SIEMENS

BASIC-Handbuch

Personal Computer PC 100

Vorläufige Ausgabe 1979/80

SIEMENS AKTIENGESELLSCHAFT

Vorwort

Bevor ein Computer eine sinnvolle Funktion ausführen kann, muß man ihm „sagen", was er tun soll. Leider verstehen Computer bis heute keine „menschliche Sprache." Der Grund dafür liegt in der Mehrdeutigkeit mancher Ausdrücke unserer Sprache.

Jedoch benötigt der Computer eindeutige Instruktionen und eine exakte Reihenfolge von auszuführenden Operationen, um eine spezifische Aufgabe erfüllen zu können. So wurden Programmiersprachen entwickelt, um die Kommunikation des Menschen mit dem Computer zu ermöglichen.

Die Programmiersprache BASIC – entwickelt an der Universität Darthmouth – hat auf dem Computergebiet breite Anwendung gefunden. Sie ist nicht nur leicht verständlich und problemlos anzuwenden, sondern sie ist vor allem äußerst leistungsfähig. Die Sprache läßt sich ausgezeichnet in Bereichen der Wirtschaft, Wissenschaft und Ausbildung anwenden.

Der Befehlsatz in BASIC besteht lediglich aus wenigen, bekannten englischen Ausdrücken. Da diese Sprache für Dialogverkehr ausgelegt ist, wird ein eingegebener Befehl wie z.B. "PRINT 2+2" sofort mit "4" beantwortet ausgegeben. Auch erübrigen sich Lochkarten und die damit verbundenen Wartezeiten. Vielmehr liegt die gesamte Computer-Kapazität sozusagen „unter Ihren Fingerspitzen."

Inhaltsverzeichnis

Herausgegeben von
Siemens AG, Bereich Bauelemente, Balanstraße 73, 8000 München 80.

Fragen über Technik, Preise und Liefermöglichkeiten richten Sie bitte an unsere Zweigniederlassungen im Inland, Abteilung VB oder an unsere Landesgesellschaften im Ausland (siehe Geschäftsstellenverzeichnis).

Um unsere Kunden unverzüglich mit detaillierten Informationen zu versorgen, haben wir uns entschlossen, eine „Vorläufige Ausgabe" zu produzieren, die wir Ihnen hiermit vorstellen. Die endgültige, redaktionell und technisch überarbeitete Ausgabe dieser Druckschrift wird derzeit erarbeitet. Sie steht auf Wunsch voraussichtlich ab 1. 10. 1979 zur Verfügung. Bitte benutzen Sie bei Bedarf die beiliegende Bestellkarte.

1. Starten und Betreiben

Um BASIC erstmals zu laden, geben Sie "5" ein, wenn das Monitor-Prompt-Zeichen auf dem Display angezeigt wird (<).
BASIC wird antworten mit:

```
< 5 >
MEMORY SIZE? ^
```

Falls Sie nach MEMORY SIZE? die RETURN-Taste (= Wagenrücklauf) drücken, wird BASIC den gesamten zusammenhängenden Speicher ab der Adresse 512 (2Ø0 Hexadezimal) nach oben, belegen, den es finden kann. BASIC wird mit der Suche aufhören, nachdem es ein Byte ROM oder eine nichtexistierende Speicherzelle gefunden hat.

Falls Sie BASIC nur einem Teil des Speichers zuweisen wollen, geben Sie die Zahl der Speicherbytes, die Sie zuweisen wollen, dezimal ein. Das könnte z. B. dann getan werden, wenn Sie einen Teil des Speichers für ein Maschinensprache-Unterprogramm verwenden wollen. Die höchste Adresse ist 4Ø95 (ØFFF hex) im 4K System. Schließen Sie die Eingabe durch Drücken der Taste RETURN ab.
BASIC wird dann fragen:

```
WIDTH? ^
```

Damit wird nur für PRINT-Anweisungen die Länge der Ausgabezeile festgelegt. Geben Sie die Zahl der Zeichen für die Zeilenlänge in Dezimalform ein, die der Drucker oder ein anderes Ausgabegerät braucht. Das kann jede Zahl zwischen 16 und 255 sein und hängt vom Terminal ab. Falls keine Antwort gegeben wird (d. h. RETURN wird gedrückt), wird die Zeilenlänge auf 2Ø Zeichen festgelegt.
Nun wird BASIC ausgeben:

```
XXXX BYTES FREE
SIEMENS PC 100 BASIC
```

Anmerkung!
"XXXX" ist die Zahl der Bytes für das Programm, Variable, Matrix-Speicherung und String-Platz.

BASIC befindet sich nun im Kommandoeingabemodus, was durch den BASIC-Cursor "(^)" angezeigt wird. Im Kapitel Programmieren könnten Sie nun mit BASIC beginnen. Aber bitte lesen Sie die folgenden Abschnitte zuerst, um zu verstehen, wie Sie in BASIC eintreten, es wieder verlassen und wie der BASIC-Cursor funktioniert.

Achtung!
Der Eintritt in BASIC mit der Taste 5 bewirkt, daß der gesamte zugewiesene Speicher mit AA (Hexadezimal) beschrieben wird, beginnend ab Adresse 532. Das zerstört natürlich alle alten BASIC-Programme, die sich vielleicht in diesem Teil des Speichers befanden, wie z. B. Daten im PC 100-Textpuffer oder im Maschinensprache-Unterprogramm. Vergewissern Sie sich, daß Sie alle noch benötigten Daten oder Programme in diesem Bereich gesichert haben, bevor Sie mit der Taste 5 nach BASIC eintreten. Texte im Textpuffer oder Maschinenunterprogrammen können durchaus gemeinsam mit BASIC im Speicher stehen. Sie müssen dazu solche Routinen oder Texte in den oberen Bereich des Speichers legen und BASIC die höchste Adresse unterhalb Ihres Textpuffers übergeben.

1.1 Beenden

Um BASIC zu beenden und in den PC 100-Monitor zurückzukehren, betätigen Sie die Taste ESC und zwar immer dann, wenn der BASIC-Cursor angezeigt wird. Sie können auch während des Ablaufes eines Programms BASIC verlassen; dazu drücken Sie die F1-Taste und dann die ESC-Taste. Das Betätigen der RESET-Taste wird auch den Eintritt in den PC 100-Monitor veranlassen, genauso wie ein Hardware-RESET des PC 100. (Hardware RESET = Netzschalter betätigen).

1.2 Wiedereintritt

Sie können immer, wenn der Monitor das Promt-Zeichen " < " anzeigt, durch Betätigen der Taste 6 wieder in BASIC eintreten. Jedoch bleibt in diesem Fall ein vorhandenes BASIC-Programm im Speicher erhalten. PC 100 wird auf die Betätigung der Taste 6 wie folgt reagieren:

```
^ 6 >
```

1.3 BASIC Cursor

Der Basic-Cursor (^), der immer an der Stelle 1 der Anzeige erscheint, wenn BASIC im Kommandoeingabemodus ist, zeigt an, daß ein BASIC-Befehl eingegeben werden kann. Die zuletzt angezeigten Daten, die vom vorigen Befehl stammen, bleiben erhalten um einen Zusammenhang der Information zwischen dem vorherigen Befehl oder den angezeigten Daten zu gewährleisten (ausgenommen davon ist die Stelle 1). Das ist besonders nützlich, wenn der Drucker ausgeschaltet ist, um Papier zu sparen. Sobald das erste Zeichen des neuen Befehls eingegeben wird, wird die Anzeige mit Ausnahme der neu eingegebenen Zeichen gelöscht. Der Cursor wandert dann – in Abhängigkeit von dem eingegebenen Zeichen – über die Anzeige, um die Eingabestelle zu kennzeichnen. Er erscheint nicht bei der Druckerausgabe, damit werden auch Daten in Stelle 1 erhalten.

Achtung!

Das Minuszeichen etwaiger negativer Werte, die von der Stelle 1 an angezeigt werden, wird im Kommandoeingabemodus vom Cursor überschrieben. Bei direkten Befehlen wird, bevor der Cursor angezeigt wird und falls der Drucker eingeschaltet ist, das Minuszeichen kurz aufleuchten. Falls der Drucker ausgeschaltet ist, wird es überhaupt nicht erscheinen. Um das Minuszeichen zu speichern, sollte immer ein führendes Leerzeichen eingegeben werden, bevor der Wert angezeigt wird.

1.4 Kontrolle des Druckers

Im Kommandoeingabemodus kann der Drucker aus- oder eingeschaltet werden, indem man Taste PRINT mit gedrückter CTRL-Taste betätigt (CTRL PRINT). Der Ein-/Auszustand des Druckers wird nach Eingeben von CTRL PRINT angezeigt. Falls der Drucker ausgeschaltet ist, werden Anweisungen im BASIC Kommandoempfangsmodus und Datenausgabe von PRINT-Kommandos nur zum Display geleitet. Falls der Drucker eingeschaltet ist, werden alle Kommandos und Daten von PRINT-Anweisungen sowohl zum Drucker als auch zum Display geleitet.
Bei ausgeschaltetem Drucker können wichtige Daten aber immer noch durch die Verwendung des PRINT! – Befehl zum Drucker geleitet werden.
Analog werden bei INPUT-Anweisungen Daten in Abhängigkeit des Druckerzustandes ausgegeben.
Eine INPUT! – Anweisung wird Daten auch dann am Drucker ausgeben, wenn er ausgeschaltet ist.
Vorm Einlesen eines BASIC-Programms von der Kassette sollte der Drucker ausgeschaltet sein.

1.5 TTY – Zeilenlängen

Die maximale Zeilenlänge beträgt 72 Zeichen. Falls Sie mehr Zeichen in einer Zeile einzugeben versuchen, wird ein BELL(ASCII–7)– (Klingel) ausgeführt und die weiter eingetippten Zeichen werden nicht wieder ausgegeben (Echo). Sie können nun entweder die DEL-Taste betätigen um einen Teil der Zeile oder mit Taste @ die gesamte Zeile löschen. Das Zeichen, das Sie eingegeben haben, ist nicht mehr in die Zeile eingefügt worden, da es auf der Zeichenstelle 1 nach dem Zeilenende steht.

2. Programmieren

Ziel dieses Kapitels ist, Sie in die Grundgedanken und Anwendungsmöglichkeiten von BASIC einzuführen, um es Ihnen zu ermöglichen, erste Programme zu schreiben. Zur detaillierten Einarbeitung machen wir Sie auf viele BASIC-Bücher des Fachhandels aufmerksam.

Wir empfehlen Ihnen, jedes Beispiel dieses Kapitels gleich dann auszuführen, wenn es vorgestellt wird. Das wird Ihr Gefühl für BASIC und seine Anwendungsmöglichkeiten vertiefen.

Anmerkung!

Nachdem BASIC gestartet ist, kann immer dann, wenn der Cursor (^) ganz links auf der Anzeige steht, eine BASIC-Anweisung geschrieben werden. Alle Kommandos an BASIC müssen mit einem "RETURN" abgeschlossen werden. Das Drücken der RETURN-Taste teilt BASIC mit, daß Sie das Eintragen des Kommandos beendet haben. Falls Sie einen Tippfehler machen, müssen Sie die Taste DEL (Delete) oder RUBOUT am Fernschreiber betätigen, um das zuletzt eingegebene Zeichen zu löschen. Durch wiederholtes Betätigen der DEL-Taste werden vorangegangene Zeichen gelöscht. Die Taste @ löscht die gesamte eingetragene Zeile.

Tabelle 1

BASIC-Kommandos

Befehle

CLEAR
CONT
FRE
LIST
LOAD
NEW
PEEK
POKE
POS
RUN
SAVE

Programm-Anweisungen

DEF
DIM
END
FOR
GOSUB
GOTO
IF . . . GOTO
IF . . . THEN
LET
NEXT
ON . . . GOSUB
ON . . . GOTO
REM
RESTORE
RETURN
STOP
USR
WAIT

Ein-/Ausgabe-Anweisungen

DATA
INPUT
INPUT!
PRINT
PRINT!
?
READ
SPC
TAB

String-Funktionen

ASC
CHR$
GET
LEFT$
LEN
MID$
RIGHT$
STR$
VAL
+

Arithmetische Funktionen

ABS
ATN [1])
COS
EXP
INT
LOG
RND
SGN
SIN
SQR
TAN

Logische Verknüpfungen

AND
NOT
OR

[1]) Obwohl die Funktion ATN in PC 100 BASIC nicht enthalten ist, wird der ATN-Befehl beachtet. (siehe Abschnitt 4.9).

2.1 Direkte Befehle

Nun versuchen Sie, die folgende Zeile einzugeben:

PRINT 1Ø–4 (Abschluß mit RETURN-Taste)

BASIC wird sofort ausgegeben:

```
 6
```

Die PRINT-Anweisung, die Sie eingegeben haben, wurde sofort nach dem Betätigen der RETURN-Taste ausgeführt. Das bezeichnet man als direktes Kommando. BASIC berechnet den Ausdruck nach PRINT und gibt dann den Wert aus, in diesem Fall den Wert "6."

Nun versuchen Sie, dies einzugeben:

PRINT 1/2,3 * 1Ø (* bedeutet Multiplikation /Division)

BASIC wird ausgeben:

```
 .5            3Ø
```

Geben Sie folgende Zeile ein:

PRINT 19+4

BASIC wird antworten:

```
 2 3
```

Nun versuchen Sie es mit:

? 19+4

BASIC wird ausgeben:

```
 2 3
```

Sie erkennen ganz richtig, daß Sie sowohl mit "PRINT", als auch mit "?" zum selben Ergebnis kommen. Sie können also überall, wo ein "PRINT"-Kommando steht, bei der Eingabe auch das Kurzzeichen "?" verwenden.

Wie Sie sehen, kann BASIC sowohl dividieren, multiplizieren, addieren und subtrahieren. Merken Sie sich bitte, daß ein "," (Komma) innerhalb der PRINT-Anweisung dazu verwendet wurde, zwei statt einen Wert auszudrucken. Die Anweisung teilt eine Zeile in Spalten von zehn Zeichen Breite ein. Das "," veranlaßt BASIC, zum nächsten 1Ø-Zeichen-Feld des Terminals zu springen, um dort den Wert 3Ø zu drucken.

2.2 Indirekte Befehle

Es gibt einen anderen Befehlstyp, den man „indirekten Befehl" nennt. Jeder indirekte Befehl beginnt mit einer Zeilennummer. Eine Zeilennummer ist jede ganze Zahl zwischen Ø und 63999.

Versuchen Sie, diese Zeilen einzugeben:

```
1Ø PRINT 2+3
2Ø PRINT 2–3
```

Eine Folge von indirekten Befehlen wird „Programm" genannt.

Statt indirekte Befehle sofort auszuführen, speichert BASIC indirekte Befehle ab. Sobald Sie RUN eingeben, wird BASIC zuerst den indirekten Befehl mit der kleinsten Zeilennummer, ausführen; dann den Befehl mit der nächstgrößeren Zeilennummer usw., soviele Befehle wie eingegeben wurden werden ausgeführt. Im obigen Beispiel gaben wir zuerst die Zeile 1Ø und dann die Zeile 2Ø ein. Es ist jedoch gleich, in welcher Reihenfolge Sie indirekte Anweisungen eingeben. BASIC ordnet Sie immer korrekt in aufsteigender Reihenfolge der Zeilennummern.

Geben wir nun ein:

RUN

So wird BASIC

```
 5
-1
```

ausgeben.

2.3 Auflisten eines Programms

Falls wir einen Ausdruck des kompletten Programms wünschen, das sich zur Zeit im Speicher befindet, geben wir ein:

LIST

BASIC wird mit

```
1Ø PRINT 2+3
2Ø PRINT 2-3
```

antworten.

2.4 Löschen einer Zeile

Manchmal ist es wünschenswert, eine ganze Zeile eines Programms zu löschen. Dies wird durch das Eingeben der Zeilennummer ~~und der zu löschenden Zeile~~ erreicht, gefolgt von einem Drücken der RETURN-Taste,
Tippen Sie ein:

1Ø
~~LIST~~ RETURN

und BASIC wird mit

```
2Ø PRINT 2-3
```

antworten.

Wir haben nun die Zeile 1Ø im Speicher gelöscht.

2.5 Ersetzen einer Zeile

Sie können die Zeile 1Ø nicht nur löschen, sondern auch durch eine neue Zeile ersetzen, indem Sie eine neue Zeile 1Ø eingeben und RETURN betätigen.

Geben Sie ein:

1Ø PRINT 3-3
LIST

BASIC wird antworten mit:

```
1Ø PRINT 3-3
2Ø PRINT 2-3
```

Es ist nicht empfehlenswert, Zeilen ohne Abstand durchgehend zu numerieren. Es kann notwendig werden, zwischen zwei existierenden Zeilen eine neue Zeile einzufügen. Ein Abstand von 1Ø zwischen den Zeilennummern ist im Allgemeinen ausreichend.

2.6. Löschen eines Programms

Wollen Sie das komplette Programm löschen, das sich gerade im Speicher befindet, geben Sie "NEW" ein. Falls Sie ein Programm nicht mehr benötigen und ein neues einlesen wollen, denken Sie daran, vorher "NEW" einzugeben.

Geben Sie ein:

NEW

BASIC wird antworten:

```
^ EW
```

Nun geben Sie ein:

LIST

BASIC wird antworten mit

```
^ IST
```

Anmerkung!

Am Drucker erscheint "NEW" bzw. "LIST" während auf der Anzeige das "N" bzw. "L" durch den Cursor überschrieben wird.

Führende Nullen werden nie gedruckt, d.h. die Stelle vor dem Dezimalpunkt ist niemals Null. Nullen am Ende werden auch nie ausgewiesen. Falls nur eine Ziffer übrigbleibt, nachdem alle Endnullen unterdrückt sind, wird kein Dezimalpunkt gedruckt. Das Vorzeichen des Exponenten ist " + " für positive und " – " für negative Exponenten. Die zwei Stellen des Exponenten werden immer gezeigt, das bedeutet, daß Nullen im Exponentenfeld nicht unterdrückt werden. Der Wert einer Zahl, die so dargestellt wird, ist die Zahl links vom " E " mal 1Ø hoch der Zahl rechts vom " E ".

Gleichgültig welches Format gewählt wird, ein Zwischenraum wird immer nach einer Zahl gedruckt. BASIC prüft, ob die ganze Zahl noch in die aktuelle Zeile paßt. Falls das nicht mehr der Fall ist, wird vor dem Druck der Zahl eine Wagenrücklauf-Zeilenvorschub-Kombination ausgegeben.

Beispiele:

Zahl	Ausgabeformat
+1	1
−1	−1
6523	6523
−23.46Ø	−23.46Ø
1E2Ø	1E+2Ø
−12.3456E−7	−1.23456E−Ø6
1.234567E−1Ø	1.234567E−1Ø
1ØØØØØØØØØ	1E+Ø9
999999	999999
.1	.1
.Ø1	.Ø1
.ØØØ123	1.23E−Ø4

Eine Zahl, die von der Tastatur eingegeben wird, oder eine Zahlenkonstante in einem BASIC-Programm, kann soviele Stellen wie gewünscht haben, bis zur Maximallänge einer Zeile (72 Zeichen), jedoch werden nur die ersten 10 Stellen berücksichtigt, und die zehnte Stelle wird aufgerundet.

z.B.

```
PRINT 1.23456789876543210
 1.2345679
```

2.9 Eingabe von Variablen mit dem INPUT-Befehl

Nun folgt ein Programmbeispiel, worin ein Wert über die Tastatur eingelesen wird und dazu verwendet wird, ein Ergebnis zu berechnen und auszugeben.

```
1Ø INPUT R
2Ø PRINT 3.14159* R * R
RUN
? 1Ø
 314.159
```

Was geschieht? Sobald BASIC auf die INPUT-Anweisung stößt, gibt es ein Fragezeichen ("?") auf dem Display aus und wartet nun, bis Sie eine Zahl eingeben. Sobald Sie das getan haben (im obigen Beispiel wurde 1Ø eingegeben), wird die Programmausführung bei der nächsten Anweisung fortgesetzt, nachdem der Variablen (R) der aktuelle Wert (1Ø) zugewiesen wurde. Im obigen Beispiel wird nun Zeile 2Ø ausgeführt. Wenn die Formel in der PRINT-Anweisung berechnet wird, wird die Variable " R ", sooft sie in der Formel auftaucht, durch den Wert 1Ø ersetzt. Somit wird die Formel zu

3.14159 * 1Ø * 1Ø oder 314.159

Falls Sie es nicht schon erkannt haben: Das Programm berechnet die Fläche eines Kreises mit dem Radius " R ". Falls wir die Fläche für verschiedene Radien berechnen wollten, könnten wir das Programm jedesmal neu starten.

2.7 Ausdrucken von Daten

Oft ist es wünschenswert, Text mit den ausgedruckten Antworten auszugeben, um etwa die Bedeutung der Zahlen zu erläutern.

Geben Sie ein:

PRINT "1/2=",1/2

und BASIC wird antworten mit:

```
1/2 =        .5
```

Wie schon im Kapitel 2.1 „Direkte Befehle" erläutert, bedeutet ein in eine PRINT-Anweisung eingefügtes "," zum Beginn eines neuen Zehnerfeldes zu springen, bevor der dem "," folgende Wert gedruckt wird.

Falls wir einen ";" (Strichpunkt) anstelle eines Kommas verwenden, wird der nächste Wert unmittelbar nach dem vorangegangenen gedruckt.

Anmerkung!

Zahlen werden immer mit mindestens einem Zwischenraum gedruckt. Jeder Text, der gedruckt werden soll, muß immer zwischen Anführungszeichen stehen.

Probieren Sie folgende Beispiele:

```
1. PRINT "1/2=" ;1/2
   1/2=  .5
2. PRINT 1,2,3
    1          2
    3
3. PRINT 1;2;3
    1 2 3
4. PRINT -1;2;-3
   -1 2 -3
```

2.8 Zahlenformat

Wir werden nun etwas abschweifen, um das Format von Zahlen in BASIC zu erklären. Zahlen werden intern mit über neun Stellen Genauigkeit gespeichert. Wenn eine Zahl gedruckt wird, werden nur neun Ziffern ausgegeben. Jede Zahl kann auch einen Exponenten haben (eine Potenz von 10 als Multiplikand).

Die größte Zahl, die in PC 100 BASIC darstellbar ist, ist

$1.70141183 * 10^{38}$,

während die kleinste positive Zahl

$2.93873588 * 10^{-39}$

ist.

Wenn eine Zahl ausgedruckt wird, definieren die folgenden Regeln das Format:

1. Falls die Zahl negativ ist, wird ein Minuszeichen (–) ausgedruckt. Falls die Zahl positiv ist, wird ein Zwischenraum gedruckt.
2. Falls der Absolutwert der Zahl eine ganze Zahl zwischen 0 und 999 999 999 ist, wird die Zahl als ganze Zahl gedruckt.
3. Falls der Absolutwert der Zahl größer gleich 0.01 oder kleiner gleich 999 999 999 ist, wird die Zahl im Festkommaformat ohne Exponent gedruckt.
4. Falls die Zahl nicht unter die Gruppen 2 und 3 fällt, wird die wissenschaftliche Darstellung gewählt.

Das Format der wissenschaftlichen Darstellung (scientific notation) sieht allgemein so aus:

SX.XXXXXXXXESTT (Jedes X steht für eine ganze Zahl zwischen 0 und 9)

Das führende S ist das Vorzeichen der Zahl: Ein Zwischenraum für eine positive und ein " – " für eine negative Zahl. Eine von Null verschiedene Ziffer wird vor dem Dezimalpunkt gedruckt. Danach kommt der Dezimalpunkt und dann die anderen acht Stellen der Mantisse. Sodann wird ein "E" gedruckt (für Exponent), danach das Vorzeichen (S) des Exponenten; weiter die zwei Stellen (TT) des eigentlichen Exponenten.

Es gibt aber eine einfachere Möglichkeit:
Wir hängen eine weitere Zeile an das Programm an.

```
30 GOTO 10
RUN
?10
 314.159
?3
 28.27431
?4.7
 69.3977231
?
^
```

Durch das Einfügen einer "GOTO"-Anweisung an das Ende unseres Programms haben wir es dazu gebracht, nach jedem Ausdruck des Ergebnisses an die Zeile 10 zurückzukehren. Das könnte beliebig oft gehen, aber wir entschlossen uns, nach der Berechnung des dritten Kreises aufzuhören. Dies wurde durch Betätigen der RETURN-Taste nach der Eingabeaufforderung (Eingabe einer Leerzeile) erreicht.

2.10 Variablennamen

Den Buchstaben "R", den wir gerade in unserem Programm verwendet haben, nennt man eine "Variable". Ein Variablenname kann jeder Buchstabe des Alphabetes sein; ihm kann ein beliebiges alphanumerisches Zeichen (Buchstaben A bis Z, Zahlen 0 bis 9) folgen. Weitere alphanumerische Zeichen nach den beiden ersten werden ignoriert.

Hier einige Beispiele für erlaubte und verbotene Variablenamen:

Erlaubt	Verboten	Anmerkungen
A	%	Erstes Zeichen muß ein Buchstabe sein
Z1	Z1A Z1B	gleiche Variablen, da jedes Zeichen nach den beiden ersten ignoriert wird
TP PSTGS	TO	Variablenamen dürfen keine reservierten Worte sein.
COUNT	RGOTO	Variablenamen dürfen keine reservierten Worte beinhalten.

2.11 Reservierte Worte

Worte, die als BASIC-Anweisungen verwendet werden, sind für diesen spezifischen Anwendungsfall reserviert. Sie dürfen diese Worte nicht innerhalb eines Namens oder als Variablennamen verwenden. "ENDE" zum Beispiel wäre falsch, da "END" ein reserviertes Wort ist.

In Tabelle 2 sind alle reservierten Worte des PC 100 – BASIC zusammengefaßt.

Tabelle 2

Reservierte Worte

ABS	FN	LOAD	READ	STOP
AND	FOR	LOG	REM	STR$
ASC	FRE	MID$	RESTORE	TAB
ATN	GET	NEW	RETURN	TAN
CHR$	GOSUB	NEXT	RIGHT$	THEN
CLEAR	GOTO	NOT	RND	TO
CONT	IF	NULL	RUN	USR
COS	INPUT	ON	SAVE	VAL
DATA	INT	OR	SGN	WAIT
DEF	LEFT$	PEEK	SIN	
DIM	LEN	POKE	SPC	
END	LET	PO$	SQR	
EXP	LIST	PRINT	STEP	

2.12 Wertzuweisungen

Neben der Wertzuweisung an Variable durch eine INPUT-Anweisung kann man eine Wertzuweisung auch durch eine LET- oder Zuordnungsanweisung durchführen.
Probieren Sie die folgenden Beispiele aus:

```
A=5
PRINT A,A*2
  5          1Ø
LET Z=7
PRINT Z,Z-A
  7          2
```

Wie sie aus den Beispielen sehen, kann man das "LET" in einer Zuordnungsanweisung auch entfallen lassen. BASIC „merkt" sich die Werte, die durch diese Art von Anweisung den Variablen zugewiesen worden sind. Dieser „Merkprozeß" benötigt Speicherplatz, um die Daten zu speichern. Die Werte der Variablen werden gelöscht, und der zur Speicherung verwendete Speicherplatz wird freigegeben, wenn folgendes geschieht:

- ○ Eine neue Zeile wird in das Programm eingefügt oder eine alte Zeile wird gelöscht.
- ○ Ein CLEAR-Kommando wird eingegeben.
- ○ Ein RUN-Kommando wird eingegeben.
- ○ NEW wird eingegeben.

Jede Variable im Programm muß einen bestimmten Wert besitzen. Falls Sie einer Variablen keinen Wert zuweisen – falls Sie im Programm zum 1. Mal auftaucht – erhält sie automatisch den Wert "Null."

Probieren Sie:

```
PRINT Q,Q+2,Q*2
Ø            2          Ø
```

2.13 Bemerkungen

Die REM-Anweisung (REM für engl. „Remark" = „Bemerkung") wird zum Einfügen von Kommentaren oder Hinweisen in ein Programm verwendet. Sobald BASIC eine REM-Anweisung entdeckt, wird der Rest der Zeile ignoriert.

Dies dient als Unterstützung des Programmierers und desjenigen, der das Programm später lesen und verstehen muß. Eine weitere Funktion hat diese Anweisung nicht.

2.14 Bedingungsprüfungen

Angenommen wir wollen ein Programm schreiben, das prüft, ob eine Zahl gleich Null ist. Mit den bisher bekannten Anweisungen wäre dies nicht möglich. Wir benötigen eine Anweisung, die einen bedingten Sprungbefehl zu einer anderen Anweisung durchführt. Die "IF-THEN"-Anweisung tut genau das.

Tragen Sie folgendes Programm ein:
(Vergessen Sie nicht, vorher "NEW" einzugeben).

```
1Ø INPUT B
2Ø IF B=Ø THEN 5Ø
3Ø PRINT "UNGLEICH NULL"
4Ø GOTO 1Ø
5Ø PRINT "NULL"
6Ø GOTO 1Ø
```

Sobald das Programm eingegeben und gestartet ist, wird es nach einem Wert für B fragen. Geben Sie irgendeinen Wert ein. Der PC 100 wird dann zur IF-Anweisung gelangen. Zwischen dem "IF" und dem "THEN"-Teil der Anweisung stehen zwei Ausdrücke, die durch eine "Relation" getrennt sind.

Relation	Bedeutung
=	gleich
>	größer als
<	kleiner als
<>	ungleich
< = oder = <	kleiner gleich
= > oder > =	größer gleich

Die IF-Anweisung ist entweder richtig oder falsch, je nachdem ob die beiden Ausdrücke die Relation erfüllen oder nicht. Falls beispielsweise in dem Programm, das soeben eingegeben wurde, die Ø für B eingetippt worden wäre, wäre die IF-Anweisung wahr, da Ø = Ø ist. In diesem Fall würde die Programmausführung bei der Zeile 5Ø fortgesetzt, da 5Ø die Zahl hinter THEN ist. Somit würde "NULL" ausgedruckt und im Programm zur Zeile 10 zurückgegangen werden. (Auf Grund der GOTO-Anweisung in Zeile 6Ø)

Angenommen, für B wäre eine 1 eingegeben worden. Da 1 = Ø falsch ist, ist die IF-Anweisung falsch, und die Programmausführung geht mit der nächsten Zeile weiter. Somit würde "UNGLEICH NULL" ausgedruckt werden und die GOTO-Anweisung in Zeile 4Ø würde das Programm zur Zeile 1Ø zurückschicken.

Nun versuchen Sie das Programm, das zwei Zahlen vergleicht:

```
1Ø INPUT A,B
2Ø IF A < = B THEN 5Ø
3Ø PRINT "A IST GROESSER"
4Ø GOTO 1Ø
5Ø IF A < B THEN 8Ø
6Ø PRINT "SIE SIND GLEICH GROSS"
7Ø GOTO 1Ø
8Ø PRINT "B IST GROESSER"
9Ø GOTO 1Ø
```

Sobald dieses Programm abläuft, wird Zeile 1Ø zwei Zahlen vom Terminal einholen. Ist A größer B, so wird die Relation A < = B falsch sein. Dadurch wird die nächste Anweisung ausgeführt werden, die "A IST GROESSER" ausdruckt; Zeile 4Ø schickt dann den Computer zu Zeile 1Ø zurück, um wieder zu beginnen.

Falls A denselben Wert wie B hat, ist in Zeile 2Ø die Relation A < = B wahr – somit geht's zur Zeile 5Ø. In Zeile 50 ist, da A gleich B ist, die Relation A < B falsch; deswegen gehen wir zur folgenden Anweisung, die "SIE SIND GLEICH GROSS" ausdruckt. Dann schickt uns Zeile 7Ø wieder zurück zum Anfang.

Ist A kleiner B, ist in Zeile 2Ø die Relation A < = B wahr und wir gehen zur Zeile 5Ø. In Zeile 5Ø ist A < B wahr und es geht zur Zeile 8Ø, wo "B IST GROESSER" gedruckt wird. Zeile 9Ø schickt uns wieder an den Anfang.

Versuchen Sie die beiden letzten Programme einige Male. Zum leichteren Verständnis können Sie nun Ihr eigenes Programm mit der IF . . .THEN-Anweisung schreiben. Eigene Programme auszuprobieren und aus den Fehlern zu lernen ist der leichteste Weg, um die Arbeitsweise von BASIC zu verstehen. Denken Sie daran, daß Sie bei der Eingabe-Abfrage die RETURN-Taste drücken müssen, um die Programme anzuhalten.

2.15 Schleifen

Ein Vorteil von Computern ist die Fähigkeit, wiederkehrende Aufgaben auszuführen. Sehen wir uns nun genauer an, wie das funktioniert.
Angenommen, wir benötigen eine Tabelle der Wurzeln von 1 bis 1Ø. Die BASIC-Funktion zur Wurzelberechnung ist "SQR" (Square root); die Form ist SQR(X), wobei X die Zahl ist, deren Wurzel berechnet werden soll. Wir können das Programm wie folgt schreiben:

```
 1Ø PRINT 1,SQR (1)
 2Ø PRINT 2,SQR (2)
 3Ø PRINT 3,SQR (3)
 4Ø PRINT 4,SQR (4)
 5Ø PRINT 5,SQR (5)
 6Ø PRINT 6,SQR (6)
 7Ø PRINT 7,SQR (7)
 8Ø PRINT 8,SQR (8)
 9Ø PRINT 9,SQR (9)
1ØØ PRINT 1Ø,SQR (1Ø)
```

Dieses Programm wird funktionieren, es ist aber furchtbar aufwendig. Wir können das Programm durch Einsatz der IF-Anweisung beträchtlich verbessern.

```
1Ø N = 1
2Ø PRINT N, SQR(N)
3Ø N = N+1
4Ø IF N < = 1Ø THEN 2Ø
```

Sobald dieses Programm abläuft, wird seine Ausgabe genauso wie das des oberen Programms mit den zehn Anweisungen aussehen.

Sehen wir uns an, wie es funktioniert.

In Zeile 1Ø finden wir eine LET-Zuweisung (das Wort "LET" wurde weggelassen), die der Variablen N den Wert 1 zuweist. In Zeile 2Ø drucken wir N und dessen Quadratwurzel, jeweils mit dem derzeitigen Wert von N. So wird die Zeile zu 2Ø PRINT 1, SQR (1), und das Ergebnis wird ausgegeben. In Zeile 3Ø finden wir eine LET-Zuweisung, die auf den ersten Blick ziemlich ungewöhnlich aussieht. Mathematisch ist die Gleichung N = N+1 unsinnig. Wir müssen uns jedoch an die wichtige Tatsache erinnern, daß in einer LET-Zuweisung das Symbol " = " nicht Gleichheit bedeutet. In diesem Fall bedeutet " = ": "Zu ersetzen durch . . ."
Diese Anweisung nimmt also nun den aktuellen Wert von N und addiert 1 dazu. Somit wird N, nachdem wir das erste Mal durch Zeile 3Ø gekommen sind, zu 2.

In Zeile 4Ø ist N < = 1Ø wahr, da ja N nun 2 ist. Damit bringt uns die IF . . . THEN-Anweisung zurück zur Zeile 2Ø, mit dem Wert 2 für N.
Das Ergebnis dieses Programms ist also, daß die Zeilen 2Ø bis 4Ø wiederholt werden, wobei jedesmal 1 zum Wert von N addiert wird. Hat N in Zeile 2Ø den Wert 1Ø, wird es durch die nächste Zeile zu 11 erhöht. Dies bewirkt eine falsche Anweisung in Zeile 4Ø, und da nach ihr keine weiteren Anweisungen folgen, stoppt das Programm.

Diese Technik ist unter den Begriffen "Schleifenbildung" oder "Iteration" bekannt. Da sie beim Programmieren häufig angewandt wird, gibt es spezielle BASIC-Anweisungen dafür. Wir sehen sie im folgenden Programm:

```
1Ø FOR N = 1 TO 1Ø
2Ø PRINT N, SQR (N)
3Ø NEXT N
```

Das Ausgabeergebnis dieses Programms wird genau das gleiche sein, wie das der beiden vorigen Programme.

In Zeile 1Ø wird N gleich 1 gesetzt.
Zeile 2Ø druckt den Wert N und dessen Quadratwurzel. In Zeile 3Ø sehen wir eine neue Anweisung. Die NEXT N-Anweisung veranlaßt die Addition von 1 zu N und danach einen Rücksprung zu der Anweisung, die der FOR-Anweisung folgt. Die Gesamtfunktion ist die gleiche wie im vorigen Beispiel.

Beachten Sie, daß die Variable nach dem "FOR" genau dieselbe wie nach dem "NEXT" ist. Dies muß nicht unbedingt "N" sein; jede Variable könnte verwendet werden, solange sie dieselbe in sowohl der "FOR"- als auch der "NEXT"-Anweisung ist. Es könnte etwa wie im vorstehenden Programm "N" überall durch "Z1" ersetzt werden, die Funktion wäre exakt dieselbe.

Nehmen wir an, daß wir eine Tabelle der Quadratwurzeln jeder geraden Zahl zwischen 1Ø und 2Ø ausdrucken wollen. Das folgende Programm erfüllt diese Aufgabe:

```
1Ø N = 1Ø
2Ø PRINT N, SQR (N)
3Ø N = N+2
4Ø IF N < = 2Ø THEN 2Ø
```

Beachten Sie die ähnliche Struktur dieses und des auf Seite 14 aufgelisteten Programms, das die Wurzeln der Zahlen zwischen 1 und 1Ø berechnet. Dieses Programm kann auch mit der eben vorgestellten "FOR"-Schleife geschrieben werden.

```
1Ø FOR N = 1Ø TO 2Ø STEP 2
2Ø PRINT N, SQR (N)
3Ø NEXT N
```

Sie werden bemerken, daß der Hauptunterschied zwischen diesem und dem vorherigen Programm, bei dem die FOR-Schleife angewandt wurde, das Hinzufügen der Anweisung STEP 2 ist.

So sagt BASIC zu N jedesmal 2 anstelle der 1 im vorigen Programm zu addieren. Falls in einer "FOR"-Anweisung kein "STEP" gegeben wird, nimmt BASIC an, daß jedesmal 1 addiert werden soll. Nach STEP kann ein beliebiger Ausdruck stehen.

Nehmen wir an, wir möchten rückwärts von 1Ø bis 1 zählen. Ein Programm, das so funktioniert, würde wie folgt aussehen:

```
1Ø I = 1Ø
2Ø PRINT I
3Ø I = I–1
4Ø IF I > = 1 THEN 2Ø
```

Beachten Sie, daß wir I daraufhin prüfen, ob es größer oder gleich dem Endwert ist. Das kommt daher, daß wir nun in negativer Richtung zählen. In den vorangegangenen Beispielen war das Gegenteil der Fall, und somit prüfen wir die Variable, ob sie kleiner oder gleich dem Endwert war.

Die vorher gezeigte STEP-Anweisung kann auch mit negativen Zahlen arbeiten und erzielt dieselben Resultate. Das kann im gleichen Format wie in den anderen Programmen erfolgen:

```
1Ø FOR I = 1Ø TO 1 STEP -1
2Ø PRINT I
3Ø NEXT I
```

"FOR"-Schleifen können auch verschachtelt werden.

```
1Ø FOR I = 1 TO 5
  2Ø FOR J = 1 TO 3
    3Ø PRINT I,J
  4Ø NEXT J
5Ø NEXT I
```

Beachten Sie bitte, daß zuerst "NEXT J", dann "NEXT I" kommt. Der Grund dafür ist, daß die J-Schleife innerhalb der I-Schleife liegt.

Folgendes Programm ist nicht korrekt; starten Sie es und beobachten Sie, was geschieht.

```
1Ø FOR I = 1 TO 5
  2Ø FOR J = 1 TO 3
    3Ø PRINT I,J
  4Ø NEXT I
5Ø NEXT J
```

Es funktioniert nicht, weil alle Kenntnis über die J-Schleife verloren ist, wenn NEXT I berechnet wird. Dazu kommt es, da die J-Schleife teilweise innerhalb der I-Schleife liegt.

2.16 Matrixoperationen

Es ist oft praktisch, auf jedes Element in einer Zahlentabelle zugreifen zu können. In BASIC kann man dies durch den Gebrauch von Matrizen tun.

Eine Matrix ist eine Zahlentabelle. Der Name dieser Tabelle (Matrixname) ist irgendeine zugelassene Variable, etwa "A". Der Matrixname "A" ist eindeutig und von der einfachen Variablen "A" unterschieden; man kann beide im gleichen Programm verwenden. Um ein Element der Tabelle auszuwählen, müssen wir "A" indizieren: um etwa das I-te Element auszuwählen, setzen wir I in Klammern "(I)" und schreiben diese Indizierung nach dem "A". Somit bedeutet "A (I)" das I-te Element der Matrix "A."

"A (I)" ist nur ein Element der Matrix A; man muß BASIC jedoch mitteilen, wieviel Speicherplatz es für die gesamte Matrix berücksichtigen muß. Dies wird mit einer "DIM"-Anweisung durchgeführt; das Format ist etwa "DIM A (15)". In diesem Fall haben wir für die Matrix A soviel Platz reserviert, daß der Index I von Ø bis 15 gehen kann.

Indizes von Matrizen beginnen immer bei Ø; im obigen Beispiel haben wir also 16 Zahlen in der Matrix A zugelassen.

Falls "A (I)" in einem Programm verwendet wird, ohne vorher die Dimension festzulegen, reserviert BASIC Platz für 11 Elemente (Ø bis 1Ø)

Als ein Anwendungsbeispiel für Matrizen sehen wir uns das folgende Programm an, das acht von Ihnen einzugebende Zahlen sortiert.

```
 1Ø DIM A (8)
 2Ø FOR I=1 TO 8
 3Ø INPUT A (I)
 5Ø NEXT I
 7Ø F=Ø
 8Ø FOR I=1 TO 7
 9Ø IF A (I) <= A (I+1) THEN 14Ø
1ØØ T = A (I)
11Ø A (I) = A (I+1)
12Ø A (I+1) = T
13Ø F = 1
14Ø NEXT I
15Ø IF F = 1 THEN 7Ø
16Ø FOR I = 1 TO 8
17Ø PRINT A (I)
18Ø NEXT I
```

Sobald Zeile 1Ø ausgeführt ist, hat BASIC Platz für 9 numerierte Werte geschaffen, A (Ø) bis A (8). Die Zeilen 2Ø bis 5Ø holen die unsortierte Liste des Anwenders. Das eigentliche Sor-

tieren wird durch „Abarbeiten" der Zahlenliste und Umstellen von jeweils zwei Zahlen, die nicht in der richtigen Reihenfolge stehen, erledigt. "F" wird verwendet, um festzuhalten, daß Umstellungen durchgeführt worden sind; falls welche gemacht wurden, sagt Zeile 15Ø zurückzugehen und weiter zu prüfen.

Falls keine Zahlen mehr umgestellt werden müssen, nachdem sie also alle in Ordnung sind, drucken Zeile 16Ø bis 18Ø die sortierte Liste aus. Beachten Sie, daß ein Index einer Matrix ein beliebiger Ausdruck sein kann.

2.17 Unterprogramme

Falls Sie in einem Programm dieselbe Befehlsfolge an verschiedenen Plätzen benötigen, könnten Sie immer dieselben Anweisungen an diesen Stellen schreiben.

Die "GOSUB"- und "RETURN"-Anweisungen können dazu verwendet werden, dieses Duplizieren zu vermeiden. Sobald ein "GOSUB" entdeckt wird, springt BASIC zu der Zeile, deren Nummer nach GOSUB steht. Jedoch merkt sich BASIC, wo es im Programm vor dieser Verzweigung war. Sobald nun eine "RETURN"-Anweisung entdeckt wird, geht BASIC zurück zur ersten Anweisung, die dem letzten ausgeführten "GOSUB" folgt.

Verfolgen Sie bitte das Programm:

```
 1Ø PRINT"EINGABE EINER ZAHL";
 3Ø GOSUB 1ØØ
 4Ø T=N
 5Ø PRINT "ZWEITE ZAHL";
 7Ø GOSUB 1ØØ
 8Ø PRINT "DIE SUMME IST"; T+N
 9Ø END
1ØØ INPUT N
11Ø IF N=INT (N) THEN 14Ø
12Ø PRINT "ZAHL MUSS GANZZAHLIG SEIN"
13Ø GOTO 1ØØ
14Ø RETURN
```

Dieses Programm fragt nach zwei Zahlen, (die ganzzahlig sein müssen) und druckt dann ihre Summe aus. Das Unterprogramm innerhalb dieses Programms geht von den Zeilen 1ØØ bis 14Ø. Das Unterprogramm holt eine Zahl ein und fragt, falls sie nicht ganzzahlig ist, nach einer neuen Zahl. Es wird weiterfragen, bis eine ganze Zahl eingegeben wird.

Das Hauptprogramm druckt "EINGABE EINER ZAHL" und ruft dann das Unterprogramm auf, um den Zahlenwert in N zu bekommen. Sobald aus dem Unterprogramm (nach Zeile 4Ø) zurückgekehrt wird, wird der Eingabewert in der Variablen T gesichert. Dies geschieht deshalb, damit der erste Zahlenwert nicht verloren geht, sobald das Unterprogramm ein zweites Mal aufgerufen wird.
Dann wird "ZWEITE ZAHL" gedruckt, und der zweite Wert wird durch ein nochmaliges Aufrufen des Unterprogramms eingeholt.
Sobald ein zweites Mal aus dem Unterprogramm zurückgekehrt wird, wird "DIE SUMME IST", gefolgt von der Summe gedruckt. T enthält den Wert der ersten und N den Wert der zweiten eingegebenen Zahl.

2.18 Anhalten eines Programms

Die nächste Anweisung des Programms ist eine "END"-Anweisung. Dies veranlaßt das Programm, die Ausführung in Zeile 9Ø zu unterbrechen. Falls wir die "END"-Anweisung aus dem Programm entfernen, würden wir in das Programm ab Zeile 1ØØ "hineinlaufen". Dies ist unerwünscht, da wir erneut zum Eingeben einer Zahl aufgefordert werden. Falls wir das tun, würde das Unterprogramm versuchen zurückzukehren; und da kein "GOSUB" das Unterprogramm aufgerufen hat, wird ein RG-Fehler auftreten. Jedes "GOSUB", das in einem Programm ausgeführt wird, sollte ein entsprechendes "RETURN" haben, das später ausgeführt wird.

Auch das Gegenteil kann vorkommen:
Ein "RETURN" sollte nur dann ausgeführt werden, wenn es Teil eines durch "GOSUB" aufgerufenen Unterprogramms ist.

Sowohl "STOP" als auch "END" können dazu verwendet werden, ein Programm von seinen Unterprogrammen abzutrennen. "STOP" wird die Mitteilung ausdrucken, in welcher Zeile "STOP" steht.

2.19 Eingeben von Daten

Angenommen, Sie müssen in Ihr Programm Werte eingeben, die sich nicht bei jedem Programmablauf ändern, die Sie aber, falls nötig, leicht ändern können. BASIC besitzt zu diesem Zweck spezielle Anweisungen, nämlich "READ" und "DATA".

Betrachten Sie das folgende Programm:

```
 1Ø PRINT "RATEN SIE EINE ZAHL"
 2Ø INPUT G
 3Ø READ D
 4Ø IF D = – 999999 THEN 9Ø
 5Ø IF D <> G THEN 3Ø
 6Ø PRINT "RICHTIG"
 7Ø END
 9Ø PRINT "NEIN, NOCHMALS"
 95 RESTORE
1ØØ GOTO 1Ø
11Ø DATA 1, 393, –39, 28, 391, –8, Ø, 3.14, 9Ø
12Ø DATA 89, 5, 1Ø, 15, –34, –999 999
```

Während die "READ"-Anweisung ausgeführt wird, ist das Ergebnis der einer INPUT-Anweisung. Es wird die Zahl jedoch nicht vom Terminal, sondern aus der "DATA"-Anweisung gelesen.

Sobald das erste Mal eine Zahl für ein READ benötigt wird, wird die erste Zahl in der ersten DATA-Anweisung übergeben. Wird zum zweiten Mal eine Zahl benötigt, wird die zweite Zahl der ersten DATA-Anweisung übergeben. Sobald auf diese Weise der gesamte Inhalt der ersten DATA-Anweisung gelesen ist, wird die zweite DATA-Anweisung benützt. DATA wird also immer auf diese Weise der Reihe nach gelesen; Sie können in Ihrem Programm beliebig viele DATA-Anweisungen stehen haben.

Der Sinn des Programms ist ein Spiel:

Sie versuchen eine der Zahlen, die in den DATA-Anweisungen enthalten sind, zu erraten. Jede Zahl, die Sie eintippen, vergleichen wir mit allen Zahlen der DATA-Anweisungen, bis wir eine finden, die Sie richtig geraten haben.

Falls mehr Werte gelesen werden, als Zahlen in den DATA-Anweisungen stehen, erscheint ein "Daten leer" (OD)-Fehler. Deshalb prüfen wir in Zeile 4Ø, ob – 999 999 gelesen wurde. Diese Zahl soll nicht geraten werden, sie dient als Zeichen dafür, daß alle Daten (eventuell richtig geratene) gelesen wurden. Somit wissen wir, daß die geratene Zahl falsch war, sobald wir – 999 999 lesen.

Bevor wir nun zur Zeile 1Ø für einen weiteren Versuch zurückgehen, müssen wir die READ's dazu bringen, wieder mit der ersten Zahl der DATA-Anweisungen zu beginnen. Diese Funktion führt die Anweisung RESTORE aus. Nachdem RESTORE ausgeführt wurde, wird die nächste Zahl, die durch READ gelesen wird, wieder die erste Zahl der ersten DATA-Anweisung sein.

DATA-Anweisungen können beliebig im Programm verstreut sein. Nur READ-Anweisungen benützen die DATA-Anweisungen eines Programms; sobald sonst innerhalb eines Programms auf sie gestoßen wird, werden sie ignoriert.

2.20 Strings

Eine Reihe von Zeichen bezeichnet man als "String" SIEMENS, PC 100 und DIES IST EIN TEST sind Strings. String-Variablen können, wie bei numerischen Variablen, spezifische Werte zugewiesen werden. String-Variable werden durch ein "$" nach dem Variablenamen von numerischen Variablen unterschieden.

Versuchen Sie als Beispiel:

```
A$  = "SIEMENS PC 100"
PRINT A$
SIEMENS PC 100
```

In diesem Beispiel haben wir der String-Variablen A$ den String-Wert "SIEMENS PC 100" zugewiesen. Beachten Sie, daß wir den Zeichenstring, dem wir A$ zugewiesen haben, zwischen Anführungsstrichen gesetzt haben.

LEN

Nachdem wir nun A$ einen String-Wert zugewiesen haben, können wir herausfinden, wie lang dieser Wert ist (d.h. wieviele Zeichen er enthält).

```
PRINT LEN(A$), LEN ("MIKROCOMPUTER")
 14              13
```

Die "LEN"-Funktion ergibt eine ganze Zahl, die der Zeichenanzahl eines Strings entspricht. Die Anzahl von Zeichen innerhalb eines Strings kann von Ø bis 255 gehen. Ein String, der Null Zeichen enthält, wird "Null"-String genannt. Bevor einer String-Variablen in einem Programm ein Wert zugewiesen wird, ist sie ein Null-String. Die Ausgabe eines Null-Strings auf dem Terminal bewirkt, daß weder ein Zeichen abgedruckt noch der Cursor oder Druckknopf zur nächsten Spalte bewegt wird.

Versuchen Sie:

```
PRINT LEN (Q$); Q$;3
 Ø      3
```

Man kann einen Null-String auch durch

`Q$ = ""` erzeugen.

Das Setzen einer String-Variablen zum Null-String kann dazu verwendet werden, den String-Platz, der durch einen nicht Null-String verbraucht wurde, freizubekommen.

LEFT$

Es ist oft gewünscht, auf Teile eines Strings Zugriff zu haben, um mit diesen etwas zu tun. Nachdem wir nun A$ den Wert "SIEMENS PC 100" gegeben haben, wollen wir vielleicht nur die ersten sieben Zeichen von A$ ausdrucken.

```
PRINT LEFT$ (A$,7)
SIEMENS
```

LEFT$ ist eine Stringfunktion, die einen String zurückgibt, der nur aus den links stehenden N Zeichen des Originalstrings besteht.
Hier ein weiteres Beispiel:

```
For N = 1 TO LEN (A$) : PRINT LEFT$ (A$,N) : NEXT N
S
SI
SIE
SIEM
SIEME
SIEMEN
SIEMENS
SIEMENS
SIEMENS P
SIEMENS PC
SIEMENS PC
SIEMENS PC 1
SIEMENS PC 1Ø
SIEMENS PC 1ØØ
```

Da A$ 14 Zeichen hat, wird diese Schleife mit N = 1,2,3 . . . 13, 14 ausgeführt werden. Das erste Mal wird nur das erste Zeichen gedruckt, das zweite Mal die beiden ersten Zeichen usw.

RIGHT$

Eine weitere String-Funktion, die "RIGHT$" genannt wird, übergibt die rechten N Zeichen eines String-Ausdruckes. Ersetzen Sie das "LEFT$" des vorigen Beispieles durch "RIGHTS" und sehen Sie, was passiert.

MID$

Es gibt auch eine String-Funktion, die es uns erlaubt, Zeichen aus der Mitte eines Strings zu entnehmen.

Versuchen Sie:

```
FOR N=1 TO LEN (A$) : PRINT MID$ (A$,N) : NEXT N
SIEMENS PC 100
IEMENS PC 100
EMENS PC 100
MENS PC 100
ENS PC 100
NS PC 100
S PC 100
 PC 100
PC 100
C 100
 100
100
10
0
```

"MID$" übergibt einen String, der von der N-ten Stelle von A$ bis zum Ende (letztes Zeichen) von A$ geht. Die erste Stelle eines Strings ist Stelle 1 und die letzte mögliche Stelle eines Strings ist die Stelle 255.

Oftmals ist es gewünscht, nur das N-te Zeichen aus einem String herauszuholen. Dies kann durch den Aufruf von MID$ mit drei Argumenten geschehen. Das dritte Argument gibt die Zahl der zu übergebenden (herauszuholenden) Zeichen an.
Etwa:

```
FOR N=1 TO LEN (A$) : PRINT MID$ (A$,N,1),MID$(A$,N,2) : NEXT N
S               SI
I               IE
E               EM
M               ME
E               EN
N               NS
S                 S
                    P
P               PC
C               C
                    1
1               10
0               00
0               0
```

VERKETTUNG

Strings können durch die Anwendung des "+"-Operators verkettet (zusammengefügt) werden.
Versuchen Sie:

```
B$ = "BASIC FUER" + "  " + A$
PRINT B$
BASIC FUER SIEMENS PC 100
```

Verkettung ist besonders dann nützlich, wenn Sie einen String auseinandernehmen und dann mit kleinen Änderungen wieder zusammenfassen wollen.

Beispiel:

```
C$= LEFT$ (B$,10)+"-"+MID$ (B$,12,7)+"-"+RIGHT$ (B$,6)
PRINT C$
BASIC FUER-SIEMENS-PC 100
```

VAL und STR$

Manchmal möchte man eine Zahl in ihre String-Darstellung und umgekehrt wandeln. "VAL" und "STR$" führen diese Funktionen aus.
Versuchen Sie:

```
STRING$= "567.8"
PRINT VAL (STRING $)
  567.8
STRING$= STR$ (3.1415)
PRINT STRING$, LEFT$ (STRING$,5)
  3.1415                3.14
```

"STR$" kann dazu verwendet werden, Zahlen zur Ausgabe zu formatieren. Sie können eine Zahl in einen String verwandeln und dann LEFT$, RIGHT$, MID$ und Verkettung anwenden, um die Zahl wie gewünscht zu formatieren. "STR$" kann auch bequem dazu verwendet werden, um herauszufinden, wieviele Druckspalten eine Zahl brauchen wird.
Etwa:

```
PRINT LEN (STR$ (3.157))
 6
```

CHR$ und ASC

CHR$ ist eine String-Funktion, die einen Ein-Zeichen-String übergibt, der das alphanumerische ASCII Äquivalent des Argumentes enthält, nach der Umwandlungstabelle Abschnitt 4.6 ASC nimmt das erste Zeichen eines Strings und wandelt es in den ASCII-Dezimalwert um.

Eine der häufigsten Anwendungen von CHR$ ist, ein spezielles Zeichen zu einem Terminal zu senden. Das am häufigsten gebrauchte Zeichen ist das Zeichen BEL (ASCII:7). Der Abdruck dieses Zeichens wird eine Glocke an einigen Terminals anschlagen lasse oder auf vielen Sichtgeräten einen Piepser ertönen lassen. Das kann man zur Einleitung einer Fehlermeldung tun.
(Beispiel: PRINT CHR$ (7);)

Weitere STRING-Betrachtungen

1. Ein String kann Ø bis 255 Zeichen enthalten. Alle String-Variablenamen enden mit einem Dollar ($)-Zeichen; etwa A$, B9$, K$, HALLO$.
2. String-Matrizen können genau wie numerische Matrizen dimensioniert werden. Beispielsweise erzeugt DIM A$ (1Ø,1Ø) eine String-Matrix mit 121 Elementen, elf Reihen mal elf Spalten (Reihe Ø bis 1Ø und Spalte Ø bis 1Ø).
   Jedes Element der String-Matrix ist ein kompletter String der bis zu 255 Zeichen lang sein kann.

Name	Beispiel	Zweck/Anwendung
DIM	25 DIM A$ (1Ø,1Ø)	Reserviert Platz für einen Zeiger und Länge für jedes Element der String-Matrix. Kein String-Raum wird zugewiesen.
LET	27 LET A$= FØØ"+VS	Weist einer String-Variablen einen String-Ausdruck zu. LET kann man weglassen.
= > < <=oder =< >=oder => <>		Stringvergleichsoperationen. Verglichen wird auf der Grundlage von ASCII-Codes, ein Zeichen pro Vergleich, bis ein Unterschied entdeckt wird. Falls während des Vergleichs zweier Strings das Ende des einen erreicht wird, wird der kürzere String als kleiner betrachtet. Beachten Sie, daß "A " größer als "A" ist, da der anhängende Zwischenraum berücksichtigt wird.
+	3Ø LET Z$=R$+Q$	Stringverkettung Der resultierende String muß weniger als 256 Zeichen haben, sonst erscheint ein LS-Fehler.
INPUT	4Ø INPUT X$	Liest einen String von der Tastatur ein. Strings müssen hier nicht in Anführungszeichen stehen, falls sie es nicht sind, werden führende Zwischenräume ignoriert und der String durch ein "," oder einen ":" beendet.
READ	5Ø READ X$	Liest einen String von Data-Anweisungen innerhalb des Programms. Strings müssen nicht in Anführungszeichen stehen; aber falls sie es nicht sind, werden sie durch ein "," oder einen ":" beendet und führende Zwischenräume werden ignoriert. Siehe DATA- für das Format von String-Daten.
PRINT	6Ø PRINT X$ 7Ø PRINT "FØØ"+A$	Bringt den Stringausdruck auf den Display/Drucker

3. Anweisungsdefinitionen

3.1 Spezielle Tastenfunktionen

Zeichen	Anwendung
@	Löscht die soeben eingetragene aktuelle Zeile und bringt eine Wagenrücklauf/Zeilenvorschub-Kombination.
DEL	Löscht das zuletzt eingetippte Zeichen.
RETURN	Ein RETURN muß am Ende jeder eingegebenen Zeile stehen. Bringt den Druckkopf oder den Sichtgerät-Cursor in die erste (ganz linke) Position in einer Zeile, und führt einen Zeilenverschub aus.
F1	Unterbricht die Ausführung eines Programms oder eines List-Kommandos. F1 wirkt nach Beendigung einer Anweisungsausführung oder, falls es ein List-Kommando unterbricht, nach dem Ausdruck einer kompletten Zeile. In beiden Fällen wird zur BASIC-Kommandoebene zurückgegangen und ^ ausgegeben. Druckt "BREAK IN XXXX" aus, wobei XXXX die Zeilennummer der als nächstes auszuführenden Anweisung ist. Falls die Taste weiter gedrückt wird, wird ein weiterer F1 empfangen. Am Fernschreiber (TTY) gibt es keine F1-Taste; jedoch funktioniert die F1-Taste auf der PC-100-Tastatur auch dann, wenn Fernschreiber angeschlossen ist.
:	Ein Doppelpunkt trennt Anweisungen in einer Zeile. Doppelpunkte kann man in direkten und indirekten Kommandos verwenden. Die einzige Beschränkung der Anweisungsanzahl innerhalb einer Zeile ist die Zeilenlänge. Man kann nicht mit GOTO oder GOSUB in die Mitte einer Zeile springen.
?	Fragezeichen entspricht einem PRINT. Beispielsweise ist ? 2+2 gleichwertig mit PRINT 2+2. Fragezeichen können auch in indirekten Anweisungen verwendet werden. 1Ø ?X wird beim Auflisten als 1Ø PRINT X ausgegeben.
$	Ein $-Zeichen, angehängt an einen Variablennamen, macht die Variable zu einem String.
%	Eine Variable bei deren Namen das %-Zeichen an der 2. Stelle steht, kann nur ganzzahlige Werte annehmen.
!	Ein !-Zeichen, angehängt an eine PRINT- oder INPUT-Anweisung, bewirkt, daß auch bei ausgeschaltetem Drucker ein Ausdruck erfolgt.
ESC	Betätigen der ESC-Taste gibt die Kontrolle an das Monitorprogramm
CTRL/PRINT	Gleichzeitiges Betätigen beider Tasten schaltet den PC 100 Drucker ein oder aus.
E/R	Gleichzeitiges Betätigen beider Tasten bewirkt RESET-Funktion (Programmverlust). Diese Tastenkombination gibt es nur beim Komplettgerät.
T	Umschalten von der PC 100-Tastatur auf Fernschreiber-Tastatur.
S	Taste zum schrittweise „abarbeiten" eines Programms (nur für Monitorbetrieb)
F 2	Taste nicht im BASIC-Betrieb zu verwenden.
PRINT	Betätigen der PRINT-Taste druckt Anzeige am PC 100 Drucker aus (auch bei ausgeschaltetem Drucker)

3.2 Operatoren

Symbol	Beispiel	Zweck/Verwendung
=	A = 10Ø LET Z=2,5	Weist einer Variablen einen Wert zu. LET kann weggelassen werden.
–	B=–A	Negation. Beachten Sie, daß Ø –A Subtraktion bedeutet, während –A Negation darstellt.
^ (Taste F3)	10Ø PRINT X^3	Hochrechnung (entspricht X*X*X im Beispiel), Ø^Ø = 1, Ø hoch irgendetwas = Ø. A^B, mit A negativ und B nicht geradzahlig ergibt FC-Fehler
*	14Ø X=R* (B*D)	Multiplikation
/	15Ø PRINT X/1.3	Division
+	16Ø Z=R+T+Q	Addition
–	17Ø J=1ØØ–I	Subtraktion

Berechnen von Ausdrücken (Regeln)

1. Operationen höherer Wertigkeit werden vor Operationen niedrigerer Wertigkeit ausgeführt. Das bedeutet, daß Multiplikationen und Divisionen vor Additionen und Subtraktionen ausgeführt werden. Beispielsweise ergibt die Formel 2+1Ø/5=4, nicht 2.4. Sollten gleichwertige Operationen in einer Formel stehen, wird die linke zuerst ausgeführt: 6–3+5=8, nicht –2.
2. Die Reihenfolge, in der Operationen ausgeführt werden, kann immer explizit durch die Anwendung von Klammern angegeben werden. Um etwa zu 5 die Zahl 3 zu addieren und dies dann durch 4 zu teilen, würden wir (5+3) /4 anwenden, was 2 ergibt. Falls wir statt dessen 5+3/4 genommen hätten, würden wir als Ergebnis 5.75 erhalten (5 plus 3/4).

Die Wertigkeiten von Operatoren sind im folgenden angegeben, mit der höchsten Wertigkeit beginnend:

1) Ausdrücke in Klammern werden immer zuerst berechnet		
2) ^ (Taste F3)	Potenzieren	
3) Negation	–X wobei X eine Formel sein kann	
4) * und /	Multiplikation und Division	
5) + und –	Addition und Subtraktion	
6) Relationen	= gleich <> ungleich < kleiner als > größer als =< oder <= kleiner gleich => oder >= größer gleich	*Anmerkung!* *Die Relationen sind absolut gleichwertig.*
Logische Operatoren		
7) NOT	Logisches bitweises "NICHT" wie Negation, NOT nimmt nur die rechtsstehende Formel als Argument	
8) AND	Logisches bitweises "UND"	
9) OR	Logisches weises "ODER"	

Eine Relation kann Teil eines Ausdrucks sein. Ergebnisse von Relationen sind entweder wahr (–1) oder falsch (Ø). So ist (5=4)=Ø, (5=5)=–1, (4>5)=Ø (4<5)=–1 usw.

Der THEN-Teil einer IF-Anweisung wird dann ausgeführt, wenn die Formel nach dem IF ungleich Ø ist. Somit bedeutet IF X THEN soviel wie IF X <> Ø THEN . . .

Symbol	Beispiel	Sinn/Anwendung
=	1Ø IF A=15 THEN 4Ø	Ausdruck ist gleich Ausdruck
<>	7Ø IF A<>Ø THEN 5	Ausdruck ist nicht gleich Ausdruck
>	3Ø IF B>1ØØ THEN 8	Ausdruck größer Ausdruck
<	16Ø IF B < 2 THEN 1Ø	Ausdruck kleiner Ausdruck
<=,=<	18Ø IF 1ØØ < = B+C THEN 1Ø	Ausdruck kleiner gleich Ausdruck
>=,=>	19Ø IF Q=> R THEN 5Ø	Ausdruck größer gleich Ausdruck
AND	2 IF A < 5 AND B < 2 THEN 7	Falls Ausdruck 1 (A<5) und Ausdruck 2 (B<2) beide wahr sind, Verzweigung zu Zeile 7
OR	IF A <1 OR B<2 THEN 2	Falls entweder Ausdruck 1 (A 1) oder Ausdruck 2 (B 2) wahr sind wird zu Zeile 2 verzweigt.
NOT	IF NOT Q3 THEN 4	Falls "NOT Q3" wahr ist, (da Q3 falsch ist) Verzweigung zu Zeile 4 *Anmerkung: NOT–1=Ø (NOT WAHR=FALSCH)*

AND, OR und NOT können zur Bitmanipulation und zur Ausführung Bool'scher Operationen verwendet werden.

Diese Operatoren wandeln ihre Argumente in eine vorzeichenbehaftete 16-Bit-Zahl im Bereich von –32768 bis +32767 (Negative Zahlen im Zweier-Komplement). Dann führen Sie die angegebene Operation aus und übergeben das Ergebnis im selben Zahlenbereich. Falls die Argumente nicht in diesem Bereich liegen, erscheint ein "FC"-Fehler.

Die Wahrheitstabelle zeigt die logische Beziehung zwischen Bits:

Operator	Argument 1	Argument 2	Ergebnis
AND	1 Ø 1 Ø	1 1 Ø Ø	1 Ø Ø Ø
OR	1 1 Ø Ø	1 Ø 1 Ø	1 1 1 Ø
NOT	1 Ø	– –	Ø 1

Beispiele:
(In den Beispielen sind führende Nullen nicht angegeben).

63 AND 16=16	'Da 63 Binär 111 111 und 16 binär 1ØØØØ bedeutet, ist das Ergebnis des AND binär 1ØØØØ oder dezimal 16
15 AND 14=14	15 bedeutet binär 1111 und 14 binär 111Ø, somit ist 15 und 14 binär 111Ø oder dezimal 14
–1 AND 8= 8	–1 bedeutet binär 1111111111111111 und 8 bedeutet binär 1ØØØ, somit ist das Ergebnis 1ØØØ oder dezimal 8
4 AND 2=Ø	4 bedeutet binär 1ØØ und 2 binär 1Ø; damit ist das Ergebnis Ø, da keine der Bitstellen der beiden Argumente ein "1" bit im Ergebnis bewirkt.
4 OR 2=6	Binär 1ØØ, geodert mit 1Ø ergibt binär 11Ø oder 6 dezimal
1Ø OR 1Ø=1Ø	Binär 1Ø1Ø geodert mit binär 1Ø1Ø ergibt binär wieder 1Ø1Ø (1Ø dez.)
–1 OR –2 = –1	Binär 1111111111111111 (–1) geodert mit binär 111111111111110 (–2) ergibt 1111111111111111 oder –1
NOT Ø = –1	Das Bitkomplement einer binären Null an 16 Stellen sind 16 log 1 (1111111111111111) oder –1. Ebenso ist NOT –1 = Ø.
NOT X	NOT X entspricht –(X+1). Dies ergibt sich, da man, um das Zweierkomplement zu bilden, das Bit- (Einser-) komplement bildet und 1 addiert.
NOT 1 = –2	Das Einserkomplement von 1 ist 1111111111111110, das gleich – (1+1), oder, –2, ist.

Ein typischer Anwendungsfall der bitweisen Operationen ist das Testen einzelner Bits in Speicherstellen, die etwas über den Zustand externer Geräte aussagen.
Das Bit an der Stelle 7 ist das höchstwertigste, das Bit an der Stelle Ø ist das niederwertigste Bit eines Bytes. Nehmen wir beispielsweise an, daß das Bit 1 der Speicherzelle 41987 "Ø" ist, wenn die Tür eines Raumes geschlossen ist und "1", falls sie geöffnet ist.

Das folgende Programm wird "EINBRUCHALARM" ausdrucken, sobald die Tür geöffnet wird.

```
1Ø IF NOT (PEEK (41987) AND 2) THEN 1Ø
```

Diese Zeile wird immer wieder ausgeführt, bis Bit 1 (das aus der Speicherzelle durch logisches UND mit 2 = ØØØØØØ1Ø Binär markiert wurde) eine logische 1 wird. Sobald dies geschieht, gehen wir zur Zeile 20

```
2Ø PRINT "EINBRUCHALARM"
```

Zeile 2Ø wird "EINBRUCHALARM" ausdrucken.

Wir können jedoch die Anweisung in Zeile 1Ø durch eine WAIT-Anweisung ersetzen, die den gleichen Effekt hat.

```
1Ø WAIT 41987,2
```

Diese Zeile verzögert die Ausführung der nächsten Anweisung, bis Bit 1 der Speicherzelle 41987 logisch 1 wird.
WAIT ist viel schneller als die entsprechende IF-Anweisung und benötigt auch weniger Speicherplatz.

Das folgende Beispiel ist eine weitere nützliche Anwendung von Relationen:

```
124 A=- (B>C)*B- (B < =C) *C
```

Diese Anweisung setzt die Variable A auf MAX (B,C), das heißt auf den Wert der größeren der beiden Variablen B und C.

3.3 Befehle

Ein Befehl wird normalerweise gegeben, nachdem der BASIC-Cursor angezeigt wird. Dies nennt man die "Befehlsebene" (Command level). Befehle können als Programmanweisungen verwendet werden. Gewisse Befehle, wie LIST, NEW und LOAD beenden die Programmausführung, sobald sie beendet sind.

Anweisung	Syntax/Funktion	Beispiel
CLEAR	CLEAR Löscht alle Programmvariablen, setzt FOR- und GOSUB-Zustände zurück und macht einen Daten-RESTORE.	CLEAR
CONT	CONT Startet die Programmausführung wieder, nachdem ein "F1" eingetippt oder eine STOP-Anweisung erkannt wurde. Sie können nach einem Fehler, nach einer Programmänderung oder bevor Ihr Programm gelaufen ist, nicht mit CONT weiterstarten. Eine der Hauptanwendungen des CONT ist die Fehlersuche. Angenommen es wird an irgend einer Stelle, nachdem Ihr Programm gestartet ist fälschlich nichts ausgedruckt. Das kann etwa dadurch passieren, daß Ihr Programm lange Zeit für Berechnungen benötigt; es kann aber auch sein, daß Ihr Programm in eine „Endlosschleife" gekommen ist. Eine Endlosschleife ist eine Reihe von BASIC-Anweisungen, aus denen es keinen Ausgang gibt. PC 100 wird diese Anweisungen immer und immer wieder ausführen, bis Sie eingreifen oder die Versorgungsspannung des PC 100 ausschalten. Falls Sie Ihr Programm in einer Endlosschleife vermuten, drücken Sie die Taste F1. Die Zeilennummer der Anweisung, die BASIC gerade ausgeführt hat, wird nun angezeigt. Nachdem BASIC "^" ausgegeben hat, können Sie PRINT verwenden, um sich die Werte Ihrer Variablen ausdrucken zu lassen. Nachdem Sie diese Werte geprüft haben, stellen Sie vielleicht fest, daß Ihr	CONT

Anweisungsdefinitionen

Anweisung	Syntax/Funktion	Beispiel
CONT	Programm korrekt arbeitet. Sie können dann CONT eingeben, um Ihr Programm an der Stelle wo es unterbrochen wurde, weiterlaufen zu lassen; Sie können auch direkt eine GOTO-Anweisung eingeben um die Programmausführung an einer anderen Zeile wieder aufzunehmen. Sie könnten auch Zuweisungen (LET-Anweisungen) verwenden, um einigen Variablen neue Werte zuzuweisen. Denken Sie daran, daß, falls Sie Ihr Programm mit "F1" unterbrochen haben, keine Fehler haben und keine neuen Zeilen eingeben dürfen, falls Sie später mit "CONT" weitermachen wollen. Falls Sie es trotzdem versuchen, wird das Programm nicht weiter ausgeführt und Sie bekommen einen "CN" (Continue Not)-Fehler. Einen direkten Befehl kann man auch nicht mit CONT fortsetzen. CONT macht immer in der nächsten auszuführenden Anweisung nach der "F1"-Unterbrechung weiter.	CONT
FRE	FRE Ø Gibt die Zahl der Speicherstellen aus, die momentan von BASIC nicht benötigt werden.	27Ø PRINT FRE (Ø)
LIST	LIST((Startzeile)(–(–Endzeile))) LIST druckt das eingetragene Programm aus, wahlweise an der angegebenen Zeile beginnend. LIST kann durch "F1" unterbrochen werden, wobei BASIC die angefangene Zeile noch fertig ausdruckt. BASIC druckt nur Zeile 1ØØ BASIC druckt die Zeilen 1ØØ-1ØØØ BASIC druckt die laufende Zeile bis zur Zeile 1ØØØ BASIC druckt von Zeile 1ØØ bis Programmende	LIST LIST 1ØØ LIST 1ØØ-1ØØØ LIST –1ØØØ LIST 1ØØ–
LOAD	LOAD Lädt das Programm von der Kassette. Nach dem Laden gibt LOAD wie gewohnt ^ aus. Siehe Abschnitt 4.9 für weitere Informationen.	LOAD
NEW	NEW Löscht das laufende Programm und alle Variablen.	NEW
PEEK	PEEK (Adresse) Die PEEK-Funktion übergibt den Inhalt der Speicherzelle mit der Adresse I dezimal. Der übergebene Wert liegt zwischen >=Ø und <=255. Falls I nicht innerhalb Ø und 65535 liegt, erscheint "FC"-Fehler. Ein Zugriff auf eine nicht existierende Speicherzelle ergibt einen Zufallswert.	365 PRINT PEEK (I)
POKE	POKE Adresse, Byte Die POKE-Anweisung speichert das Byte (J) (zweites Argument) in der Speicherstelle, deren Adresse (I) im ersten Argument übergeben wurde. Das zu speichernde Byte muß zwischen >=Ø und <=255 liegen, sonst tritt ein FC-Fehler auf. Eine unüberlegte Anwendung der POKE-Anweisung kann das Programm zerstören oder zu fehlerhaften BASIC- und Monitor-Funktionen führen. Sie müssen BASIC neu starten. Ein POKE auf eine nicht vorhandene Speicherzelle ist unschädlich. Eine der POKE-Anwendungen ist die Übergabe von Argumenten an Unterprogramme in Maschinensprache (Siehe Abschnitt 4.7). Sie können PEEK und POKE auch zum Schreiben einer Speichertestroutine in BASIC verwenden.	357 POKE I,J

Anweisung	Syntax/Funktionen	Beispiel
POS	Pos (Ausdruck) Übergibt die derzeitige Position des Cursors am Display, das ganz linke Zeichen hat die Position Null. Ein Leeroperand wie (Ø), (1) usw. muß angegeben werden.	26Ø PRINT POS (1)
RUN	RUN (Zeilennummer) Startet das derzeit im Speicher befindliche Programm, beginnend mit der niedrigsten Zeilennummer. RUN löscht alle Variablen (führt ein CLEAR aus) und setzt den Datenzeiger zurück (führt ein Restore aus). Falls Sie Ihr Programm angehalten haben und es an irgendeiner Stelle wieder weiterlaufen lassen wollen, können Sie direkt eine GOTO-Anweisung zur gewünschten Zeile geben oder ein CONT, um nach einer Unterbrechung fortzufahren. Startet die Programmausführung an der angegebenen Zeilennummer.	RUN RUN 2ØØ
SAVE	SAVE Speichert das im PC 100-Speicher befindliche Programm auf Kassette. Das Programm im Speicher wird dabei nicht verändert. Man kann mit diesem Kommando mehr als ein Programm abspeichern. Weitere Informationen im Anhang I.	SAVE

3.4 Programm-Anweisungen

Anmerkung: In der folgenden Beschreibung der Anweisungen bedeuten die Argumente B,C,V und W numerische Variable, X bedeutet einen numerischen Ausdruck, X$ bedeutet einen STRING-Ausdruck und ein I oder J bedeuten einen Ausdruck, der bei Ausführung der Anweisung zu einer ganzen Zahl gekürzt wird. Stutzen heißt, daß die Stellen nach dem Komma abgeschnitten werden (z.B. 3.9 wird zu 3, 4.Ø1 zu 4 usw.)
Ein Ausdruck ist eine Reihe von Variablen, Operatoren, Funktionsaufrufen und Konstanten, die nach ihrer Ausführung (nach den Wertigkeiten berechnet) eine numerische oder eine STRING-Variable ergeben.
Eine Konstante ist entweder eine Zahl (3.14) oder ein Zeichen eines STRING's („FØØ").

Anweisung	Syntax/Funktion	Beispiel
DEF	DEF FN X (Argumentliste)=Ausdruck Der Anwender kann mit der DEF-Anweisung Funktionen definieren, ähnlich den bereits eingebauten Funktionen (SQR, SGN, ABS usw.). Der Name der Funktion ist "FN", gefolgt von irgendeinem zulässigen Variablenamen, etwa: FNX, FNGG, FNKO, FNR2. Anwenderdefinierte Funktionen müssen in eine Zeile passen. Eine Funktion kann irgendein Ausdruck sein, aber darf nur ein Argument haben. Im Beispiel sind B und C Variable, die im Programm verwendet werden. Das Ausführen der DEF-Anweisung definiert die Funktion. Anwenderdefinierte Funktionen können durch den Einsatz einer weiteren DEF-Anweisung für die gleiche Funktion redefiniert werden. "V" nennt man eine Dummy-Variable. Ausführung dieser Anweisung, die der oberen folgt, wird Z zu 3/B+C machen, aber der Wert von V bleibt unverändert.	1ØØ DEF FNA (V)=V/B+C 11Ø Z=FNA(3)
DIM	DIM-Variable (Größe 1,(Größe 2...))... Reserviert Platz für die Matrizen. Durch die DIM-Anweisung werden alle Matrixelemente zu Null gesetzt.	113 DIM A(3), B(1Ø)

Anweisung	Syntax/Funktion	Beispiel
DIM	Matrizen können mehr als eine Dimension haben. Bis zu 255 sind erlaubt, aber da in einer Zeile nur 72 Zeichen geschrieben werden können, ist das praktische Maximum etwa 34 Dimensionen.	114 DIM R3 (5,5),D$ (2,2,2)
	Matrizen können dynamisch während der Programmausführung dimensioniert werden. Falls eine Matrix nicht explizit mit einer DIM-Anweisung dimensioniert ist, wird sie als eindimensionale Matrix mit Elementen von Ø bis 1Ø(11Elementen) angenommen.	115 DIM Q1 (N),Z(2*I)
	Falls diese Anweisung im Programm vor einer DIM-Anweisung für A entdeckt wird, ist es als ob DIM A(1Ø)-Anweisung vor Zeile 117 ausgeführt worden wäre. Alle Indizes beginnen ab Null(Ø), was bedeutet, daß etwa DIM X (1ØØ) effektiv 1Ø1 Matrix-Elemente reserviert.	117A(8)=4
END	END Beendet die Programmausführung ohne eine BREAK-Botschaft auszudrucken. (Siehe STOP) CONT nach einer Endanweisung veranlaßt die Programmwiederaufnahme an der Anweisung nach der END-Anweisung. END kann überall im Programm eingesetzt werden.	999 END
FOR	FOR Variable = Ausdruck TO Ausdruck STEP Ausdruck (Siehe NEXT-Anweisung) V wird auf den Wert des Ausdrucks nach dem Gleichheitszeichen gesetzt, in diesem Fall 1. Dieser Wert wird Startwert genannt. Dann werden die Anweisungen zwischen FOR und NEXT ausgeführt. Der Endwert ist der Wert des Ausdrucks, der TO folgt (hier 9.3). Die Schrittweite ist der Wert des Ausdrucks der STEP folgt. Sobald die NEXT-Anweisung entdeckt wird, wird die Schrittweite zur Variablen addiert.	3ØØ FOR V=1 TO 9.3 STEP .6
	Ohne STEP-Wert wird 1 angenommen, falls der Schritt positiv und der neue Wert der Variablen kleiner gleich dem Endwert ist (9.3) oder falls die Schrittweite negativ und der neue Wert der Variablen größer gleich dem Endwert ist, wird die der FOR-Anweisung folgende Anweisung ausgeführt. Andernfalls wird die Anweisung die der NEXT-Anweisung folgt, ausgeführt. Alle FOR-Schleifen führen die Anweisung zwischen dem FOR und dem NEXT mindestens einmal aus, sogar für Fälle wie FOR V=1 TO Ø.	31Ø FOR V=1 TO 9.3
	Beachten Sie, daß auch Ausdrücke (Formeln) für die Start-, End- und Schrittwerte in einer FOR-Schleife erlaubt sind. Die Werte der Ausdrücke werden nur einmal berechnet bevor der Kern der FOR...NEXT-Schleife ausgeführt wird.	315 FOR V=1Ø*N TO 3.4/Q STEP SQR(R)
	Sobald die Anweisung nach dem NEXT ausgeführt wird, ist die Schleifenvariable nie gleich dem Endwert, sondern gleich dem Wert, der die FOR... ...NEXT-Schleife zum Abbruch brachte.	320 FOR V=9 TO 1 STEP-1
	Die Anweisungen zwischen dem FOR und seinem zugehörigen NEXT werden in beiden Beispielen (310 und 320) neun mal ausgeführt.	
	Fehler: Verschachteln Sie keine FOR... ...Next-Schleifen mit der gleichen Indexvariablen.	
	Das Verschachteln von FOR-Schleifen ist nur durch die Größe des verfügbaren Speichers begrenzt (siehe Abschnitt 4.3).	330 FOR W=1 TO 1Ø: FOR W=1 TO 5:NEXT W: NEXT W

Anweisungsdefinitionen

Anweisung	Syntax/Funktion	Beispiel
GOSUB	GOSUB Zeilennummer Verzweigt zur angegebenen Anweisung (91Ø), bis ein RETURN gefunden wird; dann wird bei der GOSUB folgenden Anweisung weitergemacht. Verschachteln von GOSUB's ist nur durch den verfügbaren Speicherplatz beschränkt.	1Ø GOSUB 91Ø
GOTO	GOTO Zeilennummer Verzweigt zu der angegebenen Anweisung	5Ø GOTO 1ØØ
IF...GOTO	IF Ausdruck GOTO Zeilennummer Entspricht dem IF...THEN mit der Ausnahme, daß nach IF...GOTO eine Zeilennummer folgen muß, während nach IF...THEN entweder eine Zeilennummer oder eine andere Anweisung folgen kann.	32 IF X<=Y+ 23.4 GOTO 92
IF...THEN	IF Ausdruck THEN Anweisung Verzweigt zur angegebenen Anweisung, falls die Relation stimmt. (Siehe auch IF...GOTO)	IF X=1Ø THEN 5 2Ø IF X<Ø THEN PRINT „X KLEINER NULL"
IF...THEN	**Achtung:** das "Z=A" wird nie ausgeführt, denn, falls die Relation stimmt, wird BASIC zur Zeile 5Ø verzweigen; falls die Relation falsch ist, wird BASIC die Zeilen nach der Zeile 25 ausführen. In diesem Beispiel wird, falls X kleiner Null ist, die PRINT-Anweisung ausgeführt; die folgende GOTO-Anweisung wird zur Zeile 35Ø verzweigt. Falls X Null oder positiv war, wird BASIC die Zeilen nach der Zeile 26 ausführen.	25 IF X=5 THEN 5Ø:Z=A 26 IF X<Ø THEN PRINT "FEHLER,X NEGATIV": GOTO 35Ø
LET	(LET) Variable=Ausdruck Weist einer Variablen einen Wert zu. "LET" kann weggelassen werden.	3ØØ LET W=X 310 V=5.1
NEXT	NEXT Variable (,Variable)... Bezeichnet das Ende einer FOR-Schleife. Falls keine Variable angegeben wird, bezieht sich das NEXT auf die zuletzt ausgeführte FOR-Schleife. Ein NEXT kann verwendet werden, um das Ende mehrerer FOR-Anweisungen zu bezeichnen. Gleichwertig zu NEXT V: NEXT W.	340 NEXT V 345 NEXT 35Ø NEXT V,W
ON...GOSUB	ON Ausdruck GOSUB Zeile (,Zeile) Entspricht dem "ON...GOTO" mit dem Unterschied, daß ein Unterprogrammaufruf (GOSUB) ausgeführt wird statt eines GOTO. RETURN eines GOSUB verzweigt zu der Zeile, die der ON...GOSUB-Anweisung folgt.	11Ø ON I GOSUB 5Ø,6Ø
ON...GOTO	ON Ausdruck GOTO Zeile (,Zeile)... Verzweigt zu der Zeile, die durch die I-te Zahl nach dem GOTO angegeben wird. Gleichbedeutend mit: IF I=1 THEN GOTO 1Ø IF I=2 THEN GOTO 2Ø IF I=3 THEN GOTO 3Ø IF I=4 THEN GOTO 4Ø. Falls I Null ist, oder I versucht, eine nicht existierende Zeile zu adressieren (I>=5 in diesem Fall), wird die Anweisung nach der ON-Anweisung ausgeführt. Falls jedoch I>255 oder <Ø ist, wird ein FC-Fehler gemeldet. Es können soviele Zeilennummern wie in eine Zeile hineinpassen, angegeben werden. Diese Anweisung wird zu Zeile 4Ø verzweigen falls der Ausdruck kleiner Null ist, zur Zeile 5Ø falls er gleich Null ist und zur Zeile 6Ø falls er größer Null ist.	1ØØ ON I GOTO 1Ø,2Ø, 3Ø,4Ø 1Ø5 ON SGN(X)+2 GOTO 4Ø,5Ø,6Ø

Anweisung	Syntax/Funktion	Beispiel
REM	REM Beliebiger Text Erlaubt dem Programmierer Kommentare in sein Programm zu bringen. REM-Anweisungen werden nicht ausgeführt, aber man kann sie als Sprungziel wählen. Eine REM-Anweisung wird durch ein Zeilenende, nicht durch ein ":" beendet. In diesem Fall wird durch BASIC das V=Ø nie ausgeführt.	5ØØ REM NEUER BEGINN V=Ø 5Ø5 REM NEUER BEGINN =Ø:V=Ø
RESTORE	RESTORE Erlaubt das erneute Lesen von DATA-Anweisungen. Nach einem RESTORE wird der erste gelesene Wert der sein, der als erster in der ersten DATA-Anweisung im Programm geschrieben wurde. Der zweite gelesene Wert wird der zweite in der ersten DATA-Anweisung sein usw., wie in einer normalen READ-Operation.	51Ø RESTORE
RETURN	RETURN Veranlaßt die Rückkehr zu der Anweisung, die der zuletzt ausgeführten GOSUB-Anweisung folgt.	5Ø RETURN
STOP	STOP Veranlaßt den Stop der Programmausführung und den Eintritt in den Kommando-Modus. Druckt "BREAK IN LINE 9ØØ" aus (In diesem Beispiel). CONT nach einem STOP verzweigt zu der dem STOP folgenden Anweisung.	9ØØ STOP
USR(W)	USR (Argument) Ruft ein vom Anwender in Maschinensprache geschriebenes Unterprogramm mit dem Argument W auf (siehe Abschnitt 4.8).	2ØØ V=USR(W)
WAIT	WAIT Adresse, Maske (,Select) Diese Anweisung liest den Inhalt der Adresse I, führt ein Exklusiv-ODER von K und dem Zustand der Maske aus und verrUNDet das Ergebnis mit J, bis ein von Null verschiedenes Ergebnis erscheint. Die Programmausführung geht bei der der WAIT-Anweisung folgenden Anweisung weiter. Falls die WAIT- Anweisung nur zwei Argumente hat, wird K zu Null angenommen. Falls Sie auf ein Bit warten bis es Null wird, sollte an der entsprechenden Stelle in K eine 1 stehen, J und K müssen >=Ø und <=255 sein.	8Ø5 WAIT I,J,K 8Ø6 WAIT I,J

3.5 Ein-/Ausgabe-Anweisungen

Anweisung	Syntax/Funktion	Beispiel
DATA	DATA Wert (,Wert...) Legt Daten fest, die von links nach rechts gelesen werden. Die Daten werden vom Programm also so gelesen wie sie geschrieben wurden. Strings können von DATA-Anweisungen gelesen werden. Falls Sie im STRING führende Leerzeichen (Blanks), Doppelpunkte (:) oder Kommas (,) wünschen, müssen Sie den STRING in doppelte Anführungszeichen setzen. Es ist nicht erlaubt in STRING-Felder Anführungszeichen zu schreiben. (" "BASIC" ") ist verboten.	1Ø DATA 1, 3,–1E3,.Ø4 2Ø DATA "FØØ",ZØØ

Anweisung	Syntax/Funktion	Beispiel
INPUT	INPUT("Aufforderungsstring";) Variable (,Variable)...	3Ø INPUT V,W,W2
	Verlangt Daten von der Tastatur (die eingegeben werden müssen). Jeder Wert muß vom Programm durch ein Komma (,) getrennt werden. Dem letzten eingegebenen Wert sollte ein Wagenrücklauf folgen. Ein "?" wird als Aufforderungszeichen ausgegeben. Als Antwort auf eine INPUT-Anweisung dürfen nur Konstante, wie etwa 4.5E–3 oder "LOTTE" eingegeben werden. Falls die INPUT-Anweisung mehr Daten als eingegeben wurden benötigt, wird "??" ausgegeben, und der Rest der Daten sollte eingegeben werden.	
	Falls mehr Daten als benötigt eingegeben werden, wird die Warnung "EXTRA IGNORED" ausgegeben. Strings müssen im gleichen Format, wie bei der DATA-Anweisung schon erläutert, eingegeben werden.	
	Ein Aufforderungsstring wird geschrieben, sofern gewünscht, bevor Daten von der Tastatur erwartet werden. Falls auf eine INPUT-Anweisung nur Wagenrücklauf eingegeben wird, kehrt BASIC in den Kommando-Modus zurück. Die Eingabe von CONT nach einer Unterbrechung eines INPUT-Kommandos wird das Programm erneut an der INPUT-Anweisung starten.	4Ø INPUT "A EINGEBEN";A 5Ø INPUT "WERT";V
INPUT!	Eine INPUT!-Anweisung wird auch dann Daten am Drucker ausgeben, wenn dieser ausgeschaltet ist.	3ØØ INPUT!X,Y,Z
PRINT	PRINT Ausdruck (,Ausdruck)	36Ø PRINT X,Y,Z
	Gibt den Wert von Ausdrucken am Display oder Drucker aus. Falls die Reihe der auszugebenden Werte nicht mit einem Komma (,) oder Strichpunkt (;) endet, wird ein Wagenrücklauf/Zeilenvorschub ausgeführt, nachdem alle Werte gedruckt worden sind. STRINGS, in Anführungszeichen (") eingeschlossen, können auch angegeben werden. Falls ein Strichpunkt zwei Werte trennt, werden die Werte nebeneinander ausgedruckt. Falls ein Komma nach einem Ausdruck in der Programmzeile steht und der Schreibkopf in Stellung 11 oder weiter rechts ist, wird eine Wagenrücklauf/Zeilenvorschubkombination ausgeführt. Falls der Schreibkopf vor der Position 11 steht, werden Zwischenräume gedruckt, bis der Kopf am Beginn einer neuen Zehnerspalte steht. Falls ein Blank-(Leerzeichen)-String in Anführungszeichen steht (wie in Zeile 37Ø), wird ein Wagenrücklauf/Zeilenvorschub ausgeführt.	37Ø PRINT " " 38Ø PRINT X,Y; 39Ø PRINT "WERT IST";A 4ØØ PRINT A2,B,
	STRING-Ausdrücke können durch PRINT ausgegeben werden.	41Ø PRINT MID$(A$,2);
PRINT!	Bei ausgeschaltetem Drucker können Daten durch Verwendung der PRINT!-Anweisung zum Drucker geleitet werden.	42Ø PRINT!A$
READ	READ Variable (,Variable)	49Ø READ V,W
	Weist Daten, die in einer DATA-Anweisung angegeben sind, den Variablen zu. Der erste gelesene Wert wird der erste Wert der ersten DATA-Anweisung des Programms sein. Der zweite Wert wird der zweite Wert der ersten DATA-Anweisung sein usw. Sobald alle Daten der ersten DATA-Anweisung gelesen worden sind, wird der nächste Wert, der gelesen wird, der erste Wert in der zweiten DATA-Anweisung sein. Der Versuch, mehr Daten zu lesen als in DATA-Anweisungen hinterlegt sind, wird einen OD-Fehler (out of DATA) zur Folge haben.	

Anweisung	Syntax/Funktion	Beispiel
SPC	SPC Ausdruck Druckt I Zwischenraum-(Leer-)Zeichen auf dem Terminal. Kann nur in einer Print-Anweisung verwendet werden. I muß zwischen =>Ø und <=255 liegen, sonst tritt ein FC-Fehler auf.	25Ø PRINT SPC(I)
TAB(I)	TAB Ausdruck Springt zur angegebenen Print-Stelle (Spalte) des Druckers. Kann nur in PRINT-Anweisungen verwendet werden. Null ist die ganz linke Stelle des Terminals, 19 die ganz rechte. Falls der Drucker rechts von Position I steht, wird nichts gedruckt. I muß >=Ø und <=255 sein. Falls I>19 ist, wird der Drucker die benötigte Anzahl von Zeilen überspringen, um zur gewünschten Position zu kommen.	24Ø PRINT TAB(I)

3.6 String-Funktionen

Anweisung	Syntax/Funktion	Beispiel
ASC	ASC (Ausdruck) Druckt den numerischen ASCII-Wert des ersten Zeichens des Strings X$ aus. Siehe Abschnitt 4.6 ASCII-Zeichen-Wert-Umwandlungstabelle. Falls X$ ein Null-String ist, erscheint ein FC-Fehler.	3ØØ PRINT ASC(X$)
CHR$	CHR$ (Ausdruck) Gibt einen Ein-Zeichen-String aus, in dem das alphanumerische ASCII-Zeichen steht, dessen numerischer Wert im Argument (I) steht. I muß ≥ Ø und ≤ 255 sein, um eine Fehlermeldung zu vermeiden. Außerdem werden am Drucker und an der Anzeige nur Zeichen mit einem Dezimalwert 32 bis 94 sinnvoll dargestellt. Siehe Abschnitt 4.6.	275 PRINT CHR$(I)
GET	GET X$ Holt ein Zeichen von der Tastatur. Falls Daten an der Tastatur anliegen, werden sie der Variablen, die in der GET-Anweisung spezifiziert ist, zugewiesen. Falls keine Daten erhältlich sind, wird GET nicht erwartet und BASIC setzt das Programm fort. GET kann nur als indirektes Kommando verwendet werden.	1Ø GET A$
LEFT$	LEFT$ (String, Länge) Übergibt die linken I Zeichen des String-Ausdrucks X$. Falls I<=Ø oder >255 ist, wird ein FC-Fehler auftauchen.	31Ø PRINT LEFT$ (X$,I)
LEN	LEN(String) Übergibt die Länge des String-Ausdrucks X$ (Zeichen, Bytes). Nicht abdruckbare Zeichen und Zwischenräume werden mitgezählt.	22Ø PRINT LEN(X$)
MID$	MID$ (String, Start(,Länge)) Wird MID$ mit zwei Argumenten aufgerufen, werden die Zeichen des String-Ausdrucks X$ ab der Stelle I übergeben. Falls I größer als LEN(X$) ist, übergibt MID$ einen Nullstring (Länge Null). Falls I<=Ø oder >255 ist, erscheint ein FC-Fehler. Wird MID$ mit drei Argumenten aufgerufen, wird ein String übergeben, der aus Zeichen des String-Ausdrucks X$ besteht. Er beginnt mit dem I-ten Zeichen und ist J Zeichen lang. Falls I > LEN(X$) ist, übergibt MID$ einen Nullstring. Falls I oder J<=Ø oder >255 sind, erscheint ein FC-Fehler. Falls J mehr Zeichen angibt als im String übrig sind, werden alle Zeichen vom I-ten Zeichen an übergeben.	33Ø PRINT MID$ (X$,I) 34Ø PRINT MID$ (X$,I,J)

Anweisung	Syntax/Funktion	Beispiel
RIGHT$	RIGHT$ (String, Länge) Übergibt die rechten I Zeichen des String-Ausdrucks X$. Ist I<=Ø oder >255, tritt ein FC-Fehler auf. Ist I>=LEN(X$), dann wird RIGHT$ den ganzen String X$ übergeben.	32Ø PRINT RIGHT$ (X$,I)
STR$	STR$ Ausdruck Übergibt einen String, der die Zeichen des numerischen Ausdrucks X enthält. Beispielsweise ist STR$(3.1)="3.1".	29Ø PRINT STR$ (X)
VAL	VAL Ausdruck Übergibt eine Zahl, die dem String-Ausdruck X$ entspricht. z.B. VAL("3.1")=3.1. Falls das erste von einem Leerzeichen verschiedene Zeichen kein (+),(–) (Plus,Minus) Zeichen, keine Ziffer oder kein Dezimalpunkt (.) ist, wird Null übergeben.	28Ø PRINT VAL (X$)

3.7 Arithmetische Funktionen

Anweisung	Syntax/Funktion	Beispiel
ABS	ABS (Ausdruck) Übergibt den absoluten Wert des Ausdrucks X. ABS übergibt X, falls X>=Ø ist, sonst –X.	12Ø PRINT ABS(X)
ATN	ATN (Ausdruck) Übergibt den Arcus-Tangens des Ausdrucks X. Das Ergebnis wird in Bogenmaß ausgegeben im Bereich von $-\pi/2$ bis $+\pi/2$ ($\pi/2$ = 1.57Ø8). Siehe Abschnitt 4.9.	21Ø PRINT ATN(X)
COS	COS (Ausdruck) Übergibt den Cosinus des Ausdrucks X, wobei X im Bogenmaß angenommen wird. (Siehe SIN)	2ØØ PRINT COS(X)
EXP	EXP (Ausdruck) Übergibt die Eulersche Konstante e (= 2.71828) hoch dem Ausdruck X(e^X). Das maximale Argument (X), das keinen Überlauf verursacht ist 88.Ø296919.	15Ø PRINT EXP(X)
INT	INT (Ausdruck) Übergibt die größte ganze Zahl, die <= dem Ausdruck X ist. z.B. INT(.23)=Ø, INT(7)=7, INT (–.1)=–1, INT(–2)=–2, INT(1.1)=1. Der folgende Ausdruck rundet X auf D dezimale Stellen: INT(X*1Ø^D+.5)/1Ø^D	14Ø PRINT INT(X)
LOG	LOG (Ausdruck) Übergibt den natürlichen Logarithmus des Ausdrucks X(Basis e). Um einen Logarithmus zu einer anderen Basis zu berechnen, benutzt man die Formel: LOG(X)/LOG(Y). Beispiel: Der alg. Logarithmus zur Basis 1Ø von 7 ist LOG(7)/LOG(1Ø).	16Ø PRINT LOG(X)
RND	RND (Parameter) Erzeugt eine Zufallszahl zwischen Ø und 1. Der Parameter X steuert die Generierung von Zufallszahlen wie folgt: X<Ø startet eine neue Reihe von Zufallszahlen, wobei X verwendet wird. Der Aufruf von RND mit	17Ø PRINT RND(X)

Anweisung	Syntax/Funktion	Beispiel
RND	dem gleichen X startet die gleiche Zufallszahlenreihe. X=Ø übergibt die zuletzt generierte Zufallszahl. Wiederholte Aufrufe mit RND(Ø) übergeben immer die gleiche Zufallszahl. X>Ø erzeugt eine neue Zufallszahl zwischen Ø und 1. Beachten Sie, daß (B–A)*RND(1)+A eine Zufallszahl zwischen A und B erzeugen kann.	
SGN	SGN (Ausdruck) Übergibt 1 falls X>Ø, Ø falls X=Ø und –1 falls X<Ø	23Ø PRINT SGN(X)
SIN	SIN (Ausdruck) Übergibt den Sinus des Ausdrucks X. X wird im Bogenmaß angenommen. Beachten Sie: COS(X)=SIN(X+3.14159/2); 1Rad=18Ø/ñ =57.2958 Grad; damit ist der Sinus von X in Grad=SIN(X/57.2958).	19Ø PRINT SIN(X)
SQR	SQR (Ausdruck) Übergibt die Quadratwurzel des Ausdrucks X. Ein FC-Fehler erscheint, falls X kleiner Null ist.	18Ø PRINT SQR(X)
TAN	TAN (Ausdruck) Übergibt den Tangens des Ausdrucks A. X im Bogenmaß.	2ØØ PRINT TAN(X)

3.8 Abgeleitete Funktionen

Die Funktionen, die nicht Bestandteil von BASIC sind, können mit den existierenden BASIC-Funktionen dargestellt werden.

Funktion	Funktion, dargestellt mit BASIC
SEKANS	SEC(X) = 1/COS(X)
KOSEKANS	CSC(X) = 1/SIN(X)
KOTANGENS	COT(X) = 1/Tan(X)
ARKUSSINUS	ARCSIN(X) = ATN(X /SQR(– X * X + 1))
ARKUSKOSINUS	ARCCOS (X) =–ATN(X /SQR(– X * X + 1))+1.57Ø8
ARKUSSEKANS	ARCSEC(X) = ATN(SQR(X * X – 1)) + (SGN(X)– 1) * 1.57Ø8
ARKUSKOSEKANS	ARCCSC(X) = ATN(1/SQR(X * X – 1)) + (SGN(X)– 1) * 1.57Ø8
ARKUSKOTANGENS	ARCCOT(X) =–ATN(X) + 1.57Ø8
HYPERBELSINUS	SINH(X) = (EXP(X)– EXP(–X))/2
HYPERBELKOSINUS	COSH(X) = (EXP(X)+ EXP(–X))/2
HYPERBELTANGENS	TANH(X) =–EXP (–X)/ (EXP(X)+EXP(–X)) *2+1
HYPERBELSEKANS	SECH(X) = 2/ (EXP(X)+EXP(–X))
HYPERBELKOSEKANS	CSCH(X) = 2/ (EXP(X)–EXP(–X))
HYPERBELKOTANGENS	COTH(X) = EXP(–X) / (EXP(X)–EXP(–X)) *2+1
INVERSER HYPERBELSINUS	ARCSINH(X) = LOG(X+SQR(X*X+1))
INVERSER HYPERBELKOSINUS	ARCCOSH(X) = LOG(X+SQR(X*X–1))
INVERSER HYPERBELSEKANS	ARCSECH(X) = LOG((SQR(–X*X+1)/X)
INVERSER HYPERBELKOSEKANS	ARCCSCH(X) = LOG((SGN(X)*SQR(X*X+1)+1)/X)
INVERSER HYPERBELKOTANGENS	ARCCOTH(X) = LOG((X+1) / X–1))/2
INVERSER HYPERBELTANGENS	ARCTANH(X) = LOG((1+X) / (1–X))/2

4. Betriebsinstruktionen

4.1 Inbetriebnahme PC 100 KIT

Bevor Sie die BASIC ROM's in die Hand nehmen, sollten Sie, um Schäden am PC 100 oder an angeschlossenen Geräten zu vermeiden, folgende Vorsichtsmaßnahmen beachten:

Nicht abgedeckte Baugruppe!

Gegenstände die in die Baugruppe fallen oder auf ihr abgelegt werden, können den Drukker, die Anzeige und andere Bauteile beschädigen. Gelangt Flüssigkeit in die Baugruppe, können Kurzschlüsse entstehen.

MOS-Teile!

Mikrokomputerbauteile werden in Metall-Oxid-Silizium-Technologie (MOS) hergestellt. Unachtsames Anlegen hoher Spannungen können MOS-Bauteile zerstören.

Vorsichtsmaßnahmen:

- ○ Entladen Sie Ihren Körper von statischen Aufladungen, indem Sie einen geerdeten Punkt berühren (etwa ein geerdetes Gerätegehäuse). Diese Vorsichtsmaßnahme ist besonders wichtig, falls Sie im Raum Teppichböden und eine niedrige relative Luftfeuchtigkeit haben.
- ○ Überprüfen Sie, ob alle Meßgeräte, alle peripheren Geräte und alle elektrischen Werkzeuge (z.B. Lötkolben) ordnungsgemäß geerdet sind.

Offene Spannungen!
Die Versorgungsspannungen (5V–; 24 V–) liegen an vielen Stellen der Baugruppe offen. Ein Kurzschluß dieser Stellen gegen Masse kann falsche Funktionen oder dauernden Schaden zur Folge haben.

Schalten Sie die Spannungen des PC 100 ab. Prüfen Sie die Anschlüsse der beiden BASIC-ROM's, ob sie gerade sind und ob keine Fremdkörper dazwischen stecken. Durch Gegenhalten unter den Sockeln der Baugruppen einstecken.
ROM Nummer 3252 in den Sockel Z 25 und ROM Nummer 3226 in den Sockel Z 26. Beachten Sie die richtige Lage der Bausteine und prüfen Sie, ob sie korrekt in den Sockeln stecken.

4.2 Fehlermeldungen

Nach Auftreten eines Fehlers geht BASIC in die Befehlsebene zurück und zeigt ^ an. Werte von Variablen und das Programm bleiben erhalten, aber das Programm kann nicht fortgesetzt werden und alle GOSUB und FOR-Zusammenhänge sind verloren.

Das Format der Fehlermeldungen:

Direkte Anweisung	?XX ERROR
indirekte Anweisung	?XX ERROR IN YYYYY

In beiden Beispielen wird für "XX" der Fehlercode eingesetzt. "YYYYY" wird die Zeilennummer sein in der der Fehler in der indirekten Anweisung auftrat.

Fehlercode	Bedeutung
BS	Falsche Indizierung (Bad Subscript). Es wurde versucht, sich auf ein Matrix-Element zu beziehen, das außerhalb der Dimension der Matrix liegt. Dieser Fehler kann auch auftreten, wenn die falsche Dimension als Matrix-Bezug gewählt wird. z.B. LET A(1,1,1)=Z wenn aber A mit DIM A(2,2) dimensioniert wurde.
CN	Folgefehler (Continue Error). Versuch ein nicht existierendes Programm, ein Programm nach einem Fehler oder nachdem eine neue Zeile in das Programm eingetragen wurde, fortzusetzen.
DD	Doppelte Dimensionierung (Double Dimension). Nachdem eine Matrix dimensioniert wurde, wurde noch eine Dimensionsanweisung für die gleiche Matrix entdeckt. Dieser Fehler tritt dann oft auf, falls einer Matrix die vorher definierte Dimension 1Ø gegeben wurde, da auf eine Anweisung wie A(I)=3 gestoßen wurde und erst später im Programm ein DIM A(1ØØ) gefunden wurde.

Fehlercode	Bedeutung
FC	Funktionsaufruffehler (Function Call Error). Der Parameter, der an eine mathematische oder String-Funktion übergeben wurde, war außerhalb des Bereichs. FC-Fehler können auftreten bei 1) einem negativen Matrizen-Index (LET A(−1)=Ø) 2) einem zu großen Matrizen-Index (>32767) 3) einem negativem oder Null-Argument für LOG 4) negativem Argument für SQR 5) A^B mit negativen A und ungradzahligem B 6) einem USR-Aufruf, ohne die Adresse des Maschinensprachen-Unterprogramms übergeben zu haben. 7) Aufrufen von MID$, LEFT$, RIGHT$, WAIT, PEEK, POKE, TAB, SPC oder ON...GOTO mit unrichtigen Argumenten.
ID	Direkt Verboten (Illegal Direct). Sie dürfen eine INPUT, DEF oder GET-Anweisung nicht als direktes Kommando verwenden.
LS	Zu langer String (Long String). Mit dem Verkettungsoperator wurde versucht, einen String mit mehr als 255 Zeichen herzustellen.
NF	NEXT ohne FOR. Die Variable in einer NEXT-Anweisung gehört zu keiner vorher ausgeführten FOR-Anweisung.
OD	Keine Daten da (Out of Data). Eine READ-Anweisung wurde ausgeführt, aber alle DATA-Zuweisungen im Programm sind schon gelesen worden. Das Programm versuchte, zuviele Daten zu lesen oder zu wenig Daten wurden in das Programm eingeschlossen.
OM	Speicher voll (Out of Memory). Programm zu groß, zu viele Variable, zu viel For-Schleifen, zu viele GOSUB's, zu komplizierter Ausdruck oder eine Kombination dieser Fälle.
OV	Überlauf (Overflow). Das Ergebnis einer Berechnung war zu groß und im BASIC-Zahlenformat nicht mehr darstellbar. Falls ein Unterlauf auftritt, wird als Ergebnis Null angenommen und die Programmausführung ohne Fehlermeldung fortgesetzt.
RG	RETURN ohne GOSUB. Es wurde auf eine RETURN-Anweisung gestoßen ohne vorherige Ausführung einer GOSUB-Anweisung.
SN	Syntax-Fehler. Fehlende Klammern in dem Ausdruck, nicht zulässige Zeichen in einer Zeile, unkorrekte Satzzeichen usw.
ST	Zeitweiliger Stringfehler (String Temporaries). Ein Stringausdruck war zu komplex. Machen Sie zwei kürzere Strings daraus.
TM	Arten falsch (Type dismatch). Die linke Seite einer Zuweisung war eine unnumerische Variable, und die rechte Seite war ein String oder umgekehrt; oder einer Funktion, die ein String-Argument erwartete, wurde eine Zahl übergeben oder umgekehrt.
UF	Undefinierte Funktion. Man bezog sich auf eine Anwender-Funktion, die nie definiert wurde.
US	Undefinierte Anweisung (Undefined Statement). In einem GOTO, GOSUB oder THEN wurde eine Anweisung eingegeben die nicht existiert.
/ Ø	Division durch Null.

4.3 Speicherplatz-Sparen

Folgende Hinweise sollen Ihnen helfen, Ihr Programm zu kürzen und damit Speicherplatz zu sparen:

4.3.1 Verwenden Sie Mehrfachanweisungen in einer Zeile. Pro Zeile eines Programms benötigt BASIC fünf Bytes zusätzlich. Zwei dieser fünf Bytes enthalten binär die Zeilennummer. Dies bedeutet, daß unabhängig von der Stellenzahl Ihre Zeilennummer (Minimum ist Ø, Maximum ist 63999) die gleiche Anzahl von Bytes gebraucht wird. Indem Sie soviele Anweisungen wie möglich in eine Zeile packen, sparen Sie Bytes ein.

4.3.2 Löschen Sie alle unnötigen Zwischenräume in Ihrem Programm.
Zum Beispiel benötigt
```
10 PRINT X, Y, Z
```
drei Bytes mehr als
```
10 PRINTX,Y,Z
```
Anmerkung: Alle Zwischenräume zwischen der Zeilennummer und dem ersten Zeichen werden ignoriert.

4.3.3 Löschen Sie alle REM-Zeilen. Jede REM-Anweisung benötigt mindestens ein Byte plus der Anzahl der Kommentarzeichen.
Zum Beispiel benötigt die Anweisung
```
130 REM DIES IST EIN KOMMENTAR
```
23 Bytes im Speicher.
In der Anweisung
```
140 X=X+Y:REM NEUE SUMME
```
benötigt REM 15 Bytes einschließlich des Doppelpunktes vor dem REM.

4.3.4 Verwenden Sie Variable statt Konstante. Angenommen, Sie verwenden die Konstante 3.14159 zehnmal in Ihrem Programm.
Falls Sie eine Anweisung 10 P=3.14159 in das Programm einfügen und jedesmal P statt 3.14159 verwenden, wenn es benötigt wird, werden Sie 40 Bytes sparen. Das Programm wird auch schneller ausgeführt werden.

4.3.5 Ein Programm muß nicht mit END am Ende abgeschlossen werden. Sie können also die END-Anweisung am Ende des Programms löschen.

4.3.6 Verwenden Sie die gleichen Variablen erneut. Falls Sie eine Variable T haben, die ein Zwischenergebnis in einem Teil Ihres Programms aufnehmen soll und Sie später in Ihrem Programm wieder ein Zwischenergebnis haben, verwenden Sie wieder T. Oder falls Sie vom Anwender JA oder NEIN erwarten, verwenden Sie bei zwei Fragen zu verschiedenen Zeiten Ihres Programms auch dieselbe zeitliche Variable A$, um das Ergebnis zu speichern.

4.3.7 Verwenden Sie GOSUB's, um Teile Ihres Programms, die identische Auswirkungen haben, auszuführen.

4.3.8 Verwenden Sie auch die mit Null indizierten Werte von Matrizen, etwa A(0),B(0,X).

Information über Speicherzuweisungen!

Einfache numerische Variable (keine Matrizen), wie V benötigen sechs Byte. 2 Byte für den Variablennamen, 4 für den Wert. Einfache (keine Matrizen) String-Variable benötigen auch 6 Bytes. 2 für den Variablennamen, 2 für die Länge und 2 für einen Zeiger. Matrizenvariable benötigen mindestens 12 Bytes. 2 Bytes werden für den Variablennamen, 2 für die Größe der Matrix, 2 für die Zahl der Dimension und für jede Dimension 4 Bytes für jedes Element, gebraucht. String-Variable benötigen ferner ein Byte für jedes Zeichen des Strings. Dies gilt immer, gleich ob die String-Variable eine einfache ist wie A$ oder ein Element einer String-Matrix wie Q1$(5,2).

Sobald eine Funktion mit DEF definiert wird, werden sechs Bytes benötigt, um die Definition zu speichern. Reservierte Worte, wie FOR, GOTO oder NOT und die Namen der eingebauten Funktionen, wie COS, INT und STR$ benötigen nur ein Byte Programmspeicher. Alle weiteren Zeichen in Programmen benötigen ein Byte pro Zeichen. Sobald ein Programm ausgeführt wird, wird dem Stack (Stapelspeicher) folgender Platz zugewiesen:

- Jede aktive FOR...NEXT-Schleife benötigt 22 Bytes
- Jedes aktive GOSUB (eines, das noch kein RETURN gefunden hat) benötigt 6 Bytes.
- Jede Klammer, die in einem Ausdruck gefunden wird, benötigt 4 Bytes und jedes berechnete Zwischenergebnis in einem Ausdruck benötigt 12 Bytes.

4.4 Erhöhung der Arbeitsgeschwindigkeit

Die im Folgenden gegebenen Hinweise sollen die Ausführungszeit Ihres BASIC-Programms verringern helfen. Beachten Sie, daß einige dieser Hinweise dieselben sind wie die, die zum Speicherplatz-Sparen gegeben wurden. Dies heißt, daß Sie in vielen Fällen gleichzeitig schnellere und kürzere Programme erhalten können.

4.4.1 Erhöhung der Arbeitsgeschwindigkeit (Wichtigster Hinweis)
Verwenden Sie Variable statt Konstante. Es braucht länger, eine Konstante in ihre Gleitkommadarstellung (Floating Point) umzuwandeln, als den Wert einer einfachen oder einer Matrix-Variablen zu holen. Dies ist besonders in FOR...NEXT-Schleifen oder in anderen wiederholt auszuführenden Teilen wichtig.

4.4.2 Löschen Sie alle unnötigen Zwischenräume und REM's in Ihrem Programm. Dies verkürzt die Ausführungszeit ein wenig, da BASIC sonst Zwischenräume und REM-Anweisungen ignoriert oder darüber hinwegspringen muß.

4.4.3 Variable, die während der Programmausführung zuerst erkannt wurden, werden am Beginn der Variablenliste abgespeichert. Dies bedeutet, daß eine Anweisung wie 5 A=Ø:B=A:C=A die Variable A zuerst, dann B und dann C in die Symboltabelle bringen wird (Angenommen, Zeile 5 ist die erste ausgeführte Zeile Ihres Programms). Findet BASIC später im Programm einen Verweis zu A, muß es nur im Anfang der Symboltabelle suchen um A zu finden; es muß zweimal suchen, um B zu finden und dreimal absuchen, um C zu finden.

4.4.4 Verwenden Sie NEXT-Anweisungen ohne Index-Variable. NEXT ist eine Spur schneller als NEXT I, da nicht geprüft werden muß, ob die im NEXT angegebene Variable die gleiche ist wie die in der letzten FOR-Anweisung.

4.5 Umschreiben von BASIC-Programmen

Obwohl BASIC in vielen Computern fast identisch ist, gibt es einige Ungleichheiten, die Sie beachten sollten, falls Sie ein BASIC-Programm, das nicht in PC 100-BASIC geschrieben wurde, umsetzen wollen.

4.5.1 Matrix-Indizes. Einige BASIC's verwenden "[" und "]" nicht "(" und ")", um Matrix-Indizes zu bezeichnen.

4.5.2 Strings. Viele BASIC's zwingen Sie, die Länge von Strings zu dimensionieren (deklarieren), bevor Sie sie verwenden. Sie sollten alle Dimensionierungsanweisungen dieser Art aus dem Programm entfernen. In einigen dieser BASIC's deklariert eine Anweisung der Form DIM A$(I,J) eine String-Matrix mit J Elementen der Länge I. Wandeln Sie DIM-Anweisungen dieser Art in die des PC 100-BASIC's um: DIMA$(J).

PC 100-BASIC verwendet "+" für Stringverkettungen, nicht "," oder "&".
PC 100-BASIC verwendet LEFT$, RIGHT$ und MID$, um Teile aus Strings zu erzeugen. Andere BASIC's verwenden A$(I) um auf das I-te Zeichen des Strings zuzugreifen und A$(I,J) um einen Teilstring von A$ von der Zeichenposition I zur Zeichenposition J zu schaffen.
Wandeln Sie wie folgt:

Andere	PC 100
A$(I)	MID$(A$,I,1)
A$(I,J)	MID$(A$,I,J–I+1)

Dies setzt voraus, daß die Referenz zu einem Unterstring von A$ ein Ausdruck ist und auf der rechten Seite einer Anweisung steht. Falls die Referenz zu A$ auf der linken Seite einer Anweisung steht und X$ der String-Ausdruck ist, der Zeichen in A$ ersetzen soll, wandeln Sie wie folgt um:

Andere	PC 100
A$(I)=X$	A$=LEFT$(A$,I–1)+X$+MID$(A$,I+1)
A$(I,J)=X$	A$=LEFT$(A$,I–1)+X$+MID$(A$,J+1)

4.5.3 Mehrfachzuweisungen. Einige BASIC's erlauben Anweisungen der Form: 5ØØ LET B=C=Ø. Diese Anweisung würde die Variablen B und C auf Null setzen.
Im PC 100-BASIC ist die Auswirkung eine völlig andere. Alle Gleichheitszeichen rechts von dem ersten würden als logische Vergleichsoperatoren interpretiert. Dies würde die Variable B auf –1 setzen, falls C gleich Null wäre. Falls C ungleich Null wäre, würde B zu Null gesetzt. Der einfachste Weg, Anweisungen wie diese umzuschreiben ist wie folgt:
5ØØ C=Ø:B=C

4.5.4 Einige BASIC's verwenden "/" statt ":", um Mehrfachanweisungen in einer Zeile zu trennen.
Ändern Sie "/" zu ":".

4.5.5 Programme, die die MAT-Funktion, die es in einigen BASIC's gibt, verwenden, müssen mit FOR...NEXT-Schleifen für die passende Funktion umgeschrieben werden.

4.6 ASCII – Zeichen – Codes

Dezimal	Zeichen	Dezimal	Zeichen	Dezimal	Zeichen
ØØØ	NUL	Ø43	+		
ØØ1	SOH	Ø44	,	Ø86	V
ØØ2	STX	Ø45	-	Ø87	W
ØØ3	ETX	Ø46	.	Ø88	X
ØØ4	EOT	Ø47	/	Ø89	Y
ØØ5	ENQ	Ø48	Ø	Ø9Ø	Z
ØØ6	ACK	Ø49	1	Ø91	[
ØØ7	BEL	Ø5Ø	2	Ø92	√
ØØ8	BS	Ø51	3	Ø93	]
ØØ9	HT	Ø52	4	Ø94	∧
Ø1Ø	LF	Ø53	5	Ø95	←
Ø11	VT	Ø54	6	Ø96	`
Ø12	FF	Ø55	7	Ø97	a
Ø13	CR	Ø56	8	Ø98	b
Ø14	SO	Ø57	9	Ø99	c
Ø15	SI	Ø58	:	1ØØ	d
Ø16	DLE	Ø59	;	1Ø1	e
Ø17	DC1	Ø6Ø	<	1Ø2	f
Ø18	DC2	Ø61	=	1Ø3	g
Ø19	DC3	Ø62	>	1Ø4	h
Ø2Ø	DC4	Ø63	?	1Ø5	i
Ø21	NAK	Ø64	@	1Ø6	j
Ø22	SYN	Ø65	A	1Ø7	k
Ø23	ETB	Ø66	B	1Ø8	l
Ø24	CAN	Ø67	C	1Ø9	m
Ø25	EM	Ø68	D	11Ø	n
Ø26	SUB	Ø69	E	111	o
Ø27	ESCAPE	Ø7Ø	F	112	p
Ø28	FS	Ø71	G	113	q
Ø29	GS	Ø72	H	114	r
Ø3Ø	RS	Ø73	I	115	s
Ø31	US	Ø74	J	116	t
Ø32	SPACE	Ø75	K	117	u
Ø33	!	Ø76	L	118	v
Ø34	"	Ø77	M	119	w
Ø35	#	Ø78	N	12Ø	x
Ø36	$	Ø79	O	121	y
Ø37	%	Ø8Ø	P	122	z
Ø38	&	Ø81	Q	123	{
Ø39	'	Ø82	R	124	/
Ø4Ø	(	Ø83	S	125	}
Ø41	)	Ø84	T	126	~
Ø42	*	Ø85	U	127	DEL

LF=Line-Feed (Zeilenvorschub)
FF=Form-Feed (Papiervorschub)
CR=Carriage Return(Wagenrücklauf)
DEL=(Rubout am TTY)

4.7 Maschinensprache – Unterprogramme

PC 100-BASIC erlaubt einem Benutzer mit Hilfe der Funktionen USR(W) Unterprogramme in Maschinensprache einzubinden. Diese Funktion erlaubt die Übergabe eines Parameters von BASIC zum Unterprogramm.
Zuerst muß ausreichend Speicherplatz für das Unterprogramm bereitgestellt werden. PC 100-BASIC verwendet immer den gesamten RAM-Speicher, beginnend ab der Adresse 53Ø Dezimal (212 Hexadezimal) bis zur vom Anwender vorgegebenen Grenze. Um die Speicherzuweisung zu begrenzen, sollte der Anwender BASIC's Abfrage MEMORY SIZE?

mit der Zahl kleiner 3566 beantworten, falls wir ein 4k-System haben (4Ø96–53Ø). Dadurch wird am oberen Ende des RAM's ausreichend Platz für das Unterprogramm gelassen. Falls der Anwender auf die Frage MEMORY SIZE? 2Ø48 antwortet, bleiben an der Spitze des RAM-Bereiches 1518 Bytes für Unterprogramme in Maschinensprache frei. Beachten Sie, daß Unterprogramme in Maschinensprache immer in Speicheradressen liegen, die größer als die von BASIC verwendeten sind.
Die USR(W)-Funktion wird einen Parameter (W) an ein Unterprogramm übergeben. Er wird in einen Gleitkommawert gewandelt in den Gleitkomma-Akkumulator bei $A9 geschrieben. Das Format des Gleitkomma-Akkumulators ist:

Adresse	Inhalt
$A9	Exponent+$81($8Ø falls Mantisse=ØØ)
$AA bis $AD	Mantisse (normalisiert, Bit 7 des MSB's ist gesetzt) $AA=MSB, $AD=LSB
$AE	Vorzeichen der Mantisse (positiv, falls Mantisse positiv)

Ein Parameter, der von BASIC an ein Maschinenspracheunterprogramm übergeben wird, kann von den Unterprogrammen in eine 2-Byte-Integer-Zahl gekürzt und in $AC(MSB) und $AD(LSB) hinterlegt werden. Falls der Parameter größer als 32767 oder kleiner als –32768 ist, wird ein FC-Fehler auftreten.
Die Adresse des Unterprogramms, das eine Gleitkommazahl in eine gerade Zahl oder umgekehrt wandelt, steht in den Zellen $BØØ6, $BØØ7.

Ein Parameter, der von einem Maschinensprache-Unterprogramm an BASIC übergeben wird, wird zu einer Gleitkommazahl gewandelt. Die Adresse des Unterprogramms, das eine Integer- in eine Gleitkommazahl wandelt, steht in den Zellen $BØØ8,$BØØ9. Das ganzzahlige MSB muß im Akku, das ganzzahlige LSB im Y-Register übergeben werden. Die Startadresse des Anwender-Maschinen-Unterprogramms muß in den Zellen $Ø4 (LSB) und $Ø5 (MSB) gespeichert werden, bevor man die USR-Anweisungen ausführt. Dies wird im Allgemeinen mit dem POKE-Kommando gemacht. Beachten Sie, daß mehr als ein Maschinenunterprogramm von BASIC durch Ändern der Startadresse in $Ø4 und $Ø5 aufgerufen werden kann.

4.8 Speichern auf Kassette

PC 100-BASIC-Programme können durch Anwendung der BASIC-Kommandos SAVE und LOAD auf eine Kassette gespeichert und von dort zurückgeholt werden. Bevor Sie diese Operation ausführen, vergewissern Sie sich, daß Sie die Anweisung für den Anschluß des Recorders und die Inbetriebnahme in der PC 100-Bedienungsanleitung beachtet haben.

4.8.1 So speichern Sie ein BASIC-Programm

○ Legen Sie eine Kassette ein und spulen Sie das Band manuell dorthin, wo die Aufzeichnung beginnen soll. Denken Sie daran den Zähler zu setzen.

○ Inhalt des Speicherplatzes 41993 von 8 in 64 oder 128 ändern:

Poke 41993, 64 bzw.
Poke 41993, 128

Kontrolle: PRINT PEEK (41993)

○ Geben Sie SAVE ein. BASIC wird antworten mit:

OUT =

○ Geben Sie ein T für Tonband ein, und BASIC wird anzeigen:

OUT=T F=

○ Geben Sie den Dateinamen mit maximal 5 Zeichen ein. Falls der Dateiname PNAME ist, wird BASIC anzeigen:

OUT=T F=PNAME T=

○ Geben Sie die Nummer des Recorders 1 oder 2 ein (noch kein RETURN)

○ Nun schalten Sie den Recorder auf Aufnahme und geben RETURN: Falls Sie mit Fernsteuerung arbeiten, gehen Sie nach den Anweisungen in der PC 100-Bedienungsanleitung vor.

- ○ Sobald die Aufzeichnung beendet ist, wird BASIC anzeigen:

  ^

- ○ Schalten Sie die Aufnahme am Recorder aus.

4.8.2 So laden Sie ein BASIC-Programm von der Kassette

- ○ Legen Sie die Kassette ein und fahren Sie Ihr Band etwa 5 Zählerwerkstellen vor Beginn des gewünschten Files (Datei).

- ○ Sobald Sie im BASIC sind, geben Sie LOAD ein. BASIC wird antworten:

  IN=

- ○ Tippen Sie T für Tonband ein. BASIC wird anzeigen:

  IN=T F=

- ○ Geben Sie den Dateinamen von dem Programm ein, das Sie lesen wollen. (PNAME im vorigen Beispiel). BASIC wird anzeigen:

  IN=T F=PNAME T =

- ○ Geben Sie die Recordernummer 1 oder 2 ein (mit RETURN). BASIC wartet jetzt auf die Eingabe vom Band.

- ○ Schalten Sie den Recorder auf Wiedergabe. Beachten Sie die Angaben in der PC 100 Bedienungsanleitung.
  Während die Datei gelesen wird, wird jede Zeile angezeigt, falls der Drucker an ist, auch gedruckt. Falls beim Laden gedruckt wird, muß Zelle 41933 mit 128 beim Abspeichern geladen gewesen sein. Die Datei, die geladen wird, wird keine schon eingegebene BASIC-Anweisungen überschreiben, ausgenommen Sie haben die gleichen Zeilennummern.

- ○ Nach Beenden des Ladens wird BASIC anzeigen:

  ^

- ○ Schalten Sie nun den Recorder aus.

PC 100-BASIC-Programme können auch mit dem PC 100-Editor gespeichert und gelesen werden. Jedoch muß eine Regel beachtet werden, falls das Programm von BASIC in Zukunft wieder gelesen werden soll. Wenn BASIC ein Programm auf Kassette speichert, fügt es ein CTRL/Z-Zeichen nach der letzten Zeile ein.
Der PC 100-Editor wird das CTRL/Z nicht zurückladen, wenn er das Programm wieder lädt. Deswegen muß der Anwender ein CTRL/Z als letztes Zeichen einfügen, bevor er ein Programm mit dem Editor speichert, da BASIC das CTRL/Z-Zeichen als Enderkennung benötigt.

4.9 ATN-Implementierung

Die ATN-Funktion ist als Befehl im PC 100 Basic vorhanden. Sie muß aber initiert werden, bevor sie benutzt werden kann. Dazu dient das nachfolgende BASIC-Unterprogramm:

```
60000 REM  INIT ATN PC 100
60010 POKE 188,189
60020 POKE 189,15
60030 POKE 127,128
60040 POKE 128,15
60050 RESTORE
60060 READ A
60070 READ B
60080 IF B < 0 THEN RETURN
60090 POKE A,B
60100 A = A + 1
60110 GOTO 60070
60120 DATA  3968,11,118,179,131,189,211,121,30
60130 DATA  244,166,245,123,131,252,176,16
60140 DATA  124,12,31,103,202,124,222,83
60150 DATA  203,193,125,20,100,112,76,125
60160 DATA  183,234,81,122,125,99,48,136
60170 DATA  126,126,146,68,153,58,126,76
60180 DATA  204,145,199,127,170,170,170,19
60190 DATA  129,0,0,0,0,165,174,72
60200 DATA  16,3,32,184,204,165,169,72
60210 DATA  201,129,144,7,169,251,160,198
60220 DATA  32,78,200, 169, 128,160,15,32
60230 DATA  68,205,104,201,129,144,7,169
60240 DATA  78,160,206,32,143,197,104,16
60250 DATA  3,76,184,204,96,-1
```

Dieses Programm kann wie ein normales BASIC-Programm auf Kassette gespeichert werden und kann ebenso wie andere BASIC-Programme mit anderen Programmen gebunden werden. Deshalb wurden auch hohe Zeilennummern verwendet.

Dieses Programm muß nur einmal durchlaufen. Dann kann die ATN-Funktion des PC 100 benutzt werden. Das Programm muß daher auch nicht ständig im BASIC-Programmspeicher des PC 100 bleiben, obwohl es natürlich bequemer ist, es zusammen mit dem Programm zu laden, mit dem es verwendet wird.

Wird dieses Programm zusammen mit einem anderen Programm geladen, so können nach Initierung der ATN-Funktion die Programmzeilen des ATN-Programms gelöscht oder überschrieben werden, wenn der Speicherplatz benötigt wird.

Wenn die ATN-Funktion nicht mehr benötigt wird, kann der belegte Speicher durch das folgende Programm wieder freigemacht werden:

```
60300 REM  DELETE ATN PC 100
60310 POKE 188,135
60320 POKE 189,191
60330 POKE 127,0
60340 POKE 128,16
60350 RETURN
```

Unsere Geschäftsstellen

Bundesrepublik Deutschland und Berlin (West)

Siemens AG
Salzufer 6–8
Postfach 110560
1000 Berlin 11
☎ (030) 3939-1, Tx 1810-278
FAX (030) 3939-2630

Siemens AG
Contrescarpe 72
Postfach 107827
2800 Bremen 1
☎ (0421) 364-1, Tx 245451
FAX (0421) 364-687

Siemens AG
Lahnweg 10
Postfach 1115
4000 Düsseldorf 1
☎ (0211) 3030-1, Tx 8581301
FAX (0211) 3030-506

Siemens AG
Gutleutstraße 31
Postfach 4183
6000 Frankfurt 1
☎ (0611) 262-1, Tx 414131
FAX (0611) 262-2253

Siemens AG
Lindenplatz 2
Postfach 105609
2000 Hamburg 1
☎ (040) 282-1, Tx 2162721
FAX (040) 282-2210

Siemens AG
Am Maschpark 1
Postfach 5329
3000 Hannover 1
☎ (0511) 199-1, Tx 922333
FAX (0511) 199-2799

Siemens AG
N 7, 18 (Siemenshaus)
Postfach 2024
6800 Mannheim 1
☎ (0621) 296-1, Tx 462261
FAX (0621) 296-222

Siemens AG
Richard-Strauss-Straße 76
Postfach 202109
8000 München 2
☎ (089) 9221-1, Tx 529421

Siemens AG
Von-der-Tann-Straße 30
Postfach 4844
8500 Nürnberg 1
☎ (0911) 654-1, Tx 622251
FAX (0911) 654-3436, 3466

Siemens AG
Martin-Luther-Straße 25
Postfach 359
6600 Saarbrücken 3
☎ (0681) 3008-1, Tx 4421431
FAX (0681) 3008-254

Siemens AG
Geschwister-Scholl-Straße 24
Postfach 120
7000 Stuttgart 1
☎ (0711) 2076-1, Tx 723941
FAX (0711) 2076-706

Siemens Bauteile Service
Gründlacher Straße 260
Postfach 146
8510 Fürth-Bislohe
☎ (0911) 3001-1, Tx 623818

Europa

Belgien
Siemens S.A.
chaussée de Charleroi 116
B-1060 Bruxelles
☎ (02) 5373100, Tx 21347

Bulgarien
RUEN,
Technisches Beratungsbüro
der Siemens Aktiengesellschaft
San Stefano 14/16
BG-1504 Sofia 4
☎ 457082, Tx 22763

Dänemark
Siemens A/S
Borupvang 3
DK-2750 Ballerup
☎ (02) 656565, Tx 35313

Finnland
Siemens Osakeyhtiö
Mikonkatu 8
Fach 8
SF-00101 Helsinki 10
☎ (90), 1626-1, Tx 12465

Frankreich
Siemens Société Anonyme
39–47, boulevard Ornano
B.P. 109
F-93203 Saint-Denis CEDEX 1
☎ (16-1) 8206120, Tx 620853

Griechenland
Siemens Hellas E.A.E.
Voulis 7
P.O.B. 601
Athen 125
☎ (01) 3293-1, Tx 216291

Großbritannien
Siemens Limited
Siemens House
Windmill Road
Sunbury-on-Thames
Middlesex TW 16 7HS
☎ (09327) 85691, Tx 8951091

Irland
Siemens Limited
8, Raglan Road
Dublin 4
☎ (01) 684727, Tx 5341

Island
Smith & Norland H/F
Nóatún 4
P.O.B. 519
Reykjavik
☎ 28322, Tx 2055

Italien
Siemens Elettra S.p.A.
Via Fabio Filzi, 25/A
Casella Postale 4183
I-20124 Milano
☎ (02) 6248, Tx 36261

Jugoslawien
Generalexport
Masarikova 5/XV
Poštanski fah 223
YU-11001 Beograd
☎ (011) 684866, Tx 11287

Luxemburg
Siemens Société Anonyme
17, rue Glesener
B.P. 1701
Luxembourg
☎ 49711-1, Tx 3430

Niederlande
Siemens Nederland N.V.
Wilhelmina van Pruisenweg 26
Postbus 16068
NL-2500 BB Den Haag
☎ (070) 782782, Tx 31373

Norwegen
Siemens A/S
Østre Aker vei 90
Postboks 10, Veitvet
N-Oslo 5
☎ (02) 153090, Tx 18477

Österreich
Siemens Aktiengesellschaft
Österreich
Apostelgasse 12
Postfach 326
A-1031 Wien
☎ (0222) 7293-0, Tx 11866

Polen
PHZ Transactor S.A.
ul. Stawki 2
P.O.B. 276
PL-00-950 Warszawa
☎ 398910, Tx 815554

Portugal
Siemens S.A.R.L.
Avenida Almirante Reis, 65
Apartado 1380
Lisboa 1
☎ (019) 538805, Tx 12563

Rumänien
Siemens birou
de consultaţii tehnice
Strada Edgar-Quinet 1
R-70106 Bucureşti 1
☎ 151825, Tx 11473

Schweden
Siemens Aktiebolag
Avd. elektronikkomponenter
Norra Stationsgatan 69
Box 23141
S-10435 Stockholm
☎ (08) 241700, Tx 11672

Schweiz
Siemens-Albis AG
Freilagerstraße 28
Postfach
CH-8047 Zürich
☎ (01) 2473111, Tx 52131

Spanien
Siemens S.A.
Orense, 2
Apartado 155
Madrid 20
☎ (91) 4552500, Tx 27769

Tschechoslowakei
EFEKTIM,
Technisches Beratungsbüro Siemens AG
Anglická ulice 22
P.O.B. 1087
CS-11000 Praha 1
☎ 258417, Tx 122389

Türkei
Simko Ticaret ve Sanayi A.Ş.
Meclisi Mebusan Caddesi,
55/35, Findikli
P.K. 64, Tophane
Istanbul
☎ 452090, Tx 22290

Ungarn
Intercooperation AG,
Siemens Kooperationsbüro
Böszörményi út 9–11
P.O.B. 1525
H-1126 Budapest
☎ (01) 154970, Tx 224133

Union der Sozialistischen Sowjetrepubliken
Vertretung der Siemens AG
Kurssowoj Pereulok, Dom 1/1,
Kwartira 4,
Wchod Sojmonowskij Projezd
Postf. 77, Internationales Postamt
SU-Moskau G 34
☎ 2027711, Tx 7413

Afrika

Ägypten
Siemens Resident Engineers
33, Dokki Street
P.O.B. 775
Cairo
☎ 982671, Tx 321

Äthiopien
Siemens Ethiopia Ltd.
Ras Bitwoded Makonen Building
P.O.B. 5505
Addis Ababa
☎ 151599, Tx 21052

Algerien
Siemens Algérie S.A.R.L.
3, Viaduc du Duc des Cars
B.P. 224, Alger-Gare
Alger
☎ 615966/67, Tx 52817

Libyen
Siemens Resident Engineers
Libyan Arab People Socialist Jamahiriya
P.O.B. 46
Tripoli
☎ 41534, Tx 20029

Marokko
SETEL S.A.
Immeuble Siemens
km 1, Route de Rabat
Casablanca-Ain Sebâa
☎ 351025, Tx 21914

Nigeria
Siemens Nigeria Ltd.
Siemens House
Industrial estate 3 f,
Block A
P.O.B. 304, Apapa
Oshodi (Lagos)
☎ 842502, Tx 21357

Sudan
National Electrical
& Commercial Company
P.O.B. 1202
Khartoum
☎ 80818, Tx 642

Südafrika
Siemens Limited
Siemens House,
Corner Wolmarans and
Biccard Streets, Braamfontein
P.O.B. 4583
Johannesburg 2000
☎ (011) 7159111, Tx 58-7721

Tunesien
Sitelec S.A.,
Immeuble Saâdi - Tour C
Route de l'Ariana
Tunis-El Menzah TN
☎ 231526, Tx 12326

Zaire
Siemens Zaire S.P.R.L.
1222, Avenue Tombalbaye
B.P. 9897
Kinshasa 1
☎ 22608, Tx 21377

Amerika

Argentinien
Siemens Sociedad Anónima
Avenida Pte. Julio A. Roca 516
Casilla Correo Central 1232
RA-1067 Buenos Aires
☎ 300411, Tx 121812

Bolivien
Sociedad Comercial é Industrial
Hansa Limitada
Calle Mercado esquina Yanacocha
Cajón Postal 1402
La Paz
☎ 354445, Tx 5261

Brasilien
Siemens S.A.
Sede Central
Avenida Mutinga, 3650
Caixa Postal 1375
BR-05110 São Paulo
☎ (011) 2610211
Tx 11-23633, 11-23641

Chile
Gildemeister S.A.C.,
Area Siemens
Amunátegui 178
Casilla 99-D
Santiago de Chile
☎ 82523,
Tx TRA SGO 392, TDE 40588

Ecuador
Siemens S.A.
Avenida América y
Hernández Girón s/n.,
Casilla 3580
Quito
☎ 454000, Tx 22190

Kanada
Siemens Electric Limited
Montreal Office
7300 Trans-Canada Highway
P.O.B. 7300
Pointe Claire, Québec H9R 4R6
☎ (514) 6957300,
Tx 5-822778

Kolumbien
Siemens S.A.
Carrera 65, No. 11–83
Apartado Aéreo 80150
Bogotá 6
☎ 2610477, Tx 44750

Mexico
Siemens S.A.
Poniente 116, No. 590
cd. Ind. Vallejo
Apartado Postal 15064
México 15, D.F.
☎ 5670722, Tx 1772700

Uruguay
Conatel S.A.
Ejido 1690
Casilla de Correo 1371
Montevideo
☎ 917331, Tx 934

Venezuela
Siemens S.A.
Avenida Principal,
Urbanización Los Ruices
Apartado 3616
Caracas 101
☎ (02) 348531, Tx 25131

Vereinigte Staaten von Amerika
Siemens Corporation
186 Wood Avenue South
Iselin, New Jersey 08830
☎ (201) 494-1000
Tx WU 844491
TWX WU 7109980588

Asien

Afghanistan
Afghan Electrical Engineering
and Equipment Limited
Alaudin, Karte 3
P.O.B. 7
Kabul 1
☎ 40446, Tx 35

Bangladesch
Siemens Bangladesh Ltd.
74, Dilkusha Commercial Area
P.O.B. 33
Dacca 2
☎ 244381, Tx 824

Hongkong
Jebsen & Co., Ltd.
Prince's Building, 23rd floor
P.O.B. 97
Hong Kong
☎ 5225111, Tx 73221

Indien
Siemens India Ltd.
134A, Dr. Annie Besant Road, Worli
P.O.B. 6597
Bombay 400018
☎ 379906, Tx 112373

Indonesien
Representative Siemens AG
Jl. Kebon Sirih 4
P.O.B. 2469
Jakarta Pusat
☎ 351051, Tx 46222

Irak
Siemens Iraq Consulting Office
P.O.B. 3120
Baghdad
☎ 98198, Tx 2393

Iran
Siemens Sherkate S. (K.)
Khiabane Takhte Djamshid 32,
Siemenshaus
Teheran 15
☎ (021) 614-1, Tx 212351

Japan
Nippon Siemens K.K.
Gotanda Fujikura Building,
11–20, Nishigotanda 2-chome,
Shinagawa-ku
P.O.B. 68, Osaki
Tokyo 141-00
☎ (03) 4902171, Tx 22808

Korea (Republik)
Siemens Electrical
Engineering Co., Ltd.
Daehan Building, 8th floor,
75, Susomun-dong, Chung-ku
C.P.O.B. 3001
Seoul
☎ 7783431, Tx 23229

Kuwait
Abdul Aziz M. T. Alghanim Co.
& Partners
Abdulla Fahad Al-Mishan Building
Al-Sour Street
P.O.B. 3204
Kuwait, Arabia
☎ 423336, Tx 2131

Libanon
Ets. F. A. Kettaneh S.A.
(Kettaneh Frères)
Medawar
P.B. 110242
Beyrouth
☎ 251040, Tx 20614

Malaysia
Guthrie Engineering (Malaysia)
Sdn. Bhd.,
Electrical &
Communications Division
17, Jalan Semangat
P.O.B. 30
Petaling Jaya
☎ 773344, Tx 37573

Pakistan
Siemens Pakistan Engineering
Co. Ltd.
Ilaco House, Abdullah Haroon Road
P.O.B. 7158
Karachi 3
☎ 516061, Tx 2820

Philippinen
Engineering Equipment, Inc.
Machinery Division,
Siemens Department
E. Rodriguez Avenue
Murphy, Quezon City
P.O.Box 7160
Airport Exchange Office
Manila International Airport
Philippines 3120
☎ 773011,
Tx RCA 7222382, EEC 3695

Saudi-Arabien
Arabia Electric Ltd.
Head Office
P.O.B. 4621
Jeddah
☎ 59521, Tx 401864

Singapur
Siemens Components PTe. Ltd.
Promotion Office
19B – 45B, Jalan Tenteram
Singapore 12
☎ 550811, Tx RS 21000

Syrien
Syrian Import Export & Distribution
Co., S.A.S. SIEDCO
Port Saïd Street
P.O.B. 363
Damas
☎ 13431, Tx 11267

Taiwan
Delta Engineering Ltd.
42, Hsu Chang Street, 8th floor
P.O.B. 58497
Taipei
☎ 3114731, Tx 21826

Thailand
B. Grimm & Co., R.O.P.
1643/4, Phetburi Road (Extension)
P.O.B. 66
Bangkok 10
☎ 2524081, Tx 2614

Yemen (Arab. Republik)
Tihama Tractors
& Engineering Co. Ltd.
P.O.B. 49
Sanaa
☎ 2462, Tx 217

Australien

Australien
Siemens Industries Limited
Melbourne Office
544 Church Street
Richmond, Vic. 3121
☎ (03) 4297111, Tx 30425

6.79

SIEMENS AKTIENGESELLSCHAFT

Bestell-Nr. B 2057
Printed in Germany
KG 8795.