Dieses Kapitel erläutert alle Befehle, Statements und Funktionen des BASIC der TI-70-Serie. Alle Befehlswörter sind in alphabetischer Aufstellung gelistet.

Inhaltsverzeichnis

REGELN IN DIESEM KAPITEL

Alle Beschreibungen der BASIC Befehlswörter erfolgen nach demselben Schema. Anordnung und Regeln in den einzelnen Beschreibungen sind nachfolgend erklärt.

Anordnung

Sinn des Befehlswortes wird zunächst erklärt.

Der Abschnitt über das Eingabeformat gibt Aufschluß über die vollständige Syntax des Befehls.

Die Beschreibung erklärt den Gebrauch oder die Wirkung des Befehls und zeigt Möglichkeiten für seine Anwendung auf.

Die Beispiele geben Aufschluß über die Anwendung eines Befehlswortes, wenn dies angebracht ist.

Der Querverweis-Abschnitt nimmt Bezug auf ähnliche oder komplementäre Befehle, soweit ein Zusammenhang besteht.

Eingabeformat-Regeln

Im Abschnitt über das Eingabeformat gelten folgende Regeln.

- BEFEHLE werden in Großbuchstaben gedruckt.
- **Variablen** werden kursiv dargestellt.
- Wahlweise möglichen Eingaben werden in Klammern [ ] wiedergegeben.
- Alle Klammern sind erforderlich. Kann eine Eingabe wahlweise erfolgen, so müssen auch dazugehörige Klammern gesetzt werden, wenn sie erfolgt.
- Eingaben, die wiederholt werden können, sind durch runde Klammern gekennzeichnet (...).

ABS

Eingabeformat

ABS(*numerischer Ausdruck)*

Beschreibung

Die ABS-Funktion gibt unabhängig vom Vorzeichen den absoluten Betrag eines *numerischen Ausdruckes* an. Das Ergebnis von ABS ist damit immer positiv oder gleich Null.

Beispiele

```
370 PRINT ABS(42.3): PAUSE
```
Zeigt 42,3 an

```
140 VV = ABS(-6.124)
```
Setzt VV gleich 6.124

ACCEPT

Das ACCEPT Statement unterbricht den Ablauf des Programms und ermöglicht so die Eingabe von Daten über die Tastatur. ACCEPT erlaubt die Eingabe von Daten von jeder Anzeigenposition aus, löscht die gesamte oder Teile der Anzeige, limitiert die Zahl und Art der zulässigen Eingabe und ersetzt einen fehlenden Wert für die Eingabevariable.

Eingabeformat

ACCEPT [[AT (Spalte)] [SIZE *(numerischer Ausdruck)*] [ERASE ALL] [VALIDATE *(Datenart)*] [NULL (AUSDRUCK)],] *Variable*

Beschreibung

Die allgemeine Form des ACCEPT Statements

ACCEPT *Variable*

ordnet die über die Tastatur eingegebenen Daten der **Variablen** zu. Die Variable kann je nach Art der eingegebenen Daten entweder numerisch oder ein String sein. Sie können bis zu 80 Zeichen eingeben, jede abschließende Leerstelle wird ignoriert.

Die Anzeige wird gelöscht von der gegenwärtigen Kursorposition bis zum Ende der 80spaltigen Zeile. Die Eingabe beginnt in Spalte 1, wenn nicht der Kursor durch ein vorangegangenes Eingabe/Ausgabe Statement anderswo gesetzt wurde; in diesem Fall beginnt die Eingabe dort, wo der Kursor steht.

Anmerkung: Wollen Sie einen String eingeben, der ein Komma, ein Anführungszeichen oder führende oder abschließende Leerstellen enthält, so müssen Sie den String in Anführungszeichen setzen. Ein Anführungszeichen innerhalb eines Strings wird durch zwei nebeneinander gesetzte Anführungszeichen dargestellt.

Wenn ein ACCEPT Statement Eingaben erwartet, löscht (CLR) lediglich das Eingabefeld; (CTL) (↑) (Ausgangspunkt) und (CTL) (←) (Tabulatorrücklauf) bewegen den Kursor zum Anfang des Eingabefelds, (CTL) (→) hat keine Wirkung.

ACCEPT

Möglichkeiten

Sie können eine oder mehrere der folgenden Optionen beliebig mit dem ACCEPT Statement verwenden. Setzen Sie vor jede Option eine Leerstelle (wenn nicht schon die geschlossene Klammer ")" einer vorhergehenden Option hier steht) und achten Sie darauf, ein Komma vor die letzte Option vor der **Variablen** zu setzen.

- AT (*Spalte*) – setzt den Kursor in die Spalte, die durch den gerundeten Wert der **Spalte** bestimmt wird. Der Eingabebereich reicht von der Kursorposition bis zum Ende der 80spaltigen Zeile.

  Spaltenwerte sind zulässig im Bereich von 1 bis 31. Ist der gerundete Wert der **Spalte** außerhalb dieses Bereichs, wird die Fehlermeldung **Bad value** angezeigt.

- SIZE (*numerischer Ausdruck*) – limitiert die Maximalzahl von Zeichen, die eingegeben werden können. SIZE läßt die Eingabe von maximal so vielen Zeichen zu, wie der Betrag des **numerischen Ausdrucks** angibt.

  Ist der **numerische Ausdruck** positiv, wird das Eingabefeld gelöscht, bevor eine Eingabe zugelassen wird.

  Ist der **numerische Ausdruck** negativ, wird das Eingabefeld nicht gelöscht. Dadurch können Sie einen zuvor mit einem DISPLAY oder PRINT Statement in die Anzeige geschriebenen Wert übernehmen.

  Der Kursor steht an der ersten Stelle des Eingabefeldes für weitere Eingabe/Ausgabe Statements.

  Beachten Sie, daß die SIZE Option nur die Maximumanzahl von Zeichen bestimmt, die eingegeben werden können. Ungeachtet der bestimmten Größe kann das Eingabefeld über die 80spaltige Zeile hinaus nicht erweitert werden.

- ERASE ALL – löscht die gesamte 80spaltige Anzeigenzeile vor Annahme einer neuen Eingabe.

- VALIDATE (*Datenart*) – erlaubt nur die Eingabe der durch die **Datenart** bestimmten Zeichen. Wird mehr als eine **Datenart** bestimmt, so werden Zeichen aus jeder Art zugelassen.

  Datenart kann eine der nachfolgenden Arten sein.

Art	**erlaubte Eingabe**
ALPHA	alle alphabetischen Zeichen
UALPHA	nur Großbuchstaben
DIGIT	alle Ziffern (0-9)
NUMERIC	alle Ziffern (0-9), den Dezimalpunkt (.), das Pluszeichen (+), das Minuszeichen (-) und den Großbuchstaben E
ALPHANUM	alle alphabetischen Zeichen und Ziffern
UALPHANUM	nur Großbuchstaben und Ziffern

  Anmerkung: Wenn Sie die Datenarten UALPHA und UALPHANUM verwenden, wird jeder eingegebene Kleinbuchstabe automatisch in einen Großbuchstaben verwandelt.

  Datenart kann auch ein String-Ausdruck sein; hier wird jedes Zeichen oder jede Zeichenkombination als Eingabe zugelassen.

- NULL (*Ausdruck*) – schafft einen vom Ausdruck bestimmten Nullwert. Wenn Sie (ENTER) drücken und das Eingabefeld ist leer, wird die nicht erfolgte Eingabe automatisch der **Variablen** zugeordnet. Beachten Sie, daß die VALIDATE Option keine Wirkung auf den Nullwert hat.

Zusätzliche Eingabemethoden

Es gibt zusätzlich noch zwei Arten, Daten während der Durchführung eines ACCEPT Statements einzugeben. Sie können:

- die (FN) Taste benutzen, um Befehle und anwenderdefinierte Strings einzugeben.

- einen numerischen Ausdruck eingeben, wenn die **Variable** numerisch ist. Der Ausdruck wird berechnet und das Ergebnis der **Variablen** zugeordnet.

Beispiele

100 ACCEPT AT(5)ERASE ALL,T

Löscht die Anzeige und läßt die Eingabe der Daten von Spalte 5 bis zum Ende der Zeile zu. Die Daten werden der Variablen T zugeordnet.

320 ACCEPT VALIDATE("YN") SIZE(1),A$

Läßt für ein nur ein Zeichen fassendes Feld die Variablen Y oder N zu. Das Zeichen wird der Variablen A$ zugeordnet.

430 ACCEPT AT(3)SIZE(-5) VALIDATE(DIGIT,"+-"),X

Läßt eine Eingabe bis zu 5 Zeichen ab Spalte 3 für die Variable X zu. Die eingegebenen Zeichen müssen Ziffern sein oder die Zeichen + oder -. Das Eingabefeld wird nicht gelöscht, da die SIZE Anweisung negativ ist.

570 ACCEPT NULL(PI),C

Läßt die Eingabe von Daten für die Variable C zu. Drücken Sie (ENTER), obwohl keine Daten eingegeben wurden, wird der Wert von π in C gespeichert.

210 PRINT "ADDRESS:";:ACCEPT AT(10),ADDR$

Zeigt in der Anzeige ADDRESS: an und stellt den Kursor hinter den Dialog. Die Eingabe von Daten für die Variable ADDR$ ist zugelassen. Beachten Sie, daß eine Pause mit DISPLAY nicht nötig ist, da der Strichpunkt eine schwebende Druckbedingung enthält.

Querverweis

INPUT, LINPUT

ACOS

Eingabeformat

ACOS*(numerischer Ausdruck)*

Beschreibung

Die ACOS*(Arkuskosinus)* Funktion berechnet den Winkel, dessen Kosinus dem *numerischen Ausdruck* entspricht. Das Ergebnis wird in dem zuletzt vor Benutzen dieser Funktion eingestellten Winkelmodus (RAD, DEG oder GRAD) berechnet. Nachstehend ist der Wertebereich für die Arkuskosinusfunktion in den drei Winkelmodi angegeben.

Einheit	Bereich des resultierenden Winkels
Altgrad	$0 \leq ACOS(X) \leq 180$
Radiant	$0 \leq ACOS(X) \leq PI$
Neugrad	$0 \leq ACOS(X) \leq 200$

Beispiele

```
100 DEG
```
Wählen des Altgrad-Modus
```
110 PRINT ACOS(1): PAUSE
```
Zeigt 0 an.
```
220 RAD
```
Wählen des Bogenmaß-Modus
```
230 T = ACOS(0.75)
```
Setzt T gleich 0,7227342478

ACOSH

Die ACOSH Funktion berechnet den hyperbolischen Arkuskosinus eines numerischen Ausdrucks.

Eingabeformat

ACOSH (*numerischer Ausdruck*)

Beschreibung

Die Funktion ACOSH (hyperbolischer Arkuskosinus) gibt die Zahl wieder, deren hyperbolischer Kosinus ein **numerischer Ausdruck** ist. Die Definition des hyperbolischen Arkuskosinus lautet:

ACOSH(X) = LN(X + SQR(X*X−1)

Beispiele

100 PRINT ACOSH(1):PAUSE

Druckt **0** aus (den hyperbolischen Arkuskosinus von 1).

230 A=ACOSH(4/3)

Setzt A gleich .7953654612 (dem hyperbolischen Arkuskosinus von 4/3).

Querverweis

ASINH, ATANH, COSH, SINH, TANH

ADDMEM UNTERPROGRAMM

Mit dem ADDMEM Unterprogramm wird die Speicherkapazität des 8K RAM Speichererweiterungsmoduls an den verfügbaren Arbeitsspeicher angehängt.

Eingabeformat

CALL ADDMEM

Beschreibung

Das ADDMEM Unterprogramm erweitert den Arbeitsspeicher, um so den Speicher eines RAM Moduls mit einzubeziehen. Der Computer betrachtet die verbundene Speicherkapazität nun als fortlaufenden Arbeitsspeicher.

Das ADDMEM Unterprogramm kann nur als Befehl verwendet, nicht jedoch in einem Programm angewendet werden.

Die durch das Hinzufügen des Modulspeichers an den Arbeitsspeicher erhaltene Speicherkapazität beträgt 16 KBytes.

Die Fehlermeldung **E31 No RAM** wird angezeigt, falls bei Ausführung das CALL ADDMEM Befehls kein 8K RAM angeschlossen ist.

Wenn der Speicher eines 8K RAM Speichererweiterungsmodul an den Arbeitsspeicher angehängt wird, behält das System gespeicherte BASIC Programmzeilen ; es werden jedoch alle Programme in PASCAL gelöscht, die im Speicher waren.

Bevor ein ADDMEM Unterprogramm ausgeführt wird, betrachtet der Computer den Speicher in einem 8K RAM als getrennten Speicherbereich. Der Inhalt des Arbeitsspeichers kann mit einem PUT Befehl in das Modul gespeichert werden. Nach Ausführung des ADDMEM Unterprogramms hält der Computer jedoch das Modul für einen Teil des Arbeitsspeichers und der Modulspeicher wird nicht mehr als separater Speicherbereich betrachtet.

UNTERPROGRAMM ADDMEM

Der Speicher in einem 8K RAM Speichererweiterungsmodul bleibt solange an den vorhandenen Speicherraum angeschlossen, bis ein NEW ALL Befehl ausgeführt wird, der Computer ohne eingesetztes Modul eingeschaltet wird, die Batterien entnommen oder das System initialisiert wird.

Querverweis

GET, PUT

ASC

Eingabeformat

ASC(*String-Ausdruck*)

Beschreibung

Die ASC-Funktion gibt den dezimalen ASCII-Zeichenkode an, der dem ersten Zeichen des durch den *String-Ausdruck* spezifizierten Strings entspricht. Die Fehlernachricht `E23 Bad argument` wird angezeigt, wenn der *String-Ausdruck* ein Null-String ist. Eine Liste der ASCII-Zeichenkodes wird in Anhang E angegeben. Die ASC-Funktion ist die Umkehrung der CHR$-Funktion.

Querverweis

CHR$

Beispiele

```
100 PRINT ASC ("A"): PAUSE
```
Druckt 65
```
120 B = ASC("1")
```
Setzt B gleich 49
```
790 DISPLAY ASC("HALLO"): PAUSE
```
Zeigt 72 an

ASIN

Eingabeformat

ASIN(*numerischer Ausdruck)*

Beschreibung

Die ASIN*(Arkussinus)* Funktion berechnet den Winkel, dessen Sinus dem *numerischen Ausdruck* entspricht. Das Ergebnis wird in dem zuletzt vor Benutzen dieser Funktion eingestellten Winkelmodus (RAD, DEG oder GRAD) berechnet. Nachstehend ist der Wertebereich für die Arkussinusfunktion in den drei Winkelmodi angegeben.

Einheit	Bereich des resultierenden Winkels
Altgrad	$-90 \leq \text{ASIN}(X) \leq 90$
Radiant	$-PI/2 \leq \text{ASIN}(X) \leq PI/2$
Neugrad	$-100 \leq \text{ASIN}(X) \leq 100$

Beispiele

```
140 DEG
```
Wählen des Altgrad-Modus
```
160 PRINT ASIN(1): PAUSE
```
Zeigt 90 an.
```
240 RAD
```
Wählen des Bogenmaß-Modus
```
250 B = ASIN(0.9)
```
Setzt B gleich 1.119769515

ASINH

Die ASINH Funktion berechnet den hyperbolischen Arkussinus eines numerischen Ausdrucks.

Eingabeformat

ASINH (**numerischer Ausdruck**)

Beschreibung

Die Funktion ASINH (hyperbolischer Arkussinus) berechnet die Zahl, deren hyperbolischer Sinus ein **numerischer Ausdruck** ist. Die Definition des hyperbolischen Arkussinus lautet:

ASINH(X) = LN(X + SQR(X*X + 1))

Beispiele

100 PRINT ASINH(0):PAUSE

Druckt **0** aus.

230 A=ASINH(4/3)

Setzt A gleich 1.098612289 (dem hyperbolischen Arkussinus von 4/3).

Querverweis

ACOSH, ATANH, COSH, SINH, TANH

ATANH

Die ATANH Funktion berechnet den hyperbolischen Arkustangens eines numerischen Ausdrucks.

Eingabeformat

ATANH (**numerischer Ausdruck**)

Beschreibung

Die Funktion ATANH (hyperbolischer Arkustangens) berechnet die Zahl, deren hyperbolischer Tangens ein **numerischer Ausdruck** ist. Die Definition des hyperbolischen Arkustangens lautet:

ATANH(X) = .5*LN((1 + X)/(1 – X))

Beispiele

100 PRINT ATANH(0):PAUSE

Druckt **0** aus.

230 A=ATANH(4/7)

Setzt A gleich .6496414921 (dem hyperbolischen Arkustangens von 4/7).

Querverweis

ACOSH, ASINH, COSH, SINH, TANH

ATN

Eingabeformat

ATN *(numerischer Ausdruck)*

Beschreibung

Die ATN (Arkustangens) Funktion berechnet den Winkel, dessen Tangens dem *numerischen Ausdruck* entspricht. Das Ergebnis wird in dem zuletzt vor Benutzen dieser Funktion eingestellten Winkelmodus (RAD, DEG oder GRAD) berechnet. Nachstehend ist der Wertebereich für die Arkustangensfunktion in den drei Winkelmodi angegeben.

Einheit	Bereich des resultierenden Winkels
Altgrad	$-90 < ATN(X) < 90$
Radiant	$-PI/2 < ATN(X) < PI/2$
Neugrad	$-100 < ATN(X) < 100$

Beispiele

```
130 GRAD
```
Wählen des Neugrad-Modus
```
140 PRINT ATN(30): PAUSE
```
Druckt 97.87871952
```
810 RAD
```
Wählen des Bogenmaß-Modus
```
820 Q=ATN(2.5)
```
Setzt Q gleich 1.19028995

BREAK

Eingabeformat

BREAK [Zeilennummern-Liste]

Beschreibung

Das BREAK Statement dient dazu, den Programmablauf an bestimmten Stellen im Programm, die als Stopp-Stellen bezeichnet werden, zu unterbrechen. Stopp-Stellen können auf zwei Arten spezifiziert werden. Wenn zum BREAK Statement keine *Zeilennummern-Liste* angegeben wird, tritt eine Stopp-Stelle auf, wenn das BREAK Statement ausgeführt wird. Bei vorhandener *Zeilennummern-Liste* werden Stopp-Stellen unmittelbar vor den Zeilen gesetzt, welche die *Zeilennummern-Liste* angibt. Das Drücken der **[BREAK]**-Taste unterbricht den Programmablauf genauso, als wenn ein BREAK Statement ausgeführt worden wäre.

Wenn eine Stopp-Stelle auftritt, wird die Nachricht `Break` angezeigt.

Eine Stopp-Stelle vor einer Programmzeile verbleibt so lange im Programm bis ein UNBREAK Statement ausgeführt wird, um das BREAK Statement aufzuheben, oder bis die Zeile editiert oder gelöscht wird.

Das BREAK Statement ist bei der Fehlersuche nützlich. Wenn das Programm mit einer Stopp-Stelle unterbrochen wurde, können Variablen gedruckt und Berechnungen durchgeführt werden, um festzustellen, warum das Programm nicht korrekt arbeitet. Der CONTINUE-Befehl kann benutzt werden, um den Programmablauf fortzusetzen.

Querverweis

CONTINUE, ON BREAK, UNBREAK

Beispiele

```
150 BREAK
```
Verursacht eine Stopp-Stelle, wenn das BREAK Statement ausgeführt wird.
```
100 BREAK 120, 130
```
Erzeugt Stopp-Stellen vor Ausführung der Zeilen 120 und 130.
```
BREAK 10, 400, 130
```
Erzeugt Stopp-Stellen vor Ausführung der Zeilen 10, 130 und 400.

CALL

Das CALL Statement überträgt die Programmsteuerung vom Hauptprogramm auf ein bestimmtes Unterprogramm. Ist das Unterprogramm ausgeführt, geht die Programmkontrolle zum ersten Statement über, das dem CALL Statement folgt.

Eingabeformat

CALL *Unterprogramm-Name*

CALL *Unterprogramm-Name* (*Argumentenliste*)

Beschreibung

Die beiden möglichen Eingabeformate für das CALL Statement werden nachfolgend erläutert.

- CALL **Unterprogramm-Name** – überträgt die Programmsteuerung auf das erste Unterprogramm mit dem angegebenen **Unterprogramm-Namen**. Der **Unterprogramm-Name** muß der Name eines existierenden Unterprogramms sein, da sonst die Fehlermeldung **E13 Not found** angezeigt wird.

- CALL **Unterprogramm-Name (Argumentenliste)** – Mit der **Argumentenliste** werden Daten aus einem oder mehreren Unterprogrammen übertragen. Die Liste kann aus einem oder mehrere durch Kommas getrennte Argumente bestehen. Zahl und Art der Argumente in der **Argumentenliste** muß den Parametern in der **Parameterliste** des Unterprogramms entsprechen, da sonst die Fehlermeldung **E23 Bad argument** oder **E1 Syntax** in der Anzeige erscheint.

CALL

Unterprogramm-Prioritäten

Die Reihenfolge, nach der der Computer ein bestimmtes Unterprogramm sucht, läuft folgendermaßen ab.

1. Eingebaute Unterprogramme wie ADDMEM, ERR, I0 und KEY
2. BASIC Unterprogramme, die durch SUB definiert sind
3. Unterprogramme aus Software-Modulen

Jedes eingebaute Unterprogramm wird unter dem eigenen Namen eigens in diesem Kapitel behandelt. BASIC Unterprogramme sind in Kapitel 4 und in diesem Kapitel unter SUB beschrieben.

Beispiel

```
100 CALL KEY (K,S)
```

Ruft das eingebaute Unterprogramm KEY auf.

Querverweis

ADDMEM, ERR, GET, I0, KEY, PUT, SUB, SUBEND, SUBEXIT

CHR$

Eingabeformat

CHR$*(numerischer Ausdruck)*

Beschreibung

Die CHR$-Funktion zeigt das Zeichen an, dessen ASCII-Zeichen dem Wert des *numerischen Ausdruckes* entspricht. Die CHR$-Funktion ist die Umkehrung der ASC-Funktion. Eine Liste der ASCII-Zeichenkodes für jedes Zeichen des Standardzeichensatzes gibt Anhang E an.

CHR$ dient dazu, mit PRINT oder DISPLAY Statements und Zeichen aus dem erweiterten Zeichensatz, der nicht über die Tastatur zugänglich ist, anzuzeigen (siehe Anhang E). Mit Peripheriegeräten dient CHR$ zur Ausübung von Kontrollfunktionen, wie zum Beispiel, den Drucker zum Seitenvorschub zu veranlassen.

Querverweis

ASC

Beispiele

```
840 PRINT CHR$(72): PAUSE
```
Druckt H

```
900 X$=CHR$(33)
```
Setzt X$ gleich!.

CLOSE

Eingabeformat

CLOSE #*Dateinummer* [,DELETE]

Beschreibung

Das CLOSE Statement hebt den Zusammenhang zwischen einer Datei und Ihrer momentanen *Dateinummer* auf. Das Programm hat zu dem Gerät oder der Datei so lange keinen Zugriff mehr, bis es (sie) wieder eröffnet wird. Sobald eine Datei abgeschlossen ist, kann die *Dateinummer* einem anderen Gerät oder einer anderen Datei zugeordnet werden. Bei jedem Versuch, eine nicht eröffnete Datei mit CLOSE abzuschließen, erfolgt eine Fehlermeldung.

Jeder der nachfolgenden Vorgänge schließt alle offenen Dateien ab.

- Editieren eines Programmes oder eines Unterprogrammes.
- Eingabe eines NEW, RENUMBER, RUN, OLD, SAVE oder VERIFY Befehls.
- Auflistung des Programmes oder eines Unterprogrammes.
- Aufruf des ADDMEM-Unterprogrammes.
- Abschalten des Systems oder Drücken der Resettaste.

Der normale Programmabschluß schließt ebenfalls alle offenen Dateien ab.

Einige Peripherie-Geräte bieten die Möglichkeit, durch Anhängen von DELETE an das CLOSE Statement, die Datei sofort zu löschen, wenn sie abgeschlossen wird. Der Gebrauch von DELETE wird im Handbuch der jeweiligen Peripherie-Geräte beschrieben.

Querverweis

OPEN

Beispiel

```
790 CLOSE # 6
```
Schließt Datei Nummer 6 ab.

CONTINUE

Eingabeformat

CONTINUE *[Zeilennumer]*

Beschreibung

Den CONTINUE (oder CON)-Befehl benutzt man, um nach einer Stoppstelle den Programmablauf erneut zu starten. Der Start eines Programmes oder eines Unterprogrammes kann bei der angegeben *Zeilennummer* erfolgen. Die Angabe der *Zeilennummer* muß sich auf eine Zeilennummer im Hauptprogramm beziehen, wenn das Hauptprogramm unterbrochen wurde. Bei der Unterbrechung eines Unterprogrammes muß sich die Angabe für die *Zeilennummer* auf eine Zeilennummer im Unterprogramm beziehen. Wird eine nicht zulässige *Zeilennummer* verwendet, kann dies zu nicht vorherbestimmbaren Ergebnissen führen.

Die folgenden Vorgänge lassen die Fortsetzung des Programmablaufes nach einer Stoppstelle mit CONTINUE nicht zu:

- Editieren eines Programmes oder eines Unterprogrammes.
- Eingabe eines NEW, RENUMBER, RUN, OLD, SAVE oder VERIFY Befehls.
- Auflistung des Programmes mit einem Peripherie-Gerät.
- Aufruf des ADDMEM-Unterprogrammes.
- Abschalten des Systems oder Drücken der Resettaste.

Querverweis

BREAK

COS

Eingabeformat

COS*(numerischer Ausdruck)*

Beschreibung

Die COS-Funktion berechnet den Kosinus des *numerischen Ausdrucks*. Das Ergebnis wird in dem zuletzt vor Benutzen dieser Funktion eingestellten Winkelmodus (RAD, DEG oder GRAD) berechnet. Schlagen Sie in Anhang B nach, um sich über Grenzwerte für den *numerischen Ausdruck* zu informieren.

Beispiele

```
140 GRAD
```
Wählen des Neugrad (GON) Winkelmodus
```
150 PRINT COS(30): PAUSE
```
Druckt .8910065242
```
240 RAD
```
Wählen des Bogenmaß Winkelmodus
```
300 T=COS(PI)
```
Setzt T gleich -1

COSH

Die COSH Funktion berechnet den hyperbolischen Kosinus eines numerischen Ausdrucks.

Eingabeformat

COSH (*numerischer Ausdruck*)

Beschreibung

Die Funktion COSH (hyperbolischer Kosinus) berechnet den hyperbolischen Kosinus eines **numerischen Ausdrucks**. Die Definition des hyperbolischen Kosinus lautet:

COSH(X) = .5*(EXP(X)+EXP(–X)) .5 ($e^{x} + e^{-x}$)

Beispiele

100 PRINT COSH(0):PAUSE

Druckt **1** aus.

230 T=COSH(0.75)

Setzt T gleich 1.294683285.

Querverweis

ACOSH, ASINH, ATANH, SINH, TANH

DATA

Eingabeformat

DATA *Datenliste*

Beschreibung

Das DATA Statement wird zusammen mit dem READ Statement benutzt, um den Variablen Werte zuzuordnen. Wenn ein READ Statement ausgeführt wird, werden die Werte aus der *Datenliste* den Variablen aus der Variablen-Liste des READ Statements zugeordnet. Die *Datenliste* besteht aus numerischen oder String-Konstanten, die durch Kommata voneinander getrennt werden. Führende und abschließende Leerstellen werden unterdrückt. Eine String-Konstante, welche Kommata oder führende oder abschließende Leerstellen enthält, muß in Anführungszeichen gesetzt werden. Ein Anführungszeichen innerhalb des Strings wird durch zwei Anführungszeichen dargestellt. Ein Null-String wird durch zwei aufeinanderfolgende Kommata wiedergegeben.

Ein DATA Statement muß als einziger Befehl in einer Zeile stehen. Es kann an jeder beliebigen Stelle in einem Programm oder Unterprogramm erscheinen. Wenn ein Programm mehr als ein DATA Statement enthält, erfolgt der Lesevorgang nach Zeilennummern in ansteigender Reihenfolge.

Das RESTORE Statement dient dazu, DATA Statements wiederholt zu lesen, oder die Reihenfolge zu ändern, in der DATA Statements gelesen werden sollen.

Querverweis

READ, RESTORE

DATA

Beispiel

Das nachstehende Programm liest und zeigt mehrere numerische - und String-Konstanten an.

```
100 FOR A=1 TO 5
110 READ B, C
120 PRINT B; C: PAUSE 1.1
130 NEXT A
```

Die Zeilen 100 bis 130 lesen zehn Datenelemente und zeigen ihre Werte an.

```
140 DATA 2, 4, 6, 7, 8
150 DATA 1, 2, 3, 4, 5
160 DATA """DIES SIND ANFUEHRUNGSZEICHEN"""
170 DATA "HIER, KEINE ANFUEHRUNGSZEICHEN"
180 DATA AUCH KEINE ANFUEHRUNGSZEICHEN
190 FOR A=1 TO 7
200 READ B$
210 PRINT B$: PAUSE 1.1
220 NEXT A
```

Die Zeilen 190 bis 220 lesen sieben Datenelemente und zeigen sie an.

```
230 DATA 1, ZAHL, ,TI
```


DEG

Eingabeformat

DEG

Beschreibung

Das DEG Statement stellt den Winkelmodus auf Altgrad ein. Sobald der DEG Winkelmodus gewählt wurde, werden alle eingegebenen und berechneten Winkel in Altgrad gemessen. Dieser Modus wird geändert und auf das Bogenmaß eingestellt, wenn NEW ALL eingegeben, oder das System initialisiert wird.

Querverweis

GRAD, RAD

DELETE

Eingabeformat

DELETE { *Zeilen-Gruppe [,Zeilengruppe...]* / *"Gerät. Dateienname"* }

Beschreibung

Das DELETE (oder DEL) Statement benutzt man, um Zeilen aus einem im Programmspeicher enthaltenen Programm zu entfernen, oder um eine Datei in einem externen Speichermedium zu löschen. Die *Zeilen-Gruppe* gibt Programmzeilen an, die gelöscht werden sollen, und darf folgendes enthalten:

Zeilen-Gruppe	Wirkung
Eine einzelne Zeilennr.	Entfernt diese Zeile.
Zeilennummer –	Entfernt diese und alle folgenden Zeilen.
– Zeilennummer	Entfernt diese und alle Zeilen davor.
Zeilennr. - Zeilennr.	Entfernt alle dazwischenliegenden Zeilen einschließlich der angegebenen.

Der Befehl DELETE *Zeilen-Gruppe* darf nicht Bestandteil eines Programmes sein.

Wenn die *Zeilen-Gruppe* eine einzelne Zeilennummer spezifiziert, die nicht existiert, wird die Fehlernachricht `W11 Line number error` angezeigt. Nach Drücken der **[ENTER]**-Taste oder der **[CLR]**-Taste, werden jedoch alle noch verbleibenden *Zeilen-Gruppen* gelöscht. Wenn die Anfangszeile dieser Gruppe nicht existiert, wird die nächsthöher numerierte Zeile als Anfangszeile interpretiert. Wenn die Endzeile einer Gruppe nicht existiert, wird die nächsttiefer numerierte Zeile als Endzeile interpretiert.

Gerät. Dateiname benutzt man, um eine Datei aus einem externen Speichermedium zu löschen. Als Angabe für das *Gerät* ist eine diesem zugeordnete Nummer einzugeben, die zwischen 1 und 255 betragen darf. Mit dem *Dateinamen* wird eine bestimmte Datei identifiziert.

DELETE

Gerät. Dateiname darf ein String-Ausdruck sein. Bezüglich der Gerätenummer für ein Peripheriegerät und wegen Form und Einschränkungen zu den *Dateinamen* schlagen Sie bitte im Handbuch für das jeweilige Peripheriegerät nach.

Dateien mit Datenstruktur können Sie aus einigen Peripheriegeräten durch Verwendung von DELETE im CLOSE Statement löschen. Weitere Informationen hierzu finden Sie im Peripherie-Gerätehandbuch.

Querverweis

CLOSE,

Beispiele

`DELETE 10-50, 90, 110-220`
Löscht die Zeilen 10 bis 50, die Zeile 90 und die Zeilen 110 bis 220.

`DELETE 900-`
Löscht alle Zeilen von 900 bis zum Ende des Programms.

`DELETE -500, 750`
Löscht alle Zeilen bis 500 und Zeile 750.

`DELETE "1.DATEI"`
Löscht "DATEI" aus Gerät 1.

DIM

Eingabeformat

DIM *Datenfeld-Name (Ganzzahl 1 [,Ganzzahl2] [,Ganzzahl 3]) [,...]*

Beschreibung

Das DIM Statement spezifiziert die Eigenschaften eines Datenfeldes und reserviert den erforderlichen Speicherplatz dafür. *Der Datenfeld-Name* darf aus einer String oder einer numerischen Variablen bestehen. Die Zahl der Werte in der Klammer, die dem *Datenfeld-Namen* folgt, bestimmt die Dimension des Datenfeldes. Datenfelder mit bis zu drei Dimensionen sind zulässig. Die Werte in den Klammern stellen die Maxima der Werte dar, welche die Indices in den einzelnen Dimensionen annehmen dürfen.

Der niedrigste Indexwert ist Null. Daher beträgt die Zahl der Elemente in jeder Dimension um eins mehr, als das Indexmaximum angibt. Zum Beispiel ist ein Datenfeld, das durch DIM A(6) definiert ist, ein eindimensionales Datenfeld, mit den sieben Elementen A(0) bis A(6). Wenn ein Datenfeld mittels seines DIM Statements nicht dimensioniert wurde, beträgt das Maximum für jeden Index 10.

Beim Start eines Programmes wird jedes Element eines numerischen Datenfeldes gleich Null gesetzt und jedes Element eines String-Datenfeldes wird einem Null-String gleichgesetzt.

Ein Datenfeld kann nur einmal dimensioniert werden. Ein DIM Statement muß im Programm bei einer niedrigeren Programmzeilennummer erscheinen, als jedes andere Statement, das sich auf dieses Datenfeld bezieht. Bemerkungen (REM) und Nachbemerkungen (!) sind die einzigen zulässigen Statements, die nach den DIM Statement in einer Mehrfach-Statement-Zeile auftreten dürfen. Ein DIM Statement darf nicht in einem IF THEN ELSE Statement erscheinen.

Beispiele

```
430 DIM X$(30)
```

Reserviert im Speicher des Computers Speicherplatz für ein Datenfeld namens X$ mit 31 Elementen. Jedem Element wird ursprünglich als Wert ein Null-String zugeordnet.

```
430 DIM D(100),B(10,9)
```

Reserviert im Speicher des Computers Speicherraum für 101 Elemente des mit D benannten Datenfeldes und 110 (11 x 10) Elemente des mit B benannten Datenfeldes. Jedem Element wird ursprünglich der Wert Null zugeordnet.

DISPLAY

Das DISPLAY Statement zeigt die (den) in der Druck-Liste aufgeführten Wert(e) an. (Durch die verschiedenen mit DISPLAY möglichen Optionen ist dieser Befehl vielseitiger einsetzbar als das PRINT Statement.)

Eingabeformat

DISPLAY [[*AT(Spalte)*] [*ERASE ALL*]
[*SIZE(numerischer Ausdruck)*] [*USING Zeilennummer,*]]
Druck-Liste

DISPLAY [[*AT(Spalte)*] [*ERASE ALL*]
[*SIZE(numerischer Ausdruck)*] [*USING String-Ausdruck,*]]
Druck-Liste

Beschreibung

Das DISPLAY Statement formatiert den (die) in der **Druck-Liste** aufgeführten Wert(e) und zeigt ihn (sie) an.

Die Druck-Liste kann aus einem oder mehreren durch Komma oder Strichpunkt getrennten numerischen oder String-Ausdrücken bestehen. Weitere Hinweise, wie Kommas und Strichpunkte den Platz zwischen den angezeigten Werten beeinflussen, finden Sie bei PRINT in diesem Kapitel.

Möglichkeiten

Die mit DISPLAY verfügbaren Möglichkeiten geben Ihnen die Kontrolle über das Ausgabeformat der angezeigten Information. Die Optionen können beliebig eingefügt sein und müssen durch eine Leerstelle eingeleitet werden (wenn nicht bereits vor der Option die geschlossene Klammer ")" einer vorhergegangenen Option steht).

- AT(*Spalte*) – setzt das erste Zeichen der angezeigten Information in die Spalte, die durch den gerundeten Wert der **Spalte** bestimmt wurde.

  Spaltenwerte sind zulässig im Bereich von 1 bis 80. Ist die **Spalte** größer als 80, gibt der TI-74 die Fehlermeldung **E4 Bad value** wieder.

Die Auswertung der TAB Funktion und der Komma-Separatoren erfolgt relativ zu der durch die AT Option bestimmten Startposition. Gehen jedoch die Zeichen, die angezeigt werden sollen, über Spalte 80 hinaus, so beginnt die angezeigte Information bei Spalte 1 und nicht in der durch die AT Anweisung, TAB Funktion oder Komma-Separator spezifizierten Spalte.

Die AT Option kann von der SIZE Option beeinflußt werden.

Wenn AT weggelassen wurde, beginnt die Auswertung bei Position 1, wenn nicht ein schwebendes Eingabe/Ausgabe Statement existiert und die TAB Funktion und die Komma-Separatoren relativ zu Spalte 1 sind.

- ERASE ALL – löscht die Zeile, bevor Daten angezeigt werden.
- SIZE (*numerischer Ausdruck*) – begrenzt die Gesamtzahl der angezeigten Zeichen auf den absoluten Wert des **numerischen Ausdrucks**.

  Ist der **numerische Ausdruck** größer als die Restanzahl der Anzeigestellen, erstreckt sich das Anzeigefeld von der momentanen Anzeigeposition bis an das Ende der Zeile. Die Länge des Anzeigefeldes wird nun zur neuen Satzlänge für die Auswertung der TAB Funktion und der Komma-Separatoren in der **Druck-Liste**. Das Anzeigefeld wird vor Anzeige neuer Daten immer gelöscht.

  Wird die SIZE Option weggelassen und die Zeichen, die angezeigt werden sollen, gehen über Spalte 80 hinaus, können Sie die Information nur ansehen, wenn Sie ein PAUSE ALL Statement vor der Ausgabezeile einbauen. Sie können nun den ersten Ausdruck ansehen und mit [ENTER] oder [CLR] den nächsten Abschnitt. Wird jedoch die SIZE Option verwendet und die Zeichen gehen über Spalte 80 hinaus, so wird der Text auf 80 Spalten beschränkt oder auf die Anzahl der Zeichen, die durch SIZE bestimmt wurde, was in jedem Fall kürzer als 80 Spalten bedeutet.

Wenn die SIZE Option weggelassen wird, bleibt die Anzeige ungelöscht (wenn nicht ERASE ALL angewendet wurde), bevor neue Daten eingegeben werden. Wenn der **Druck-Liste** kein Abschluß-Separator folgt, wird die Anzeige nach Abschluß des DISPLAY Statements vom letzten angezeigten Zeichen an bis zum Ende der Zeile gelöscht.

- USING – kann verwendet werden, um ein exaktes Format für die Datenausgabe zu spezifizieren. Wenn USING angewandt wird, muß es als letzte Anweisung in der Liste erscheinen. Zur Beschreibung der Definition von Formaten und der Wirkung auf die Datenausgabe mit dem DISPLAY Statement schlagen Sie bitte unter IMAGE und USING nach.

Beispiele

120 DISPLAY AT(7),Y

Zeigt den Wert von Y ab Spalte 7 an und löscht alles, was nach dieser Zahl in der Anzeige steht. Tatsächlich erscheint der Wert in Spalte 8, da ein Vorzeichen in Spalte 7 stehen würde.

150 DISPLAY N

Gibt den Wert von N in Spalte 1 der Anzeige an und löscht den Rest der Anzeige.

190 DISPLAY ERASE ALL,B

Löscht die gesamte Anzeige, bevor der Wert von B angezeigt wird.

370 DISPLAY AT(C) SIZE(19),X

Löscht 19 Zeichen ab Position C und zeigt beginnend von Position C den Wert von X an.

Querverweis

IMAGE, PAUSE, PRINT, TAB, USING

END

Eingabeformat

END

Beschreibung

Das END Statement schließt ein Programm ab und kann auch anstelle eines STOP Statement benutzt werden. Obgleich das END Statement überall im Programm stehen darf, wird es gewöhnlich in die letzte Zeile eines Hauptprogrammes gesetzt. Das END Statement ist nicht unbedingt erforderlich. Der Programmablauf hält von selbst an, nachdem die Zeile mit der höchsten Zeilennummer ausgeführt ist.

Das END Statement schließt alle offenen Dateien ab.

Querverweis

STOP

EOF

Eingabeformat

EOF*(Dateinummer)*

Beschreibung

Die EOF-Funktion dient dazu, um festzustellen, ob das logische Dateiende erreicht ist. Der Wert für die *Dateinummer* gibt an, welche Datei überprüft werden soll, und muß die Nummer einer offenen Datei sein. EOF zeigt die momentane Position in einer Datei wie folgt an:

Wert	Position
0	Dateiende nicht erreicht
-1	Logisches Dateiende erreicht

Das logische Ende einer Datei (im Gegensatz zum physikalischen, also tatsächlichen, von wo ab kein weiterer Raum mehr für Eingaben zur Verfügung steht) ist dann erreicht, wenn die ganze Information, die die Datei enthält, eingelesen ist.

Bei der Verwendung einer offenen (schwebenden) Eingabebedingung (siehe Kapitel 4), zeigt das EOF Statement nicht an, ob schwebende Daten im Speicher verbleiben.

Querverweis

INPUT (mit Dateien)

Beispiele

```
140 PRINT EOF(3): PAUSE
```

Druckt -1, wenn Datei Nr. 3 ihr logisches Ende erreicht hat und 0, wenn das logische Ende nicht erreicht ist.

```
710 IF EOF(27) THEN 1150
```

Wenn das Ende der Datei # 27 erreicht ist, wird der Programmablauf mit Zeile 1150 fortgesetzt.

ERR UNTERPROGRAMM

Eingabeformat

CALL ERR*(Fehlerkode, Fehlertyp [,Dateinummer, Zeilennummer])*

Beschreibung

Das ERR Unterprogramm zeigt den Fehlerkode, den Fehlertyp und gegebenenfalls die Dateinummer und Zeilennummer des zuletzt aufgetretenen Fehlers an. Falls eine Fehlerbedingung auftritt, kann man auch ein Teilprogramm (siehe ON ERROR) aufrufen, das den CALL ERR Befehl enthält. Die Fehlerbedingung wird gelöscht, wenn dieses Fehlerteilprogramm über einen Rücksprung (RETURN) verlassen wird.

Die *Fehlerkodes* reichen von 0 bis 127. Die Bedeutung der einzelnen Fehlerkodes werden in Anhang G angegeben.

Der *Fehlertyp* ist immer dann 0, wenn der *Fehlerkode* nicht 0 ist, wodurch ein Eingabe/Ausgabe (E/A) Fehler angezeigt wird. Bei einem E/A Fehler wird der *Fehlertyp* durch einen E/A *Fehlerkode* spezifiziert, der vom E/A Gerät vorgegeben wird. Der Bereich für E/A Fehlerkodes erstreckt sich von 1 bis 255.

Die *Dateinummer* ist 0, wenn es sich nicht um einen E/A Fehler handelt. Bei einem E/A Fehler wird die *Dateinummer* angezeigt, die in dem Statement angegeben ist, durch welches die Fehlerbedingung ausgelöst wird.

Als *Zeilennummer* wird jene angezeigt, welche bei Auftreten der Fehlerbedingung ausgeführt wurde. Nicht immer befindet sich die Fehlerursache in dieser Zeile, da ein Fehler auch als Folge von Werten oder logischen Entscheidungen auftreten kann, welche sich aus Zeilen an völlig anderer Stelle im Programm ergeben.

Falls keine Fehlerbedingung(en) vorliegen, ergibt das Resultat für CALL ERROR für alle Werte Nullen.

Querverweis

ON ERROR, RETURN (mit ON ERROR)

Beispiele

```
170 CALL ERR(A,B)
```
Setzt A dem *Fehlerkode* und B dem *Fehlertyp* des letzten unkorrigierten Fehlers gleich.

```
390 CALL ERR(W,X,Y,Z)
```
Setzt W dem *Fehlerkode* X dem *Fehlertyp*, Y der *Dateinummer* und Z der *Zeilennummer* des letzten unkorrigierten Fehlers gleich.

EXP

Eingabeformat

EXP*(numerischer Ausdruck)*

Beschreibung

Die EXP-Funktion ermittelt das Ergebnis für e^x, mit x als *numerischen Ausdruck.*

Der Wert für *e* beträgt 2,71828182846

Beispiele

```
150 Y=EXP(7)
```
Setzt Y der siebten Potenz von e gleich, deren Betrag sich zu 1096.633158 ergibt.

```
390 L=exp(3*PI)
```
Setzt L gleich $e^{3\pi}$.

FOR TO STEP

Eingabeformat

FOR *Kontrollvariable = Anfangswert TO Grenze [STEP Inkrement]*

Beschreibung

Das FOR TO STEP Statement verwendet man zur Programmierung von sich wiederholenden (iterativen) Prozessen (Schleifen), welche eine Reihe von Statements eine bestimmte Anzahl von Durchläufen ausführen. Die *Kontrollvariable* ist eine nicht indizierte, numerische Variable, die zur Zählung der Schleifendurchläufe dient. Der *Anfangswert*, die *Grenze* und das *Inkrement* sind numerische Ausdrücke.

Wenn das FOR Statement ausgeführt wird, erfolgt die Zuordnung des *Anfangswertes* als *Kontrollvariable*. Wenn der *Anfangswert* den *Grenzwert* überschreitet, wird die Schleife nicht ausgeführt und der Programmablauf mit dem Statement fortgesetzt, das dem NEXT Statement folgt. Anderenfalls werden die dem FOR Statement folgenden Statements ausgeführt, bis das zugehörige NEXT Statement ausgeführt wird. Dann wird das *Inkrement* zur *Kontrollvariablen* addiert. Falls die *Kontrollvariable* die *Grenze* nicht überschreitet, kehrt der Programmablauf zu den auf das FOR Statement folgenden Statements zurück.

Überschreitet die *Kontrollvariable* die *Grenze*, so verzweigt der Programmablauf zum auf NEXT folgenden Statement. Die *Kontrollvariable* nimmt dann den Wert an, den sie vor dem letzten Schleifendurchlauf hatte, zuzüglich des *Inkrements*.

Eine Schleife, die vollständig in einer anderen enthalten ist, nennt man verschachtelte Schleife. Verschachtelte Schleifen müssen voneinander verschiedene Kontrollvariablen verwenden. Der Programmablauf kann mit GOTO und GOSUB aus der Schleife heraus verzweigt werden, oder mit IF THEN ELSE verzweigen und anschließend wieder in die Schleife zurückspringen. Wenn ein NEXT Statement vor dem zu ihm gehörenden FOR Statement ausgeführt wird, erfolgt eine Fehlermeldung.

FOR TO STEP

STEP spezifiziert das *Inkrement*, das zur *Kontrollvariablen* addiert wird, jedesmal wenn die Schleife durchlaufen ist. Wird die STEP-Anweisung weggelassen, beträgt der Wert des *Inkrementes* 1. Wenn das *Inkrement* negativ ist, wird die *Kontrollvariable* bei jedem Schleifendurchlauf um dessen Wert vermindert und die *Grenze* muß niedriger als der *Anfangswert* sein. Die Schleife wird nicht durchgeführt, wenn der Anfangswert kleiner als die *Grenze* ist. Andernfalls wird die Schleife so lange wiederholt, bis die *Kontrollvariable* die *Grenze* unterschreitet.

Querverweis

NEXT

Beispiele

```
140 FOR A=1 TO 5 STEP 2
    :
190 NEXT A
```

Die Statements zwischen FOR und NEXT A werden dreimal nacheinander ausgeführt, wobei A die Werte 1, 3 und 5 annimmt. Wenn die Schleife abgeschlossen wird, hat A den Wert 7.

```
250 FOR J=7 TO -5 STEP -.5
    :
350 NEXT J
```

Die Statements zwischen FOR und NEXT J werden fünfundzwanzig Mal nacheinander ausgeführt, wobei J die Werte 7, 6.5, 6, ..., -4, -4.5 und -5 annimmt. Wenn die Schleife abgeschlossen wird, hat J den Wert -5.5

```
700 FOR X=1 TO 2 STEP -1
    :
780 NEXT X
```

Die Schleife wird nicht ausgeführt, weil das *Inkrement* negativ ist und der Anfangswert schon jetzt kleiner als die *Grenze* ist.

FORMAT

Eingabeformat

FORMAT *Gerät*

Beschreibung

Das FORMAT Statement initialisiert das momentan im externen Speichergerät enthaltene Medium. **Durch das Formatieren werden alle zuvor gespeicherten Daten gelöscht.**

Für die Bezeichnung des *Geräts* ist die dem Gerät zugeordnete Nummer, die zwischen 1 und 255 betragen kann, anzugeben. Bezüglich dieser Nummer schlagen Sie bitte im jeweiligen Peripherie-Gerätehandbuch nach.

Anmerkung

Das Formatieren eines angeschlossenen Kassetten-Recorders ist nicht notwendig.

FRE

Die FRE Funktion gibt Aufschluß über die gegenwärtige Verfügbarkeit des Speichers im Computer.

Eingabeformat

FRE (*numerischer Ausdruck*)

Beschreibung

Mit Hilfe der FRE Funktion erhalten Sie Informationen über Speicherbelegung und Verwendung:

- Wieviel Speicherbereich ist momentan für Programm- und Datenspeicherung verfügbar.
- Wieviel Speicherbereich wird momentan von dem im Speicher befindlichen Programm belegt.

Der Wert des **numerischen Ausdrucks** wählt die Informationsart wie folgt aus.

Wert	Bedeutung
0	Speicherplatz, der für Programm- und Datenspeicherung vorhanden ist (**nicht** zum Betrieb des Systems reservierter Speicherplatz).
1	Belegter Speicherplatz des im Speicher befindlichen Programms. Der angezeigte Wert enthält 11 Bytes für die Programmverwaltung.

Beispiel

```
300 A=FRE(1)
```

Setzt A gleich der Anzahl von Bytes, die zur Speicherung des laufenden Programms benötigt werden.

GET UNTERPROGRAMM

Das GET Unterprogramm ersetzt den Inhalt des Systemspeichers durch Informationen aus einem RAM Modul.

Eingabeformat

CALL GET (*Image-Zahl*)

Beschreibung

Das GET Unterprogramm ruft eine Kopie eines System RAM Image aus einem 8K RAM Modul auf. Der Begriff "Image" bezieht sich auf den gesamten Inhalt eines 8K RAM Erweiterungsmoduls mit Programmzeilen, Variablen und ungenutztem Speicherplatz.

Die Image-Zahl kann 1 oder -1 sein. Mit 1 wird der Modulinhalt in den Speicher 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 kopiert oder ausgetauscht werden, überprüft der Computer, ob das Modul ein Image von Systemspeicher enthält. Ist dies nicht der Fall, zeigt der Computer eine Fehlermeldung an.

Beispiel

CALL GET(1)

Kopiert den Modulinhalt in den Systemspeicher.

CALL GET(-1)

Tauscht den Modulinhalt mit dem Systemspeicher aus.

Querverweis

PUT, ADDMEM

GOSUB

Eingabeformat

GOSUB *Zeilennummer*

Beschreibung

Das GOSUB Statement überträgt die Steuerung des Programmablaufes zu dem Teilprogramm, das bei der angegebenen *Zeilennummer* beginnt. Die Statements des Teilprogrammes werden ausgeführt, bis ein RETURN Statement erreicht wird. Ein RETURN Statement überträgt die Steuerung des Programmablaufes sofort an das auf GOSUB folgende Statement.

Teilprogramme können in einem Programm beliebig oft aufgerufen werden und können sich selbst und andere Teilprogramme aufrufen. Bitte beachten Sie: Ein **Teilprogramm** ist ein abhängiger Teil eines Hauptprogrammes und darf nicht mit den grundsätzlich davon verschiedenen **Unterprogrammen** verwechselt werden. Unterprogramme sind eigenständige Programme, deren Eigenschaft bei SUB, SUBEND und SUBEXIT (sowie CALL) beschrieben sind. Daher kann GOSUB die Steuerung des Programmablaufes nicht in ein Unterprogramm hinein oder daraus heraus übertragen.

Querverweis

ON GOSUB, RETURN

Beispiel

```
100 GOSUB 200
```

Überträgt die Steuerung des Programmablaufes zu Zeile 200. Das Statement in Zeile 200 und alle darauf folgenden werden ausgeführt, bis ein RETURN Statement angetroffen wird. RETURN überträgt die Steuerung des Programmablaufes zum Statement, das auf GOSUB folgt.

GOTO

Eingabeformat

GOTO *Zeilennummer*

Beschreibung

Das GOTO Statement überträgt die Steuerung des Programmablaufes (nicht bedingt) zu einer anderen *Zeile* in einem Programm. Wenn ein GOTO Statement ausgeführt wurde, geht die Steuerung an das erste Statement in der angegebenen *Zeile* über.

Wie auch GOSUB kann GOTO die Steuerung des Programmablaufes nur zu Teilprogrammen verzweigen, nicht aber auf Unterprogramme einwirken. Bitte beachten Sie, die bei GOSUB gegebenen grundsätzlichen Hinweise.

Beispiel

```
100 GOTO 300
```

Überträgt die Steuerung des Programmablaufes zur Zeile 300.

GRAD

Eingabeformat

GRAD

Beschreibung

Das GRAD Statement stellt den Winkelmodus auf die Einheit Neugrad ein. Sobald der Grad Winkelmodus gewählt wurde, werden alle eingegebenen und berechneten Winkel in Neugrad gemessen. Dieser Modus wird geändert und auf das Bogenmaß eingestellt, wenn [NEW ALL] eingegeben oder das System initialisiert wird.

Querverweis

DEG,RAD

IF THEN ELSE

Eingabeformat

IF *Bedingung* THEN *Folge-1* [ELSE *Folge-2*]

Beschreibung

Das IF THEN ELSE Statement führt eine von zwei vorherbestimmten Folgen, in Abhängigkeit von der Erfüllung der vorgegebenen Bedingung aus. Wenn die *Bedingung* erfüllt ist, wird *Folge-1* ausgeführt. Wenn die *Bedingung* nicht erfüllt ist, wird *Folge-2* ausgeführt. Wenn ELSE weggelassen ist und die *Bedingung* nicht erfüllt ist, geht die Steuerung des Programmablaufes zur nächsten Zeile über.

Die *Bedingung* kann entweder ein Vergleichsausdruck oder ein numerischer Ausdruck sein. Wenn ein Vergleichsausdruck ausgewertet wird, ergibt sich 0 für falsch und -1 für wahr. Bei der Auswertung eines numerischen Ausdruckes, wird das Ergebnis 0 als falsch und ungleich 0 als wahr interpretiert.

Folge-1 und *Folge-2* können Zeilennummern, Statements oder durch Doppelpunkte getrennte Gruppen von Statements sein. Wenn eine Zeilennummer angegeben ist, wird die Steuerung des Programmablaufes zu dieser Zeilennummer übertragen. Im Falle der Angabe von Statements, werden diese ausgeführt.

Das IF THEN ELSE Statement muß in einer Zeile stehen und wird mit dem Ende dieser Zeile abgeschlossen. IF THEN ELSE Statements können verschachtelt werden, indem ein weiteres IF THEN ELSE Statement in *Folge-1* oder *Folge-2* auftritt. Wenn verschachtelte IF THEN ELSE Statements nicht die gleiche Anzahl von THEN und ELSE Anweisungen enthalten, so wird jedes ELSE als zum nächsten ungepaarten THEN gehörend betrachtet.

IF THEN ELSE Statements dürfen keine DIM, IMAGE, SUB oder SUBEND Statements enthalten.

Beispiele

```
100 IF Y<5 THEN 150
```

Wenn der Wert von Y kleiner als 5 ist, wird das erste Statement aus Zeile 150 ausgeführt. Falls Y größer oder gleich 5 ist, wird das dem Vergleich unmittelbar folgende Statement ausgeführt.

IF THEN ELSE

```
140 IF WERT=0 THEN 200
150 PRINT "UNGLEICH NULL":PAUSE 2
```

Wenn WERT gleich null ist, wird die Steuerung des Programmablaufes zu Zeile 200 übertragen. Falls WERT nicht null ist, wird UNGLEICH NULL angezeigt und der Programmablauf für 2 Sekunden bis zur Ausführung des nächsten Statements unterbrochen.

```
230 IF X>5 THEN GOSUB 300 ELSE X=X+5
```

Wenn der Wert für X größer als 5 ist, wird GOSUB 300 ausgeführt (also zum in Zeile 300 beginnenden Teilprogramm verzweigt). Sobald das Teilprogramm durchlaufen ist, wird die Steuerung des Programmablaufes an die dem IF THEN ELSE Statement folgende Zeile übertragen. Ist X kleiner oder gleich 5, dann wird X=X+5 gesetzt und die Steuerung geht zur folgenden Zeile über.

```
250 IF Q THEN C=C+1:GOTO 500 ELSE L=L/C:GOTO 300
```

Wenn Q ungleich Null(wahr) ist, wird C gleichgesetzt C+1 und die Steuerung des Programmablaufes zu Zeile 500 übertragen. Ist Q gleich Null (falsch) wird L gleichgesetzt L/C und die Steuerung geht zu Zeile 300 über.

```
290 IF A$="J" THEN COUNT=COUNT + 1:DISPLAY AT(4),
    "KEHREN WIR ZURUECK!":GOTO 200
```

Wenn A$ gleich "J" ist, wird die Variable COUNT um 1 vermehrt, eine Nachricht angezeigt und die Steuerung des Programmablaufes zu Zeile 200 verzweigt. Wenn A$ ungleich "J" ist, geht die Steuerung zur folgenden Zeile über.

```
350 IF STD=< 40 THEN BEZUEGE=STD * LOHN
    ELSE BEZUEGE=STD*LOHN+.5*LOHN*(STD-40):MS=1
```

Wenn STD kleiner oder gleich 40 ist, wird die Variable BEZUEGE dem Produkt aus STD und LOHN gleichgesetzt und die Steuerung des Programmablaufes geht zur nächsten Zeile über. Ist STD größer als 40, wird BEZUEGE gleichgesetzt STD * LOHN+.5 * LOHN * (STD-40), MS wird gleich 1 gesetzt und die Steuerung an die nächste Zeile übertragen.

```
700 IF A=1 THEN IF B=2 THEN C=3 ELSE D=4
```

Wenn A gleich 1 und B gleich 2 ist, wird C gleich 3 gesetzt und die Steuerung des Programmablaufes geht zur nächsten Zeile über. Ist A gleich 1 und B ungleich 2, wird D gleich 4 gesetzt und die Steuerung an die nächste Zeile übertragen. Für A ungleich 1 schreitet die Steuerung zur nächsten Zeile fort.

IMAGE

Eingabeformat

IMAGE *String-Konstante*

Beschreibung

Das IMAGE Statement benutzt man, um ein Ausgabeformat zu definieren. Dieses Format wird dargestellt, indem man die Zeilennummer des IMAGE Statements durch eine USING Anweisung in ein DISPLAY oder PRINT Statement einsetzt (siehe USING in diesem Kapitel). Die *String-Konstante* kann in Anführungszeichen gesetzt werden. Wenn die *String-Konstante* nicht in Anführungszeichen gesetzt wird, werden führende und abschließende Leerstellen unterdrückt.

Das IMAGE Statement muß als einziges in einer Zeile stehen und darf nur in dem Programm oder Unterprogramm stehen, von dem es herangezogen wird. Wenn ein IMAGE Statement im Programm auftritt, schreitet der Programmablauf sofort zur nächsten Programmzeile fort.

Die Definition eines Formates besteht aus Formatierungsfeldern und direkten Feldern. Wenn ein PRINT oder DISPLAY Statement eine Definition für ein Format heranzieht, werden die Formatierungsfelder durch die Werte der Druckdateneinheiten ausgefüllt und die direkten Felder werden unverändert dargestellt. Nachstehend wird die Definition von Formaten erklärt.

Definition von Formaten

Die drei Zeichen, die zur Definition eines Formates benutzt werden können, sind das Ziffernsymbol (#), der Dezimalpunkt (.) und das Potenzensymbol(^). Das Ziffernsymbol definiert die Position eines Zeichens im Formatierungsfeld. Es wird beim Ausdruck ersetzt, durch ein Zeichen der ASCII Darstellung des Wertes der Druckdateneinheit. Der Dezimalpunkt dient im dezimalen Formatierungsfeld dazu, die Position des Dezimalpunktes zu spezifizieren. Das Potenzensymbol dient in einem exponentiellen Formatierungsfeld dazu, die Zahl der Positionen, in denen der Exponent erscheint anzugeben. Alle anderen Zeichen werden vom FORMAT Statement direkt (unverändert) übernommen.

Die fünf möglichen Arten der Felder in einer Formatierungsdefinition heißen ganzzahlig, dezimal, exponentiell, String und direkt. Die für die einzelnen Arten geltenden Regeln werden nachstehend aufgeführt.

Ganzzahliges Feld

- Bis zu 14 signifikante Ziffern dürfen angegeben werden.
- Ein Ganzzahlfeld besteht aus Ziffernsymbolen (#).
- Wenn eine Zahl das Feld nicht ausfüllt, erscheint sie rechtsbündig.
- Wenn eine Zahl die Feldlänge überschreitet, werden statt ihres Wertes Sternchen gedruckt (*).
- Reale Zahlen werden auf die nächste ganze Zahl gerundet.
- Bei einer negativen Zahl wird eine Ziffernposition für das Minuszeichen benutzt.

Dezimales Feld

- Bis zu 14 gültige Ziffern dürfen angegeben werden.
- Das Dezimalfeld besteht aus Ziffernsymbolen und einem Dezimalpunkt. Der Dezimalpunkt darf an beliebiger Stelle im Formatierungsfeld stehen.
- Die Zahl wird mit dem Dezimalpunkt an der angegebenen Position angezeigt.
- Wenn der ganzzahlige Teil des Wertes länger ist, als der ganzzahlige Teil im Format, werden Sternchen anstelle des Wertes gedruckt.
- Die Zahl wird entsprechend der rechts vom Dezimalpunkt vorhandenen Stellen gerundet.
- Bei negativen Zahlen muß im Format wenigstens ein Ziffernsymbol links vom Dezimalpunkt stehen, um das Minuszeichen anzeigen zu können.

Exponentielles Feld

- Bis zu 14 gültige Ziffern dürfen angegeben werden.
- Ein Exponentialfeld besteht aus einem dezimalen oder ganzzahligen Feld für die Definition der Mantisse, worauf vier oder fünf Potenzsymbole für die Definition des Exponenten folgen. Werden weniger als drei Symbole angegeben, so werden diese direkt angezeigt. Bei mehr als fünf Symbolen dienen die ersten fünf der Definition des Exponenten und die restlichen Symbole werden direkt angezeigt.
- Die Zahl wird entsprechend der Definition für die Mantisse gerundet.

- Wenn die Definition der Mantisse Stellen links vom Dezimalpunkt enthält, bleibt eine Stelle immer dem negativen Vorzeichen vorbehalten und bei positiven Zahlen frei.

String-Feld

- Die Länge des Feldes wird nur durch die Länge des Strings begrenzt, der das Format definiert.
- Ein String-Feld besteht aus einem ganzzahligen, dezimalen oder exponentiellen Feld. Zusätzlich zu den Ziffernsymbolen definieren der Dezimalpunkt und die Potenzensymbole die Positionen der Zeichen.
- Wenn der String kürzer als das Feld ist, erscheint er linksbündig.
- Wenn der String länger als das Feld ist, werden an dessen Stelle Sternchen gedruckt (*).

Direkte Felder

- Die Länge des Feldes wird nur durch die Länge des Strings begrenzt, der das Format definiert.
- Ein direktes Feld wird unverändert angezeigt. Es besteht aus Alpha-numerischen Zeichen, die nicht als Formatierungszeichen interpretiert werden. D.h. alle Zeichen mit Ausnahme von: (#), (.) und (^).
- Direkte Felder erscheinen in gedruckter Ausgabe genauso wie bei der Definition ihres Formates.

Querverweis

DISPLAY, PRINT, USING

Beispiele

Das folgende Programm druckt unter Verwendung des IMAGE Statements zwei Zahlen pro Zeile.

```
100 FOR ZAEHLVARIABLE=1 TO 6
110 READ A,B
120 PRINT USING 150, A,B:PAUSE 2
130 NEXT ZAEHLVARIABLE
140 DATA -99, -9.99, -7, -3.459, 0, 0, 14.8, 12.75, 795, 852,
    -984, 64.7
150 IMAGE ERGIBT FORMATIERT ### UND ##.##
```

Das Ergebnis mit den vorgegebenen Werten ist nachstehend wiedergegeben.

Werte		Darstellung
−99	−9.99	ERGIBT FORMATIERT −99 UND −9.99
−7	−3.459	ERGIBT FORMATIERT −7 UND −3.46
0	0	ERGIBT FORMATIERT 0 UND .00
14.8	12.75	ERGIBT FORMATIERT 15 UND 12.75
795	852	ERGIBT FORMATIERT 795 UND *****
−984	64.7	ERGIBT FORMATIERT *** UND 64.70

Ein ähnliches Programm ermöglicht den Gebrauch von Buchstaben mit IMAGE LIEBE #####. Hier ist das Ergebnis mit einigen Namen.

Wert	Darstellung
UTE	LIEBE UTE
KARIN	LIEBE KARIN
ANGELA	LIEBE *****

Das folgende Programm zeigt eine Anwendung für IMAGE. Es liest sieben Zahlen und zeigt diese mit ihrer Summe an. Die Beträge werden mit bündigen Dezimalpunkten gedruckt.

```
100 IMAGE DM #####.##
110 IMAGE "     #####.##"
```

Die Zeilen 100 und 110 bestimmen die Darstellung. Die Zeile 110 erhält im IMAGE Statement 2 Leerstellen. Um diese Leerstellen zu erhalten, müssen die Ziffernsymbole in Anführungszeichen gesetzt werden.

```
120 DATA 233.45, -147.95, 8.4, 37.263, -51.299, 85.2, 464
130 SUMME = 0
140 FOR A=1 TO 7
150 READ BETRAG
160 SUMME=SUMME+BETRAG
170 IF A=1 THEN PRINT USING 100, BETRAG:PAUSE ELSE
    PRINT USING 110, BETRAG:PAUSE
```

Die Werte werden mittels des IMAGE Statements formatiert angezeigt.

```
180 NEXT A
190 PRINT USING "DM #####.## GESAMT.",
    SUMME:PAUSE
```

In Zeile 190 wird das Format direkt in einem PRINT Statement vorgegeben. Die Anführungszeichen haben in diesem Falle keine Wirkung, müssen aber vorhanden sein.