodetabelle (1 bis 8)

Funktion:
Als Vordergrundfarbe (ink - Tinte) wird für alle folgenden Ausgaben die Farbe verwendet, deren Farbcode dem in der Anweisung bereitgestellten Wert entspricht.

Hinweis:
Im Kommandomodus wird dieselbe Wirkung durch Drücken der Taste [COLOR] (oder gleichzeitiges Drücken der Tasten [CONTR] [T]) und anschließende Betätigung der gewünschten Farbcodierung entsprechenden Zifferntaste (außer 3 und 4) erreicht. Geben Sie keine Ziffer von 1 bis 8 ein, kann die Arbeit mit dem Heimcomputer nicht fortgesetzt werden.

Beispiel:

```
10 PAPER 1 :CLS
20 INK 2
30 PRINT TAB(10); "UEBERSCHRIFTEN"
40 PRINT
50 INK 7
60 PRINT TAB(19); "TEXT"
```

Als Hintergrundfarbe wird Schwarz eingestellt. Der erste Text erscheint rot auf schwarz, der zweite Text wird zyan (hellblau) geschrieben. Zwischen beiden Ausschriften ist eine Leerzeile eingefügt. Es soll jetzt gezeigt werden, wie die Bildschirmrandfarbe eingestellt werden kann.

Format:
BORDER *farbcode*
farbcode - numerischer Ausdruck mit ganzzahligem Wert entsprechend Farbcodetabelle

Funktion:
Die Bildschirmrandfarbe wird entsprechend dem Wert des Farbcodes eingestellt.

Hinweis:
1. Durch gleichzeitiges Drücken der Tasten [CONTR] [E] und anschließende Betätigung der gewünschten Farbcodierung entsprechenden Zifferntaste (außer 3 und 4) wird im Kommandomodus dieselbe Wirkung erreicht. Die Eingabe einer Ziffer von 1 bis 8 ist für die Fortsetzung der Arbeit mit dem Heimcomputer nach Drücken der Tasten [CONTR] und [E] unbedingt erforderlich.
2. Durch die BORDER-Anweisung wird stets der 24-Zeilen-Modus für alle folgenden Ausgaben wirksam.

Beispiel:

```
10 A=5
20 BORDER A
```

Der Bildschirmrand leuchtet blau, nachdem die Anweisungen ausgeführt wurden.

Lokale Farbeinstellung
Die bisher erläuterten (globalen) Farbeinstellungen sind für die folgenden Ausgaben wirksam. Sie können aber für die Ausgabe der einzelnen Elemente der Ausgabeliste einer PRINT- oder einer „PRINT AT“-Anweisung lokal andere Farben vorschreiben.

Format:
PRINT *farbfestlegung*; [*ausgabeliste* [*endezeichen*]]
farbfestlegung - Vorschrift zur lokalen Änderung von Vorder- und/oder Hintergrundfarbe
ausgabeliste - siehe Abschnitt 4.4 und 4.15
endezeichen - Komma oder Semikolon

Funktion:
Als *farbfestlegung* sind erlaubt zur lokalen Änderung der

- Vordergrundfarbe	**INK** *farbcode*
- Hintergrundfarbe	**PAPER** farbcode
- Vorder- und Hintergrundfarbe	**INK** *farbcode1*, **PAPER** *farbcode2*

Entsprechend den Werten der numerischen Ausdrücke *farbcode* bzw. *farbcode1* und *farbcode2* werden lokal für die Ausgabe der Elemente der Ausgabeliste anderer Vorder- und/oder Hintergrundfarben wirksam. Zugelassen für die numerischen Ausdrücke sind Werte von 1 bis 8. Nach Ausführung der PRINT- Anweisung gelten wieder die vorher eingestellten Farben.

Hinweise:
1. Ist die Ausgabeliste mit einem Komma oder Semikolon als Endezeichen beendet, erfolgt die Rückschaltung auf die global eingestellten Farben nach dem Endezeichen.

2. Steht kein Endezeichen nach der Ausgabeliste, werden die global eingestellten Farben erst in der Zeile, in der die nächste Ausgabe erfolgt, wirksam.

Beispiel:

```
10 INK 7:PAPER 1:CLS
20 PRINT TAB(10);
30 PRINT INK 8;PAPER 5;"HERVORHEBUNG";:PRINT
40 PRINT
50 PRINT TAB(10);"AKTUELLE FARBEN"
```

In Zeile 10 werden global Zyan als Vordergrund- und Schwarz als Hintergrundfarbe eingestellt und der Anzeigebereich gelöscht. Der Text HERVORHEBUNG wird, beginnend auf Position 10, weiß auf blau ausgegeben. Durch das Semikolon als Endekennzeichen der Ausgabeliste werden sofort nach Beendigung der Ausgabe wieder die globalen Farben wirksam. Durch die nachgestellte PRINT-Anweisung wird daher der Rest der Zeile schwarz gefärbt. Nach Ausgabe einer Leerzeile erscheint der Text AKTUELLE FARBEN zyan auf schwarz.
Auch bei der Anweisung PRINT AT können lokal andere Farben vorgeschrieben werden.

Format:
PRINT [*farbfestlegung*;] **AT**(*zeile*,*spalte*);*ausgebeliste*
farbfestlegung - Vorschrift zur lokalen Änderung von Vorder- und/oder Hintergrundfarbe (vgl. PRINT-Anweisung)
zeile, *spalte* - siehe Abschnitt 4.15
ausgabeliste - siehe Abschnitt 4.15

Funktion:
Für die Ausgabe der Elemente der Ausgabeliste können durch die *farbfestlegung* lokal andere Vorder- und/oder Hintergrundfarben eingestellt werden. Nach Ausführung der Anweisung PRINT AT gelten wieder die global eingestellten Farben.

Beispiel:

```
10 INK 7:PAPER 1:CLS
20 PRINT AT(17,10);"SPIELKARTENFARBEN"
30 PRINT INK2;PAPER8;AT(19,14);CHR$(201);" ";
   CHR$(203)
40 PRINT INK1;PAPER8;AT(19,20);CHR$(204);" ";
   CHR$(202)
```

Der Text SPIELKARTENFARBEN erscheint in den aktuellen globalen Farben zyan auf schwarz. In der Bildschirmzeile 19 werden die Spielkartenkennzeichen rot bzw. schwarz auf weißem Hintergrund ausgegeben. Im folgenden Bild sind noch einmal alle Ausgaben der Beispielprogramme zusammengefaßt.

4.17. Interner BASIC-Arbeitsspeicher

CLEAR	Löschen von Variablen und Festlegung von Speicherbereichen
FRE	Größe der freien Speicherbereiche bestimmen

Anhang E enthält eine Übersicht über den Arbeitsspeicher des BASIC-Interpreters. Der Platz im Arbeitsspeicher wird für die Ablage Ihres BASIC-Programms, die Ablage der numerischen Variablen und Zeichenketten-variablen sowie zur Organisation der Arbeit des BASIC-Interpreters benötigt. In der Regel brauchen Sie sich um die Aufteilung des Platzes im Arbeitsspeicher nicht zu kümmern. Wenn der Speicherplatz aber knapp wird, ist diese Aufteilung für Sie von Interesse.

Speicherplatzbedarf für Programme und Variable

BASIC-Programme
Alle eingegebenen Programmzeilen werden vorübersetzt im Arbeitsspeicher des BASIC-Interpreters abgelegt. Dabei werden alle Schlüsselwörter durch 1 Byte lange Abkürzungen ersetzt, die übrigen Zeichen mit Ausnahme der Zeilennummer und des anschließenden Leerzeichens werden übernommen. Außerdem werden mit jeder Programmzeile die 2 Bytes lange Adresse der Folgezeile und die in 2 Bytes verschlüsselte Zeilennummer abgespeichert. Am Schluß jeder Zeile steht ein Endekennzeichen (1 Byte). Bezeichnen m die durchschnittliche Länge einer vorübersetzten Programmzeile und z die Anzahl der Programmzeilen, werden etwa (5 + m) - z Bytes für die Speicherung eines BASIC-Programms benötigt.

Möglichkeiten zur Einsparung von Speicherplatz sind:
- Eingabe mehrerer durch Doppelpunkt getrennter BASIC-Anweisungen in einer Programmzeile
- Weglassen nicht erforderlicher Leerzeichen
- Kurze Variablennamen
- Kürzung der Kommentare.

Variable
Für numerische Werte werden 4 Bytes, für Zeichenketten wird Speicherplatz entsprechend ihrer Länge benötigt. Zusätzlich ist noch Speicherplatz für die Organisation des Zugriffs auf die Werte erforderlich. In der folgenden Tabelle ist der Gesamtbedarf (in Bytes) zusammengestellt.

Typ	Speicherplatz	
	in Tabelle der Variablen	im Zeichenketten-speicherbereich
numerische Variable	6	
Zeichenkettenvariable	6	l
numerisches Feld	5+2*d+4*a	
Zeichenkettenfeld	5+2*d+4*a	summe_l

Dabei bedeuten
- l - Länge der Zeichenkettenvariablen
- d - Dimension des Feldes
- a - Anzahl aller Feldelemente

summe-l - Summe der Länge aller gespeicherten Zeichenketten.

Der benötigte Zeichenkettenspeicherbereich reduziert sich, wenn Zeichenkettenvariablen oder -feldelementen im BASIC-Programm Zeichenkettenkonstanten zugewiesen werden. Felder sollten entsprechend der tatsächlich benötigten Größe dimensioniert werden, um Speicherplatz zu sparen. Beachten Sie bitte bei der Speicherplatzbedarfsermittlung, daß der BASIC-Interpreter für seine Arbeit einen freien Bereich von mindestens 70 Bytes benötigt. Sie müssen außerdem berücksichtigen, daß für die Speicherung von Zwischenergebnissen bei der Ausführung von Zeichenkettenoperationen Platz im Zeichenkettenspeicherbereich reserviert werden muß.

Löschen von Variablen und Festlegen von Speicherbereichen

Format:
CLEAR [*laenge* [,*ende*]]
laenge - ganzzahliger numerischer Ausdruck (0 bis 32767), der die Größe des Zeichenkettenspeicherbereiches in Bytes festlegt
ende - ganzzahliger numerischer Ausdruck (...32767,-32767...), der die (dezimale) Adresse des letzten Speicherplatzes des Arbeitsspeichers des BASIC-Interpreters bestimmt

Funktion:
Alle bereits vereinbarten Variablen, die Tabellen der einfachen Variablen, der Zeichenkettenvariablen und der Feldvariablen werden gelöscht. Numerische Variable erhalten den Wert 0, Zeichenkettenvariable die leere Zeichenkette (Länge Null) zugewiesen
Standardmäßig ist der Zeichenkettenspeicherbereich 256 Bytes groß. Er kann entsprechend der Größe des Ausdrucks *laenge* im Rahmen des verfügbaren Speicherplatzes vergrößert oder verkleinert werden.
Die letzte Adresse des Arbeitsspeichers des BASIC-Interpreters wird bei Neustart durch die Eingabe nach MEMORY SIZE festgelegt. Entsprechend dem Wert des Ausdrucks *ende* kann diese Adresse verändert

werden. Ist der Wert von *ende* positiv (- 32767), wird die letzte Adresse des BASIC-Arbeitsspeichers gleich diesem Wert gesetzt; ist er kleiner als Null, ergibt sich diese Adresse durch Addition von 65536 zu diesem Wert. Die zulässigen Werte des Ausdrucks *ende* hängen von der Gerätekonfiguration ab.

	kleinster *ende*-wert[1])		größter
	ROM-BASIC	RAM-BASIC	*ende*-Wert
Grundausstattung	1200	11440	16 383
1 RAM-Erweiterungs-modul	1200	11440	32 767
2 RAM-Erweiterungs-module	1200	11440	49 151(-16 385)

Hinweise:

1 Vor Ausführung des Kommandos ausgeführte DIM-Anweisungen werden wirkungslos.
2. Durch ein CLEAR-Kommando wird gleichzeitig eine RESTORE-Anweisung ausgeführt.
3. Ein CLEAR-Kommando darf nicht in einem mit GOSUB aufgerufenen Programmteil stehen.
4. Der Zeichenkettenspeicherbereich kann mit diesem Kommando bei Auftreten der Fehlermeldung OS vergrößert werden.
5. Die Speicherplätze zwischen dem - durch den Wert des Ausdruckes *ende* festgelegten - Ende des Arbeitsspeichers des BASIC-Interpreters und dem Ende des verfügbaren Schreib-Lese-Speichers (RAM) können für andere Verwendungszwecke (z. B. zur Speicherung von Maschinencodeprogrammen, die aus einem BASIC-Programm aufgerufen werden) benutzt werden.
6. Durch ein NEW-Kommando werden die Größe des Zeichen-kettenspeicherbereiches und das Ende des Arbeitsspeichers nicht verändert.

Beispiel:

```
OK
CLEAR 300,16000
```

[1]) Zeichenkettenspeicherbereich 0 Bytes, extrem kurze Programme.

Nach Ausführung dieses Kommandos ist der Zeichenkettenspeicherbereich 300 Bytes groß. Die Adresse des letzten Speicherplatzes des Arbeitsspeichers des BASIC-Interpreters ist 16000. Größe der freien Speicherbereiche bestimmen

Format:

FRE(*argument*)

argument - beliebiger numerischer Ausdruck (Wert von 0 bis 255) oder Zeichenkettenausdruck

Typ: BASIC-Funktion

Funktion:

Ist das Argument ein numerischer Ausdruck, kann aus dem Funktionswert die Größe des freien Bereiches (in Bytes) ermittelt werden. Ist das Argument ein Zeichenkettenausdruck, ergibt sich aus dem Funktionswert der freie Platz im Zeichenkettenspeicherbereich. Der Funktionswert ist eine ganze Zahl zwischen -32768 und 32767. Ist er kleiner als Null, ist die Zahl der freien Speicherplätze gleich der Summe aus 65536 und dem Funktionswert, sonst ist sie gleich dem Funktionswert.

Beispiele:

```
OK
>PRINT FRE(0)
```

Die Größe des freien Bereiches (in Bytes) wird ausgegeben.

```
100 CLEAR 256
110 A=FRE("")
120 PRINT A
130 INPUT ST$
140 PRINT FRE("")
```

Es wird von einer Größe des Zeichenkettenbereichs von 256 Bytes ausgegangen. Dieser Wert wird in Zeile 110 ermittelt und mit der Anweisung in Zeile 120 ausgegeben. Durch die INPUT-Anweisung in Zeile 130 kann eine Zeichenkette eingegeben werden, die in diesem Bereich gespeichert wird. Der dadurch verringerte freie Zeichenkettenspeicher-bereich wird anschließend angezeigt.

4.18. Externspeicherung von Programmen und Daten

CSAVE Speichern von Programmen
CSAVE* Speichern von Feldern
CLOAD Laden von Programmen
CLOAD* Laden von Feldern

Sie haben im Abschnitt 3.3 schon kennengelernt, wie Sie Anwenderprogramme, die auf Magnetbandkassette gespeichert sind, in den Heimcomputer laden können. Weitere Möglichkeiten der Arbeit mit dem Kassettengerät werden in diesem Abschnitt erläutert.

Kommandos und Anweisungen zum Speichern

Format:
CSAVE "*progname*"
progname - Name (1 bis 8 Zeichen)

Funktion:
Mit diesem Kommando wird das gesamte im Arbeitsspeicher des BASIC-Interpreters stehende Programm unter der Bezeichnung *progname* in vorübersetzter Form auf Kassette gespeichert.

Hinweise:
1. Ein mit dem CSAVE-Kommando abgespeichertes Programm können Sie mit dem CLOAD-Kommando wieder einlesen.
2. Eine weitere Möglichkeit zur Speicherung von Programmen ist im Abschnitt 5.4 beschrieben.

Beispiel:

```
OK
>CSAVE "TEST"
```

Mit diesem Kommando kann ein Programm unter dem Namen TEST gespeichert werden.
Zum Speichern von Feldern dient die folgende Anweisung.

Format:
CSAVE* "*name*";*feldname*
name - Name (1 bis 8 Zeichen)

feldname - Name eines numerischen Feldes oder eines Zeichenkettenfeldes

Funktion:
Es werden alle Elemente des Feldes *feldname* auf Kassette unter dem angegebenen Namen abgespeichert.

Hinweise:
1. Sie können nur ein Feld mit einer Anweisung abspeichern.
2. Mit der CLOAD*-Anweisung können Sie das abgespeicherte Feld wieder einlesen. Ein Beispiel ist im Anschluß an die Erläuterung der Bedienhandlungen angegeben

Bedienhandlungen beim Speichern von Programmen und Feldern

1. Legen Sie die Kassette, auf der Ihr Programm oder Ihre Daten abgespeichert werden sollen, in das Kassettengerät!
   Spulen Sie die Kassette an die Stelle, an der Ihre Aufnahme beginnen soll!
2. Befindet sich der Computer im Kommandomodus, geben Sie bitte ihr Kommando zum Speichern ein, ohne die Taste nach Abschluß ihrer Eingabe zu drücken!
3. Stellen Sie Ihr Kassettengerät auf Aufnahme! Schalten Sie, wenn möglich, die automatische Regelung der Aussteuerung Ihres Recorders ein. Starten Sie die Aufnahme!
4. Jetzt kann mit der Ausführung des Kommandos oder der Anweisung zum Speichern von Programmen bzw. Feldern begonnen werden. Dazu müssen Sie, wenn Sie das Kommando im Kommandomodus eingegeben haben, jetzt die [ENTER]-Taste drücken. Zuerst wird ein etwa fünf Sekunden langer Vorton aufgezeichnet. Er hilft Ihnen später beim Wiederauffinden Ihres Programms bzw. Ihrer Daten. Nach der Übertragung eines Datenblocks von 128 Bytes rückt der Kursor auf dem Bildschirm um eine Position vor.
5. Wenn alle Daten aufgezeichnet sind, erscheint auf dem Bildschirm die Frage

   ```
   VERIFY (Y/N)?
   ```

   (verify - überprüfen, Y(es) - ja, N(o) - nein).
   Sie müssen jetzt die Aufnahme mit Ihrem Kassettengerät beenden.

6. Wenn Sie ihre Aufzeichnung noch einmal kontrollieren wollen, drücken Sie die Taste [Y] , wenn nicht, die Taste [N].
7. Haben Sie die Frage mit [Y] beantwortet, erscheint auf dem Bildschirm die Aufforderung

   ```
   REWIND
   ```

   (rewind - zurückspulen).

   Spulen Sie die Kassette bitte bis zum Programmanfang zurück. Schalten Sie Ihr Kassettengerät auf Wiedergabe!
   Wenn Sie den Vorton hören, drücken Sie bitte die [ENTER]-Taste. Ihre Aufzeichnung wird zur Kontrolle noch einmal gelesen. Das Kontrollesen ist beendet, wenn auf dem Bildschirm die Eingabeaufforderung „>“ erscheint oder wenn die Ausführung der nächsten Anweisung in Ihrem BASIC-Programm beginnt.
8. Sie können jetzt das Kassettengerät wieder ausschalten.

Hinweise:

1. Notieren Sie bitte den Namen, unter dem Sie Ihr Programm oder Daten abgespeichert haben! Bei Feldern sollten Sie sich auch Dimension und Typ merken. Falls Ihr Kassettengerät über ein Bandzählwerk verfügt, schreiben Sie sich den Zählerstand bei Aufzeichnungsbeginn und -ende auf. Zwischen dem Ende der einen und dem Beginn der nächsten Aufzeichnung sollten Sie auf Ihrer Kassette ausreichend Platz freihalten. Das erleichtert Ihnen später das Wiederauffinden Ihrer Programme und Daten.
2. Tritt beim Kontrollesen Ihres aufgezeichneten Programms oder Feldes ein Fehler auf, so bewegt sich der Kursor auf dem Bildschirm nicht mehr weiter. Sie können dann das Kontrollesen fortsetzen, wenn Sie so reagieren, wie es bei der Fehlermeldung

   ```
   BOS-error: bad record
   ```

   im Anhang H beschrieben ist. Gegebenenfalls ist die Aufzeichnung zu wiederholen.

Die beiden folgenden Beispiele zeigen, wie aus Programm- und Kommandomodus ein Feld bzw. ein Programm abgespeichert werden kann.

Beispiele:

```
10 DIM A(50)
20 FOR I =0 TO 50
30 A(I)=1:PRINT A(I),
40 NEXT I:PRINT
50 PRINT "1. BAND POSITIONIEREN! "
60 PRINT "2. AUFNAHMETASTEN DRUECKEN! "
70 PRINT "3. CONT-TASTE DRUECKEN! "
80 PAUSE
90 CSAVE* "DATEN";A
100 PRINT
110 PRINT "KASSETTENGERAET AUSSCHALTEN! "
```

Das Feld A wird mit den Werten 0 bis 50 belegt. Die CSAVE*- Anweisung wird erst ausgeführt, wenn das Programm durch Drücken der [CONT]-Taste nach der PAUSE-Anweisung fortgesetzt wird.
Zur Übung sollten Sie dieses Programm unter dem Namen TEST aus dem Kommandomodus abspeichern.

```
OK
>CSAVE "TEST"  [ENTER]

               nach Betätigung der Aufnahmetasten drücken!

VERIFY (Y/N)   [Y] REWIND

               [ENTER]
               nach Bandpositionierung und Betätigung der
               Wiedergabetaste drücken!
```

Kommandos und Anweisungen zum Laden

Format:
CLOAD "*progname*"
progname - Name (1 bis 8 Zeichen)

Funktion:
Ein mit dem CSAVE-Kommando abgespeichertes Programm kann durch das CLOAD-Kommando in den Arbeitsspeicher des BASIC-Interpreters geladen werden.

Hinweise:

1. Vor dem Laden eines Programms ist normalerweise der Programmspeicher mit einem NEW-Kommando zu löschen.
2. Ohne vorheriges NEW können mit CLOAD-Kommandos nacheinander weitere Programmteile in den Programmspeicher geladen werden. Bereits im Programmspeicher stehende Programmteile müssen dabei kleinere Zeilennummern haben als die noch zu ladenden Programmteile.

Beispiel:

```
OK
CLOAD "TEST"
```

Abgespeicherte Felder können mit der folgenden Anweisung eingelesen werden.

Format:

CLOAD* "*name*";*feldname*

name - Name (1 bis 8 Zeichen)

feldname - Name eines numerischen Feldes oder eines Zeichenkettenfeldes

Funktion:

Die Werte eines mit einer CSAVE*-Anweisung abgespeicherten Feldes können mit dieser Anweisung in das Feld f*eldname* eingelesen werden. Der *feldname*, unter dem die Daten abgespeichert sind, muß mit dem in der CLOAD*-Anweisung nicht übereinstimmen. Typ und Dimensionierung des abgespeicherten Feldes und des Feldes, in das gelesen werden soll, müssen gleich sein

Hinweise:

1. Beim Einlesen von Zeichenkettenfeldern ist darauf zu achten, daß der benötigte Zeichenkettenspeicherbereich ausreichend groß festgelegt ist. Er ist gegebenenfalls mit einem CLEAR-Kommando abweichend von der Standardgröße (256 Bytes) einzurichten.
2. Das Laden von Feldern ist in der Regel nur im Programmodus sinnvoll.

Bedienhandlungen beim Laden von Programmen und Feldern

Die Bedienhandlungen beim Laden von Programmen sind bereits im Abschnitt 3.3 erläutert worden. Für das Laden von Feldern muß die Kassette vor Ausführung der CLOAD*-Anweisung positioniert und das Kassettengerät auf Wiedergabe geschaltet sein.
Das folgende Beispiel demonstriert die Anwendung der CLOAD*-Anweisung.

Beispiel:

```
10 DIM B(50)
20 PRINT " 1 . BAND POSITIONIEREN! "
30 PRINT " 2. WIEDEGABETASTE DRUECKEN! "
40 INPUT " 3. WENN VORTON BEGINNT, ENTER-TASTE
DRUECKEN! ";E$
50 CLOAD* "DATEN";B
60 PRINT
70 PRINT "KASSETTENGERAET AUSSCHALTEN!"
80 FOR I=0 TO 50
90 PRINT B(I),
100 NEXT I
```

Das im letzten Beispiel unter dem Namen DATEN abgespeicherte Feld soll gelesen werden. Das Feld B, in das eingelesen werden soll, wird dazu entsprechend der Größe des abgespeicherten Feldes dimensioniert. Durch die INPUT-Anweisung in Zeile 40 wird die Programmausführung so lange unterbrochen, bis die [ENTER]-Taste gedrückt wird. Danach beginnt der Computer, die CLOAD*-Anweisung auszuführen.

4.19. Testunterstützung

TRON Einschalten der Ablaufverfolgung (Tracemodus)

TROFF Ausschalten der Ablaufverfolgung

Im Tracemodus wird Ihnen zusätzlich zu den übrigen Ausgaben angezeigt, in welcher Reihenfolge die Zeilen Ihres BASIC-Programms ausgeführt werden. Die Zeilennummern erscheinen in spitzen Klammern < > eingeschlossen auf dem Bildschirm. Dadurch können Sie Sprünge und Verzweigungen bei der Ausführung Ihres Programms verfolgen. Der Tracemodus hilft Ihnen in der Testphase, logische Fehler in Ihrem Programm zu erkennen. Die beiden angegebenen Kommandos können auch sinnvoll im Programmodus verwendet werden. Sie ermöglichen Ihnen dann, den Tracemodus nur bei der Abarbeitung bestimmter kritischer Programmteile einzuschalten.

Format: **TRON**

Funktion:
Durch das Kommando wird die dynamische Ablaufverfolgung (Tracemodus) eingeschaltet.

Format: **TROFF**

Funktion:
Der Tracemodus wird mit diesem Kommando ausgeschaltet.

4.20. Fehlermeldungen

Haben Sie bei der Eingabe eines Kommandos oder einer Anweisung durch die Verwendung eines falschen Formats (z. B. weil ein Zeichen vergessen wurde) einen Fehler gemacht, wird Ihnen dieser angezeigt, und der BASIC-Interpreter kehrt in den Kommandomodus zurück. Angezeigt wird auch, wenn der BASIC-Interpreter eine formal richtige Anweisung nicht ausführen kann (z. B. weil zu wenig Speicherplatz vorhanden ist). Die Fehlermeldungen des BASIC-Interpreters sind im Anhang H zu-sammengestellt. Ist der Fehler in einer Anweisung eines Programms bemerkt worden, können Sie die fehlerhafte Anweisung korrigieren und das Programm neu starten. Es ist aber auch möglich, durch Eingabe von

GOTO *zeilennummer*

an einer anderen sinnvollen Stelle (siehe auch Abschnitt 4.5) fortzufahren. Beachten Sie in diesem Fall jedoch, daß der Zusammenhang aller GOSUB ... RETURN- oder FOR ... NEXT-Anweisungen erst bei Neustart des Programms wiederhergestellt wird.

Im Anhang H finden Sie außerdem eine Zusammenstellung der Fehlermeldungen des Betriebssystems und der Fehlermeldungen bei der Kassettenarbeit. Alle Meldungen und die dazugehörigen Erläuterungen sind so abgefaßt, daß sie Ihnen helfen, schnell die Fehlerursache zu erkennen und zu beseitigen. Falls Ihnen trotzdem die Fehlersuche nicht klar sein sollte, lesen Sie bitte noch einmal die Abschnitte dieses Handbuches durch, in denen die betreffende Anweisung erläutert wird.

5. Fortgeschrittene BASIC-Programmierung

5. 1. Direkter Speicherzugriff

PEEK Lesen von 1 Byte
POKE Schreiben von 1 Byte
DEEK Lesen von 2 Bytes
DOKE Schreiben von 2 Bytes

PEEK, POKE und DEEK, DOKE sind Anweisungen, mit deren Hilfe Sie direkt auf den Speicher des "robotron Z 9001" zugreifen können. POKE und DOKE dienen dem Schreiben auf Speicherplätze, PEEK und DEEK sind Aufrufe von Standardfunktionen und dienen dem Lesen von Speicherplätzen.

Schreiben auf Speicherplätze

Format:
POKE *adresse*,*byte*
adresse - numerischer Ausdruck (ganzzahlig) mit einem Wert von -32768 bis 32767
byte - numerischer Ausdruck (ganzzahlig) mit einem Wert von 0 bis 255

Funktion:
Mit dieser Anweisung wird ein Byte (8 Bits) auf den Speicherplatz mit der angegebenen Adresse geschrieben.

Format:
DOKE *adresse*,*wort*
adresse - numerischer Ausdruck (ganzzahlig) mit einem Wert von -32768 bis 32767
wort - numerischer Ausdruck (ganzzahlig) mit einem Wert von -32768 bis 32767

Funktion:
Hiermit werden zwei Bytes auf die Speicherplätze *adresse* und *adresse*+1 (niederwertiges und höherwertiges Byte) geschrieben.

Hinweise:
1. Mit beiden Anweisungen können Sie auch auf den Bild- und Farbspeicher zugreifen.
2. Beide Anweisungen werden benötigt, wenn Sie vom BASIC- Programm aus Unterprogramme in der Maschinensprache generieren (erzeugen) wollen.

Lesen von Speicherplätzen

Format:
PEEK(*adresse*)
adresse - numerischer Ausdruck (ganzzahlig) mit einem Wert von -32768 bis 32767

Typ: BASIC-Funktion

Funktion:
Der Inhalt des Speicherplatzes mit der angegebenen Adresse wird als Funktionswert angenommen (Wert zwischen 0 und 255).

Format:
DEEK(*adresse*)
adresse - numerischer Ausdruck (ganzzahlig) mit einem Wert von -32768 bis 32767

Typ: BASIC-Funktion

Funktion:
Der Inhalt der Speicherplätze *adresse* und *adresse*+1 wird als Funktionswert angenommen (Wert zwischen, -32768 und 32767).

Hinweis:
Sind in den vorhergehenden Anweisungen und Funktionen die Werte von *adresse* und *wort* negativ, so werden sie vom BASIC-Interpreter als 65536+*wert* interpretiert. Damit ist dann ein Zugriff auf den gesamten Speicher des "robotron Z 9001" möglich; mit -32766 wird also der Speicherplatz 32770 angesprochen. Über die Funktion PEEK besteht auch die Möglichkeit, wichtige Adressen des Betriebssystems des "robotron Z 9001" zu lesen, wie z. B. die der Systemuhr. Außer der Übernahme der Zeit ins BASIC-Programm kann auf diesem Weg unter anderem der Zufallszahlengenerator „zufällig“ initialisiert werden.

Beispiel:
In folgendem Programm wird bei jedem Programmstart eine neue Zufallszahlenreihe begonnen.

```
10 !Initialisierung durch Zugriff auf Sekundenz
   aehler
20 Z=RND(-PEEK(31)-1)
30 !Ausgabe einer Zufallszahl
40 PRINT RND(1)
```

5.2. Zugriff zu Bild- und Farbspeicher

Der Bildspeicher des "robotron Z 9001" beginnt auf der Adresse 60416 (0EC00H), der Farbspeicher (bei Vorhandensein des Farbmoduls) auf Adresse 59392 (0E800H). Die Adressen für die einzelnen Bildschirmpositionen entnehmen Sie bitte dem Anhang C. Um auf Zeile 10, Spalte 10, des Bildschirms zu schreiben, könnten folgende Anweisungen verwendet werden (Zählung beginnt bei Null!):

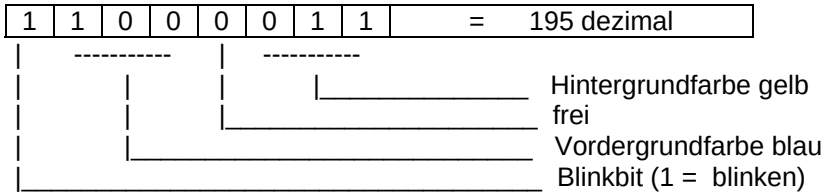
```
POKE -(65536-60826),ASC("a")

POKE -4710,97
```

Beide Anweisungen schreiben in die angegebene Position den Kleinbuchstaben a. Die Adresse der Bildschirmposition kann mit der Tabelle im Anhang C folgendermaßen bestimmt werden:

Zeilenanfang:	60816	bzw.	-4720
+ Spalte	+ 10		+ 10
Bildschirm-position	60826		-4710
Verwendung in Anweisung	-(65536-60826)		-4710

Das Zeichen a wird auf diese Weise in der für diese Position ein gestellten Vorder- und Hintergrundfarbe ausgegeben.

Sie können nun durch Setzen des zu Zeile 10 und Spalte 10 gehörenden Bytes des Farbspeichers die Farben, mit denen a ausgegeben wird, beeinflussen. Der Farbmodul des "robotron Z 9001" bietet ihnen aber noch mehr Möglichkeiten. Sie können das Zeichen auch blinken lassen. Dabei werden hardwaremäßig Vorder- und Hintergrundfarbe ständig vertauscht.
Wollen Sie durch POKE-Anweisungen einzelne Zeichen blinken lassen, so muß die interne Codierung im Farbattributspeicher beachtet werden:



Wichtig ist dabei, daß die Codierung der Farben intern mit *farbcode*-1 erfolgt (*farbcode* entsprechend Tabelle in Abschnitt 4.16.), schwarz also mit Null usw.
Mit den Anweisungen

```
POKE -4710,97
POKE -5734,195
```

blinkt also das Zeichen a auf Zeile 10, Spalte 10, des Bildschirmes blau auf gelbem Hintergrund.

Auch mit den Steuerzeichen des Betriebssystems können Sie den Farbspeicher innerhalb der PRINT-Anweisung beeinflussen (siehe auch Abschnitt 7.4. und Anhang A). Besonders zweckmäßig sind hier die Zeichen CHR$(6) und CHR$(22).
Alle nach CHR$(6) ausgegebenen Zeichen erscheinen blinkend auf dem Bildschirm (für sie wird das Blinkbit im Farbspeicher gesetzt). Nach nochmaliger Ausgabe von CHR$(6) werden alle danach ausgegebenen Zeichen wieder normal dargestellt. Analoges gilt für CHR$(22), die Zeichen werden dann invers (mit vertauschten Vorder- und Hintergrundfarben) ausgegeben.

Beispiel:

```
10 INK3:PAPER1
20 NO$="NORMAL"
30 BL$="BLINKEND"
40 IN$="INVERS"
50 PRINT NO$;CHR$(22);IN$;CHR$(22)
60 PRINT NO$;CHR$(6);BL$;CHR$(6)
70 PRINT CHR$(6);NO$;BL$;CHR$(22);IN$;BL$;
   CHR$(22);CHR$(6)
OK
>RUN
NORMALINVERS
NORMALBLINKEND
NORMALBLINKENDINVERSBLINKEND
```

Die Steuerzeichen wirken nur in der PRINT-Anweisung nicht in der Anweisung PRINT AT. Um dort die Möglichkeiten zu nutzen, müßten Sie programmieren:

```
10 PRINT CHR$(22)
20 PRINT AT(3,10; "INVERS"
30 PRINT CHR$(22)
40 PRINT AT(4,10); "NORMAL"
```

Zur Farbdarstellung können Sie die Steuerzeichen ebenfalls benutzen.
Der nach CHR$(20) ausgegebene Farbcode bestimmt die Vordergrundfarbe der nachfolgenden Zeichen, der nach CHR$(21) die Hintergrundfarbe.
Die Farben werden wieder mit *farbcode*-1 angegeben, schwarz also wieder mit Null.

Mit der Anwendung dieser Steuerzeichen können Sie innerhalb einer PRINT-Anweisung eine Farbum- und rückschaltung erreichen.

Beispiel:

```
> LIST
10 SW$=CHR$(0):RO$=CHR$(1)
20 GR$=CHR$(2):GE$=CHR$(3)
30 VO$=CHR$(20):HI$=CHR$(21)
40 PRINT VO$;GR$;HI$;SW$
50 PRINT "Sie koennen auch ";VO$;RO$;
60 PRINT HI$;GE$;"rot auf gelb";
70 PRINT VO$;GR$;HI$;SW$;"schreiben. "
OK
>RUN

Sie koennen auch rot auf gelb schreiben.
OK
```

5.3. Aufruf von Maschinencodeprogrammen

CALL, **CALL*** Unterprogramme ohne Parameterübergabe
USR(X) Unterprogramme mit Parameterübergabe

Maschinencodeprogramme können vom BASIC-Interpreter aus auf zwei verschiedene Arten aufgerufen werden. Soll keine Parameterübergabe erfolgen, werden CALL bzw. CALL* verwendet, andernfalls wird USR(X) benutzt.

CALL-Unterprogramme

Format:
CALL *adresse* (dezimal)
CALL* *adresse* (hexadezimal)

Funktion:
Das Maschinencodeprogramm mit der angegebenen Startadresse wird aufgerufen.

Hinweise:

1. Die Startadresse kann entweder dezimal oder (bei CALL*) hexadezimal angegeben werden. Das Programm darf die CPU-Register nicht verändern und muß mit einer Return-Anweisung (z. B. 0C9H) enden. Wird *adresse* dezimal angegeben, so muß ihr Wert zwischen -32768 und 32767 liegen.
2. Maschinencodeprogramme lassen sich z. B. dann vorteilhaft verwenden, wenn schnelle Bewegungen auf dem Bildschirm dargestellt werden sollen.
3. Bei CALL* wird Adresse immer als Konstante angegeben; bei CALL ist ein numerischer Ausdruck zugelassen.

Beispiel:
Durch das folgende Programm wird zweimal die gleiche Folge von Bewegungen auf dem Bildschirm erzeugt, einmal über ausschließlich BASIC-Anweisungen (Zeilen 10 bis 70) und zum anderen in den Zeilen 90 bis 120 über einen Aufruf eines Maschinencodeprogramms und die Einstellung der Geschwindigkeit über BASIC-Anweisungen. Das Maschinencodeprogramm muß ab Adresse 600 im Speicher stehen. (Wie Maschinencodeprogramme im Speicher bereitgestellt werden können, erfahren Sie im Abschnitt 8.)

```
10 FOR I=0 TO 100 STEP 15
20 FOR AD=-420 TO -4201
30 FOR J=0 TO I:NEXT
40 POKE AD,62
50 POKE AD-1,32
60 NEXT AD
70 NEXT I
80 !
90 FOR I=5 TO 100 STEP 15
100 POKE 611,I
110 CALL 600
120 NEXT I
```

Das aufgerufene Maschinencodeprogramm besteht aus folgenden Anweisungen (ab Adresse 600):

```
  Anweisungen          hexadezimale     dezimale
                       Codierung        Codierung
-------------------------------------------------------
     PUSH  HL          E5               229
     PUSH  BC          C5               197
     PUSH  DE          D5               213
     LD    HL,0EF70    21   70   EF     33  112  239
           H
     LD    (HL),62     36   3E          54   62
     LD    B,39        06   27           6   39
M0:  LD    C,0         0E   00          14    0
M1:  LD    D,0         16   00          22    0
M2:  DEC   D           15               21
     JR    NZ,M2       20   FD          32  253
     DEC   C           0D               13
     JR    NZ,M1       20   F8          32  248
     LD    (HL),20H    36   20          54   32
     INC   HL          23               35
     LD    (HL),62     36   3E          54   62
     DJNZ  M0          10   EF          16  239
     POP   DE          D1              209
     POP   BC          C1              193
     POP   HL          E1              225
     RET               C9              201
```

Die Bewegungsabläufe bei beiden Varianten sehen nahezu gleich aus. Müssen nun aber zwischen den Einzelbewegungen noch Abfragen bezüglich möglicher Bewegungsgrenzen, Richtungsänderungen oder Karambolagen bei Beibehaltung der hohen Geschwindigkeiten durchgeführt werden, so ist die Anwendung von Maschinencode-programmen nahezu unumgänglich.

Aufruf wo. USR(X)

Format: **USR(X)**

Typ: BASIC-Funktion

Funktion:
Mit diesem Funktionsaufruf wird ein Maschinencodeprogramm gestartet, in das der Parameter x übernommen wird und. das einen Funktionswert an das BASIC-Programm zurückgibt.
Die Werte der Parameter müssen wieder im Bereich von -32768 bis 32767 liegen. Vor dem Aufruf der Funktion ist die Startadresse des zugehörigen Maschinencodeprogrammes in zwei Systemzellen des BASIC-Interpreters (WSP+4, WSP+5) einzutragen. Das geschieht günstig über eine DOKE-Anweisung.
Die Übernahme des Parameters erfolgt, indem aus dem Maschinencodeprogramm die Routine (Programmteil) EPRVL3 des BASIC-Interpreters aufgerufen wird (CALL EPRVL3). Der Wert des Parameters steht dann im Registerpaar DE, er muß im Bereich von -32768 bis 32767 liegen. Die Übermittlung des Rückgabewertes an das BASIC-Programm erfolgt durch Aufruf der Routine FRE3 (JP FRE3); er muß vorher in den Registern A und 8 (B niederwertiger Teil) wieder als vorzeichenbehaftete Intgerzahl (ganze Zahl) bereitgestellt werden.
Das Maschinencodeprogramm darf andere CPU-Register nicht verändern.

Adressen:

	RAM-BASIC	ROM-BASIC
WSP+4	2B04H(11012)	304H (772)
EPRVL3	0C6FH	0C96FH
FRE3	13B1H	OD0B1H

Hinweis:
Neben der schnellen Ausführung einfacher Operationen können Sie diese Funktion vor allem dann günstig anwenden, wenn Sie die Unterprogramme des Betriebssystems nutzen wollen.

Beispiel:
Das folgende Beispiel dient der Ermittlung der aktuellen Kursorposition im Bildspeicher.

```
10 DOKE 772,600
20 NR=17
30 KF=USR(NR)
40 PRINT KP
```

Anweisungen		hexadezimale Codierung			dezimale Codierung		
CALL	EPRVL3	CD	6F	C9	205	111	201
LD	C,E	4B			75		
CALL	5	CD	05	00	205	5	0
LD	A,B	78			120		
LD	B,C	41			65		
JP	FRE3	C3	B1	D0	195	177	208

5.4. Datentransfer

LIST*, **LOAD*** Aus-, Eingabe im ASCII-Code
NULL Anzahl der Dummyzeichen (bedeutungslosen Zeichen) festlegen
WIDTH Zeilenlänge festlegen
PRINT CHR$(16) Ausgaben auf einen Drucker
PRINT CHR$(14)

LIST# und LOAD# sind Kommandos, die ähnlich CSAVE und CLOAD zum Übertragen von Programmen auf Kassette und von dort in den Rechner benutzt werden können.

Ausgabe von Programmen im ASCII-Code

Format:
LIST#*gerät*"*name*"[*zeilennummer*]
gerät - Parameter, der das externe Gerät spezifiziert 0 - Bildschirm 1 - Kassette
name - Name, den das Programm auf der Kassette erhält (vgl. CSAVE, Abschnitt 4.18.)
zeilennummer - die Ausgabe erfolgt ab der angegebenen Zeilennummer oder, wenn sie nicht angegeben ist, ab Programmanfang

Funktion:
Das Kommando dient der Ausgabe des Programmtextes auf das angegebene Gerät im externen Code (ASCII).

Hinweise:
1. Im Gegensatz zu CSAVE, bei dem das Programm im internen Code ausgelagert wird, wird bei LIST# der vollständige Quelltext ausgegeben (mit den durch das Kommando NULL festgelegten Dummyzeichen und der durch WIDTH eingestellten Zeilenlänge). Für LIST# gelten nicht die mit LINES getroffenen Festlegungen; es wird intern LINES 65535 gestellt.
2. Wird die Programmübertragung nach dem LIST#-Kommando vorzeitig durch einen Fehler (z. B. IO-Error) beendet, so hat der LINES-Parameter den Wert 65535. Er muß durch eine LINES-Anweisung wieder zurückgesetzt werden.

Eingabe von Programmen im ASCII-Code

Format:
LOAD#*gerät*"*name*"
gerät - Parameter, der das externe Gerät spezifiziert (entsprechend LIST#)
name - Name der angesprochenen Datei

Funktion:
Das Kommando dient dem Laden von LIST*-gespeicherten BASIC-Programmtexten.
Hinweis:
Der wesentliche Unterschied zum Kommando CLOAD ist der, daß die einzelnen Programmzeilen in den Zwischencode gewandelt und entsprechend ihrer Numerierung (wie bei Eingabe über die Tastatur) in das schon im Speicher befindliche Programm einsortiert werden; bei CLOAD werden sie an das im Speicher stehende Programm angehängt.

Spezielle Kommandos

Format:
NULL *anzahl*
anzahl - Anzahl der am Zeilenende auszugebenden Dummyzeichen von 0 bis 255 (Standardwert: 10)

Funktion:
Mit dem Kommando wird die Anzahl der an jedem Zeilenende auszugebenden bedeutungslosen Zeichen (Dummyzeichen) festgelegt.

Hinweis:
Mit Hilfe des Kommandos NULL ist es möglich, die Arbeit von peripheren Geräten (z. B. Kassette) mit der des Rechners zu koordinieren. Bedeutsam ist das beim Kommando LOAD#, da dort unter Umständen die Zeit für das Einsortieren einer eingegebenen Zeile nicht ausreicht. Es tritt dann der "BOS-error: record not found" auf.
Die einzusortierende Datei ist dann mit einer größeren Anzahl von Dummyzeichen neu aufzuzeichnen; der Fehler tritt ab einer genügend großen Anzahl nicht mehr auf.

Format:
WIDTH *zeilenlänge*
zeilenlänge Anzahl der Zeichen, nach denen der Interpreter eine Zeilenschaltung bei der Ausgabe einfügt (0 bis 255), Standardwert: 255

Funktion:
Mit dem Kommando wird die maximale Zeilenlänge festgelegt, mit der Ausgaben auf den Bildschirm und andere externe Geräte erfolgen.

Hinweise:
1. WIDTH kann verwendet werden, um die Ausgabe bei den LIST-Kommandos und PRINT-Anweisungen zu beeinflussen.
   Es wirkt nicht bei der Eingabe und Änderung von Quelltexten, z. B. im EDIT-Modus.
2. WIDTH wird zweckmäßigerweise verwendet, um die Zeilenlänge bei Ausgabe über den Drucker festzulegen.

Ausgaben über einen Drucker

Um Programme oder Daten über einen Drucker auszugeben, müssen Sie den entsprechenden Druckermodul gesteckt haben. Nach dem Aktivieren des Druckertreibers (siehe Abschnitt 7.3.) wird der Drucker durch die Sonderfunktion CHR$(16), d. h. durch

```
PRINT CHR$(16)
```


oder durch Drücken der Tasten [CONTR][N] als Ausgabegerät zugeschaltet. Dann werden alle Zeichen, die auf dem Bildschirm ausgegeben werden, zusätzlich über den Drucker ausgegeben. Davon ausgenommen sind alle "PRINT AT"- und "POKE"-Ausgaben.
Durch nochmaliges Ausführen dieser Sonderfunktion wird die parallele Ausgabe über den Drucker wieder abgeschaltet.
Wenn der Drucker zugeschaltet ist, können Sie mit Hilfe der Sonderfunktion CHR$(14) eine Hardcopy des augenblicklichen Bildschirmbildes erhalten, d. h., nach

```
PRINT CHR$(14)
```

oder nach Drücken der Tasten [CONTR][N] wird der gesamte Bildschirminhalt zeilenweise über den Drucker ausgegeben.

5.5. Ein- und Ausgabe über Nutzerschnittstelle

INP	Eingabe über externen Kanal
OUT	Ausgabe über externen Kanal
WAIT	Warten auf Ereignis

Mit Hilfe der Anweisungen INP, OUT und WAIT können Sie auf die externen Kanäle des "robotron Z 9001" zugreifen und so eine Meßwerterfassung und Verarbeitung vornehmen oder auch Steuerungsaufgaben lösen.
Im Grundgerät stehen ihnen dafür über die Ein-/Ausgabebuchse für spezielle Anwendungen
PIO 1, Kanal B, Adresse 137, und
CTC, Kanal 1, Adresse 129,
zur Verfügung. Weitere Möglichkeiten erhalten Sie mit dem E/A-Erweiterungsmodul.

Anweisungen zur Nutzung der Kanäle

Format:
INP(*kanaladresse*)
kanaladresse - numerischer Ausdruck (ganzzahlig) mit einem Wert von 0 bis 255

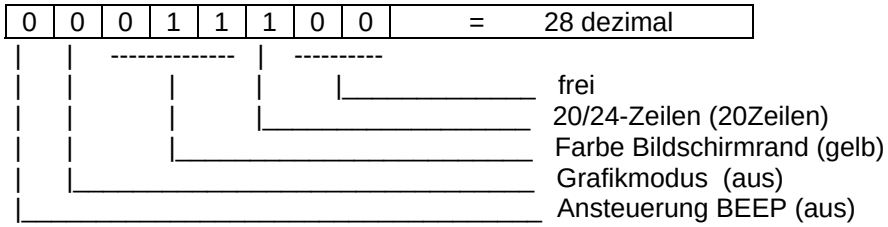
Typ: BASIC-Funktion

Funktion:
Es wird ein Byte vom E/A-Kanal mit der angegebenen Kanaladresse gelesen.

Format:
OUT *kanaldresse*, *byte*
kanaladresse - numerischer Ausdruck (ganzzahlig) mit einem Wert von 0 bis 255
byte - numerischer Ausdruck (ganzzahlig) mit einem Wert von 0 bis 255

Funktion:
Diese Anweisung realisiert die Ausgabe von einem Byte an die angegebene Kanaladresse.

Hinweis:
Über die PIO 1, Kanal A, Adresse 136, sind der Farbcode für den Bildschirmrand, der 20/24-Zeilen-Modus und die Ansteuerung von Grafikmodus und Summerton (BEEP) codiert.



Über die BORDER-Anweisung können die Bits 3 bis 5 gesetzt werden. Alle anderen Bits werden zurückgesetzt. Eine Umschaltung in den 20-Zeilen-Modus ist aber nur über eine entsprechende OUT-Anweisung realisierbar.

Beispiel:

```
OUT 136,28
```


stellt einen gelben Bildschirmrand und den 20-Zeilen-Modus ein (vgl. auch Abschnitt 4.15).

Format:
WAIT *kanaladresse*, *ausdruck* [,*ausdruck*]
kanaladresse - numerischer Ausdruck (ganzzahlig) mit einem Wert von 0 bis 255
ausdruck - numerischer Ausdruck (ganzzahlig) mit einem Wert von 0 bis 255

Funktion:
Diese Anweisung hält die Programmausführung an, bis an der angegebenen Kanaladresse ein spezieller Wert anliegt.
Der augenblicklich an der Kanaladresse anliegende Wert wird Exklusiv-ODER-verknüpft mit dem zweiten Ausdruck der Anweisung, das Ergebnis wird UND-verknüpft mit dem ersten Ausdruck. Eine Programmfortsetzung erfolgt, wenn das Ergebnis ungleich Null ist. Fehlt der zweite Ausdruck, so wird sein Wert mit Null angenommen.

Hinweis:
Mit WAIT kann das Warten auf ein bestimmtes externes Ereignis programmiert werden.

Programmierung der Kanäle
Die Nutzung der PIO-Kanäle und des CTC-Kanals als Zähler setzt gründliche Kenntnisse in der Hard- und Software voraus, da in diesen Fällen von Ihnen selbst gebaute Schaltungen an die Kanäle angeschlossen werden. Dabei müssen die entsprechenden Anschlußbedingungen unbedingt eingehalten werden. Die Nutzung des freien CTC-Kanals als Zeitgeber ist weniger kritisch, da dann kein Hardware-Anschluß vorgenommen wird.
Der Kanal B der PIO 1, Adresse 137, ist in der üblichen Weise in den Betriebsarten Byte-Ausgabe, Byte-Eingabe und Bit-Ein/Ausgabe verwendbar. Die Steuerwortadresse zur Programmierung des Kanals ist 139. Der PIO-Kanal ist natürlich auch im Interruptbetrieb verwendbar. In diesem Fall sind über entsprechende Steuerworte noch das Maskierungsregister und das Interruptvektorregister des Kanals zu programmieren.

Den Aufbau der Steuerworte, die Bedeutung der einzelnen Bits in ihnen und ihre notwendige Reihenfolge bei der Programmierung des PIO-Kanals entnehmen Sie bitte der einschlägigen Fachliteratur. Entsprechendes gilt für die Programmierung und Nutzung des freien CTC-Kanals.

Hinweis:
Die CPU des "robotron Z 9001" arbeitet im Interruptmodus 2, das I-Register ist mit 2 geladen.
Das Interruptvektorregister der CTC ist mit Null geladen, so daß die mögliche Startadresse für ein Interruptbehandlungsprogramm des Kanals 1 auf 514/515 (202H/203H) zu schreiben ist.
Die Adressen 520 bis 523 sind durch die PIO 2 belegt, PIO 1, Kanal A, arbeitet nur im Ausgabebetrieb, so daß die Startadressen der Interrupt-behandlungsprogramme für PIO 1, Kanal B, und die PIOs des E/A-Erweiterungsmoduls ab Adresse 524 (20CH) eingetragen werden können.

Nutzung des Kassetteninterface zur Tonerzeugung

Das Kassetteninterface des "robotron 9001" ist unter Nutzung des als Zeitgeber arbeitenden Kanals 0 der CTC realisiert (Adresse 128). Durch „zweckentfremdete“ Verwendung dieses Kanals lassen sich auch andere Töne als die bei der Programm- und Datenaufzeichnung entstehenden erzeugen.
Zu diesem Zweck sind nur die Zeitkonstante des Kanals mit einem entsprechenden Wert zu laden und der Zeitgeber zu starten. Je nachdem, mit welchem Vorteiler der Zeitgeber betrieben wird, lassen sich dadurch höhere oder niedere Töne erzeugen.
Ein einfaches Programm zum Erzeugen einer Melodie finden Sie im Abschnitt 6. Das nachfolgende Beispiel dient nur der Illustration der einzelnen Programmschritte zur Tonerzeugung.

Beispiel:

```
10 !Eingabe Zeitkonstante(0...255),
          Vorteiler          (0/1)
20 INPUT "Zeitkonstante, Vorteiler: ";ZK,VT
30 VT=32*VT
40 !Betriebsart CTC programmieren
50 OUT 128,ZK
60 !Zeitkonstante laden
70 !Zeitgeber starten
80 OUT 128,ZK
90 PAUSE
100 !Kanal abschalten
110 OUT 128,VT+3
120 GOTO 10
```

Wird im Beispiel die Pause Mit [CONT] abgebrochen, so endet die Tonausgabe mit dem Abschalten des Kanals.
Die erzeugten Töne sind so über die Mithörkontrolle Ihres Kassettenrecorders hörbar. Schalten Sie dazu den Recorder mit gedrückter [PAUSE]-Taste auf Aufnahme.
Mit den zusätzlichen Anweisungen

```
5 !Ansteuerung BEEP ein
6 OUT 136,128

115 !Ansteuerung BEEP aus
116 OUT 136,0
```

wird der eingebaute Summer parallel zur Ausgabe auf das Kassettengerät ein- und wieder ausgeschaltet, so daß die erzeugten Töne auf diesem Weg auch ohne Kassettengerät hörbar sind.

Hinweis:
Die Belegung des PIO-Kanals 136 im Grundzustand (nach [RESET]) ist Null. Der beim Betätigen der Taste [CONTR] [G] entstehende Ton ist über den Summer nur bei der Belegung des Bit 7 mit Null hörbar.

6. Hinweise und Beispiele zur BASIC-Programmierung

6. 1. Schrittweises Vorgehen

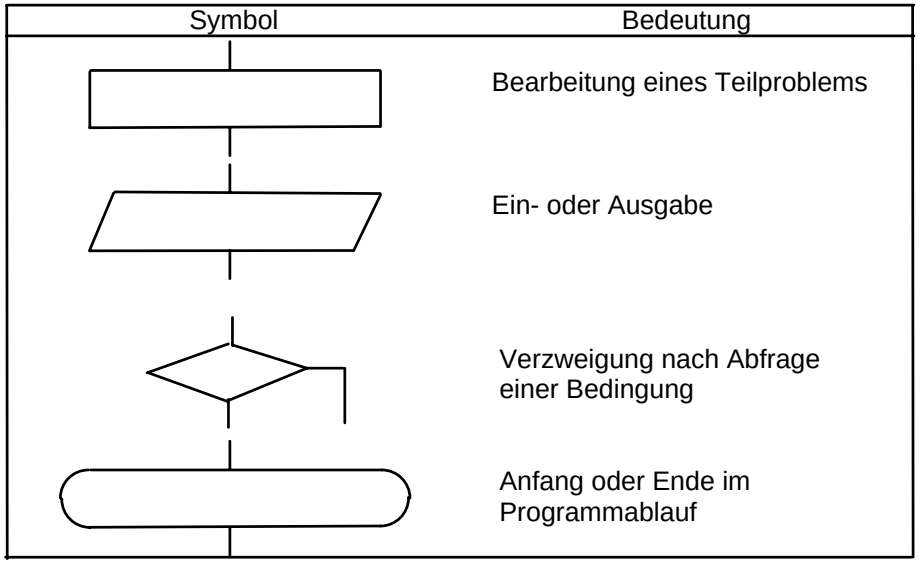
Sie haben jetzt eine Reihe von Anweisungen und Kommandos kennengelernt, mit denen Sie Ihrem Heimcomputer kleine Aufträge zur Ausführung übergeben können. Mit Folgen von solchen Anweisungen - Programmen -, die in den Heimcomputer eingegeben und dort gespeichert werden, lassen sich umfangreiche Aufgaben lösen. Kleinere Programme sind Ihnen schon in den Demonstrationsbeispielen im Abschnitt 4. vorgestellt worden. Sicher werden Sie beabsichtigen, auch für größere Probleme selbst eigene Programme zu schreiben. Wie man schrittweise zu einem Programm kommt, soll im folgenden gezeigt werden. Das Anliegen ist dabei, Ihnen zu helfen, Ihr ursprüngliches Problem in eine Reihe von besser beherrschbaren Problemen zu zerlegen.

Formulierung der Aufgabenstellung
Der Heimcomputer führt Ihre Anordnungen gewissenhaft aus. Sie sind sozusagen sein Chef. Aber das, was ihr Chef von ihnen verlangt, nämlich, daß Sie bei der Erfüllung einer übertragenen Aufgabe mitdenken und eigene Initiativen zu ihrer Lösung entfalten, können Sie von ihm nicht erwarten. Bevor Sie mit dem Programmieren beginnen, sollten Sie deshalb darüber nachdenken, was Ihr Heimcomputer machen soll.
Überlegen Sie sich, welche Größen, numerische Werte oder Folgen von Zeichen Sie zu Beginn ihrem Heimcomputer mitteilen müssen, damit er die gewünschten Ergebnisse ermitteln kann. Außerdem müssen Sie natürlich wissen, wie das Problem, von den Eingangsgrößen zu den interessierenden Ausgangsgrößen zu kommen, im Prinzip gelöst werden kann. Wenn Sie es nicht wissen, können Sie Ihrem Heimcomputer auch nicht sagen, wie er es machen soll.
Sie sollten an dieser Stelle auch bedenken, ob Ihr Heimcomputersystem so weit ausgebaut ist, daß Sie die gestellte Aufgabe damit lösen können. Muß Ihr Heimcomputer sich beispielsweise während der Lösung der Aufgabe viele Zwischenergebnisse merken, d. h. speichern, oder wollen Sie eine große Anzahl von Daten auswerten, z. B. eine umfangreiche Kartei, so ist zu überprüfen, ob Sie mit dem vorhandenen Speicherplatz auskommen oder ob Sie noch RAM-Erweiterungsmodule benötigen.

Lassen Sie bei der Betrachtung der Lösungsmöglichkeit für eine Aufgabe aber auch Rechengeschwindigkeit und -genauigkeit nicht außer Betracht.

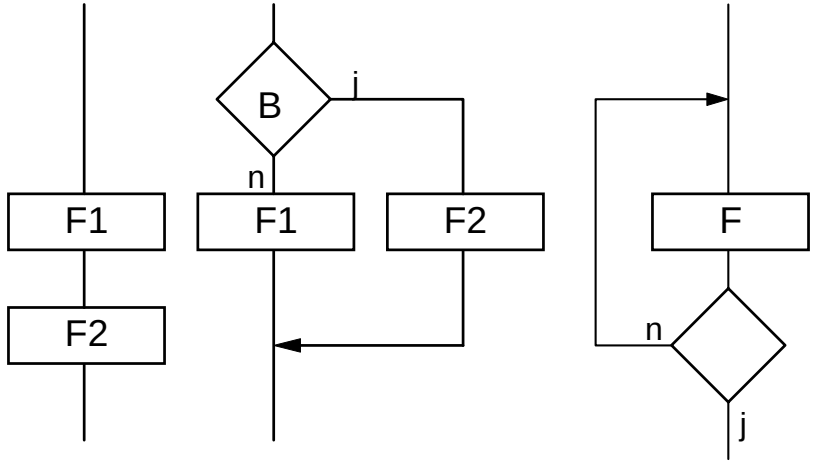
Entwurf des Programms
Der nächste Schritt besteht darin, daß Sie sich über die große Linie bei der Lösung Ihres Problemes Klarheit verschaffen. Wenn Sie Ihre Ferien planen, überlegen Sie sicher auch zuerst, wann und wohin Sie fahren wollen. Was Sie an den einzelnen Ferientagen machen, ergibt sich erst später.
Das Problem, das Sie mit dem Heimcomputer zu lösen beabsichtigen, ist zunächst in eine Reihe immer noch umfangreicher, aber schon kleinerer Teilprobleme zu zerlegen. Bei Ihrer Ferienplanung entsprechen dem vielleicht die Terminabstimmung mit Ihren Kollegen, die Quartierbestellung, die Organisation von An- und Abreise. Beim Entwurf eines Programms können Teilprobleme die Vereinbarung von Daten, ihre Ein- und Ausgabe und einzelne größere Schritte bei der Bestimmung von Zwischenergebnissen sein.
Diese Teilprobleme sollten so beschaffen sein, daß sie bei Vorgabe von Eingangsgrößen unabhängig voneinander gelöst werden können. Die als Ausgangsgrößen der Teilprobleme ermittelten Zwischenergebnisse sind bei der Lösung des Gesamtproblems dann wieder Eingangsgrößen für andere Teilprobleme.
Es ist möglich, daß bei der ersten Anfertigung ihres Gesamtproblems die von Ihnen erhaltenen Teilprobleme immer noch nicht überschaubar sind. Sie müssen dann nach denselben Gesichtspunkten weiter untergliedern, bis sich schließlich die entstehenden Probleme mit den zur Verfügung stehenden Anweisungen leicht lösen lassen.
Überlegen Sie bei der Zerlegung in Teilprobleme auch, ob es zweckmäßig ist, in Abhängigkeit von den Zwischenergebnissen den Fortgang der Problemlösung von Ihrem Heimcomputer steuern zu lassen oder ob Sie sich diese Zwischenergebnisse anzeigen lassen wollen, um dann selbst über die weitere Reihenfolge bei der Bearbeitung der Teilprobleme zu entscheiden. Im letzteren Fall spricht man auch davon, daß Sie Ihr Problem im Dialog lösen. Die Möglichkeit, so einen Dialog einfach zu programmieren, ist ein wesentlicher Vorzug Ihres Heimcomputers.
Auf der Grundlage der Zerlegung Ihres Problems muß das BASIC-Programm vorbereitet werden. Dazu sollten Sie sich für Eingangs-, Ausgangs- und Zwischengrößen geeignete Datenstrukturen überlegen. Zur Auswahl stehen Ihnen, Sie erinnern sich bestimmt, einfache

numerische Variable und Zeichenkettenvariable sowie Felder. Für die Darstellung der Zusammenhänge bei der Lösung der einzelnen Teilprobleme, d. h. der von Ihrem Programm äuszuführenden Teilfunktionen, haben sich Programmablaufpläne bewährt. Feststehende Symbole kennzeichnen einzelne Gruppen von Funktionen. Die wichtigsten sind in der nachstehenden Tabelle zusammengestellt.



Verbindungslinien geben an, in welcher Reihenfolge die einzelnen Programmteile durchlaufen werden sollen. Sie können zunächst einen recht groben Programmablaufplan zeichnen. Teilfunktionen, die an anderer Stelle noch weiter verfeinert werden sollen, markieren Sie in diesem Fall durch Doppelstriche am linken und rechten Rand des entsprechenden Kästchens. In feineren Programmablaufplänen legen Sie dann dar, wie diese Teilfunktionen ausgeführt.werden sollen.

Die dargestellten Symbole sind die Bausteine Ihrer Programmablaufpläne. Damit die "Gebäude", die Sie mit ihnen errichten, stabil stehen, sind einige Bauvorschriften zu beachten. Die wichtigsten Grundstrukturen, aus denen sich Ihre Programmablaufpläne zusammensetzen sollten, sind nachstehend angeführt.



Für den Fall, daß die Bedingungen erfüllt sind, wird jeweils der mit j gekennzeichnete Weg des Programms durchlaufen. Sind sie nicht erfüllt, wird das Programm auf dem mit n markierten Weg fortgesetzt. Als weitere Grundstrukturen sind eine Auswahl zwischen mehreren Wegen (realisierbar mit einer ON ... GOSUB- Anweisung) und eine Schleife, bei der die Abbruchbedingung am Anfang steht, erlaubt.
Anstelle eines Kästchens darf im Programmablaufplan auch wieder eine andere Grundstruktur stehen.
Die Beachtung der angegebenen Regeln wird Ihnen helfen, überschaubare, gut gegliederte und leicht zu testende Programme zu schreiben.

Schreiben des Programms

Mit dem Programmentwurf haben Sie die Hauptarbeit geschafft. Sie können nun anfangen, die Forderungen, die Sie an Ihren Heimcomputer haben, in die Sprache BASIC zu übersetzen, die er versteht; d. h., die Programmablaufpläne müssen in ein BASIC-Programm umgesetzt werden. Als Wörterbuch können Sie dabei die vorangegangenen Abschnitte dieses Programmierhandbuches nutzen.

Ausdrucksmöglichkeiten, Anweisungen, die Ihnen für die einzelnen Teilfunktionen zur Verfügung stehen, sind in der nachfolgenden Tabelle zusammengestellt.

Teilfunktion		Anweisungen
Vereinbarungen		DIM, DEF FN, CLEAR
Eingabe		INPUT, DATA/READ/RESTORE, CLOAD*, PEEK, DEEK, INP
Ausgabe		PRINT, PRINT AT, CSAVE*, POKE, DOKE, OUT, WINDOW, CLS, INK, PAPER, BORDER
Verarbeitung	Auswertungen	LET, Operationen, numerische Funktionen, Zeichenkettenfunktionen
	Auswahl	IF ... THEN ... :ELSE/GOTO..., ON ... GOTO, ON ... GOSUB
	Schleife	FOR ... NEXT
	Dialog	PRINT, PAUSE, BEEP, INPUT, INKEY$, JOYST
	Unterprogramme	GOSUB ... RETURN, CALL, CALL*, USR
	Kommentar	REM

Zu Beginn Ihres BASIC-Programms wird in der Regel das mit einer END-Anweisung abgeschlossene Hauptprogramm stehen, das Ihrem ersten groben Programmablaufplan entspricht. An den Stellen, an denen im Programmablaufplan durch Doppelstriche markierte Kästchen stehen, werden Unterprogramme aufgerufen. Die Reihenfolge, in der Sie die GOSUB-Unterprogramme in Ihrem BASIC-Programm niederschreiben, ist in der Regel gleich. Beachten Sie jedoch, daß ein Programmteil um so schneller verarbeitet wird, je weniger Programrnzeilen vor ihm stehen. Die Eingabe Ihres Programms erfolgt im AUTO-Modus. Korrekturen sind im EDIT-Modus möglich. Haben Sie häufig benötigte Unterprogramme auf Kassette abgespeichert, können Sie diese während der Programmeingabe auch mit CLOAD oder LOAD# in den Heimcomputer laden. Sicher wollen Sie nicht jedesmal Ihr gesamtes BASIC-Programm von neuem eintippen.
Vergessen Sie daher bitte nicht, es mit dem CSAVE- oder LIST#-Kommando auf Kassette abzuspeichern!

Programmtest

Ganz fertig sind Sie nach der Programmerstellung nicht. Sie müssen noch eine „Probefahrt“ (Testlauf) machen, um eventuelle Fehler zu erkennen. Beim Test gehen Sie jetzt umgekehrt wie beim Programmentwurf vor. Zunächst werden die Details, Unterprogramme und weniger umfangreiche Funktionen geprüft, erst zum Schluß das Gesamtprogramm.
Es gibt zwei typische Fehlerursachen. Sie können sich beim Programmentwurf geirrt haben, oder Sie haben bei der Programmerstellung in BASIC Fehler (z.B. auch Tippfehler) gemacht. Bei der Suche nach solchen Fehlern unterstützt Sie Ihr Heimcomputer, wie Sie aus der Tabelle entnehmen können.

Fehlerart	Hilfsmittel zur Fehlersuche
Fehler, die der Rechner merkt	Fehlermeldungen entsprechend Anhang H
Fehler, die Sie merken müssen (logische Fehler)	TRON,TROFF
	Ausgabe von Zwischenergebnissen mit PRINT, PAUSE
	Einfügen von STOP in Programm Wertzuweisung im Kommando-modus
	Programmstart mit GOTO *zeilennummer*
	Ergebnisausgabe im Kommando-modus mit PRINT

Hinweise:

Bei der Entwicklung einzelner Programme kann der Aufwand für die angegebenen Schritte unterschiedlich sein. Bei einfachen Programmen werden Sie nach einiger Übung schon mal auf das Aufschreiben von Programmablaufplänen verzichten, bei umfangreicheren kann es auch vorkommen, daß Sie erst bei Programmerstellung oder -test merken, daß Sie beim Entwurf noch etwas vergessen haben. Dann müssen Sie einzelne Schritte eventuell mehrfach wiederholen.

Beispiel:

Es soll ein Programm geschrieben werden, mit dem die Worte der Funktion y(x) = x - cos x für unterschiedliche Argumente x berechnet werden können. Dabei sind ein Anfangsargument und eine Schrittweite, die den Abstand der Argumente voneinander festlegt, zu Beginn der Auswertung einzugeben. Nach der Ausgabe eines Arguments x und des zugehörigen Funktionswertes y(x) soll durch Tastatureingabe entschieden werden, ob mit der alten Schrittweite das nächste Argument ermittelt werden soll, ob eine neue Schrittweite zur Bestimmung der folgenden Argumente einzugeben ist oder ob das Programm beendet werden kann.
Damit ist die Aufgabenstellung formuliert. Umfangreicher Speicherplatz für Zwischenergebnisse wird nicht benötigt, und der Lösung dieses Problems mit dem Heimcomputer steht nichts im Wege. Sie können mit dem Programmentwurf beginnen.

Die Gliederung des Ausgangsproblems ist in der Tabelle zusammengefaßt. Beachten Sie bitte, daß das zu schreibende Programm eine farbige Ausgabe ermöglichen soll. Verfügt Ihr Heimcomputer über keinen Farbzusatzmodul werden die Farbeinstellungen ignoriert.

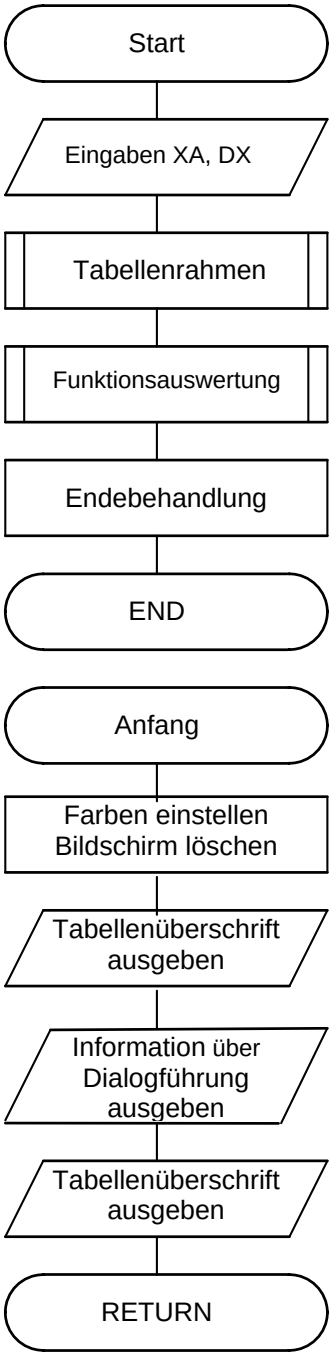
Teilprobleme	Zerlegung der Teilprobleme/Erläuterungen
Eingabe Tabellenrahmen	- Eingabe von Anfangsargument und Schrittweite - Farben wählen und Bildschirm löschen - Ausgabe der Überschrift X Y(X) in die 0. Bildschirmzeile - Ausgabe von Informationen über die Dialogführung in die 21. bis 23. Bildschirrnzeile - Festlegung des Ausgabebereiches auf die 3. bis 19. Bildschirmzeile
Funktionsauswertung	- Ausgabe von aktuellem Argument und Funktionswert - Tastaturabfrage (wird [ENTER] gedrückt, soll W zurückgegeben werden) - Programmfortsetzung in Abhängigkeit von der gedrückten Taste [W] - neues aktuelles Argument bestimmen [S] - alte Schrittweite zur Kontrolle ausgeben, neue Schrittweite eingeben, neues aktuelles Argument bestimmen [E] - Programmbeendigung
Endebehandlung	- Standardfarben einstellen - Bildschirm als Ausgabebereich definieren - Bildschirm löschen

Die erforderlichen Variablen sind aus der nächsten Tabelle ersichtlich.

Variablenname	Bedeutung
XA	Anfangsargument
DX	Schrittweite
x	aktuelles Argument
T$	Ergebnis der Tastaturabfrage

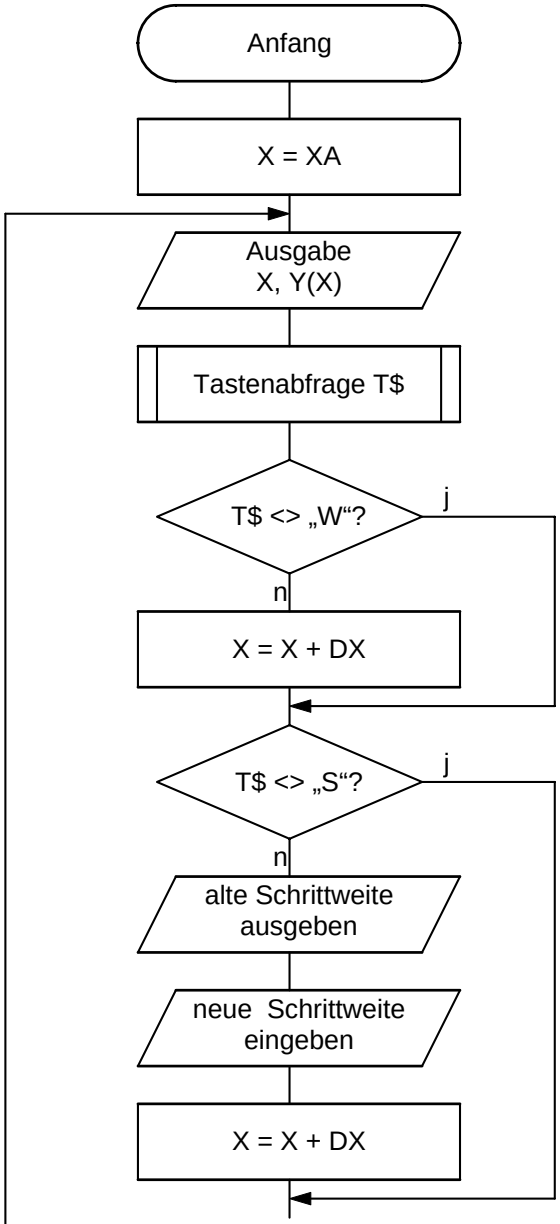
Auf der Grundlage der Aufteilung des Gesamtproblems in Teilprobleme werden die Programmablaufpläne erstellt.

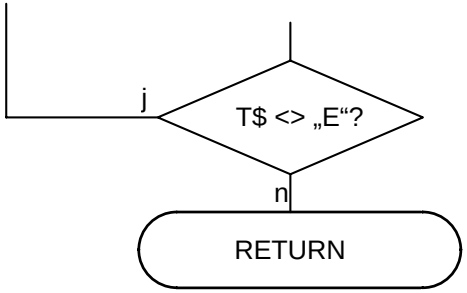
Hauptprogrammablauf



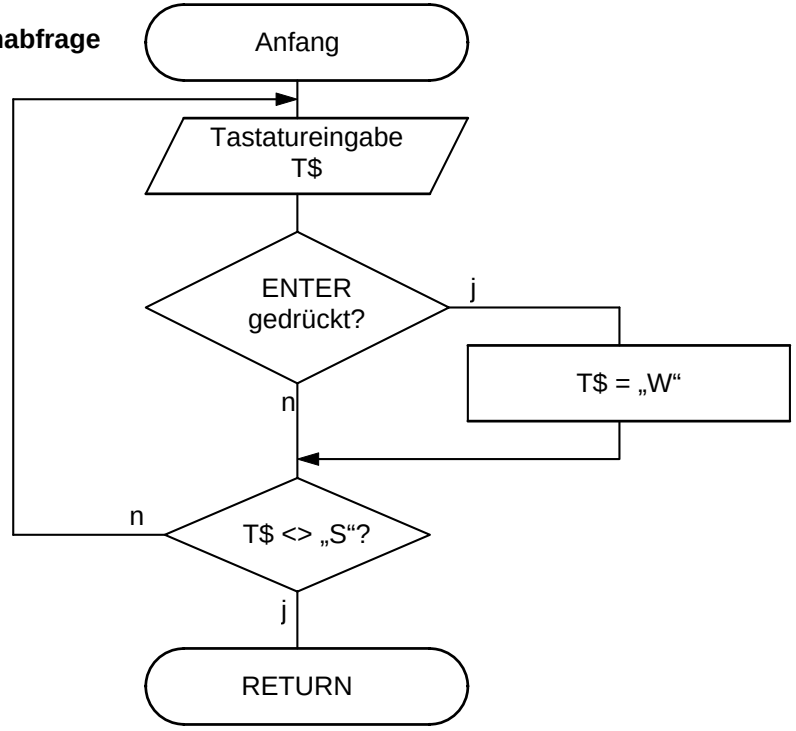
Tabellenrahmen

Funktionsauswertung





Tastenabfrage



Im nächsten Schritt werden die Programmablaufpläne in ein BASIC-Programm umgesetzt.

```
10 ! Eingaben
20 INPUT "ANFANGSARGUMENT?";XA
30 INPUT "SCHRITTWEITE?";DX
40 ! Aufruf Tabellenrahmen
50 GOSUB 130
60 ! Aufruf Funktionsauswertung
70 GOSUB 220
80 !Endebehandlung
90 INK 4:BORDER 1:PAPER 1:WINDOW:CLS
100 END
110 ! ----
120 ! Tabellenrahmen
130 INK7:BORDER 1:PAPER 1:WINDOW:CLS
140 PRINT INK2;TAB(7);"X";TAB(27);"Y(X) "
150 PRINT INK4;AT(21,0);STRING$(40,CHR$(160))
160 PRINT INK4;AT(22,0);"EINGABE:"
170 PRINT INK4;AT(23,0);"(W)-WEITER, S-NEUE
    SCHRITTWEITE, E-ENDE"
180 WINDOW 3,19,0,39
190 RETURN
200 ! ----
210 ! Funktionsauswertung
220 X=XA
230 PRINT TAB(5);X;TAB(25);X-COS(X)
240 ! Aufruf Tastenabfrage
250 GOSUB 370
260 ! Fallauswahl
270 IF T$<>"W" THEN GOTO 290
280 X=X+DX
290 IF T$<>"S" THEN GOTO 330
300 PRINT "ALTE SCHRITTWEITE:";DX
310 INPUT "NEUE SCHRITTWEITE:";DX
320 X=X+DX
330 IF T$<>"E" THEN GOTO 230
340 RETURN
350 ! ----
360 ! Tastaturabfrage
370 T$=INKEY$
380 IF T$=CHR$(13) THEN T$="W"
390 IF T$<>"W" AND T$<>"S" AND T$<>"E" THEN
   GOTO 370
400 RETURN
```

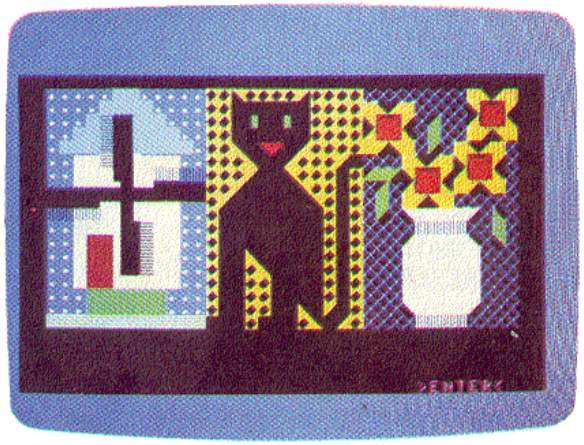
Der Programmtest kann beispielsweise mit dem Test der Tastaturabfrage beginnen. Eingangsgrößen gibt es in diesem Fall nicht. Als Zeile 391 wird das STOP-Kommando in das Programm eingefügt. Mit

```
STOP 370
```

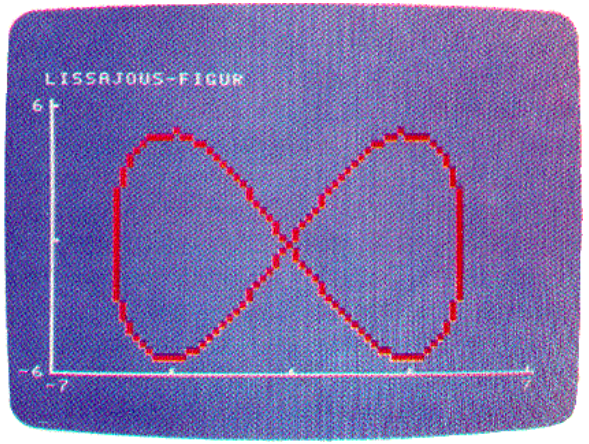
wird der Test aus dem Kommandomodus gestartet. Eine der Tasten [ENTER], [W], [S] oder [E] wird gedrückt, und nach Rückkehr in den Kommandomodus wird T$ mit PRINT T$ ausgegeben und überprüft. Ist alles richtig, wird die Zeile 391 wieder gelöscht. Danach kann die Funktionsauswertung getestet werden. Den Eingangsgrößen XA und DX werden im Kommandomodus Werte zugewiesen (z. B, durch XA=0.5:DX=0.1). Das STOP-Kommando wird in Zelle 331 eingefügt und der Text mit GOTO 220 gestartet. Anschließend können die Eingabe von Anfangsargument und Schrittweite und danach das Gesamtprogramm getestet werden. Ist dieses Programm vollständig getestet, können Sie mit ihm beispielsweise eine Nullstelle der Funktion x - cos(x) im Dialog mit Ihrem Heimcomputer bestimmen (wie?).
Wollen Sie andere Funktionen auswerten, müssen sie in Zeile 230 nur X-COS(X) durch eine andere Berechnungsvorschrift ersetzen. Bei der Berechnung des aktuellen Arguments X können sich übrigens Rundungsfehler ergeben. Vielleicht überlegen Sie sich einmal, wie diese zu vermeiden sind.

6.2. Programmtips

Sofortgrafik
Sie können auf den Bildschirm nicht nur Buchstaben und Zahlen ausgeben. Mit den Grafikzeichen ist es beispielsweise auch möglich, Bilder darzustellen und Kurven zu zeichnen.



Bilddarstellung mit Grafikzeichen



Kurvendarstellung

Mit dem folgenden einfachen Programm können Sie sofort selbst kleine Bilder darstellen.

```
10 WINDOW:CLS
20 PRINT INKEY$; :GOTO 20
```

Mit den Kursortasten positionieren Sie den Kursor und können dann ein beliebiges Zeichen (nach Drücken der [GRAFIK]-Taste auch ein Grafikzeichen) auf die markierte Stelle ausgeben. Ist mit Ihrem Heimcomputer auch eine Farbausgabe möglich, so können Sie durch Drücken der Taste [COLOR] oder der Tasten [SHIFT] [COLOR] und anschließende Eingabe eines Farbcodes auch Vorder- und Hintergrundfarbe neu wählen.

Bewegte Bilder

Nicht nur stehende, sondern auch bewegte Bilder können Sie mit Ihrem Heimcomputer erzeugen. Das Prinzip ist einfach. Sie geben auf eine Position des Bildschirms ein Zeichen aus, löschen es dann wieder durch Überschreiben mit einem Leerzeichen, geben das Zeichen erneut auf eine benachbarte Position aus und wiederholen den Vorgang. Für das Schreiben und Löschen können Sie die Anweisung PRINT AT verwenden. Wenn es besonders schnell gehen soll, schreiben Sie die Zeichen direkt mit POKE in den Bildwiederholspeicher. Wenn nötig, müssen Sie für einzelne Teile Maschinencodeprogramme verwenden (siehe Abschnitt 5.3).
Im Beispiel wird ein (roter) Ball (Code: 207) an den unsichtbaren Grenzen eines Raumes reflektiert.

```
10 PAPER 5:INK 2:CLS:PRINT INK5;AT(0,0);" "
20 Z=3:S=6:RZ=1:B$=CHR$(207)
30 ! Ballbewegung
40 IF (Z*RZ=-3)OR(Z*RZ=20) THEN RZ=-RZ
50 IF (S*RS=-3)OR(S*RS=36) THEN RS=-RS
60 PRINT AT (Z,S);" "
70 Z=Z+RZ:S=S+RS:PRINT A(Z,S);B$
80 GOTO 40
```

Anfangsposition und Richtung des Balls werden in Zeile 20 festgelegt. Die Raumgrenzen, an denen der Ball seine Richtung ändert, ergeben sich aus den Anweisungen in den Zeilen 40 und 50. Mit der Anweisung in Zeile 60 wird der "alte" Ball gelöscht. In Zeile 70 wird der "neue" Ball geschrieben.

Hexadezimal-/Dezimalumwandlung

Für Zahlendarstellungen wird heute üblicherweise ein Positionssystem, wie es auch das gebräuchliche Dezimalsystem darstellt, verwendet. Die Zeichen innerhalb so einer Darstellung haben verschiedene Stellenwerte. Beim Dezimalsystem erhält man den Zahlenwert als eine Summe von Zehnerpotenzen, multipliziert mit den an den zugehörigen Stellen stehenden Ziffern.
Beispiel für Dezimalzahl:

$$32453 = 3 \cdot 10^4 + 2 \cdot 10^3 + 4 \cdot 10^2 + 5 \cdot 10^1 + 3 \cdot 100$$

Bei Angaben in der Rechentechnik begegnen Sie auch dem Hexadezimalsystem. Der Zahlenwert ergibt sich als eine Summe von Potenzen der Zahl 16, multipliziert mit den Werten, die den Zeichen entsprechen, die an den zugehörigen Stellen stehen. Den Ziffern 1 bis 9 sind die entsprechenden Werte zugeordnet, den Buchstaben A bis F die Werte 10 bis 15.
Beispiel für Hexadezimalzahl:

$$\begin{aligned} 7EC5 &= 7 \cdot 16^3 + E \cdot 16^2 + C \cdot 16^1 + 5 \cdot 16^0 \\ &= 7 \cdot 16^3 + 14 \cdot 16^2 + 12 \cdot 16^1 + 5 \cdot 16^1 \\ &= 32453 \text{ (dezimal)} \end{aligned}$$

Die im Anhang D angegebene Tabelle soll Ihnen bei der Umwandlung beider Darstellungsarten ineinander helfen. Die Benutzung der Tabelle soll an einem Beispiel erläutert werden.

Gegeben:	Hexadezimalzahl 7EC5	
Gesucht:	Dezimalzahl	
Lösung:	1. Suche 7E00 in Spalte HEXA!	
	Aus rechter DEZ-Spalte folgt:	32256
	2. Suche C500 in Spalte HEXA!	
	Aus linker DEZ-Spalte folgt:	197
	3. Ergebnis	32453

Es ist aber auch möglich, diese Aufgabe mit einfachen BASIC-Programmen zu lösen. Die beiden folgenden Programme demonstrieren Ihnen gleichzeitig die Nutzung der Zeichenkettenfunktionen.
Dezimal- in Hexadezimaldarstellung umwandeln:

```
10 H$="":INPUT "DEZIMALZAHL: ";D
20 R=D-INT(D/16)*16
30 IF R<10 THEN H$=CHR$(48+R)+H$:ELSE
     H$=CHR$(55+R)+H$
40 IF D>15 THEN D=INT(D/16):GOTO20
50 PRINT "HEXADEZIMALZAHL: ";H$:PRINT:GOTO 10
```

Nach dem Programmstart ergibt sich z. B. folgendes Bild:

```
DEZIMALZAHL:  32453
HEXADEZIMALZAHL: 7EC5

DEZIMALZAHL:  ■
```

Hexadezimal- in Dezimaldarstellung umwandeln:

```
10 D=0:INPUT "HEXADEZIMALZAHL: ";H$: FOR I=1 TO
    LEN(H$)
20 Z=ASC(MID$(H$,I,1))
30 IF Z<50 THEN Z=Z-48:ELSE Z=Z-55
40 D=Z+D*16:NEXT I
50 PRINT "DEZIMALZAHL:   ";D:PRINT:GOTO 10
```

Tonwiedergabe

Mit dem robotron Z 9001" können Sie, wie in Abschnitt 5.5 beschrieben worden ist, Töne unterschiedlicher Höhe und Länge erzeugen. Das erlaubt es Ihnen, kleine Melodien auch ohne Musikmodul wiederzugeben.
Ein Beispiel dafür liefert das folgende Programm.

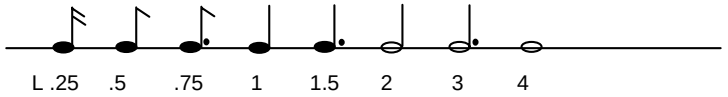
```
10 ! Kommt ein Vogel geflogen
20 DATA 172,.5,162,.5,144,1,172,1
30 DATA 172,1,172,1,193,1,193,.5
40 DATA 172,.5,162,1,193,1,193,.5
50 DATA 128,.5,144,2,172,.5,162,.5
60 DATA 144,1,172,1,172,1,172,1
70 DATA 193,1,193,.5,172,.5,162,1
80 DATA 229,1,229,1,216,2
90 ! Melodienwiedergabe
100 OUT 136,128:T=300
110 FOR I=1 TO 27
120 READ H,L:GOSUB 1000
130 NEXT
140 OUT 136,0:END
980 ! -----
990 ! Tonwiedergabe
1000 IF H=0 THEN GOTO 1020
1010 IF H>0 THEN OUT 128,7:OUT 128,H:ELSE OUT
128,39:OUT 128,-H
1020 FOR QQ=1 TO L*T:NEXT:OUT 128,3:RETURN
```

Nach dem Start des Programms hören Sie über den im Heimcomputer eingebauten Summer ein Volkslied. Eine bessere Klangqualität können Sie erreichen, wenn Sie die "Musik" über den Lautsprecher des Kassettengerätes (beim Geracord GC 6020 sind dazu eine Kassette einzulegen sowie die Pausen-, die Aufnahme- und die Wiedergabetaste zu drücken) ausgeben.
Das im Beispiel verwendete GOSUB-Unterprogramm zur Tonwiedergabe (Zeile 1000 bis 1020) können Sie in Ihren eigenen Programmen verwenden. Bevor es aufgerufen wird, müssen den Variablen H (Maß für die Tonhöhe), L (Maß für die Tonlänge) und T (Maß für das Tempo der Wiedergabe) Werte zugewiesen werden. Für T werden Werte um 300 empfohlen. Werte für H und T sind der nachfolgenden Tabelle zu entnehmen.

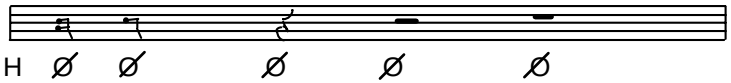
Werte von H in Abhängigkeit von der Tonhöhe

Ton	eingestrichene Oktave H	zweigestr. Oktave H	dreigestr. Oktave H
c	-18	144	72
cis/des	-17	136	68
d	-16	128	64
dis/es	243	121	60
e	229	114	57
f	216	108	54
fis/ges	204	102	51
g	193	96	48
gis/as	182	91	45
a	172	86	43
ais/b	162	81	40
h	153	76	38

Werte von L in Abhängigkeit von der Tonfolge

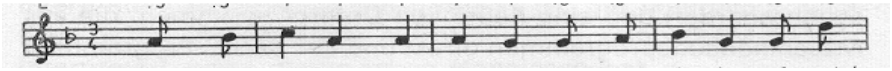


Werte von H und L bei Pausen



Mit dieser Übersicht können Sie sicher auch die übrigen Anweisungen des Beispielprogramms verstehen.
Die sich aus der Melodie des Volksliedes ergebenden Werte von H und L sind in den DATA-Anweisungen in den Zeilen 20 bis 80 des Beispiels zusammengefaßt.

H	172	162	144	172	172	172	193	193	172	162	193	193	128
L	.5	.5	1	1	1	1	1	.5	.5	1	1	.5	.5



Kommt ein Vo - gel ge - flo - gen, setzt sich nie - der auf mein'

H	144	172	162	144	172	172	172	193	193	172	162	229	229	216
L	2	.5	.5	1	1	1	1	1	.5	.5	1	1	1	2

Fuß, hat ein' Zet - tel im Schna - bel von der Mut - ter ein' Gruß.

In den Zeilen 110 bis 130 werden die Werte der DATA-Anweisungen gelesen und (so oft wie aufgrund der Länge der Melodie erforderlich) das GOSUB-Unterprogramm zur Tonwiedergabe aufgerufen.
Die OUT-Anweisungen in den Zeilen 100 und 140 schalten die Tonwiedergabe über den Summer Ihres „robotron Z9001“ ein bzw. aus (vgl. auch Abschnitt 5.5).

7. Das Betriebssystem des „robotron Z 9001“

Das Betriebssystem des „robotron Z 9001“ (Monitorprogramm) ist ein im Festwertspeicher vorhandener Grundstock von Programmroutinen. Es steht ihnen sofort nach dem Einschalten des Rechners zur Verfügung. Es enthält die Programme zur Unterstützung der Arbeit mit der Peripherie (Bildschirm, Tastatur, Kassettenrecorder) und eine Reihe von Unterprogrammen, die aus Maschinencodeprogrammen direkt und aus BASIC-Programmen über die Anweisungen CALL und USR(X) aufgerufen werden können. Das Betriebssystem realisiert außerdem eine interne Uhr und stellt Ihnen noch einige Kommandos zur Verfügung, die im OS-Modus benutzt werden können (vgl. Abschnitt 7.3). Wichtige Systemadressen, z. B. die Adressen der Uhrzellen, entnehmen Sie bitte dem Anhang E, ebenso die Speicheraufteilung und die Adressen der externen Kanäle (PIO, CTC).
Ohne gesonderte Vorkehrungen und ohne spezielle Kenntnisse machen Sie vom Betriebssystem ständig direkten oder indirekten Gebrauch (z. B. beim Laden und Abarbeiten von Anwenderprogrammen).
Für die Nutzung der Unterprogramme sind jedoch weitergehende Kenntnisse der Maschinen- und Assemblerprogrammierung notwendig.

7.1. Laden und Starten von Maschinencodeprogrammen

Maschinencodeprogramme, die in den Speicher des Rechners geladen und abgespeichert werden sollen, stehen auf der Kassette mit einem bestimmten Namen bereit, so z. B. der BASIC-Interpreter als Programm mit dem Namen BASIC. Um solche Programme einlesen zu können, geben Sie im Grundzustand des Rechners (z. B. nach dem Einschalten) über Tastatur den Dateinamen ein, z. B.:

```
> BASIC  [ENTER]
```

Auf dem Bildschirm erscheint dann die Aufforderung

```
start tape
```

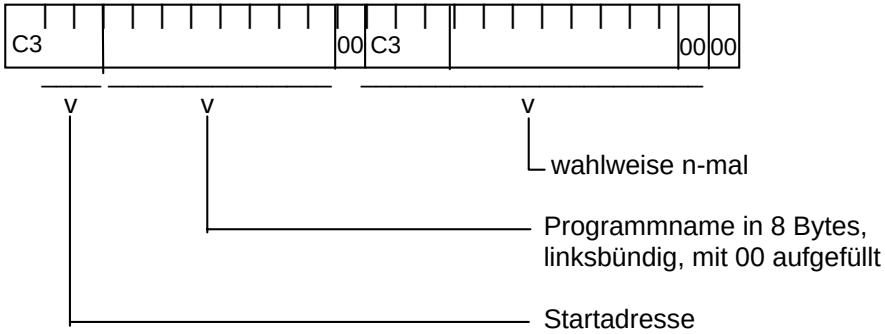
Wenn Sie jetzt feststellen, daß Sie einen falschen Programmnamen eingegeben haben, so können Sie an dieser Stelle noch durch Drücken der [STOP]-Taste wieder in den Grundzustand gelangen.

Die weiteren Arbeitsschritte entsprechen denen in Abschnitt 3.1. Mögliche Fehlermeldungen und wie Sie darauf reagieren können, finden Sie im Anhang H.

Hinweis:
Die Aufforderung

```
start tape
```

unterbleibt, wenn das angeforderte Programm, z. B. der BASIC-Interpreter, als ROM-Modul gesteckt ist. Sie unterbleibt auch nach [RESET] und erneutem Aufruf des Programms (RAM-BASIC-Interpreter), wenn dieses schon eingelesen war.
Das Betriebssystem erkennt also das Vorhandensein von im OS-Modus aufgerufenen Programmen.
Dazu muß das aufgerufene Maschinencodeprogramm ab einer „100H-Adresse“ (d. h. 300H, 400H, ..., 0E7D0H) folgendes enthalten:



Das Betriebssystem sucht nach der Eingabe eines Kommandos auf den "100H-Adressen" nach dem eingegebenen Namen und startet auf dem davor stehenden Sprungbefehl. Ist der Name nicht vorhanden, erfolgt die Ausschrift

```
start tape
```


7.2. Kommandos des Betriebssystems

CLOAD	Einlesen von Maschinencode ohne Start
TIME	Anzeigen/Setzen der Uhrzeit
ASGN	Gerätezuweisung
SAVE	Speichern von Maschinencode

Die Kommandos des Betriebssystems werden über die Eingabe ihres Namens gestartet, an den eventuell noch einige Parameter angefügt werden. Zwischen dem Namen und den Parametern muß mindestens ein Leerzeichen stehen.

Einlesen von Maschinencode

Format:
CLOAD *narne*[.*typ*l
name - Name der einzulesenden Maschinencodedatei
typ - Typ der einzulesenden Maschinencodedatei (max. 3 Zeichen, Standard: COM)

Funktion:
Mit diesem Kommando werden im OS-Modus Maschinencodedateien eingelesen.

Hinweis:
Der Dateiname *name* ist **nicht** in Anführungszeichen einzuschließen, und zwischen CLOAD und *name* muß mindestens ein Leerzeichen stehen. Mit diesem Kommando können z. B. Maschinencodeprogramme eingelesen werden, die dann aus BASIC-Programmen aufgerufen werden.
Nach der Eingabe des Kommandos (und dem abschließenden [ENTER]) erscheint die Aufforderung

```
start tape
```

Der weitere Ablauf erfolgt wie im Abschnitt 3.1 beschrieben.

Anzeigen und Setzen der Uhrzeit

Format:
TIME[*hh*:*mm*:*ss*]
hh - Stundenangabe (0 bis 23)
mm - Minutenangabe (0 bis 59)
ss - Sekundenangabe (0 bis 59)

Funktion:
Mit dem Kommando TIME ohne Parameter wird die Zeit der internen Uhr angezeigt.
Das Kommando mit Parametern dient dem Stellen der Uhr.

Hinweise:
1. Nach dem Einschalten des Rechners ist die Uhrzeit unbestimmt, die Uhr läuft aber.
   Zeitdifferenzen lassen sich also jetzt schon bestimmen. Wenn die tatsächliche Uhrzeit verwendet werden soll, ist aber die Uhr vorher zu stellen.
2. Der Zugriff zur internen Uhr vom Programm aus kann erfolgen über die Nutzung des entsprechenden Unterprogramms des Betriebssystems (siehe Anhang F) oder den direkten Zugriff zu den Systemzellen 29 bis 31 des Betriebssystems (vgl. Anhang E).
3. Die Ein- und Ausgabe zum/vom Kassettenmagnetband und das Betätigen der [RESET]-Taste führen zu kurzzeitigen Unterbrechungen des Normalbetriebes und damit zu einem "Nachgehen" der internen Uhr. Sie ist also, bevor sie verwendet wird, wieder zu stellen (falls notwendig).

Gerätezuweisung

Format:
ASGN [*log_name*:=*phys_name*]
log_name - Name eines logischen E/A-Gerätes
CONST - Konsole/Tastatur
READER - Eingabegerät
PUNCH - Ausgabegerät
LIST - Listenausgabegerät
phys_name - Name eines physischen E/A-Gerätes
CRT - Bildschirm
oder Name des Treiberprogramms auf ROM
(z. B. bei Einsatz des V24-Moduls)
bzw. auf Kassette
d. h. Name des Programms, das Treiberprogramm enthält).

Funktion:
Mit dem Kommando ASGN ohne Parameter erfolgt eine Anzeige der aktuellen Gerätezuordnung in der Form

```
>ASGN
CONST    :CRT
READER   :
PUNCH    :
LIST     :
```

Sind im Kommando Parameter angegeben, so wird der Treiber mit dem angegebenen *phys_name* initialisiert, in die Zuordnungstabelle eingetragen, und anschließend wird diese Zuordnungstabelle mit der neuen Belegung ausgegeben.

Hinweise:
1. Der eingegebene *phys_name* wird analog zu Abschnitt 7.1 im Speicher gesucht, und bei Vorhandensein wird der zugehörige Treiber initialisiert; ist *phys_name* nicht vorhanden, erfolgt Aufforderung zum Einlesen vom Kassettengerät mit

   ```
   start tape
   ```

   Der weitere Ablauf entspricht dann Abschnitt 3.1.
2. Beim Einsatz des Zusatzmoduls für den Druckeranschluß (V24-Modul) ist das entsprechende Treiberprogramm im Modul auf ROM vorhanden und wird beim Einschalten des Rechners automatisch initialisiert.

Beispiel:

```
>ASGN LIST:=CRT

CONST    :CRT
READER   :
PUNCH    :
LIST     :CRT
```

Zur Verdeutlichung der prinzipiellen Wirkungsweise können auf diese Weise nach Eingabe von [CONTR] [P] alle auf dem Bildschirm ausgegebenen Zeichen zusätzlich über den aktuellen LIST-Treiber ausgegeben werden. Es erscheinen also alle Zeichen auf dem Bildschirm doppelt. Die Rückstellung erfolgt durch erneute Betätigung von [CONTR] [P].

Speichern von Maschinencode

Dem Speichern von Maschinencode auf Magnetband dient das Kommando SAVE. Dieses Kommando steht erst nach dem Laden des Programms OS-SAVE von der Grundkassette R 0111 zur Verfügung. Wie das Kommando generiert und verwendet wird, entnehmen Sie bitte der Beschreibung dieser Grundkassette.

Format:

SAVE *name* [.*typ*]*anfangsadr*,*endadr* [,*startadr*]

name - Name der Magnetbanddatei, in die gespeichert werden soll max. 8 Zeichen)

typ Typ der Magnetbanddatei (max. 3 Zeichen) Bei Weglassen der Typangabe wird *typ* = COM gesetzt.

anfangsadr - Anfangsadresse des abzuspeichernden Bereiches

endadr - Endadresse des abzuspeichernden Bereiches

startadr - Startadresse für lauffähige Heimcomputerprogramme. Bei Weglassen von *startadr* wird *startadr* = *anfangsadr* gesetzt.

Funktion:

Speichern von Maschinencode auf Magnetbandkassette.

Hinweise:

1. Alle Angaben von Adressen müssen hexadezimal erfolgen. Der Suffix H ist wegzulassen. Die Angabe einer Adresse muß mit einer Dezimalziffer beginnen (statt A000 ist z. B. 0A000 zu verwenden).
2. Bei Abzügen von Speicherbereichen kann als Startadresse 0FFFF angegeben werden, um ein irrtümliches Starten zu verhindern; auf der Adresse 0FFFFH steht der Maschinencodebefehl RET (0C9H).
3. Das Programm OS-SAVE läßt sich mit dem Kommando SAVE nicht doppeln, da OS-SAVE verschieblich ist und über eine Initialisierungsroutine der jeweiligen Speicherkonfiguration angepaßt wird.
4. Um den RAM-BASIC-Interpreter zu doppeln, sind folgende Arbeitsschritte notwendig:
   - Laden von OS-SAVE (damit Übergang in den Extended-OS-Modus)
   - CLOAD BASIC
   - SAVE BASIC 300,2AFF,2400

Beispiel:

Die Bedienhandlungen und Ausschriften beim Auslagern einer Datei mit SAVE sind folgende:

- Nach der Eingabe des Kommandos

```
>SAVE BASIC 300,2AFF,2400
```

erscheint die Aufforderung

```
start tape
```

Danach sind der Recorder auf Aufnahme zu schalten und die [ENTER]-Taste zu betätigen.

- Nach der vollständigen Aufzeichnung der Datei erscheint die Frage

```
VERIFY ((Y)/N)?: ■
```

Geben Sie jetzt [N] [ENTER] ein, so befindet sich der Rechner nach der Ausschrift

```
50 RECORD(S) WRITTEN
NO RECORD(S) CHECKED
```

wieder im Grundzustand (jetzt Extended-OS-Modus).

Bei jeder anderen Beantwortung der Frage (z. B. nur [ENTER]), werden Sie mit

```
REWIND <--
```

zum Rückspulen des Bandes aufgefordert. Wenn Sie das getan haben, drücken Sie auf [ENTER]. Sie werden dann mit

```
start tape
```

zum Starten des Magnetbandes (Wiedergabetaste) aufgefordert. Jetzt bitte starten und [ENTER] drücken.

- Nach dem vollständigen Prüfen der richtigen Aufzeichnung erscheint die Ausschrift

```
SAVE COMPLETE
50 RECORD(S) WRITTEN
50 RECORD(S) CHECKED
```

und der Rechner befindet sich wieder im Grundzustand.

7.3. Steuerzeichen

Im Zeichensatz des "robotron Z 9001" sind neben den darstellbaren Zeichen (alphanumerische und Grafikzeichen) noch Steuerzeichen enthalten mit den Codes von 1 bis 31 (01H bis 1FH) und den Funktionen entsprechend Anhang A.
Die Steuerzeichen können über die Betätigung der dort angegebenen Tasten erzeugt werden. Einige Steuerzeichen werden nur bei der Arbeit mit dem BASIC-Interpreter ausgewertet (z. B. die für [INS], [LIST], [RUN]. Die Steuerzeichen, die eine Auswertung des danach eingegebenen Zeichens nach sich ziehen (z. B. für [COLOR]), sind aber im BASIC nur eingeschränkt anwendbar.
Die zweite Möglichkeit zur Nutzung der Steuerzeichen ist ihre Verwendung in Ausgabeanweisungen (in BASIC nur in PRINT-Anweisungen, nicht PRINT AT). Ihre Verwendung ist dann nicht eingeschränkt. Beispiele zur Anwendung der Steuerzeichen in BASIC-Programmen finden Sie in den Abschnitten 5.2 und 5.4.

7.4. Nutzung der Unterprogramme des Betriebssystems

Die vom Anwender nutzbaren Unterprogramme des Betriebssystems (vgl. Anhang F) werden in Form von Systemrufen mit einheitlichem Eintrittspunkt aufgerufen.
Die Parameter zur Identifikation des Rufes und zur Wertüber- und Rückgabe werden in den CPU-Registern übermittelt:

Rufnummer - C
Übergabeparameter - A bzw. DE
Rückgabeparameter - A bzw. BC

Fehler bei der Abarbeitung der Rufe werden durch

gesetztes C-Flag und eine
Fehlernummer im A-Register

angezeigt.
Die Rufe erfolgen einheitlich in der Form

```
CALL 5
```

Hinweise:

1. Beim Aufruf erfolgt ein automatisches Retten der Register, der Anwender-Stack wird nicht belastet.
2. Die zur Verfügung stehenden Unterprogramme mit ihren Rufnummern und verwendeten Übergaberegistern entnehmen Sie bitte dem Anhang F.
3. Werden bei der Abarbeitung der Rufe Fehler erkannt, so wird eine Fehlermitteilung entsprechend Anhang H auf dem Bildschirm erzeugt.
4. Ein Beispiel zur Nutzung eines Rufes von einem BASIC-Programm aus finden Sie in Abschnitt 5.3.

8. Maschinencode- und Assemblerprogrammierung

Maschinencodeprogramme, die vom BASIC-Interpreter aus aufgerufen werden sollen, können auf verschiedenen Wegen generiert und bereitgestellt werden. Eines ist ihnen aber gemeinsam: sie dürfen nicht auf Speicherbereichen stehen, die vom BASIC- Interpreter selbst benutzt werden.

Speichererfordernisse

Immer frei ist im Grundzustand der Bereich von 220H bis 2FFH (544 bis 767), und frei ist auch der Bereich nach dem letzten durch den Interpreter belegten RAM-Speicherplatz bis zum letzten im System verfügbaren RAM-Speicherplatz (vgl. Anhang E).
Wird nach dem Start des BASIC-Interpreters die Frage

```
MEMORY SIZE?: ■
```

mit der Eingabe einer Dezimalzahl beantwortet, so stellt diese die letzte durch den BASIC-Interpreter zu belegende Adresse dar.
Nach einer Eingabe von

```
16127
```

[ENTER]

würde also der Bereich von 16128 (3F00H) bis 16383 (3FFFH) freigehalten (Grundvariante, ohne Zusatzmoduln).
Eine entsprechende Veränderung des durch den Interpreter belegten, Speicherplatzes läßt sich auch während der Arbeit mit dem Interpreter durch die Anweisung CLEAR vornehmen. Der zweite Parameter gibt hier die letzte belegte Adresse an (vgl. Abschnitt 4.17).
Nach der Anweisung

```
CLEAR 256,16127
```

belegt der BASIC-Interpreter also den gleichen Speicherbereich wie nach Beantwortung der Frage "MEMORY SIZE?: mit 16127

Generierung der Maschinencodeprogramme in BASIC

Maschinencodeprogramme lassen sich aus BASIC-Programmen heraus als lauffähige Programme im Speicher bereitstellen. Dazu müßten Sie zunächst (auf dem Papier) den Assemblerquelltext entwerfen, den Speicherumfang Ihres Programms bestimmen und anschließend den Adreßbereich, in dem es abgearbeitet werden soll. Jetzt können Sie mit Hilfe von Assembler-Sprachbeschreibungen die hexadezimale Speicherbelegung bestimmen und davon ausgehend mit Hilfe von Anhang D die dezimale Speicherbelegung.
Die Beispiele für Maschinencodeprogramme im Abschnitt 5.3. enthalten die Startadresse, den Quelltext und die Speicherbelegung hexadezimal und dezimal. Die dezimale Speicherbelegung kann nun mit Hilfe von DATA-Anweisungen im BASIC-Programm bereitgestellt und mit wiederholten READ- und POKE-Anweisungen in die benötigten Speicherbereiche geschrieben werden.

Beispiel:

```
100 DATA 205,111,201,75,205,5
110 DATA 0,120,65, 195,177,208
120 FOR I=0 TO 11
130 READ A
140 POKE 600+I,A
150 NEXT
```


Eine solche Anweisungsfolge kann nach dem Start Ihres BASIC-Programms aufgerufen werden, wenn Sie in ihm das entsprechende Maschinencodeprogramm benötigen.
Möglich ist aber auch, daß Sie nach der einmaligen Bereitstellung des Maschinencodeprogramms dieses mit Hilfe des Betriebssystemkommandos SAVE auf Kassette auslagern. Sie können sich dann bei Bedarf das Programm noch vor dem Start des BASIC-Interpreters mit Hilfe des Betriebssystemkommandos CLOAD in den Speicher laden. Für das obige Beispiel würden die entsprechenden Kommandos (im OS-Modus) lauten

```
SAVE BEISPIEL 258,263

CLOAD BEISPIEL
```

Die Generierung beim Start des BASIC-Programms ist dann zweckmäßig, wenn die Maschinencodeprogramme relativ kurz sind. Der Nachteil der separaten Bereitstellung liegt vor allem in der zusätzlichen Arbeit mit dem Kassettengerät.

Assemblerprogrammierung

Eine weitere Möglichkeit, zu Maschinencodeprogrammen zu gelangen, steht Ihnen mit der Nutzung des EDITOR/ASSEMBLER zur Verfügung, den Sie käuflich erwerben können. Mit seiner Hilfe können sie Assemblerquelltext erstellen, ändern und auf Kassette abspeichern. Außerdem können Sie mit entsprechenden Übersetzungskommandos die erzeugten Maschinencodeprogramme im Speicher oder auf Kassette ablegen.
Die zu diesem Systemprogramm gehörenden Funktionen, Kommandos und Fehlermeldungen entnehmen Sie bitte der dazu mitgelieferten Bedienungsanleitung.

Sachwortverzeichnis

Dieses Programmierhandbuch wurde verfaßt von einem Autorenkollektiv des VEB Robotron-Meßelektronik »Otto Schön« Dresden
Dr.-Ing. Hans-Jürgen Busch
Dr.-Ing. Joachim Haase
Dr. rer. nat. Gert Keller
Dr.-Ing. Hans-Jörg Nowottne

DEWAG Dresden . ATN 33442 031/4 .
Regie: Mros; Gestaltung: Bucher
6/85a - Jt 2234/85 u. Jt 2235/85 - II-13-1