INPUT MIT TASTATUR

Eingabeformat

INPUT *[Eingabe-Dialog;] Variablen-Liste [;Eingabe-Dialog; Variablen-Liste] [,..]*

Beschreibung

Das INPUT Statement dient dazu, Daten über die Tastatur einzugeben. Wenn ein INPUT Statement ausgeführt wird, erfolgt eine Unterbrechung des Programmablaufes, bis Daten über die Tastatur eingegeben werden.

Der *Eingabe-Dialog* ist ein String-Ausdruck, dem ein Strichpunkt folgen muß. Wenn er eine String-Konstante dargestellt, muß er in Anführungszeichen gesetzt werden. Die Anzeige des *Eingabe-Dialoges* beginnt mit der momentanen Kursor-Position, so wie sich diese aus vorangegangenen Eingabe/Ausgabe Statements ergeben hat. Wenn man den *Eingabe-Dialog* wegläßt, wird an seiner Stelle ein Fragezeichen, gefolgt von einer Leerstelle angezeigt.

Nach dem Dialog wird der blinkende Kursor angezeigt. Wenn sich die Kursor-Position als größer 31 ergibt, wird die Anzeige gelöscht und der Kursor in Spalte 1 gesetzt, bevor der Dialog angezeigt wird. Wenn der *Eingabe-Dialog* die Länge von 30 Zeichen überschreitet, werden nur die ersten 30 Zeichen angezeigt.

Die *Variablen-Liste* ist eine Liste von durch Kommata getrennten Variablen. Die Variablen dürfen numerische oder String-Variablen, und wahlweise indiziert sein. Wenn mehr als eine Variable dem *Eingabe-Dialog* folgt, wird der Dialog nur für die erste Variable angezeigt. Danach erscheint das Dialog-Fragezeichen, bis der nächste *Eingabe-Dialog* auftritt. Jeder eingegebene Wert wird der zugehörigen Variablenbezeichnung zugeordnet, bevor der Computer den Dialog für den nächsten Wert anzeigt.

Bei der Eingabe numerischer Variablen ist es zulässig, einen numerischen Ausdruck anstelle einer numerischen Konstanten einzugeben. Der Ausdruck wird ausgewertet, und das Ergebnis der Variablen zugeordnet. Bei der Eingabe von String-Variablen werden führende und abschließende Leerstellen unterdrückt. Daher muß eine Stringvariable, wenn sie Kommata, führende oder abschließende Leerstellen enthält, in Anführungszeichen gesetzt werden. Anführungszeichen, die innerhalb eines in Anführungszeichen gesetzten Strings vorkommen, werden durch zwei Anführungszeichen dargestellt.

Wenn während der Dateneingabe **[SHIFT] [ENTER]** gedrückt wird, bleibt der Wert der Variablen unverändert und die Eingabe wird ignoriert. Der Programmablauf schreitet dann zum nächsten Dialog, zur nächsten Variablen, oder wenn das INPUT Statement abgeschlossen ist, zum nächsten Statement fort.

Wenn während einer Eingabe eine Fehlerbedingung auftritt, wird eine kommentierte Fehlermeldung angezeigt. Nach Drücken der **[ENTER]** Taste oder der **[CLR]** Taste, wird der Dialog erneut angezeigt und die Eingabe kann richtig wiederholt werden.

Die Eingabe von Daten wird wie folgt auf Gültigkeit überprüft:

- Wenn mehr als ein Wert zugleich eingegeben wurde, wird die Fehlermeldung `W1 at line number syntax` angezeigt und die Daten müssen erneut und einzeln eingegeben werden.
- Gibt man eine String-Konstante anstelle einer numerischen Variablen ein, wird die Fehlernachricht `W3 at line number mismatch` angezeigt und der Wert muß erneut eingegeben werden.
- Bei Eingabe einer Zahl größer als 9.9999999999999E+ 127 wird die Fehlernachricht `W25 at line number overflow` angezeigt und der Wert muß erneut eingegeben werden.
- Bei Eingabe einer Zahl kleiner als 1E-128 wird diese durch Null ersetzt, ohne daß eine Fehlermeldung erfolgt.

Bitte beachten Sie: Wenn das INPUT Statement eine Eingabe erwartet, löscht **[CLR]** nur das Eingabefeld, **[CTL]** ▲ (Ausgangspunkt) und **[CTL]** ◄ (Tabulatorrücklauf) bewegen den Kursor zum Anfang des Eingabefeldes und **[CTL]** ► (Rechts) ist wirkungslos.

Bei der Eingabe von Dezimalzahlen muß das Komma durch einen Punkt ausgedrückt werden.

INPUT MIT TASTATUR

Querverweis

ACCEPT, INPUT (mit Dateien), LINPUT

Beispiele

```
100 INPUT X
```

Läßt den Computer ein Dialog-Fragezeichen anzeigen, und wartet auf einen numerischen Eingabewert. Durch Drücken der **[ENTER]** Taste wird der Wert der Variablen zugeordnet.

```
100 INPUT X$,Y,"EINGABE Z ";Z(A)
```

Läßt den Computer das Dialog-Fragezeichen anzeigen und erwartet die Eingabe eines String Wertes für X$. Durch Drücken der **[ENTER]** Taste, wird der Wert eingegeben und X$ zugeordnet. Wiederum wird das Dialog-Fragezeichen angezeigt und der Computer erwartet die Eingabe eines numerischen Wertes für Y. Dann wird Eingabe Z angezeigt und der Computer erwartet die Eingabe eines numerischen Wertes für Z(A). Der Index wird ausgewertet, bevor der Wert dem Datenfeldelement zugeordnet wird.

MIT DATEIEN INPUT

Eingabeformat

INPUT # *Dateinummer* [,REC *numerischer Ausdruck*], *Variablen-Liste*

Beschreibung

Das INPUT # Statement dient dazu, Daten von Dateien zu lesen, die im INPUT- oder UPDATE-Modus eröffnet wurden. Jeder Variablen in der *Variablen-Liste* wird ein Wert aus der Datei zugeordnet.

Die Dateinummer kann einen Wert zwischen 0 und 255 annehmen und bezeichnet eine offene Datei oder ein Gerät. Die Dateinummer 0 bezeichnet die Tastatur und die Anzeige, und ist immer offen. (Siehe INPUT über Tastatur).

Die Variablen-Liste besteht aus einer durch Kommata getrennten Liste von Variablen. Die Variablen dürfen numerische oder String-Variablen sein und wahlweise indiziert sein. Die Datenwerte werden von der momentanen Datensatzposition aus den Variablen in der Liste der Reihe nach zugeordnet. Wenn der Datensatz an dieser Stelle nicht genügend Datenelemente enthält, wird ein weiterer Datensatz gelesen. Weitere Datensätze werden nacheinander gelesen, bis jeder Variablen ein Wert zugeordnet ist oder das Dateiende erreicht ist.

Der Computer interpretiert Daten unterschiedlich, je nachdem, ob er diese im DISPLAY oder INTERNAL-Format liest. Siehe "Verarbeitung von Dateien" Kapitel 4.

Daten im Display-Format haben die gleiche Form, wie jene Daten, die über die Tastatur eingegeben werden. Die Werte in jedem Datensatz werden durch Kommata voneinander getrennt. Führende und abschließende Leerstellen werden unterdrückt, wenn sie nicht als Teil eines Strings zwischen Anführungszeichen gesetzt sind.

Anführungszeichen, die Teil eines in Anführungszeichen gesetzten Strings sind, werden durch zwei Anführungszeichen wiedergegeben. Wenn das INPUT Statement auf zwei nacheinanderfolgende Kommata trifft, wird der entsprechenden Variablen ein Null-String als Wert zugeordnet. Jedes Datenelement wird daraufhin überprüft, daß numerische Werte nur numerischen Variablen und String-Werte nur Stringvariablen zugeordnet werden.

Daten im Internalformat sind binär kodiert, was dem Format während der internen Verarbeitung entspricht. Jedem Wert steht eine Angabe über seine Länge voran. Das INPUT Statement benutzt diese Längenangaben, um die einzelnen Werte zu separieren und um sie den Variablen zuzuordnen. Das INPUT Statement stellt zur Prüfung auf Gültigkeit nur fest ob numerische Daten zwischen 2 und 8 bytes lang sind.

Wenn ein INPUT Statement abgeschlossen wird, werden weitere Daten ignoriert, die im momentan gelesenen Datensatz möglicherweise noch enthalten sind. Das nächste INPUT Statement, welches zu dieser Datei Zugriff erlangt, liest Daten des nächsten Datensatzes. Falls jedoch die *Variablen-Liste* mit einem Komma endet, entsteht hieraus eine schwebende Input-Bedingung. Das bedeutet, daß nun die noch verbliebenen Elemente aus dem momentan ausgewerteten Datensatz gelesen werden. Das nächste INPUT Statement mit Zugriff zur Datei liest also den nächsten verfügbaren Wert.

Existiert eine schwebende Input-Bedingung wenn ein PRINT, RESTORE oder CLOSE Statement auf die Datei wirken, werden die schwebenden Daten gelöscht. Wird ein INPUT Statement erreicht, während eine schwebende Ausgabebedingung existiert, werden die Daten erst übertragen, bevor das INPUT Statement ausgeführt wird.

Die Anweisung REC *numerischer Ausdruck* benutzt man, wenn sich die *Dateinummer* auf eine Datei mit relativem Datensatzzugriff bezieht. Der *numerische Ausdruck* gibt den Datensatz an, der von der Datei gelesen werden soll. Der erste Datensatz einer Datei hat die Nummer Null. Schlagen Sie unter "Gebrauch von Peripherie-Geräten" in Kapitel 4 und im jeweiligen Peripherie-Geräte-Handbuch wegen weiterer Informationen über Dateien mit relativen Datensatzadressen und den Gebrauch der REC Anweisung nach.

Querverweis

CLOSE, INPUT, OPEN, PRINT, RESTORE

Beispiele

```
100 INPUT #1, X$
```
Ordnet den nächsten verfügbaren Wert in der mit # 1 bezeichneten Datei der Variable X$ zu.

```
250 INPUT #23,C,A,LL$
```
Ordnet die nächsten drei Werte aus der als # 23 bezeichneten Datei den Variablen X,A, und LL$ zu.

```
320 INPUT #3,A,B,C,
```
Ordnet die nächsten drei Werte aus der als # 3 bezeichneten Datei den Variablen A,B und C zu. Das Komma nach dem C erzeugt eine schwebende Inputbedingung.

INT

Die INT Funktion wandelt eine Zahl in einen ganzzahligen Wert um.

Eingabeformat

INT (*numerischer Ausdruck*)

Beschreibung

Mit der INT Funktion erhalten Sie den größten ganzzahligen Wert, der kleiner oder gleich dem **numerischen Ausdruck** ist.

Beispiele

250 P=INT(3.999999999)

Setzt P gleich 3.

470 DISPLAY AT(7),INT(4.0):PAUSE

Zeigt **4** in Spalte 8 an.

610 K=INT(-3.0000001)

Setzt K gleich -4.

Achtung Anmerkung: Die Intg-Funktion im Rechnermodus arbeitet für negative Werte unterschiedlich.

UNTERPROGRAMM IO

Das IO Unterprogramm ermöglicht die Durchführung bestimmter Operationen mit Peripheriegeräten, die nicht im BASIC des TI-74 eingebaut sind.

Eingabeformat

CALL IO *(Gerät, Befehl* [*,Status-Variable*]*)*

Beschreibung

Das IO Unterprogramm weist ein externes Gerät an, eine bestimmte Kontrolloperation durchzuführen, die im BASIC des TI-74 nicht verfügbar ist. Die mit einem externen Gerät verfügbaren Kontrolloperationen hängen von der Beschaffenheit des Gerätes ab.

Die richtige Anwendung dieses Unterprogramms erfordert Kenntnisse über die Struktur der Eingabe/Ausgabe Daten und über die besonderen Eigenschaften des Peripheriegerätes. Beispiele über die Anwendung des IO Unterprogrammes finden Sie in den Anleitungen zu den Peripheriegeräten.

Gerät erfordert die Angabe der dem Peripheriegerät zugeordneten Nummer, die zwischen 1 und 255 liegt.

Befehl ist ein numerischer Kode, der die vom Gerät auszuführende Operation bezeichnet.

Die **Status-Variable** ist eine numerische Variable, in der Informationen über das Ergebnis der Operation gespeichert ist. Wennn kein I/0 Fehler aufgetreten ist, beträgt der Wert der **Status-Variablen** Null. Ist ein Fehler aufgetreten, enthält die **Status-Variable** den entsprechenden Fehlerkode.

Die Angabe einer **Status-Variablen** beeinflußt die Antwort des Computers bei Auftreten eines I/0 Fehlers. Wird eine **Status-Variable** angegeben und ein Fehler tritt auf, wird keine Fehlermeldung angezeigt und die

Fehlerbedingung kann nicht mit ON ERROR verarbeitet werden. Wird die **Status-Variable** weggelassen und es tritt ein Fehler auf, wird eine Fehlermeldung angezeigt oder der Fehler kann mit ON ERROR verarbeitet werden.

Beispiel

140 CALL I0(1,1)

Schließt den Dialog mit Gerät 1 ab. (Der Befehlskode 1 ist eine CLOSE Operation.)

Querverweis

ON ERROR

KEY UNTERPROGRAMM

Eingabeformat

CALL KEY*(Rückgabe-Variable, Status-Variable)*

Beschreibung

Das KEY(=Taste) Unterprogramm ordnet der *Rückgabe-Variablen* den dezimalen ASCII-Kode für eine gedrückte Taste zu. Der *Rückgabe-Variablen* wird der Wert 255 zugeordnet, falls keine Taste gedrückt wurde. Eine Liste der ASCII-Kodes finden Sie in Anhang E.

Die *Status-Variable* hält den Wert fest, der den Status der gedrückten Taste wiedergibt. Ein Wert von +1 bedeutet, daß seit der letzten internen Tastaturabfrage (die auch bei KEY$, INPUT, ACCEPT etc... durchgeführt wird) eine andere Taste gedrückt wurde. Ein Wert von −1 bedeutet entsprechend, daß dieselbe Taste gedrückt wurde. Der Wert 0 besagt, daß keine Taste gedrückt wurde.

Beispiel

Der folgende Ausschnitt aus einem Programm prüft zum Beispiel in Zeile 120, wenn eine Taste gedrückt wurde, ob es sich um die Tasten **1, 2,** oder **3** handelt. Falls dies zutrifft, wird der Programmablauf entsprechend zur Zeile 1000, 2000 oder 3000 übertragen. Wenn innerhalb von ca. 5 Sekunden keine Taste, oder nicht die richtige Taste gedrückt wird, steuert das Programm die Zeile 10000 an.

```
90 PAUSE ALL
100 FOR I=1 TO 200
110 CALL KEY(K,S)
120 IF S THEN K=K-48:IF K<4 AND K>0 THEN 140
130 NEXT I:GOTO 10000
140 ON K GOTO 1000,2000,3000
1000 PRINT 1000
2000 PRINT 2000
3000 PRINT 3000
10000 PRINT 10000
```


KEY$

Eingabeformat

KEY$

Beschreibung

Das KEY$ Statement unterbricht den Programmablauf, bis eine Taste gedrückt wird. Sobald eine Taste gedrückt wurde, wird der Programmablauf fortgesetzt, und die KEY$ Funktion gibt einen aus einem Zeichen bestehenden String an, welcher der gedrückten Taste entspricht. Eine Liste der ASCII-Kodes finden Sie in Anhang E.

Wenn die [BRK] Taste gedrückt wird, während KEY$ auf Antwort wartet, erfolgt eine normale Programmunterbrechung.

Beispiel

Das folgende Programm wird fortgesetzt, wenn J gedrückt wird und unterbrochen, wenn N gedrückt wird.

```
100 PRINT "Druecke J fuer Weitermachen, N fuer Halt"
110 A$=KEY$
120 IF A$="J" OR A$="j" THEN 140
130 IF A$="N" OR A$="n" THEN 150 ELSE 110
140 PRINT "Weitermachen":PAUSE 1.5:GOTO 100
150 PRINT "HALT":PAUSE
```

LEN

Eingabeformat

LEN*(String-Ausdruck)*

Beschreibung

Die LEN Funktion gibt die Anzahl der Zeichen in einem *String-Ausdruck* an. Auch eine Leerstelle zählt als ein Zeichen.

Beispiele

```
170 PRINT LEN("ABCDE") : Pause
```
Zeigt 5 an.

```
230 X=LEN("DIES IST EIN SATZ")
```
Setzt X gleich 17.

```
910 DISPLAY LEN( "" ):PAUSE
```
Zeigt 0 an.

LET

Eingabeformat

[LET] { *numerische Variable [,numerische Variable ...] = numerischer Ausdruck* / *String-Variable [,String-Variable] = String-Ausdruck* }

Beschreibung

Mit dem LET Statement wird der Wert eines Ausdruckes einer (mehreren) spezifizierten Variablen zugeordnet. Der Computer berechnet den rechts stehenden Ausdruck und ordnet dessen Wert den links stehenden Variablen zu. Wenn mehr als eine Variable angegeben wird, müssen diese durch Kommata voneinander getrennt werden. LET kann wahlweise verwendet werden, und wird bei den Beispielen in diesem Anleitungsbuch weggelassen. Bevor die Zuordnungen erfolgen, werden alle Indices der Variablen ausgewertet.

Beispiel

```
110 LET T=4*3
```
Setzt T gleich 12.

```
170 X,Y,Z=12.4
```
Setzt X, Y und Z gleich 12.4

```
200 A=3<5
```
Setzt A gleich -1, da es 'wahr' ist, daß 3 kleiner als 5 ist.

```
350 L$, D$, B$="B"
```
Setzt L$, D$ und B$ gleich B.

LINPUT

Eingabeformat

LINPUT { *[Eingabe-Dialog;] String-Variable*
[#Dateinummer, [REC numerischer Ausdruck,]]
string-variable }

Beschreibung

Das LINPUT Statement ordnet einen vollständigen Eingabe-Datensatz, oder die restlichen Daten aus einer schwebenden Inputbedingung einer *String-Variablen* zu. Im Gegensatz zum INPUT Statement erfolgt durch LINPUT keine Aufbereitung der Eingabedaten. Dadurch werden alle Zeichen, einschließlich Kommata, führende und abschließende Leerstellen, Strichpunkte und Anführungszeichen in die *String-Variable* unverändert eingebracht.

Der *Eingabe-Dialog* ist ein String Ausdruck, auf den ein Strichpunkt folgen muß. Bei Verwendung einer String-Konstanten muß diese in Anführungszeichen gesetzt werden. Die Anzeige des *Eingabe-Dialogs* beginnt mit der momentanen Kursor-Position, so wie sich diese aus den vorangegangenen Eingabe/Ausgabe Statements ergeben hat. Wenn man den *Eingabe-Dialog* wegläßt, wird an seiner Stelle ein Fragezeichen, gefolgt von einer Leerstelle angezeigt.

Nach dem Dialog wird der blinkende Kursor angezeigt. Wenn sich die Kursor-Position als größer 31 ergibt, wird die Anzeige gelöscht und der Kursor in Spalte 1 gesetzt, bevor der Dialog angezeigt wird. Wenn der *Eingabe-Dialog* die Länge von 30 Zeichen überschreitet, werden nur die ersten 30 Zeichen angezeigt.

Man kann LINPUT auch benutzen, um Daten im Display-Format aus einer Datei oder einem Peripheriegerät zu lesen. Die *Dateinummer* muß die Nummer einer offenen Datei sein. Wenn für die angegebene Datei eine schwebende Input-Bedingung besteht, werden die restlichen Daten eines schwebenden Datensatzes gelesen. Die Fehlernachricht `Bad data` wird angezeigt, wenn der Datensatz oder der einzulesende Teil davon länger als 255 Zeichen ist.

LINPUT

Wahlweise kann die REC Anweisung in Verbindung mit Geräten benutzt werden, welche die Verwendung von Dateien zuläßt, die wahlfreie Datensätze (Direktzugriff) enthalten. Sehen Sie bitte in den einschlägigen Peripherie-Gerätebedienungsanleitung nach, um sich weiter über wahlfreie Datensätze zu informieren. Der numerische Ausdruck bezeichnet den Datensatz, zu dem der Zugriff erfolgt.

Querverweis

INPUT

Beispiele

```
300 LINPUT L$
```

Läßt den Computer das Dialogfragezeichen anzeigen, und speichert die eingegebenen Daten in L$.

```
470 LINPUT "NAME: ";NM$
```

Der Computer zeigt `NAME:` an und speichert die eingegebenen Daten in NM$.

LIST

Eingabeformat

LIST { *Zeilen-Gruppe* | *"Gerätename"* | *"Gerätename", Zeilen-Gruppe* }

Beschreibung

Mit dem LIST Befehl zeigt man Programmzeilen an. Bei fehlender Angabe der *Zeilen-Gruppe* wird das ganze Programm aufgelistet. Falls die Angabe der *Zeilen-Gruppe* erfolgte, werden nur die hierin genannten Zeilen aufgelistet. Die *Zeilen-Gruppe* kann folgende Zeilenbereiche spezifizieren: (Beachten Sie bitte die Wirkung des Bindestriches)

Zeilen-Gruppe	**Wirkung**
Einzelne Zeilennummer	Listet diese Zeile auf.
Zeilennummer-	Listet diese und alle folgenden Zeilen auf.
-Zeilennummer	Listet diese und alle vorhergehenden Zeilen auf.
Zeilennr.-Zeilennr.	Listet alle dazwischen liegenden Zeilen einschließlich der angegebenen auf.

Wenn ein *Gerätename* angegeben ist, werden die Zeilen auf dem genannten Gerät aufgelistet. Bei weggelassenem *Gerätenamen* erscheinen die Zeilen in der Anzeige. Während der Auflistung in der Anzeige, können die Zeilen editiert werden.

Um die Auflistung mit einem Gerät zu unterbrechen, drücken Sie eine beliebige Taste und halten diese gedrückt, bis die Auflistung anhält. Durch wiederholtes Drücken derselben Taste kann die Auflistung fortgesetzt werden. Jede Auflistung kann durch Drücken der **[BRK]** Taste abgeschlossen werden. Durch Drücken von **[▲]** schließt man die Auflistung mit der Anzeige ab.

LIST

Beispiele

```
LIST 100
```
Zeile 100 erscheint in der Anzeige

```
LIST 100-200
```
Listet alle Zeilen von 100 bis 200 in der Anzeige. Die nächste Zeile wird jeweils durch ENTER angezeigt.

```
LIST "12"
```
Listet das ganze Programm auf dem Peripherie-Gerät 12 (Drucker).

```
LIST "12", -200
```
Listet alle Zeilen von Programmbeginn bis 200 mit dem Peripherie-Gerät 12 auf.

LN

Eingabeformat

LN*(numerischer Ausdruck)*

Beschreibung

Die LN Funktion berechnet den natürlichen Logarithmus des *numerischen Ausdruckes*. Der *numerische Ausdruck* muß größer Null sein, da sonst die Fehlernachricht `E23 Bad argument` angezeigt wird. Die LN Funktion ist die Umkehrfunktion der EXP Funktion.

Querverweis

EXP

Beispiele

```
710 PRINT LN(3.4):PAUSE
```
Druckt den natürlichen Logarithmus von 3.4, welcher 1.223775432 beträgt.

```
850 X=LN(EXP(7.2)):PAUSE
```
Setzt X dem natürlichen Logarithmus von e hoch 7.2 gleich, wofür sich 7.2 ergibt (da sich die beiden inversen Funktionen gegenseitig aufheben).

```
910 S=LN(SQR(T))
```
Setzt S gleich dem natürlichen Logarithmus aus der Quadratwurzel von T.

LOG

Eingabeformat

LOG*(numerischer Ausdruck)*

Beschreibung

Die LOG Funktion berechnet den dekadischen Logarithmus des *numerischen Ausdruckes*. Der Wert des *numerischen Ausdruckes* muß größer Null sein, da sonst die Fehlernachricht `E23 Bad argument` angezeigt wird.

Beispiele

```
150 PRINT LOG(3.4):PAUSE
```
Druckt den dekadischen Logarithmus von 3.4, welcher .531478917 beträgt.

```
230 S=LOG(SQR(T))
```
Setzt S dem dekadischen Logarithmus der Quadratwurzel aus dem Wert T gleich.

NEW

Eingabeformat

NEW [ALL]

Beschreibung

Der NEW Befehl bereitet den Computer für ein neues Programm vor, indem das momentan im Speicher befindliche Programm und dessen Variablen gelöscht werden. Alle offenen Dateien werden abgeschlossen.

Der NEW ALL Befehl löscht das im Speicher enthaltene Programm mit seinen Variablen, weiterhin Anwender-definierte Strings, hebt die durch CALL ADDMEM herbeigeführte Erweiterung des Speicherbereiches auf, löscht alle Anzeigeindikatoren, wählt das Bogenmaß als Winkeleinheit und schließt alle offenen Dateien ab.

NEXT

Eingabeformat

NEXT *[Kontrollvariable]*

Beschreibung

Das NEXT Statement wird immer zusammen mit dem FOR TO STEP Statement verwendet, um eine Schleife zu erzeugen. Falls eine *Kontrollvariable* angegeben wird, muß diese der *Kontrollvariablen* des FOR TO STEP Statement gleich sein. Bei weggelassener *Kontrollvariable,* wird das NEXT Statement als zum letzten ungepaarten FOR TO STEP Statement gehörig interpretiert. Für das praktische Programmieren ist es nützlich, die *Kontrollvariable* anzugeben.

Wenn FOR TO STEP ... NEXT Statements verschachtelt werden, muß das NEXT Statement der inneren Schleife vor dem NEXT Statement der äußeren Schleife erscheinen.

Der Aufbau von Schleifen wird unter FOR TO STEP beschrieben.

Querverweis

FOR TO STEP

Beispiel

Das Programm beschreibt im Folgenden die Anwendung des NEXT Statements. Dabei werden die Werte 30, und −2 angezeigt.

```
100 SUMME=0
110 FOR ZAEHLVARIABLE=10 TO 0 STEP -2
120 SUMME=SUMME+ZAEHLVARIABLE
130 NEXT ZAEHLVARIABLE
140 PRINT SUMME;ZAEHLVARIABLE:PAUSE
```

NUMBER

Eingabeformat

NUMBER *[Anfangszeile] [,Inkrement]*

Beschreibung

Der NUMBER (oder NUM) Befehl erzeugt automatisch Programmzeilennummern mit festem Zeilenabstand. Diese Zeilennummern werden zum einfachen Eingeben der Programmzeile mit einer abschließenden Leerstelle angezeigt. Es ist also nur erforderlich, die Statements einzutippen. Sobald die **[ENTER]** Taste gedrückt wurde, wird die Zeile in den Speicher übertragen und die nächste Zeilennummer angezeigt.

Wenn die *Anfangszeile* und das *Inkrement* nicht angegeben wurden, beginnt die Numerierung der Zeilen bei 100 und wird aufsteigend in Zehnerschritten fortgesetzt. Anderfalls werden die Zeilen entsprechend der Angaben für *Anfangszeile* und *Inkrement* numeriert. Wenn zu einer Zeilennummer bereits eine Zeile existiert, wird diese Zeile angezeigt und kann mit den Editierungsfunktionen ersetzt oder geändert werden. Wird die Zeilennummer geändert, so wird die Folge der erzeugten Zeilennummern von der neuen Nummer an fortgesetzt.

Die Numerierung von Programmzeilen kann beendet werden, indem man mit **[ENTER]** eine Leerzeile eingibt, oder wenn man **[BREAK]** drückt, wenn eine Zeile angezeigt wird.

Querverweis

RENUMBER

Beispiele

`NUM 110`
Weist den Computer an, die Numerierung von Zeilen mit Zeile 110 zu beginnen und in Abständen von 10 fortzusetzen.

`NUM 105,5`
Weist den Computer an, die Numerierung mit Zeile 105 zu beginnen und in Abständen von 5 fortzusetzen.

NUMERIC

Eingabeformat

NUMERIC*(String-Ausdruck)*

Beschreibung

Die NUMERIC Funktion überprüft, ob ein *String-Ausdruck* die gültige Darstellung einer numerischen Konstanten ergibt. Mit NUMERIC erhält man den Wert -1, wenn der *String-Ausdruck* eine numerische Konstante gültig darstellt und den Wert 0, wenn der *String-Ausdruck* keine gültige Darstellung der numerischen Konstanten ist.

Führende und abschließende Leerstellen des *String-Ausdruckes* werden ignoriert. Man kann die NUMERIC Funktion benutzen um zu testen, ob die VAL Funktion einen String richtig verarbeiten würde, von dem man annimmt, daß er die gültige Darstellung einer Zahl ist.

Querverweis

VAL

Beispiel

Der folgende Programmteil bestimmt, ob eine Eingabe über die Tastatur die gültige Darstellung einer numerischen Konstanten ist. Wenn diese Voraussetzung nicht erfüllt ist, wird eine Fehlernachricht angezeigt, bis die Daten erneut eingegeben werden. Wenn die Daten eine numerische Konstante darstellen, werden sie der Variablen A zugeordnet.

```
100 LINPUT "EINGABE WERT: ";A$
110 IF NOT NUMERIC(A$) THEN LINPUT "FEHLER,
    EINGABE WIEDERHOLEN: "; A$:GOTO 110
120 A=VAL(A$)
```

OLD

Der OLD Befehl liest ein Programm von einem externen Speichergerät in den Speicher des Computers.

Eingabeformat

OLD *"Gerät.Dateiname"*

Beschreibung

OLD schließt alle offenen Dateien ab, löscht das momentan im Speicher befindliche Programm und liest ein neues Programm in den Speicher. Ein BASIC Programm kann mit SAVE auf **Gerät.Dateiname** gespeichert werden.

Gerät.Dateiname identifiziert das Gerät, in das das Programm gespeichert ist, und den Namen der Datei. **Gerät** bezeichnet die dem Gerät zugeordnete Nummer, die zwischen 1 und 255 liegen kann. Unter **Dateiname** versteht man die spezielle Datei. Angaben zum Gerätekode zur Form des **Dateinamens** finden Sie in den Anleitungen der jeweiligen Peripheriegeräte. Informationen zum Kassettenrekorder lesen Sie bitte in Kapitel 6 nach.

Anmerkung: Mit dem OLD Befehl können Sie keine Informationen aus einem Modul aufrufen. Auch eine Datendatei können Sie mit OLD nicht laden. Bezieht sich der **Dateiname** auf eine Datendatei, nicht auf eine Programmdatei, kann es notwendig sein, die [RESET] Taste zu drücken.

OLD

Beispiele

OLD "1.PROGRAM1"

Liest die Datei PROGRAM1 von Peripheriegerät 1 in den Speicher des Computers.

OLD "1."

Liest die nächste Datei von Peripheriegerät 1, dem Kassettenrekorder, in den Speicher des Computers. Ist die nächste Datei keine Programmdatei, erfolgt eine Fehlermeldung.

Querverweis

GET, INPUT (mit Dateien), PUT, SAVE, VERIFY

ON BREAK

Eingabeformat

ON BREAK { STOP / NEXT / ERROR }

Beschreibung

Das ON BREAK Statement bestimmt die Reaktion, die nach Auftreten einer Stoppstelle folgt. Nach der Ausführung des ON BREAK Statements werden Stoppstellen entsprechend der nachfolgend gewählten Anweisung verarbeitet.

ON BREAK STOP stellt die normale Funktion des BREAK Statements wieder her, die darin besteht, den Programmablauf zu unterbrechen und die zur STOP gehörige Standard-Nachricht anzuzeigen. Diese Möglichkeit wird beim Start eines Programmes automatisch gesetzt.

ON BREAK NEXT führt dazu, daß Stoppstellen ignoriert werden. Wenn die Anweisung für eine Stoppstelle unmittelbar der Angabe einer Zeilennummer vorausgeht, wird die Stoppstelle ignoriert und die Zeile ausgeführt. Auch das Drücken der **[BREAK]** Taste wird ignoriert. Jedoch wird durch ein BREAK Statement, dem keine Zeilenangabe folgt, das Programm unterbrochen, auch wenn das ON BREAK NEXT Statement wirksam ist. ON BREAK NEXT kann benutzt werden, um Stoppstellen zu ignorieren, die Sie zum Zweck der Fehlersuche in ein Programm eingesetzt haben. Bitte beachten Sie: Da die **[BREAK]** Taste ignoriert wird, muß die Resettaste gedrückt werden, um den Ablauf eines Programmes zu unterbrechen, bei dem kein normaler Programmhalt wirksam ist.

ON BREAK ERROR behandelt Stoppstellen so, als ob eine Fehlerbedingung aufgetreten wäre. Dies ermöglicht, die Stoppstelle mit dem ON ERROR Statement zu verarbeiten. Informationen hierzu finden Sie bei ON ERROR.

Das ON BREAK Statement bleibt wirksam, bis es durch eine anderes ON BREAK Statement geändert wird. Nach Abschluß eines Unterprogrammes ist derselbe ON BREAK Status wirksam, wie er vor Aufruf des Unterprogrammes vorlag.

KAPITEL 5
REFERENZTEIL
ON BREAK

Querverweis

BREAK, ON ERROR

Beispiel

Das nachstehende Programm verdeutlicht den Gebrauch von ON BREAK. Die Symbole] und [werden mit **[CTL]** 9 und **[CTL]** 8 eingegeben. Wenn die Nachricht `W29 Break` angezeigt wird, drücken Sie **[CLR]** und geben **[CONTINUE]** ein.

```
100 BREAK 140
```
Setzt eine Stoppstelle bei Zeile 140

```
110 ON BREAK NEXT
```
Stellt die Verarbeitung von Stoppstellen darauf ein, daß diese ignoriert werden. Es wird jeweils die nächste Zeile ausgeführt, wenn eine Stoppstelle auftritt.

```
120 BREAK
```
Trotz des Inhaltes von Zeile 110 tritt in Zeile 120 eine Stoppstelle auf. Drücken Sie **[CLR]** und **[CONTINUE].**

```
130 FOR A=1 TO 500
140 PRINT " [BRK] IST UNWIRKSAM"
150 NEXT A
```
Während der Ausführung der Zeilen 130 bis 150 ist die **[BREAK]** Taste unwirksam.

```
160 ON BREAK STOP
```
Stellt die normale Funktion der **[BREAK]** Taste wieder her.

```
170 FOR A=1 TO 500
180 PRINT "NUN IST [BRK] WIEDER WIRKSAM"
190 NEXT A
```
Während der Ausführung der Zeilen 170 bis 190 ist die **[BREAK]** Taste wieder wirksam.

ON ERROR

Eingabeformat

ON ERROR {STOP / *Zeilennummer*}

Beschreibung

Das ON ERROR Statement bestimmt die Reaktion, die nach dem Auftreten einer Fehlerbedingung während des Programmablaufes folgt. Nach der Ausführung des ON ERROR Statements wird jeder auftretende Fehler der gewählten Option entsprechend verarbeitet.

ON ERROR STOP nimmt die normale Verarbeitung von Fehlern wieder auf, die darin besteht, den Programmablauf zu unterbrechen, und eine Fehlernachricht anzuzeigen. Diese Möglichkeit wird beim Start eines Programmes automatisch gesetzt.

ON ERROR *Zeilennummer* überträgt die Steuerung des Programmablaufes zur angegebenen Zeile, wenn ein Fehler auftritt. Bei dieser *Zeilennummer* muß ein Teilprogramm zur Fehlerverarbeitung beginnen. Wenn nach dem Auftreten eines Fehlers die Steuerung des Programmablaufes zu einer Zeilennummer übertragen wurde, wird die Verarbeitung von (weiteren) Fehlern wieder auf ON ERROR STOP umgestellt. Wenn die ON BREAK ERROR Möglichkeit gewählt wurde, so wird diese durch das Auftreten des Fehlers in ON BREAK NEXT geändert. um weitere Fehler mit einem Fehlerverarbeitungsprogramm zu behandeln, muß jeweils eine weitere ON ERROR Zeilennummer-Folge ausgeführt werden.

Das ON ERROR Statement bleibt wirksam, bis es durch ein anderes ON ERROR Statement geändert wird. Nach Abschluß eines Unterprogrammes, während dessen Ablauf kein Fehler auftrat, ist derselbe ON ERROR Status wirksam, der vor Aufruf des Unterprogrammes bestand. Wenn in einem Unterprogramm ein Fehler auftrat, bleibt nach Abschluß des Unterprogrammes der Fehlerverarbeitungsstatus bestehen, der mit der Verarbeitung dieses Fehlers hervorgerufen wurde.

Das Hauptprogramm und die Unterprogramme können auf dasselbe Fehlerverarbeitungsprogramm zurückgreifen. Teilprogramme, die mit GOSUB aufgerufen werden, können nicht derart benutzt werden.

ON ERROR

Querverweis

ON BREAK, ON WARNING, RETURN (mit ON ERROR)

Beispiel

Das nachstehende Programm verdeutlicht den Gebrauch von ON ERROR.

```
100 ON ERROR 150
```
Durch einen Fehler wird die Steuerung des Programmablaufes zu Zeile 150 verzweigt.

```
110 X$="A"
120 X=VAL(X$)
```
verursacht einen Fehler.

```
130 PRINT X; "ERGIBT QUADRIERT";X * X:PAUSE 2
140 STOP
150 REM FEHLERTEILPROGRAMM
160 ON ERROR 220
```
Damit wird, wenn ein weiterer Fehler auftritt, die Steuerung des Programmablaufes zu Zeile 220 verzweigt.

```
170 CALL ERR(KODE,TYP,DATEI,ZEILE)
```
Ermittelt den Fehler durch CALL ERR.

```
180 IF ZEILE <> 120 THEN RETURN 220
```
Verzweigt die Steuerung des Programmablaufes zu Zeile 220, wenn der Fehler in einer anderen Zeile als erwartet auftritt.

```
190 IF KODE <> 29 THEN RETURN 220
```
Verzweigt die Steuerung des Programmablaufes zu Zeile 220, wenn nicht der erwartete Fehler auftritt.

```
200 X$="5"
```
Ändert den Wert von X$ in einen zulässigen Wert um.

```
210 RETURN
```
Überträgt die Steuerung des Programmablaufes zu der Zeile, in welcher der Fehler auftrat und wiederholt das Statement.

```
220 REM UNBEKANNTER FEHLER
230 PRINT "FEHLER";KODE;"IN ZEILE";ZEILE:PAUSE
```
Gibt die Art des unerwarteten Fehlers an und unterbricht den Programmablauf.

ON GOSUB

Eingabeformat

ON *numerischer Ausdruck* GOSUB *Zeilennummer 1 [,Zeilennummer 2 ...]*

Beschreibung

Das ON GOSUB Statement bestimmt, indem es den *numerischen Ausdruck* auswertet, welches Teilprogramm ausgeführt wird. Wenn der Wert des *numerischen Ausdruckes* gleich 1 ist, wird das bei der *Zeilennummer 1* beginnende Teilprogramm ausgeführt, ist der *numerische Ausdruck* gleich 2, wird das bei der *Zeilennummer 2* beginnende Teilprogramm ausgeführt u.s.w.. Jede dieser Zeilennummern muß das erste Statement eines Teilprogrammes enthalten. Wenn der *numerische Ausdruck* 0, negativ oder größer ist, als die Zahl der in der Liste enthaltenen Zeilennummern, wird die Fehlernachricht `E11 Line number error` angezeigt. Wenn der numerische Ausdruck keine ganze Zahl ergibt, wird er gerundet.

Nach Ausführung des das Teilprogramm abschließenden RETURN Statement wird die Steuerung des Programmablaufes zu dem auf ON GOSUB folgenden Statement übertragen. ON GOSUB kann nicht dazu verwendet werden, die Steuerung des Programmablaufes in ein Unterprogramm hinein oder daraus heraus zu verzweigen.

Querverweis

GOSUB, RETURN (mit GOSUB)

Beispiele

```
140 ON X GOSUB 1000, 2000, 300
```

Überträgt die Steuerung des Programmablaufes zu Zeile 1000, wenn X gleich 1 ist, zu Zeile 2000, wenn X gleich 2 ist und zu Zeile 300, wenn X gleich 3 ist.

```
240 ON P-4 GOSUB 200,250,300,800,170
```

Überträgt die Steuerung des Programmablaufes zu Zeile 200 wenn P-4 gleich 1 ist (P=5), zu Zeile 250 wenn P-4 gleich 2 ist, zu Zeile 300 wenn P-4 gleich 3 ist, zu Zeile 800 wenn P gleich 8 ist und zu Zeile 170 wenn P-4 gleich 5 ist.

ON GOTO

Eingabeformat

ON *numerischer Ausdruck* GOTO *Zeilennummer 1 [,Zeilennummer 2 ...]*

Beschreibung

Das ON GOTO Statement bestimmt, indem es den *numerischen Ausdruck* auswertet, wohin die Steuerung des Programmablaufes verzweigt wird. Wenn der Wert des *numerischen Ausdruckes* gleich 1 ist, wird die Steuerung des Programmablaufes zu *Zeilennummer 1* übertragen, ist der Wert des numerischen Ausdruckes gleich 2, wird die Steuerung des Programmablaufes zu *Zeilennummer 2* übertragen usw.. Wenn der *numerische Ausdruck* 0, negativ oder größer ist, als die Zahl der in der Liste enthaltenen Zeilennummern, wird die Fehlernachricht `E11 Line number error` angezeigt. Wenn der *numerische Ausdruck* keine ganze Zahl ergibt, wird er gerundet.

ON GOTO kann nicht dazu verwendet werden, die Steuerung des Programmablaufes in ein Unterprogramm hinein oder daraus heraus zu verzweigen.

Querverweis

GOTO

Beispiele

```
130 ON X GOTO 1000,2000,300
```

Überträgt die Steuerung des Programmablaufes zu Zeile 1000, wenn X gleich 1 ist, zu Zeile 2000, wenn X gleich 2 ist und zu Zeile 300, wenn X gleich 3 ist. Diesem Statement wäre folgendes IF THEN ELSE Statement äquivalent: `130 IF X = 1 THEN 1000 ELSE IF X = 2 THEN 2000 ELSE IF X = 3 THEN 300 ELSE PRINT "FALSCHER WERT":PAUSE:STOP`, das allerdings mehr als 80 Zeichen lang ist.

```
210 ON P-4 GOTO 200,250,300,800,170
```

Überträgt die Steuerung des Programmablaufes zu Zeile 200 wenn P-4 gleich 1 ist (P=5), zu Zeile 250 wenn P-4 gleich 2 ist, zu Zeile 300 wenn P-4 gleich 3 ist, zu Zeile 800 wenn P-4 gleich 4 ist und zu Zeile 170, wenn P-4 gleich 5 ist.

ON WARNING

Eingabeformat

ON WARNING { PRINT / NEXT / ERROR }

Beschreibung

Das ON WARNING Statement bestimmt die Reaktion, die nach dem Auftreten einer Warnung während des Programmablaufes folgt. Nach der Ausführung des ON WARNING Statements wird jede auftretende Warnung der gewählten Option entsprechend verarbeitet.

ON WARNING PRINT nimmt die normale Verarbeitung von Warnungen wieder auf, die darin besteht, eine kommentierende Fehlernachricht anzuzeigen und den Programmablauf fortzusetzen, nachdem die **[ENTER]** Taste oder die **[CLR]** Taste gedrückt wurde. Diese Möglichkeit wird beim Start eines Programmes automatisch gesetzt.

ON WARNING NEXT setzt den Programmablauf fort, ohne eine Nachricht zu drucken.

ON WARNING ERROR behandelt Warnungen so, als ob eine Fehlerbedingung aufgetreten wäre. Dies ermöglicht es, die Warnung mit dem ON ERROR Statement zu verarbeiten.

Das ON WARNING Statement bleibt wirksam, bis es durch ein anderes ON WARNING Statement geändert wird. Nach Abschluß eines Unterprogrammes ist derselbe ON WARNING Status wirksam, wie er vor Aufruf des Unterprogrammes vorlag.

Querverweis

ON ERROR

KAPITEL 5
REFERENZTEIL

ON WARNING

Beispiel

Das nachstehende Programm verdeutlicht den Gebrauch von ON WARNING.

```
100 ON WARNING NEXT
```
Eine Warnung unterbleibt und das nächste Statement wird ausgeführt.

```
110 PRINT 110,5/0:PAUSE
```
Druckt das Ergebnis ohne eine Fehlermeldung.

```
120 ON WARNING PRINT
```
Warnungen werden wieder normal verarbeitet, indem eine Fehlermeldung angezeigt und die Ausführung fortgesetzt wird.

```
130 PRINT 130, 5/0:PAUSE
```
Zeigt eine Warnung an. Nach Drücken von **[ENTER]** oder **[CLR]** wird 130 angezeigt, anschließend der genäherte Wert für 5/0.

```
140 ON WARNING ERROR
```
Warnungen werden ab jetzt wie Fehler verarbeitet.

```
150 PRINT 150; 5/0:PAUSE
```
Druckt einen Warnhinweis und verarbeitet die Warnung als Fehler.

```
160 PRINT 150:PAUSE
```
Wird nicht ausgeführt, weil die Verarbeitung nach Zeile 150 unterbrochen wurde.

OPEN

Das OPEN Statement stellt eine Kommunikation mit einem Peripheriegerät her mit dem Ziel, Daten zu übertragen.

Eingabeformat

OPEN #*"Dateinummer,"Gerät,Dateiname"*[,*Dateiorganisation*] [,*Dateityp*] [,*Satzlänge*] [,*Eröffnungs-Modus*]

Beschreibung

Das OPEN Statement ermöglicht einem BASIC Programm, Datendateien und Peripheriegeräte zu verwenden, indem es eine Verbindung zwischen der **Dateinummer** und einer Datei oder einem Gerät herstellt. In dieser Verbindung bestimmt das OPEN Statement exakt, in welcher Weise eine Datei bzw. ein Gerät verwendet werden kann (zur Ein- oder Ausgabe) und wie die Datei organisiert wird.

DAS OPEN Statement muß vor jedem anderen BASIC Statement ausgeführt werden, das Zugriff zu der mit **Dateinummer** spezifizierten Datei bzw. Gerät verlangt. Wenn sich ein OPEN Statement auf eine bereits bestehende Datei bezieht, müssen die im OPEN Statement verwendeten Attribute für **Dateiorganisation**, **Dateityp** und **Satzlänge** mit denen der Datei übereinstimmen. Bezieht sich ein OPEN Statement auf eine bereits eröffnete Datei, erfolgt eine Fehlermeldung.

Die **Dateinummer** ist eine Zahl zwischen 1 und 255, die das OPEN Statement einer Datei oder einem Gerät zuordnet. Diese **Dateinummer** wird von allen Eingabe/Ausgabe Statements verwendet, die Zugriff zur Datei oder zum Gerät haben. Die **Dateinummer** wird auf die nächste ganze Zahl gerundet. Die Dateinummer 0 ist der Tastatur und der Anzeige des Computers zugeordnet. Sie kann nicht für andere Dateien verwendet werden und ist immer offen.

Die Angabe **Gerät.Dateiname** bestimmt ein Peripheriegerät und enthält andere gerätespezifische Informationen. **Gerät.Dateiname** kann auch ein String-Ausdruck sein. **Gerät** ist die Zahl, die dem physischen Gerät zugeordnet ist und kann zwischen 1 und 255 liegen. **Dateiname** enthält Informationen für das Peripheriegerät zur Auswertung des OPEN Statements. Zum Beispiel spezifiziert **Dateiname** mit einem externen Speichergerät den Namen einer Datei. Mit anderen Geräten gibt **Dateiname** Zusatzoptionen wie Parität, Datenübertragungsrate etc.

Weitere Informationen über den Gerätekode jedes Peripheriegerätes und über die Form des **Dateinamens** finden Sie in den jeweiligen Handbüchern der Peripheriegeräte.

Dateiattribute

Die nachfolgend aufgeführten Dateiattribute können beliebig eingefügt oder weggelassen werden. Wird ein Attribut weggelassen, verwendet der Computer Standardattribute.

Die **Dateiorganisation** kann entweder sequentiell oder relativ (wahlfrei) sein. Datensätze in einer sequentiellen Datei werden nacheinander von Anfang bis Ende gelesen. Datensätze in einer RELATIVE Datei (mit wahlfreiem Zugriff) können in beliebiger Reihenfolge, auch sequentiell, gelesen oder geschrieben werden. Wird keine **Dateiorganisation** angegeben, nimmt der Computer sequentiell als Standard; lassen Sie daher, wenn Sie sequentielle Dateien wollen, die **Dateiorganisation** weg und spezifizieren Sie RELATIVE, wenn Sie wahlfreie Dateien wollen.

Dateityp kann entweder DISPLAY oder INTERNAL sein. DISPLAY bedeutet, daß die Daten im ASCII Format geschrieben sind, INTERNAL speichert die Daten im Binärformat. Binärsätze benötigen weniger Speicherplatz, werden vom Computer schneller verarbeitet und lassen eine effizientere Aufzeichnung der Daten auf externen Speichergeräten zu. Sollen die Daten jedoch gedruckt und angezeigt werden, sollten Sie das DISPLAY Format verwenden. Wenn Sie keinen **Dateityp** angeben, wird DISPLAY angenommen.

Die **Satzlänge** besteht aus dem Wort VARIABLE und einem nachfolgenden numerischen Ausdruck, der die maximale Satzlänge für die Datei spezifiziert. Die maximale Satzlänge ist abhängig vom benutzten Gerät. Wird die **Satzlänge** weggelassen, wählt das Peripheriegerät eine Standard-Satzlänge.

Der **Eröffnungs-Modus** weist den Computer an, eine Datei im UPDATE, INPUT, OUTPUT oder APPEND Modus zu verarbeiten. UPDATE bedeu-

tet, daß die Daten gelesen und auch geschrieben werden können. INPUT heißt, daß Daten nur aus der Datei gelesen werden können. OUTPUT bedeutet, daß Daten nur in die Datei geschrieben werden können.

APPEND heißt, daß Daten nur ans Ende der Datei angefügt werden können. Wird der **Eröffnungs-Modus** weggelassen, nimmt der Computer UPDATE als Standard an.

Beachten Sie, daß in einem externen Speichergerät in einer bereits existierenden Datei die schon vorhandenen Daten bei Wahl des OUTPUT-Modus von den neuen überschrieben werden.

Eröffnung von Dateien bei einem Kassettenrekorder

Das Kassetten-Interface wird als Gerät 1 in einem OPEN Statement bezeichnet. Die Attribute für eine Datei auf einem Kassettenrekorder sind nachfolgend aufgeführt.

- Die Dateien müssen sequentiell sein.
- Die Standard-Satzlänge beträgt 256 Bytes.
- Die Dateien müssen als **Eröffnungs-Modus** INPUT oder OUTPUT haben.
- Der **Dateityp** der Dateien muß DISPLAY sein.

Beachten Sie, daß Sie bei Dateien auf einem Kassettenrekorder keine RESTORE oder DELETE Befehle geben können, den **Eröffnungs-Modus UPDATE oder APPEND nicht verwenden können und der Dateityp** INTERNAL nicht erlaubt ist.

Hinweise zur Verwendung eines Kassettenrekorders für Speicherung und Aufruf von Dateien finden Sie in Kapitel 6.

Beispiele

100 OPEN #23,"2.X",INTERNAL,UPDATE

Eröffnet die Datei "X" im Peripheriegerät 2 und ermöglicht allen Eingabe/Ausgabe Statements den Zugriff zu der Datei unter Verwendung der Nummer 23. Der Dateityp ist INTERNAL. Da die Datei im UPDATE Modus eröffnet wurde, können Daten aus der Datei gelesen und auch in sie geschrieben werden.

150 OPEN #243,A$&".ABC",INTERNAL

Wenn A$ gleich "2" ist, wird in Gerät 2 eine Datei mit dem Namen ABC eröffnet. Der Dateityp ist INTERNAL, der UPDATE Modus wird als Standard gesetzt und eine Standard-Satzlänge vom Gerät vorgegeben.

200 OPEN #1, "1.DATA1",DISPLAY,OUTPUT

Eröffnet die Datei mit Namen "DATA1" auf einem Kassettenrekorder. Der Dateityp ist DISPLAY. Da die Datei im OUTPUT Modus eröffnet wurde, können Daten nur in die Datei geschrieben werden.

Querverweis

CLOSE, DELETE, INPUT, LINPUT, PRINT, RESTORE

PAUSE

Eingabeformat

PAUSE { *[numerischer Ausdruck]* / [ALL] }

Beschreibung

Das PAUSE Statement unterbricht den Programmablauf entweder für eine bestimmte Anzahl von Sekunden oder bis die **[CLR]** oder die **[ENTER]** Taste gedrückt wird. Wenn kein *numerischer Ausdruck* angegeben wurde, erscheint der Unterstreichungskursor in Spalte 1 um anzuzeigen, daß eine Pause von unbestimmter Dauer stattfindet. Währenddessen können die Tasten, mit denen der Kursor gesteuert wird, benutzt werden, um den Inhalt der 80 Zeichen umfassenden Zeile zu überprüfen. Der Programmablauf wird fortgesetzt, wenn die **[ENTER]** Taste oder die **[CLR]** Taste gedrückt werden.

Wenn der *numerische Ausdruck* angegeben wurde, findet eine Pause statt, die den Programmablauf für die Anzahl von Sekunden unterbricht, die dem Absolutwert des *numerischen Ausdrucks* entspricht. Wenn der *numerische Ausdruck* positiv ist, kann die vorgegebene Pausendauer durch Drücken der **[ENTER]** Taste oder der **[CLR]** Taste sofort beendet werden. Bei einem negativen Wert ist ein Umgehen der Pausendauer nicht möglich. Die effektive Auflösung für die Zeitangabe beträgt ungefähr 1/10 sec.. Wenn der Wert des *numerischen Ausdruckes* kleiner als 0.1 ist, findet keine Pause statt. Während einer Pause von vorgegebener Dauer wird der Kursor nicht angezeigt und der Anzeigeinhalt kann nicht beeinflußt werden.

Das PAUSE ALL Statement unterbricht den Programmablauf jedesmal, wenn nach seiner Ausführung der Anzeigeinhalt durch ein PRINT oder DISPLAY Statement verändert wird. Der Programmablauf wird fortgesetzt, wenn die **[CLR]** oder die **[ENTER]** Taste gedrückt wird. PAUSE ALL bleibt solange wirksam, bis eine PAUSE mit der vorgegebenen Länge null stattfindet.

PAUSE ALL bleibt wirksam, wenn ein Unterprogramm aufgerufen wird. Wenn PAUSE ALL durch ein Unterprogramm geändert wird, wird der vorherige Zustand nach Abschluß des Unterprogrammes wiederhergestellt.

PAUSE

Querverweis

DISPLAY, PRINT

Beispiele

```
120 PAUSE 2.2
```

Unterbricht den Progammablauf für 2,2 Sekunden bzw. bis die **[CLR]** Taste oder die **[ENTER]** Taste gedrückt wurde.

```
190 PAUSE
```

Unterbricht den Programmablauf bis zum Drücken der **[CLR]** Taste oder der **[ENTER]** Taste.

Mit dem nachstehenden Programm werden Grad Fahrenheit in Grad Celsius umgerechnet.

```
100 PRINT "EINGABE GRAD-FAHRENHEIT:";
```

Druckt den Dialog EINGABE GRAD-FAHRENHEIT:. Die schwebende Eingabebedingung, die durch den Strichpunkt nach dem PRINT Statement entsteht, bewirkt, daß der Dialog angezeigt wird, bis eine Eingabe erfolgt.

```
110 INPUT DG
120 PRINT DG;"GRADF.=";(DG-32)*5/9;"GRAD-CELSIUS":
    PAUSE
```

Zeigt das Ergebnis an. Durch das PAUSE Statement, das auf das PRINT Statement folgt, bleibt die Antwort angezeigt, bis die **[CLR]**– oder **[ENTER]** Taste gedrückt wird.

```
130 GOTO 100
```

PI

Eingabeformat

PI

Beschreibung

Die PI Funktion gibt den Wert der Kreiszahl π zu 3.14159265359 an.

Beispiel

```
130 VOLUMEN = 4/3 * PI * R ^ 3
```

Setzt die Variable VOLUMEN gleich dem Produkt aus vier Drittel π mal der dritten Potenz des Radius. Damit ist das Volumen einer Kugel mit Radius R gegeben.

POS

Eingabeformat

POS*(String1,String2,numerischer Ausdruck)*

Beschreibung

Die POS Funktion ermittelt die Stelle, an der *String 2* zum ersten Mal in *String 1* erscheint. Die Suche beginnt von der Stelle an, die mit dem *numerischen Ausdruck* spezifiziert wird. Wenn keine Übereinstimmung festgestellt wird, ist das Ergebnis gleich Null.

Beispiele

```
110 X=POS("PAN","A",1)
```

Setzt X gleich 2, weil A der zweite Buchstabe des Wertes PAN ist.

```
140 Y=POS("APAN","A",2)
```

Setzt Y gleich 3, weil A erstmals an Stelle 3, in dem Teil des Wortes APAN, der untersucht wurde, auftritt.

```
170 Z=POS("PAN","A",3)
```

Setzt Z gleich 0, da A in dem Teil von PAN, der untersucht wurde, nicht gefunden wurde.

```
290 R=POS("PABNAN","AN",1)
```

Setzt R gleich 5, da die Übereinstimmung zwischen AN und PABNAN erstmals an der Stelle 5 mit dem A beginnend auftritt.

Eingabeformat

PRINT [USING *Zeilennummer,* / *String-Ausdruck,*] *[Druck-Liste]*

Beschreibung

Das PRINT Statement verwendet man, um Daten in die Anzeige zu schreiben und diese zu formatieren. Man benutzt USING, um das Format für die Elemente der *Druck-Liste* zu spezifizieren. Eine Beschreibung der Definition von Formaten und deren Wirkung mit dem PRINT Statement finden Sie bei IMAGE und USING. Wenn die *Druck-Liste* weggelassen wird, löscht das PRINT Statement die Anzeige.

Die *Druck-Liste* besteht aus Druckelementen und Druckseparatoren (=Trennzeichen). Druckelemente sind numerische Ausdrücke und String-Ausdrücke, die angezeigt werden und TAB Funktionen, welche die Position bezüglich der Spalte in der Anzeige bestimmen. Druckseparatoren sind Kommata oder Strichpunkte, welche die Position der Druckelemente in der Anzeige bestimmen.

Druckelemente

Während der Ausführung eines PRINT Statements werden die Werte der Ausdrücke in der *Druck-Liste,* der Reihe nach von links nach rechts in den Positionen angezeigt, welche mittels der Druckseparatoren und der TAB Funktion festgelegt werden.

- Die Auswertung von *String-Ausdrücken* ergibt wieder einen String. String-Konstanten müssen zwischen Anführungszeichen gesetzt werden. Vor und nach String-Ausdrücken werden keine Leerstellen gesetzt. Falls Sie eine Leerstelle vor oder nach einem String setzen wollen, beziehen Sie diese in den String mit ein oder setzen diese separat in Anführungszeichen.
- *Numerische Ausdrücke* werden ausgewertet und mit einer nachfolgenden Leerstelle angezeigt. Positive Werte werden mit einer führenden Leerstelle (anstatt eines Pluszeichens) gedruckt und negative Zahlen erscheinen mit einem voranstehenden Minuszeichen.
- Die TAB Funktion bestimmt für das nächste Element aus der *Druck-Liste* die Anfangsposition in der Anzeige. Weitere Informationen finden Sie unter TAB.

Druckseparatoren

Zwischen zwei aufeinanderfolgende Druckelemente müssen Sie wenigstens einen Druckseparator setzen. Mehrere aufeinanderfolgende Druckseparatoren in einem PRINT Statement werden von links nach rechts ausgewertet.

- Der Strichpunkt druckt das nächste Element aus der *Druck-Liste* sofort nach dem vorangegangenen Element, ohne zwischen den Werten Leerstellen zu lassen.
- Das Komma druckt das nächste Druckelement am Anfang des nächsten Anzeigefeldes. Ein solches Anzeigefeld ist 15 Zeichen lang und beginnt jeweils in den Spalten 1,16,31,46,61 und 76 bei einer 80spaltigen Zeile. Wenn die momentane Kursorposition den Anfang des letzten Anzeigefeldes schon überschritten hat, wird nach einem Komma das nächste Anzeigeelement in der nächsten Zeile beginnend angezeigt.

Wenn ein Anzeigeelement länger ist, als der Rest der jetzigen Zeile, so wird es am Anfang der nächsten Zeile beginnend ausgedruckt. Wenn ein numerisches Anzeigeelement ohne seine abschließende Leerstelle in den Rest der jetzigen Zeile paßt, so wird es noch in dieser Zeile angezeigt. Wenn ein Anzeigeelement länger als 80 Zeichen ist, so werden die ersten 80 Zeichen in einer Zeile angezeigt und die restlichen Zeichen erscheinen zu je 80 Zeichen pro Zeile in den folgenden Zeilen.

Schwebende PRINT-Bedingungen

Wenn der *Druck-Liste* kein Komma oder Strichpunkt folgt, wird der Rest der 80 Zeichen langen Zeile gelöscht. Damit muß das nächste Eingabe/Ausgabe Statement eine neue Zeile beginnen.
Die Verwendung eines Kommas oder eines Strichpunktes am Ende der *Druck-Liste* erzeugt eine schwebende PRINT-Bedingung. Daraufhin wird der Rest der Zeile nicht gelöscht. Stattdessen schreitet der Computer nur zum Anfang des nächsten Druckfeldes fort, wenn ein Komma folgte, und läßt bei einem Strichpunkt nach dem vorangegangenen Statement keine Leerstelle frei. Die Anzeige oder Eingabe von Informationen durch das nächste E/A Statement beginnt bei der momentanen Spaltenposition, wenn diese nicht von dem Statement selbst geändert wird.

Eine schwebende PRINT-Bedingung kann benutzt werden, um einen Eingabe-Dialog für ein ACCEPT oder INPUT (mit der Anzeige) Statement zu erzeugen. Das nächste INPUT Statement schließt dann seinen Dialog an die schwebende PRINT-Bedingung an. Weitere Informationen finden Sie bei ACCEPT und bei INPUT (mit der Anzeige).

Formatierung von Zahlen

Zahlen werden entweder normal als Dezimalzahlen oder in wissenschaftlicher Schreibweise angezeigt. Die wissenschaftliche Schreibweise erscheint, wenn mehr als zehn signifikante Ziffern auftreten.

Für die Anzeige einer Zahl im normalen Dezimalzahlenformat gelten folgende Regeln:

- Ganze Zahlen werden ohne Dezimalpunkt angezeigt.
- Gebrochene Zahlen (Dezimalbrüche) werden mit Dezimalpunkt angezeigt. Abschließende Nullen bei den Nachkommastellen werden weggelassen. Wenn die Zahl aus mehr als zehn signifikanten Ziffern besteht, wird der Nachkommateil auf eine insgesamt zehnstellige Zahl gerundet.
- Eine Zahl, deren Absolutwert kleiner als 1 ist, wird ohne Null links vom Dezimalpunkt dargestellt.

Zahlen in wissenschaftlicher Schreibweise erscheinen wie folgt:

Mantisse E Exponent

Für die Anzeige einer Zahl in wissenschaftlicher Schreibweise gelten folgende Regeln:

- Die Mantisse wird mit bis zu sieben Ziffern angezeigt, wovon die erste immer links vom Dezimalpunkt steht.
- Abschließende Nullen im Nachkommateil der Mantisse werden weggelassen.
- Der Exponent wird mit E, dem vorangehenden Plus- oder Minuszeichen und zwei oder dreistelligen Exponenten angezeigt.
- Bei zweistelligen Exponenten ist die Mantisse auf sieben Ziffern begrenzt. Bei dreistelligem Exponenten ist die Mantisse auf sechs Ziffern begrenzt. Die Mantisse wird, falls es erforderlich ist, auf die geeignete Zahl von Stellen gerundet.

Querverweis

ACCEPT, DISPLAY, IMAGE, INPUT, PAUSE, TAB, USING

Beispiele

```
100 PRINT
```
Zeigt eine leere Zeile.

```
210 PRINT "DAS ERGEBNIS IST"; ERGEBNIS: PAUSE
```
Zeigt DAS ERGEBNIS IST und den Wert für das Ergebnis sofort danach.

```
320 PRINT X, Y/2:PAUSE
```
Zeigt den Wert von X und im nächsten Druckfeld den Wert von Y/2.

```
7000 PRINT "Firma: Fa";
7010 ACCEPT AT(8)SIZE(-20)  FA$
```
Zeigt Firma: Fa an und gibt Fa vor.
D.h. es wird mittels einer schwebenden PRINT Bedingung ein Dialog in die Anzeige gebracht und mit der AT Klausel bestimmt, daß das Eingabefeld bereits bei Position 8 (bei Fa) beginnt.

Eingabeformat

PRINT # *Dateinummer* [,REC *numerischer Ausdruck]*

[,USING *Zeilennummer* / *String-Ausdruck* *]* *[,Druck-Liste]*

Beschreibung

Das PRINT Statement verwendet man, um Daten in Dateien oder zu einem Peripherie-Gerät zu übertragen. Die *Dateinummer* bezeichnet eine offene Datei oder ein Gerät und muß zwischen 0 und 255 betragen. Die Datei muß im OUTPUT-,UPDATE- oder APPEND-Modus eröffnet sein. *Dateinummer* 0 ist die Anzeige und ist immer offen. Die Dateinummer wird auf eine ganze Zahl gerundet.

Die Angabe REC *numerischer Ausdruck* darf nur erfolgen, wenn sich die *Dateinummer* auf eine Datei mit wahlfreiem Zugriff bezieht. Bezüglich weiter Informationen über Dateien mit wahlfreiem Zugriff und die richtige Anwendung von REC, schlagen Sie bitte im Kapitel 4 und im jeweiligen Peripherie-Gerätehandbuch nach. Der *numerische Ausdruck* wird ausgewertet und bestimmt die spezielle Datensatz-Nummer in der Datei, in die geschrieben wird.

Man benutzt USING, um ein genaues Format für Dateien des Typs DISPLAY festzulegen. Eine Beschreibung der Definition von Formaten und deren Wirkung mit dem PRINT Statement finden Sie bei IMAGE und USING. Falls sich USING auf eine Datei des Typs INTERNAL bezieht, tritt eine Fehlerbedingung auf.

Die *Druck-Liste* besteht aus Druckelementen und Druckseparatoren (=Trennzeichen). Druckelemente sind numerische Ausdrucke und String-Ausdrücke, die gedruckt werden und TAB Funktionen, welche die Position bezüglich der Spalte beim Ausdrucken steuern. Druckseparatoren sind Kommata oder Strichpunkte, welche die Position der Druckelemente bestimmen.

Die *Druck-Liste* wird von links nach rechts ausgewertet. Die Form der Ausgabe hängt vom Dateityp (DISPLAY oder INTERNAL) ab. Der Begriff "Dateityp" ist unter OPEN und in Kapitel 4 erläutert.

Dateien im DISPLAY-Format

Während der Ausführung eines PRINT Statements, das sich auf ein Gerät oder eine Datei mit Display-Format bezieht, wird die *Druck-Liste* wie folgt ausgewertet:

- Die Auswertung von *String-Ausdrücken* ergibt wieder einen String. String-Konstanten müssen zwischen Anführungszeichen gesetzt werden. Vor und nach String-Ausdrücken werden keine Leerstellen gesetzt. Falls Sie eine Leerstelle vor oder nach einem String setzen wollen, beziehen Sie diese in den String mit ein, oder setzen diese separat in Anführungszeichen.
- *Numerische Ausdrücke* werden ausgewertet und mit einer nachfolgenden Leerstelle angezeigt. Positive Werte werden mit einer führenden Leerstelle (anstatt eines Pluszeichens) gedruckt und negative Zahlen erscheinen mit einem voranstehenden Minuszeichen.
- Die TAB Funktion bestimmt für das nächste Element aus der *Druck-Liste* die Anfangsposition in der Druckzeile. Weitere Informantionen finden Sie unter TAB.

Zwischen zwei aufeinanderfolgende Druckelemente müssen Sie wenigstens einen Druckseparator setzen. Mehrere aufeinanderfolgende Druckseparatoren in einem PRINT Statement werden von links nach rechts ausgewertet.

- Der Strichpunkt druckt das nächste Element aus der *Druck-Liste* sofort nach dem vorangegangenen Element, ohne zwischen den Werten Leerstellen zu lassen.
- Das Komma druckt das nächste Druckelement am Anfang des nächsten Druckfeldes. Ein solches Druckfeld ist 15 Zeichen lang und beginnt jeweils in den Spalten 1,16,31 usw. Wenn die momentane Kursorposition den Anfang des letzten Druckfeldes schon überschritten hat, wird nach einem Komma das nächste Druckelement in den nächsten Datensatz geschrieben.

Paßt ein Druckelement nicht mehr in den momentanen Datensatz, so wird dieser Datensatz geschrieben und das Druckelement an den Anfang des nächsten Datensatzes gesetzt. Wenn ein numerisches Druckelement ohne seine abschließende Leerstelle noch in den momentanen Datensatz paßt, so wird es in diesen Datensatz gesetzt. Ist ein Druckelement länger als die Satzlänge, so wird es, in der Satzlänge entsprechende Segmente zerlegt, bis als letztes ein Restsegment von maximal der Länge eines Satzes übrigbleibt. Diese Segmente werden dann in nacheinanderfolgende Datensätze geschrieben.

Dateien im INTERNAL-Format

Während der Ausführung eines PRINT Statements, das sich auf ein Gerät oder eine Datei mit Internalformat bezieht, wird die Druckliste wie folgt ausgewertet:

- String-Ausdrücke werden ausgewertet und als binäre String-Darstellung in den Datensatz geschrieben.
- Numerische Ausdrücke werden ausgewertet und als binäre numerische Darstellung in den Datensatz geschrieben.
- Die TAB Funktion löst einen Fehler aus, wenn sie in Verbindung mit einer Internal-Datei benutzt wird.

Zwischen zwei aufeinanderfolgende Druckelemente müssen Sie wenigstens einen Druckseparator setzen. Mehrere aufeinanderfolgende Druckseparatoren in einem PRINT Statement werden von links nach rechts ausgewertet.

- Der Strichpunkt druckt das nächste Element aus der Druck-Liste sofort nach dem vorangegangenen Element, ohne zwischen den Werten Leerstellen zu lassen.
- Das Komma funktioniert als Druckseparator genauso wie der Strichpunkt.

Paßt ein Druckelement nicht mehr in den momentanen Datensatz, so wird dieser Datensatz geschrieben und das Druckelement an den Anfang des nächsten Datensatzes gesetzt. Wenn ein Druckelement länger als die Satzlänge ist, tritt ein Fehler auf.

Schwebende PRINT-Bedingungen

Wenn die *Druck-Liste* ohne Komma oder Strichpunkt endet, wird der Datensatz sofort in die Datei übertragen. Das nächste Eingabe/Ausgabe Statement, das zu der Datei Zugriff erlangt, beginnt einen neuen Datensatz.

Die Verwendung eines Kommas oder eines Strichpunktes am Ende der *Druck-Liste* erzeugt eine schwebende PRINT-Bedingung. Wenn die *Druck-Liste* mit einem Komma oder einem Strichpunkt abgeschlossen wird, wird der momentan verarbeitete Datensatz nicht geschrieben. Der Computer schreitet zum Anfang des nächsten Druckfeldes fort, wenn das PRINT Statement mit einem Komma abgeschlossen wurde, oder läßt keinen Zwischenraum, wenn das Statement mit einem Strichpunkt endet. Das nächste Ausgabe-Statement, das zu dieser Datei Zugriff erlangt, schreibt die Daten weiter in denselben Datensatz und beginnt damit bei der momentanen Kursorposition, wenn es nicht selbst die Position ändert.

Wenn die Druck-Liste weggelassen wurde und trotzdem ein schwebender Ausgabe-Datensatz existiert, überträgt das PRINT Statement diesen schwebenden Datensatz. Wenn kein schwebender Datensatz existiert, hängt das Ergebnis vom Dateityp ab. Bei einer Datei im Displayformat, schreibt das PRINT Statement einen leeren Datensatz (der Länge Null). Bei Daten im Internalformat tritt ein Fehler auf, weil das Internalformat keine leeren Datensätze zuläßt.

Querverweis

IMAGE,INPUT(mit Dateien),OPEN,TAB,USING

Beispiele

```
150 PRINT # 32,A,B,C,
```

Schreibt die Werte der Variablen A,B und C in den nächsten Datensatz der Datei, die unter der Nummer 32 eröffnet wurde. Das abschließende Komma erzeugt eine schwebende PRINT-Bedingung. Das nächste PRINT Statement, das zur Datei 32 Zugriff erhält, schreibt in denselben Datensatz, wie das vorhergehende.

PRINT MIT DATEIEN

Das nachstehende Programm schreibt Daten in eine Datei.

```
100 OPEN #5, "1.MEINEDATEI", OUTPUT
```
Eröffnet die Datei Nr. 5. Die Datei MEINEDATEI entsteht.

```
110 DIM A(50)
```
Dimensioniert ein Datenfeld für 51 Werte.

```
120 B=0
```
Initialisiert die Summenvariable.

```
130 FOR J=1 TO 50
```
Die Zeilen 130 bis 180 führen die Dateneingabe durch.

```
140 PRINT "EINGABE WERT";
150 INPUT A(J)
160 B=B+A(J)
170 PRINT #5,A(J)
```
Der Wert von A(J)

```
180 NEXT J
190 PRINT #5,B
```
Die Summe wird in die Datei Nr.5 geschrieben.

```
200 CLOSE #5
```
Die Datei Nr. 5 wird geschlossen.

UNTERPROGRAMM PUT

Mit dem PUT Unterprogramm wird der Inhalt des Systemspeichers an ein RAM Modul geschickt.

Format

CALL PUT (*Image-Zahl*)

Beschreibung

Das PUT Unterprogramm speichert eine Kopie des System RAM Image in ein 8K RAM Modul. Der Begriff "image" bezieht sich auf den gesamten Inhalt des 8K RAM Erweiterungsmoduls mit Programmzeilen, Variablen und ungenutztem Speicherplatz.

Die **Image-Zahl** kann 1 oder -1 sein. Mit 1 wird der Speicher in das Modul kopiert, mit -1 ein Austausch von Speicherinhalten veranlaßt. Mit dieser Option können Sie das Programm des Arbeitsspeichers aufzeichnen, während Sie ein Modulprogramm aufrufen.

Wenn Modulinhalte mit dem Speicher ausgetauscht werden, überprüft der Computer, ob das Modul einen Inhalt (Image) im Speicher hat. Ist dies nicht der Fall, zeigt der Computer eine Fehlermeldung an.

Beispiel

CALL PUT(1)

Kopiert den Inhalt des Systemspeichers in das Modul.

CALL PUT(-1)

Tauscht den Modulinhalt mit dem Systemspeicher aus.

Querverweis

GET, ADDMEM