

Inhaltsverzeichnis

4.14.	spezielle Eingabefunktionen INKEY\$, JOYST	88
4.15.	Weitere Ausgabemöglichkeiten WINDOW, PRINT, TAB, SPC, POS, PRINT AT	91
4.16.	Farbige Ausgabe PAPER, INK, BORDER, PRINT, PRINT AT	101
4.17.	Interner BASIC-Arbeitsspeicher CLEAR, FRE	106
4.18.	Externspeicherung von Programmen und Daten CSAVE, CSAVE*, CLOAD, CLOAD*	111
4.19.	Testunterstützung TRON, TROFF	117
4.20.	Fehlermeldungen	117
5.	Fortgeschrittene BASIC-Programmierung	118
5.1.	Direkter Speicherzugriff PEEK, POKE, DEEK, DOKE	118
5.2.	Zugriff zu Bild- und Farbspeicher	120
5.3.	Aufruf von Maschinencodeprogrammen CALL, CALL*, USR(X)	123
5.4.	Datentransfer LIST#, LOAD*, NULL, WIDTH	127
5.5.	4 Ein- und Ausgabe über Nutzerschnittstelle INP, OUT, WAIT	130
6.	Hinweise und Beispiele zur BASIC-Programmierung	135
6.1.	Schrittweises Vorgehen	135
6.2.	Programmtips	147
7.	Das Betriebssystem des "robotron Z 9001"	154
7.1.	Laden und Starten von Maschinencodeprogrammen	154
7.2.	Kommandos des Betriebssystems CLOAD, TIME, ASGN, SAVE	156
7.3.	Steuerzeichen	161
7.4.	Nutzung der Unterprogramme des Betriebssystems	161
8.	Maschinencode- und Assemblerprogrammierung	162
	Sachwortverzeichnis	165

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.

Spielhebel 1 nach rechts gedrückt
↓
XX XXXXXXXX XX
XXX ↑
Spielhebel 1 nicht mehr nach rechts gedrückt

Mit [STOP] kann die Programmabarbeitung beendet werden.

4.15. Weitere Ausgabemöglichkeiten

WINDOW	Festlegung des Ausgabebereiches
PRINT	Ausgabeeanweisung
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

$$0 \leq \text{zeile1} \leq \text{zeile2} \leq 23$$

$$0 \leq \text{spalte1} \leq \text{spalte2} \leq 39$$

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 Cursor 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 Cursorposition verändert. Bei Ausführung einer PRINT-Anweisung steht der Cursor jeweils hinter dem gerade zuletzt ausgegebenen Zeichen.
2. Ist die Ausgabeliste einer PRINT-Anweisung nicht 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 Tabulatorfunktion angegebenen Position ausgegeben wurde, d. h., wenn vor ihrer

Ausführung die aktuelle Cursorposition 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 Cursorpositionen 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 Cursors 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=-§
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 $0 \leq \text{zeile1} \leq \text{zeile2} \leq 19$

$0 \leq \text{spalte1} \leq \text{spalte2} \leq 39$

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*);*ausgabeliste*
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 Zeichenkettenvariablen 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 $(5 + m) \cdot z$ 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 Variablenamen
- 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	Speicher platz	
	in Tabelle der Variablen	im Zeichen- ketten- 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 ROM-BASIC	ende-wert ¹⁾ RAM-BASIC	größter 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 Zeichenkettspeicherbereich 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 Zeichenkettspeicherbereiches und das Ende des Arbeitsspeichers nicht verändert.

Beispiel:

```
OK
CLEAR 300,16000
```

¹⁾ Zeichenkettspeicherbereich 0 Bytes, extrem kurze Programme.

Nach Ausführung dieses Kommandos ist der Zeichenkettspeicherbereich 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 Zeichenkettspeicherbereich. 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(ST$)
```

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 Zeichenkettspeicherbereich 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 "*progrname*"

progrname - Name (1 bis 8 Zeichen)

Funktion:

Mit diesem Kommando wird das gesamte im Arbeitsspeicher des BASIC-Interpreters stehende Programm unter der Bezeichnung *progrname* 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 Cursor 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 Kontrolllesen 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 Kontrolllesen Ihres aufgezeichneten Programms oder Feldes ein Fehler auf, so bewegt sich der Cursor auf dem Bildschirm nicht mehr weiter. Sie können dann das Kontrolllesen 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 1 =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 "programe"

programe - 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 *feldname* 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 Programmmodus 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 zusammengestellt. Ist der Fehler in einer Anweisung eines Programms be-

merkt 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 Sekundenzaehler
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")
```

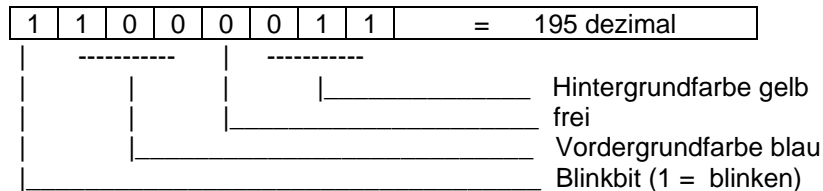
```
POKE -4710,97
```

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 Codierung		dezimale Codierung			

M0: M1: M2:	PUSH	HL	E5			229		
	PUSH	BC	C5			197		
	PUSH	DE	D5			213		
	LD	HL,0EF70H	21	70	EF	33	112	239
	LD	(HL),62	36	3E		54	62	
	LD	B,39	06	27		6	39	
	LD	C,0	0E	00		14	0	
	LD	D,0	16	00		22	0	
	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 Maschinencodeprogrammen 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 Integerzahl (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	0D0B1H

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 Cursorposition 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 Programm-anfang

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 Druckermodule 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 *kanaladresse, byte*

kanaladresse - numerischer Ausdruck (ganzzahlig) mit einem Wert von 0 bis 255

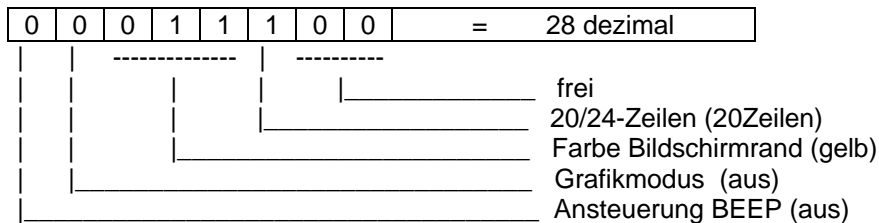
byte - numerischer Ausdruck (ganzzahlig) mit einem Wert von 0 bis 255

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 Summertone (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 Interruptbehandlungsprogramme 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 niedrigere 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 Problems 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.

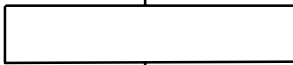
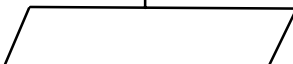
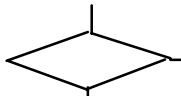
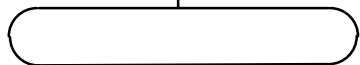
Diese Teilprobleme sollten so beschaffen sein, daß sie bei Vorgabe von Eingangsgrößen unabhängig voneinander gelöst werden können. Die als Ausgangsgrößen der Teilprobleme ermittelten Zwischenergebnisse sind bei der Lösung des Gesamtproblems dann wieder Eingangsgrößen für andere Teilprobleme.

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.

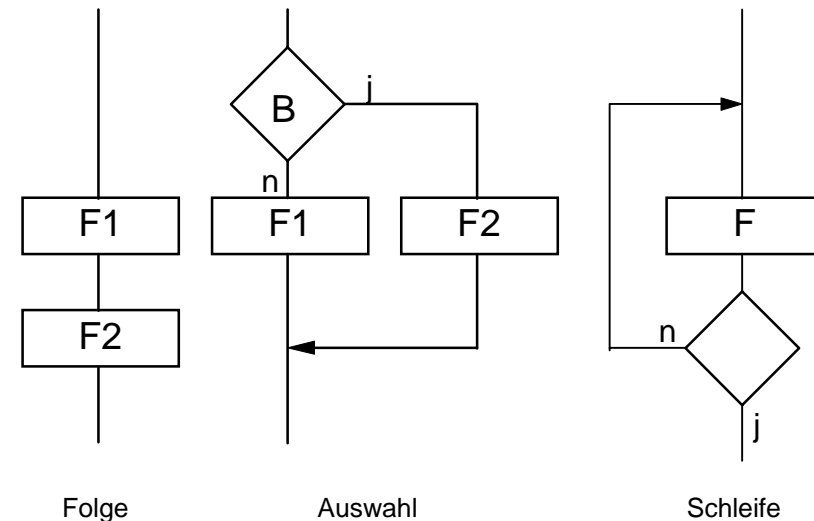
Auf der Grundlage der Zerlegung Ihres Problems muß das BASIC-Programm vorbereitet werden. Dazu sollten Sie sich für Eingangs-, Ausgangs- und Zwischengrößen geeignete Datenstrukturen überlegen. Zur Auswahl stehen Ihnen, Sie erinnern sich bestimmt, einfache numerische

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

Symbol	Bedeutung
	Bearbeitung eines Teilproblems
	Ein- oder Ausgabe
	Verzweigung nach Abfrage einer Bedingung
	Anfang oder Ende im Programmablauf

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
	Auswahl
	Schleife
	Dialog
	Unterprogramme
	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 Programmzeilen 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 Programmwurf 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 Programmwurf 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 <u>zeilennummer</u>
	Ergebnisabgabe 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 Werte 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	- Eingabe von Anfangsargument und Schrittweite
Tabellenrahmen	- 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. Bildschirmzeile - Festlegung des Ausgabegebietes auf die 3. bis 19. Bildschirmzeile

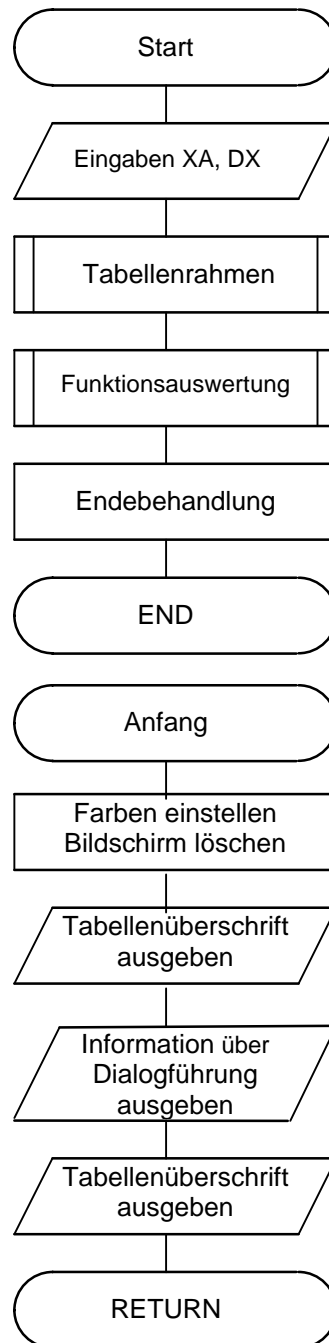
Teilprobleme	Zerlegung der Teilprobleme/Erläuterungen
Funktionsauswertung	- Ausgabe von aktuellem Argument und Funktionswert - Tastaturabfrage (wird [ENTER] gedrückt, soll W zurückgegeben werden) - Programmfortsetzung in Abhängigkeit von der gedrückten Taste [W] - neues aktuelles Argument bestimmen [S] - alte Schrittweite zur Kontrolle ausgeben, neue Schrittweite eingeben, neues aktuelles Argument bestimmen [E] - Programmbeendigung
Endebehandlung	- Standardfarben einstellen - Bildschirm als Ausgabegebiet definieren - Bildschirm löschen

Die erforderlichen Variablen sind aus der nächsten Tabelle ersichtlich.

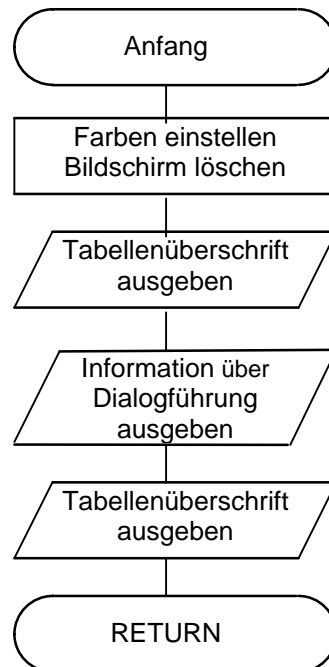
Variablenname	Bedeutung
XA	Anfangsargument
DX	Schrittweite
x	aktuelles Argument
T\$	Ergebnis der Tastaturabfrage

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

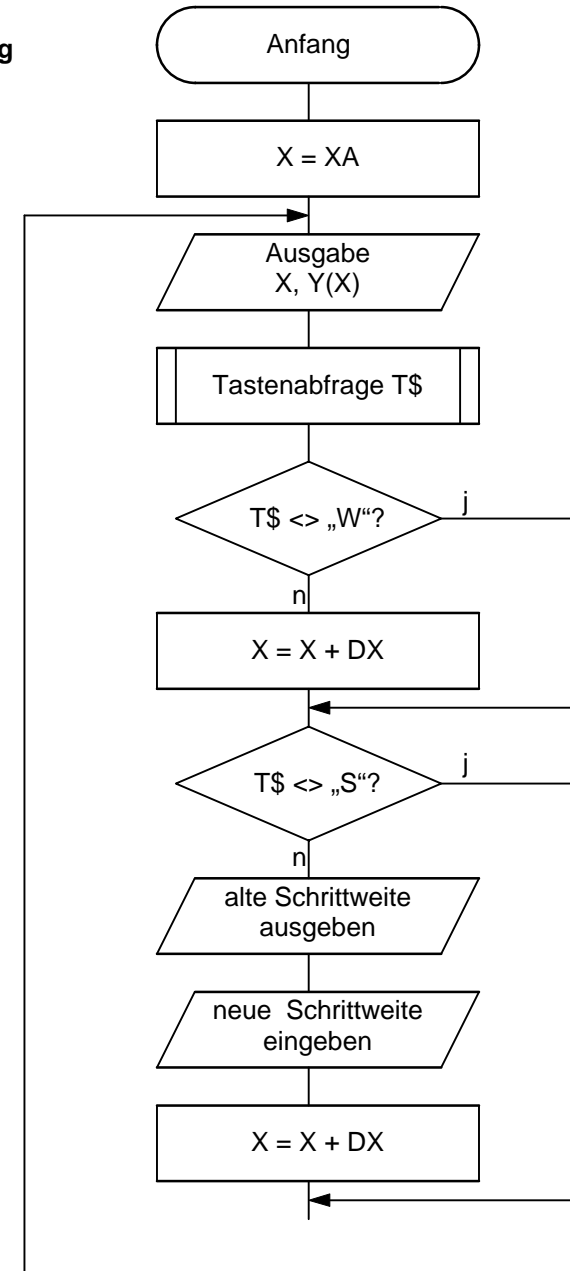
Hauptprogrammablauf

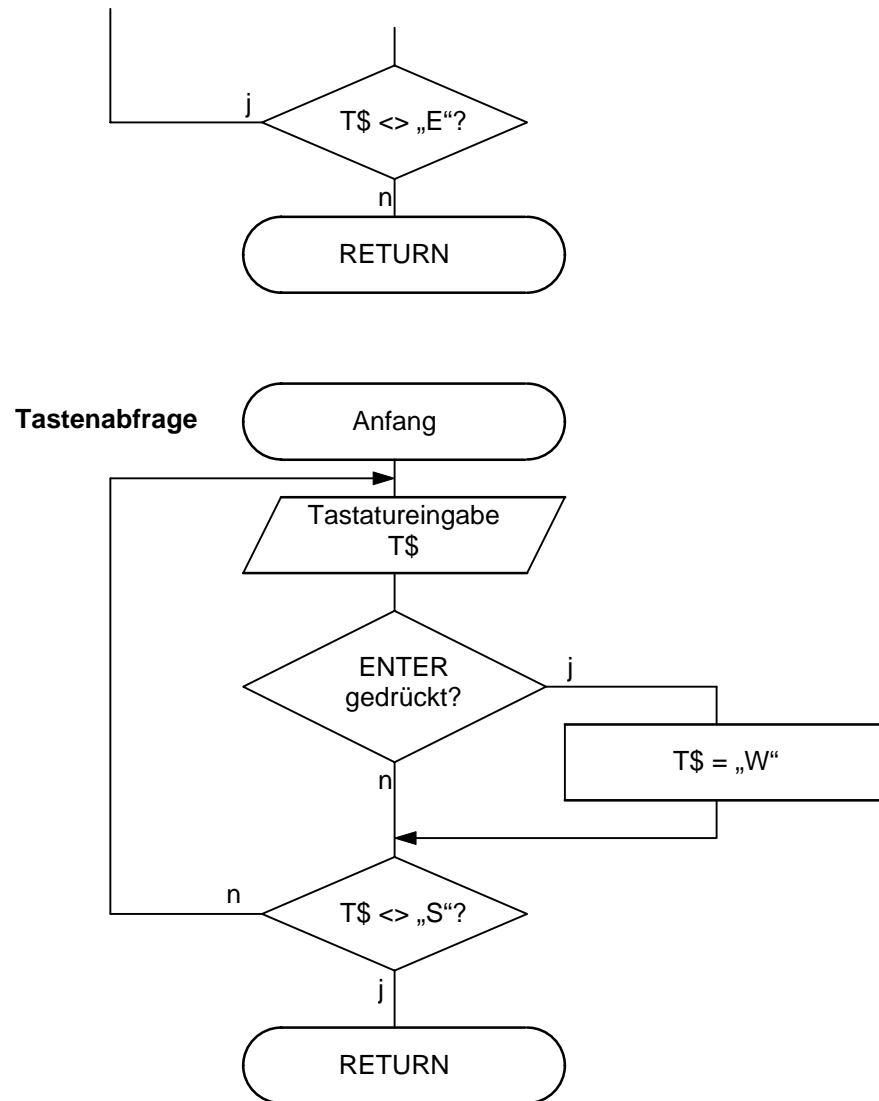


Tabellenrahmen



Funktionsauswertung





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

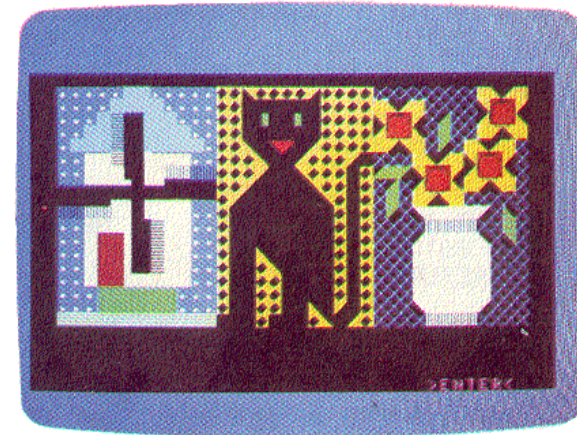
wird der Test aus dem Kommandomodus gestartet. Eine der Tasten [ENTER], [W], [S] oder [E] wird gedrückt, und nach Rückkehr in den Kommandomodus wird T\$ mit PRINT T\$ ausgegeben und überprüft. Ist alles richtig, wird die Zeile 391 wieder gelöscht. Danach kann die Funktionsauswertung getestet werden. Den Eingangsgrößen XA und DX werden im Kommandomodus Werte zugewiesen (z. B. durch XA=0.5:DX=0.1). Das STOP-Kommando wird in Zeile 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.

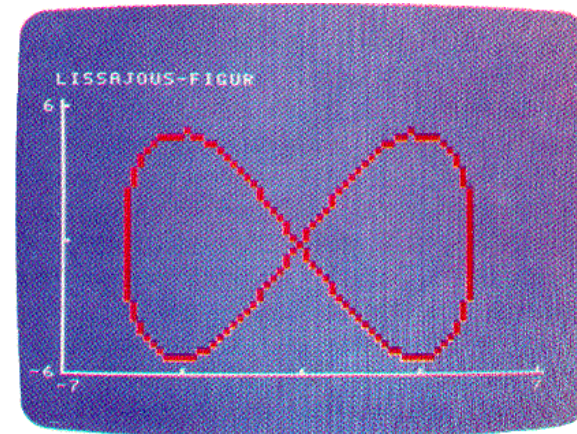
6.2. Programmtips

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 Cursor 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 Bildwiederholtspeicher. 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.

```
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:

$$32453 = 3 \cdot 10^4 + 2 \cdot 10^3 + 4 \cdot 10^2 + 5 \cdot 10^1 + 3 \cdot 10^0$$

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:

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

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

einulegen 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



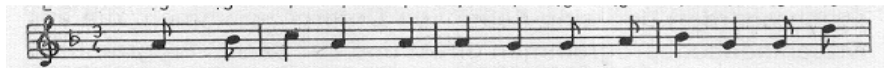
Werte von H und L bei Pausen



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	.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:

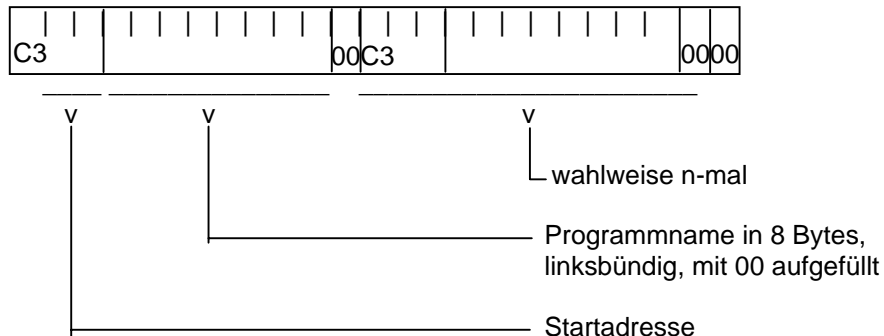
Die Aufforderung

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 *name*[*typ*]

name - Name der einzulesenden Maschinencoddatei

typ - Typ der einzulesenden Maschinencoddatei (max. 3 Zeichen, Standard: COM)

Funktion:

Mit diesem Kommando werden im OS-Modus Maschinencoddateien 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*]

<i>name</i>	- Name der Magnetbanddatei, in die gespeichert werden soll (max. 8 Zeichen)
<i>typ</i>	Typ der Magnetbanddatei (max. 3 Zeichen) Bei Weglassen der Typangabe wird <i>typ</i> = COM gesetzt.
<i>anfangsadr</i>	- Anfangsadresse des abzuspeichernden Bereiches
<i>endadr</i>	- Endadresse des abzuspeichernden Bereiches
<i>startadr</i>	- Startadresse für lauffähige Heimcomputerprogramme. Bei Weglassen von <i>startadr</i> wird <i>startadr</i> = <i>anfangsadr</i> 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 Assembler Quelltext 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 Assembler Quelltext 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

A

ABS -Funktion	32
Abspeichern	
- ,eines BASIC-Programms	111, 127
- ,eines Feldes	111
- ,eines Maschinencodeprogramms	159
Adressen	
- ,des Bildspeichers	120, Anhang C
- ,des Farbspeichers	120, 121, Anhang C
- ,des Systems	126, Anhang E
Anfangswert	56
AND -Operator	33
Anweisung	25
- ,bedingte	54
Anwenderprogramm	15
ASC -Funktion	80
ASCII-Code	80, 89, 127, Anhang A
ASGN-Kommando	157
ATN -Funktion	32
Ausdruck	
- ,einfacher	32
- ,logischer	33
- ,numerischer	32
- ,Zeichenkettenausdruck	35
- ,zusammengesetzter	33
Ausgabebereich	48, 92
Ausgabeliste	46, 93, 98
AUTO -Modus	38
Auswahlbedingung	54

B

BASIC

- ,Arbeitsspeicher	107, Anhang E
- ,Beginn der Arbeit mit	8, 10
- ,Beendigung der Arbeit mit	11
- ,Funktion	72
- , Kopieren	160
- ,RAM-BASIC	8, 109, 126
- ,ROM-BASIC	10, 109, 126
- , WBASIC	11
BEEP -Anweisung	50, 131
Bereich, freier	109, Anhang E
Bildschirm	

- ,Löschen des	24
Bildschirmrand	
- , Farbe des	99, 103
Bit	119
BORDER -Anweisung	103
BOS-error	9, 113, Anhang H
BYE -Kommando	11
Byte	10, 118

C

CALL - , CALL* -Anweisung	123
CHR\$ -Funktion	81, 130
CLEAR -Anweisung	108
[CL LN]-TASTE	14
CLOAD - , CLOAD* -Anweisung	16, 115
CLOAD -Kommando	156
CLS -Anweisung	24
[COLOR]-Taste	7, 102
CONT	
- ,Kommando	58
- ,Taste	7, 58, 61
[CONTR]-Taste	7
COS -Funktion	32
CSAVE - , CSAVE* -Anweisung	111
CTC-Programmierung	132

D

DATA -Anweisung	69
DEEK -Funktion	119
DEF FN -Anweisung	74
[DEL]-Taste	14
DELETE -Kommando	64
Dezimalzahl	13, 28, 150
Dialog	136, 139
DIM -Anweisung	67, 109
Dimension	67, 108
DOKE -Anweisung	119
Dollarzeichen \$	23, 80

E

EDIT -Modus	63
Eingaben	

- ,von Programmen	37
ELSE , siehe IF ... THEN ... ELSE	
END -Anweisung	61
Endwert	56
ERROR-Ausschrift	Anhang H
[ENTER]-Taste	4, 18, 21, 36
Erweiterungsmodul	3, 109, 130
[ESC]-Taste	7, 89, 90
EXP -Funktion	32
EXTRA IGNORED	49

F

Farbe	101
Farbcode	101
Farbattributspeicher	Anhang C
Fehlermeldung	
	17, 33, 37, 113, 117, Anhang H

Feld

- ,Element	66
- ,Index	67
- ,Variablenfeld	66, 108
- ,Zeichenkettenfeld	67, 108
Festwertspeicher	154
Format	37
FOR ... NEXT -Anweisung	56
FRE -Funktion	15, 110
Funktion, siehe BASIC	

G

Genauigkeit	28
Gleitkommazahlen	28
GOSUB ... RETURN -Anweisung	76
GOTO -Anweisung	52
Grafikzeichen	5, 27, 82, Anhang A und B
[GRAPHIC]-Taste	90
Gültigkeit von Variablen	108

H

Heimcomputer	
- ,Grundzustand des	8, 11, 160, 162
Hexadezimalsystem	150, Anhang D
Hintergrundfarbe	101, 104, 122

I

IF ... THEN ... :ELSE -Anweisung	54
Initialisierung	42, 107
INK -Anweisung	102
INKEY\$ -Funktion	88
INP -Funktion	130
INPUT -Anweisung	48, 94
[INS]-Taste	14
INSTR -Funktion	86
INT -Funktion	32
Interpreter	1
Interrupt	133

J

JOYST -Funktion	89
------------------------	----

K

Kanal	130, 132
Kanaladresse	130, Anhang E
Kassettengerät	
- ,Bedienung beim Abspeichern	112
- ,Bedienung beim Laden	15, 116
Kommando	26
Kommandomodus	2, 10
Kommentaranweisung	50
Konstante	
- ,numerische	27
- ,Zeichenkettenkonstante	29
Korrektur	9, 13, 63
Kursor	9, 92, 94, 97, 98, 126 Anhang E

L

Laden	
- ,eines BASIC-Programmes	15, 19, 114
- ,eines Feldes	116
- ,eines Maschinencodeprogramms	
	154, 156
Laufvariable	56
Leerzeichen	26, 29
LEFT\$ -Funktion	84
LEN -Funktion	83
LET -Anweisung	43

LINES -Kommando	41
LIST	
- ,Kommando	41, 47
- ,Taste	22, 39, 47
LIST* -Kommando	127
LIST# -Kommando	127
LN -Funktion	32
LOAD# -Kommando	128
Löschen	
- ,eines Programms	15, 37
- ,von Variablen	42, 108
M	
Maschinencodeprogramm	123
MEMORY SIZE	9, 108, 162
MID\$ -Funktion	85
N	
Neustart des BASIC-Interpreters	8
NEW -Kommando	15, 37, 109
NOT -Operator	33
NULL -Kommando	128
O	
OK-Ausschrift	10, 20
ON ... GOTO -Anweisung	52
ON ... GOSUB -Anweisung	78
Operation, logische	33
OR -Operator	33
OS-Ausschrift	5
OUT -Anweisung	99, 131
P	
PAPER -Anweisung	102, 104
Parameter	123
PAUSE	
- ,Anweisung	58
- ,Taste	7, 58, 61
PEEK -Funktion	119
PIO-Programmierung	132
POKE -Anweisung	118
POS -Funktion	97
PRINT -Anweisung	45, 93, 104, 129
PRINT AT -Anweisung	98, 105, 129

Priorität	33
Programm	
- ,Erstellung eines	19, 138
- ,geschütztes	15
- ,Neunumerierung	65
Programmablaufplan	136
Programmodus	2
R	
READ -Anweisung	69
?REDO FROM START	17, 49
REM -Anweisung	50
RENUMBER -Kommando	65
[RESET]-Taste	6, 8, 134
RESTORE -Anweisung	70, 109
RETURN , siehe GOSUB ... RETURN	
REWIND-Ausschrift	113, 114, 160
RIGHT\$ -Funktion	84
RND -Funktion	72, 120
Rücksprung	77
RUN	
- ,Kommando	42
- ,Taste	20, 42, 61
S	
SAVE -Kommando	159
Schlüsselwort	26, 30, 55, Anhang G
Schreib-Lese-Speicher (RAM)	3
Schrittweite	56
SGN -Funktion	32
[SHIFT]-Taste	5
[SHIFT LOCK]-Taste	5
SIN -Funktion	32
SPC -Funktion	91, 97
Speicheraufteilung	Anhang E
Speicherplatz	107, 118, 162
Spielhebel	89
Sprung	52
SQR -Funktion	32
Standardfunktion	32
STEP , siehe FOR ... NEXT	
STOP	
- ,Kommando	59
- ,Taste	5, 18, 59, 61
STR\$ -Funktion	83
STRING\$ -Funktion	86, 97

Syntaxfehler	13, 55, Anhang H
Systemuhr	119, 157
T	
TAB -Funktion	95
TAN -Funktion	32
Tastatur	4, Anhang A und B
Tastaturabfrage	88
Test	117, 140
TIME -Kommando	156
Tonwiedergabe	133, 151
TROFF -Kommando	117
TRON -Kommando	117
U	
Uhr, siehe Systemuhr	
Unterprogramm	75, 123, 161, Anhang F
USR -Funktion	125
V	
VAL -Funktion	82
Variable	29, 42, 107
Vereinbarung	
- ,eines Feldes	66
- ,einer Funktion	74
VERIFY-Ausschrift	112, 114, 160
Verkettung	35
Vordergrundfarbe	102, 104, 121
W	
WAIT -Anweisung	132
WBASIC , siehe BASIC	
WIDTH -Kommando	127, 129
WINDOW -Anweisung	92
Z	
Zeichenkette	13, 22, 35
Zeichenkettenspeicherbereich	19, 107, Anhang E
Zeilennummer	25, 38
Zufallszahl	72
Zwischencode	107, 111

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