Inhaltsverzeichnis zur Anleitung zum TOOL1

APPEND . . . . . Seite 1-2

CHANGE . . . . . Seite 1-4

DELETE . . . . . Seite 1-9

ERASE . . . . . Seite 1-11

FIND . . . . . Seite 1-13

KEEP . . . . . Seite 1-16

LINK . . . . . Seite 1-17

PLAST . . . . . Seite 1-18

PLIST . . . . . Seite 1-19

PROGRAM . . . . . Seite 1-20

RENUMBER . . . . . Seite 1-21

SPLIT . . . . . Seite 1-26

Liste der Instruktionen . . . . . Seite 1-27

APPEND

****
* Eröffnet ein neues Programm-Modul. *
****

Syntax

Hinweise:
- Es kann jeweils nur ein neues Programm-Modul eröffnet werden, d.h. ein weiteres Modul kann erst dann eröffnet werden, wenn das vorangehende Modul mindestens eine Zeile enthält.
- Es können nur BASIC - Programm-Module verwaltet werden.
- Das neu eröffnete Programm-Modul ist sofort aktives Modul.

Besondere Fehlermeldung:
- ERROR 22: Zur Eröffnung eines neuen Moduls steht kein Speicherplatz mehr zur Verfügung.

*** Bemerkung: Beachten Sie bitte im Anhang die Erläuterungen zur modularen Programmierung (Seite A-6). ***

APPEND (Beispiel)

Ein bestehendes Modul im Speicher kann um ein weiteres Modul erweitert werden:

Ausgangssituation:

MODUL 1 befindet sich im Speicher:

```
 10:"A"REM MODUL 1
 20:INPUT "D = ";D
 30:J=D:H$=""
 40:H=INT (J/16)
 50:J=J-16*H
 60:H$=MID$ ("0123
    456789ABCDEF",
    J+1,1)+H$
 70:J=H
 80:IF JTHEN 40
 90:PRINT "D = &";
    H$
100:END
```

Nach der Befehlsfolge

APPEND (enter)

10"B"REM MODUL 2 (enter)

20PRINT "TIME = ";TIME (enter)

30END (enter)

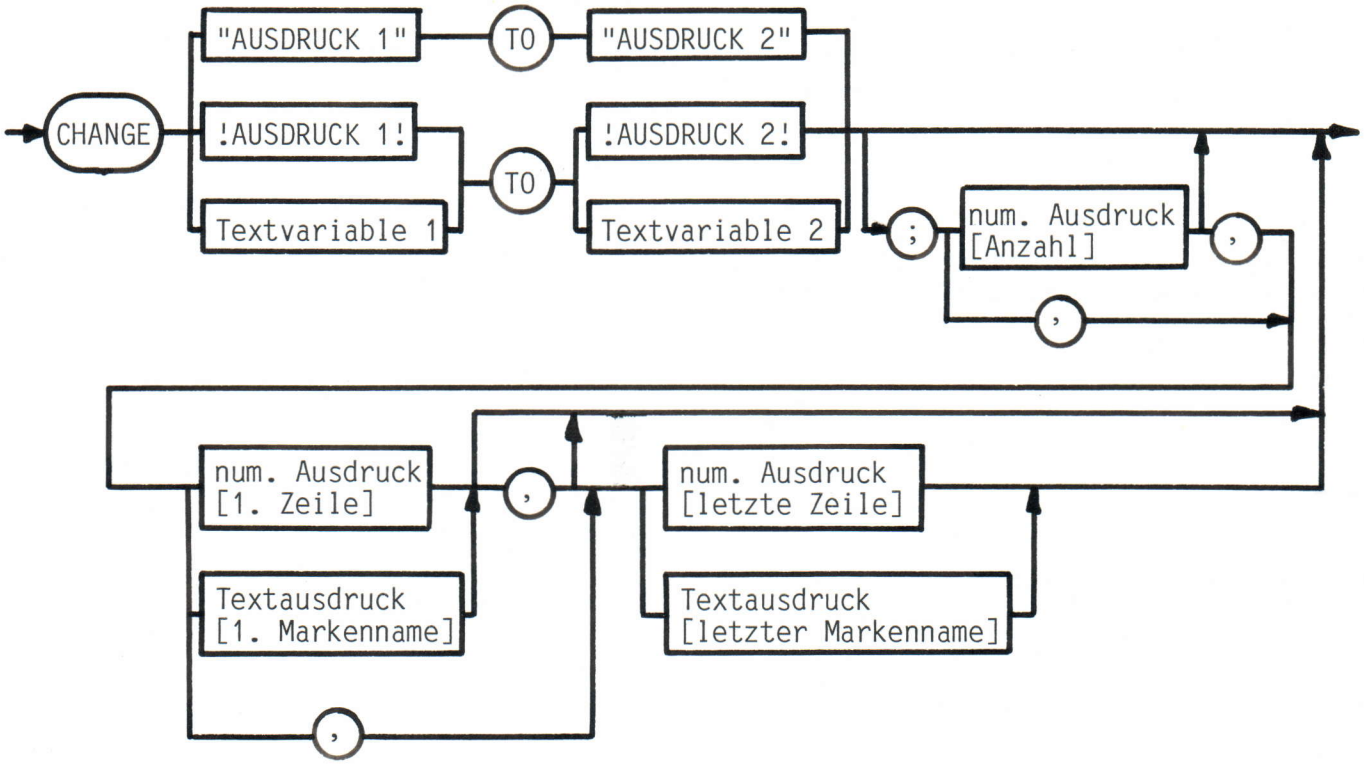
befindet sich folgendes im Speicher:

```
 10:"A"REM MODUL 1
 20:INPUT "D = ";D
 30:J=D:H$=""
 40:H=INT (J/16)
 50:J=J-16*H
 60:H$=MID$ ("0123
    456789ABCDEF",
    J+1,1)+H$
 70:J=H
 80:IF JTHEN 40
 90:PRINT "D = &";
    H$
100:END
 10:"B"REM MODUL 2
 20:PRINT "TIME =
    ";TIME
 30:END
```

CHANGE

```
*************************************************************
*                                                           *
*  Ersetzt [AUSDRUCK 1] durch [AUSDRUCK 2] im aktiven       *
*  Programm-Modul.                                          *
*  Anzahl und Bereich können angegeben werden.              *
*                                                           *
*************************************************************
```

Syntax



Parameter-Beschreibung:

- [AUSDRUCK 1] oder Textvariable 1: Ausdruck, der überschrieben werden soll.
- [AUSDRUCK 2] oder Textvariable 2: Ausdruck, der eingesetzt werden soll.

[AUSDRUCK 1] und [AUSDRUCK 2] können STRINGS oder BASIC-Schlüsselwörter sein (vergleiche Erläuterungen im Anhang, Seite A-4).
STRINGS können in Textvariablen zwischen gespeichert sein.

ACHTUNG: EIN STRING, DER IN EINER TEXTVARIABLEN ZWISCHEN GESPEICHERT IST, DARF KEINE ZEICHENFOLGE ENTHALTEN, DIE EINEM BASIC-SCHLÜSSELWORT ENTSPRICHT.

Ausdrücke zwischen Anführungszeichen (") wirken nur auf Texte innerhalb Anführungszeichen, Ausdrücke zwischen Ausrufezeichen (!) und Textvariablen wirken nur auf BASIC-Schlüsselwörter, Variablen und numerische Ausdrücke, einschliesslich Ausdrücke hinter REM.

- [Anzahl]: Mit diesem Parameter kann angegeben werden, wie oft [AUSDRUCK 1] durch [AUSDRUCK 2] ersetzt werden soll.
  [Anzahl] muss kleiner 256 sein.
  Wird * angegeben, wird so oft als möglich ausgetauscht.
  Wird nichts angegeben, wird einmal ausgetauscht.

- [1. Zeile] oder [1. Markenname]: Mit diesem Parameter kann angegeben werden, ab welcher Zeile oder ab welchem Markennamen ausgetauscht werden soll.
  Wird nichts angegeben, beginnt der Austausch bei der ersten Zeile des aktiven Programm-Moduls.

- [letzte Zeile] oder [letzter Markenname]: Mit diesem Parameter kann angegeben werden, bis zu welcher Zeile oder bis zu welchem Markennamen ausgetauscht werden soll.
  Wird nichts angegeben, endet der Austausch bei der letzten Zeile des aktiven Programm-Moduls, oder wenn die mit [Anzahl] begrenzte Menge Austausche erreicht ist.

Hinweis: Das BASIC-Schlüsselwort REM kann nur ausgetauscht werden, indem man einer Textvariablen den Wert CHR$ &F1 + CHR$ &AB zuordnet und diese Textvariable anstelle von [AUSDRUCK 1] oder [AUSDRUCK 2] angibt.

ACHTUNG: ZUM LÖSCHEN EINER GANZEN ZEILE MUSS DELETE VERWENDET WERDEN (siehe Seite 1-9) !

Besondere Fehlermeldungen:

- ERROR 17: Die Begrenzungen von [AUSDRUCK 1] und [AUSDRUCK 2] sind nicht von der gleichen Art (siehe Syntax-Diagramm).
- ERROR 67 IN xx: Zeile xx würde nach dem Austausch die maximal zulässige Länge übersteigen.
- ERROR 177: Das BASIC-Programm ist nicht in Ordnung.
- ERROR 178 IN xx: Beim Austausch in Zeile xx würden Daten überschrieben, d.h. es steht für den Austausch nicht mehr genügend Speicherplatz zur Verfügung.

Einschränkungen:

- Die numerischen Ausdrücke [Anzahl], [1. Zeile] und [letzte Zeile] dürfen nur Grundrechenoperationen enthalten. Klammerausdrücke sind verboten !
- Anführungszeichen (") können nicht ausgetauscht werden.

CHANGE (Beispiele)

Ausgangssituation:

Es befinden sich die drei nebenstehenden Module im Speicher:

```
100:"A"REM MODUL 1
110:PAUSE TIME
120:"Z"PRINT TIME
130:END
100:"B"REM MODUL 2
110:PRINT "LOGONX"
120:PRINT "X = ";X
130:REM Y=LOG X
140:PRINT "LOG = "
    ;Y
150:END
100:"C"REM MODUL 3
110:PRINT "LOG = "
    ;LOG Y
120:END
```

Beispiele von CHANGE-Anweisungen:

Instruktion	Wirkung
PROGRAM 2 (enter)	Programm-Modul 2 ist nun aktives Modul.
CHANGE !PRINT! TO !INPUT!;,120 (enter)	Ersetzt in Zeile 120 PRINT durch INPUT maximal einmal.
A$=CHR$ &F1 + CHR$ &AB (enter)	Weist A$ das TOKEN für das BASIC-Schlüsselwort REM zu (siehe Seite A-4).
CHANGE A$ TO !!;,110, (enter)	Löscht ab Zeile 110 bis zum Modulende REM maximal einmal.
CHANGE !PRINT! TO !PAUSE!;* (enter)	Ersetzt PRINT im ganzen Modul durch PAUSE so oft als möglich.
CHANGE "O" TO "**XYZ**";2 (enter)	Ersetzt im gesamten Modul "O" durch "**XYZ**" maximal zwei mal.
PROGRAM 1 (enter)	Programm-Modul 1 ist nun aktives Modul.
CHANGE !TIME! TO "PC-1500" (enter)	ERROR 17: [AUSDRUCK 1] ist durch Anführungszeichen (") begrenzt, [AUSDRUCK 2] dagegen durch Ausrufezeichen (!).
CHANGE !TIME! TO !ABS SIN (X-Y)! (enter)	Ersetzt im gesamten Modul TIME durch ABS SIN (X-Y) maximal einmal.

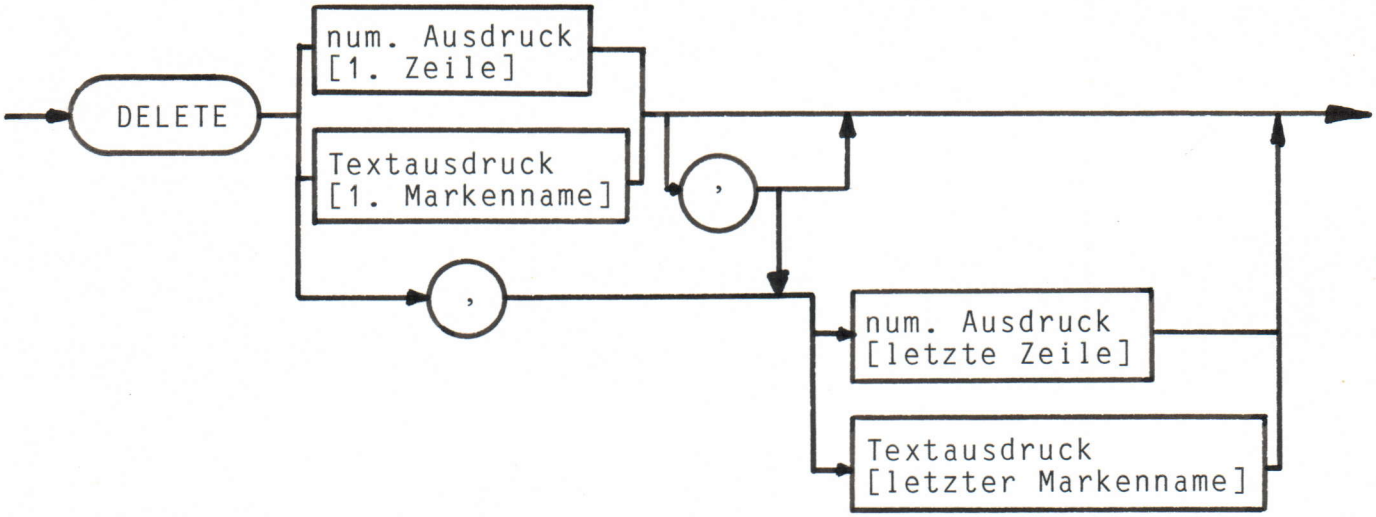
Nach diesen Anweisungen befindet sich folgendes im Speicher:

```
100:"A"REM MODUL 1
110:PAUSE ABS SIN
    (X-Y)
120:"Z"PRINT TIME
130:END
100:"B"REM MODUL 2
110:PAUSE "L**XYZ*
    *G**XYZ**NX"
120:INPUT "X = ";X
130:Y=LOG X
140:PAUSE "LOG = "
    ;Y
150:END
100:"C"REM MODUL 3
110:PRINT "LOG = "
    ;LOG Y
120:END
```

DELETE

```
*************************************************************
*                                                           *
*  Löscht im aktiven Programm-Modul die angegebene Zeile,   *
*  beziehungsweise die angegebenen Zeilen.                  *
*                                                           *
*************************************************************
```

Syntax



Parameter-Beschreibung:

- [1. Zeile] oder [1. Markenname]: Mit diesem Parameter kann angegeben werden, welche Zeile, oder ab welcher Zeile gelöscht werden soll.
  Wird dieser Parameter nicht angegeben, wird ab der ersten Zeile des aktiven Programm-Moduls gelöscht.

- [letzte Zeile] oder [letzter Markenname]: Mit diesem Parameter kann angegeben werden, bis zu welcher Zeile gelöscht werden soll.
  Wird dieser Parameter nicht angegeben, werden alle Zeile bis zum Ende des aktiven Programm-Moduls gelöscht.

Hinweise:

- Mindestens einer der beiden Parameter muss angegeben werden.

ACHTUNG: SOLL EIN GANZES PROGRAMM-MODUL GELOESCHT WERDEN, SO MUSS ERASE VERWENDET WERDEN (siehe Seite 1-11) !

DELETE (Beispiel)

Ausgangssituation:

Die nebenstehenden 4 Programm-Module befinden sich im Speicher:

```
100:"A"REM MODUL 1
110:PAUSE TIME
120:"Z"PRINT TIME
130:END
100:"B"REM MODUL 2
110:PRINT "LOGONX"
120:PRINT "X = ";X
130:REM Y=LOG X
140:PRINT "LOG = "
    ;Y
150:END
100:"C"REM MODUL 3
110:PRINT "LOG = "
    ;LOG Y
120:END
100:"D"REM MODUL 4
200:END
```

Beispiele von DELETE-Anweisungen:

Instruktion	Wirkung
PROGRAM 2 (enter)	Programm-Modul 2 ist nun aktives Modul.
DELETE "B",110 (enter)	Löscht alle Zeilen ab dem Markennamen "B" bis und mit der Zeile 110.
DELETE 130 (enter)	Löscht Zeile 130.
PROGRAM "A" (enter)	Programm-Modul 1 ist nun aktives Modul.
DELETE ,"Z" (enter)	Löscht alle Zeilen ab Modulbeginn bis und mit der Zeile mit dem Markennamen "Z".
PROGRAM 3 (enter)	Programm-Modul 3 ist nun aktives Modul.
DELETE 110, (enter)	Löscht ab Zeile 110 alle Zeilen bis zum Modulende.

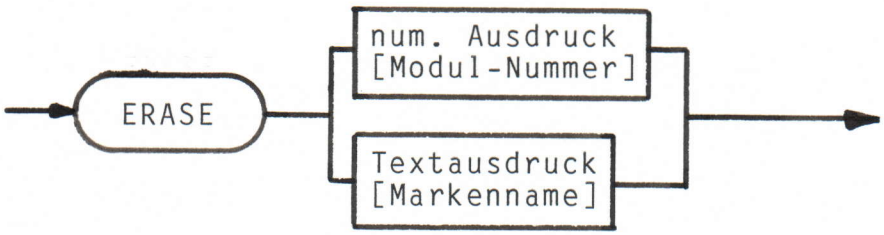
Nach diesen Anweisungen ist nun noch folgendes im Speicher:

```
130:END
120:PRINT "X = ";X
140:PRINT "LOG = "
    ;Y
150:END
100:"C"REM MODUL 3
100:"D"REM MODUL 4
200:END
```

ERASE

```
*************************************************************
*                                                           *
*  Löscht das angegebene Programm-Modul.                    *
*                                                           *
*************************************************************
```

Syntax



Parameter-Beschreibung:

- [Modul-Nummer]: Nummer des zu löschenden Programm-Moduls.
- [Markenname]: Markenname, der irgendwo im zu löschenden Programm-Modul steht.

Hinweis:

- Der freigewordene Speicherplatz steht wieder zur Verfügung.

ACHTUNG: NACH ERASE IST IMMER DAS ERSTE PROGRAMM-MODUL DAS AKTIVE MODUL !

Besondere Fehlermeldungen:

- ERROR 11: Die angegebene [Modul-Nummer] ist zu hoch, das heisst es existieren nicht soviel Module.

* Bemerkung: Beachten Sie bitte im Anhang die Erläuterungen zur modularen Programmierung (Seite A-6). *

ERASE (Beispiel)

Ausgangssituation:

Die nebenstehenden 4 Programm-Module befinden sich im Speicher:

```
100:"A"REM MODUL 1
110:PAUSE TIME
120:"Z"PRINT TIME
130:END
100:"B"REM MODUL 2
110:PRINT "LOGONX"
120:PRINT "X = ";X
130:REM Y=LOG X
140:PRINT "LOG = "
    ;Y
150:END
100:"C"REM MODUL 3
110:PRINT "LOG = "
    ;LOG Y
120:END
100:"D"REM MODUL 4
200:END
```

Nach den beiden Anweisungen:

ERASE 3 (enter)

ERASE "Z" (enter)

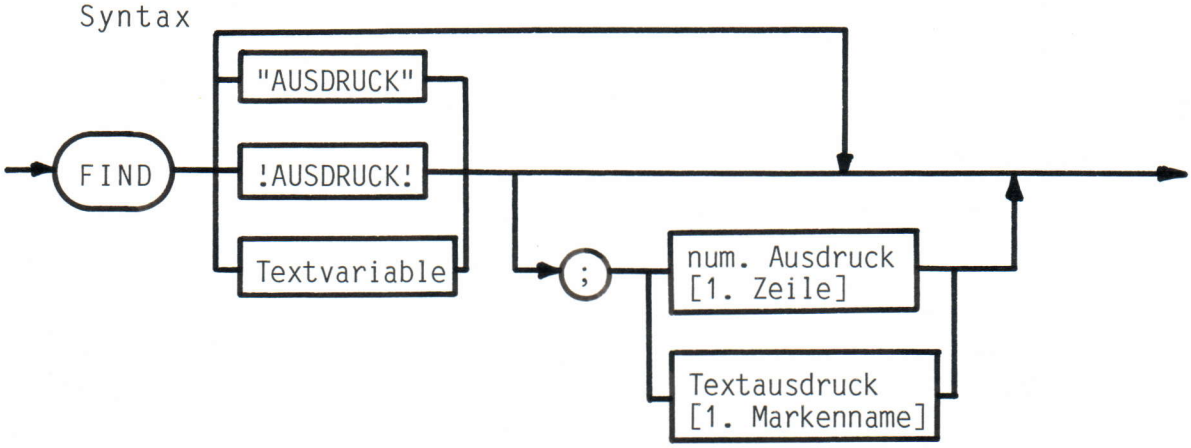
befindet sich noch folgendes im Speicher:

```
100:"B"REM MODUL 2
110:PRINT "LOGONX"
120:PRINT "X = ";X
130:REM Y=LOG X
140:PRINT "LOG = "
    ;Y
150:END
100:"D"REM MODUL 4
200:END
```

FIND

Sucht [AUSDRUCK] im aktiven Programm-Modul und positioniert den Cursor, falls der Ausdruck gefunden wird.
Es kann angegeben werden, ab welcher Zeile der Ausdruck gesucht werden soll.

Syntax



Parameter-Beschreibung:

- [AUSDRUCK]: Ausdruck, der gesucht werden soll.

  [AUSDRUCK] kann ein STRING oder ein BASIC-Schlüsselwort sein (vergleiche Erläuterungen im Anhang, Seite A-4).
  Ein STRING kann auch in einer Textvariablen zwischengespeichert sein.

  Ein Ausdruck zwischen Anführungszeichen (") kann nur innerhalb eines Textes zwischen Anführungszeichen gefunden werden.
  Ein Ausdruck zwischen Ausrufezeichen (!) oder der in einer Textvariablen zwischen gespeichert ist, kann nur BASIC-Schlüsselwörter, Variablen, num. Ausdrücke sowie Ausdrücke nach REM finden.

- [1. Zeile] oder [1. Markenname]: Mit diesem Parameter kann angegeben werden, ab welcher Zeile der [AUSDRUCK] gesucht werden soll.

Hinweise:

- Soll der selbe Ausdruck nochmals gesucht werden, so muss er nicht mehr angegeben werden.
- Wurde nach wiederholtem FIND kein weiterer Ausdruck mehr gefunden, oder wurde CL gedrückt, so kann mit der Taste ZEILE AUFWÄRTS oder ZEILE ABWÄRTS der Cursor auf den zuletzt gefundenen Ausdruck positioniert werden.

FIND (Fortsetzung)

Besondere Fehlermeldung: ERROR 40: Der [AUSDRUCK] muss angegeben werden, weil er im Zwischenspeicher nicht, oder nicht mehr vorhanden ist.

Einschränkungen:

- Der numerische Ausdruck [1. Zeile] darf nur Grundrechenoperationen enthalten. Klammerausdrücke sind verboten !
- Nach FIND ist CONT nicht mehr möglich.
- Anführungszeichen (") können nicht gefunden werden.

FIND (Beispiel)

Die nebenstehenden drei Programm-Module befinden sich im Speicher:

*** Zeile 130 in Modul 2 wurde ***
*** nachträglich mit REM versehen! ***

```
100:"A"REM MODUL 1
110:PAUSE TIME
120:"Z"PRINT TIME
130:END
100:"B"REM MODUL 2
110:PRINT "LOGONX"
120:PRINT "X = ";X
130:REM Y=LOG X
140:PRINT "LOG = "
    ;Y
150:END
100:"C"REM MODUL 3
110:PRINT "LOG = "
    ;LOG Y
120:END
```

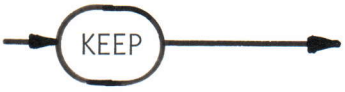
Beispiele von FIND-Anweisungen:

Instruktion	Auswirkung
PROGRAM 2 (enter)	Programm-Modul 2 ist nun aktives Modul.
FIND !PRINT! (enter)	Cursor steht auf Position 1.
CL, FIND (enter)	Cursor steht auf Position 2.
CL, FIND (enter)	Cursor steht auf Position 3.
CL, FIND (enter)	BASIC-Bereitschaftszeichen in der Anzeige: [AUSDRUCK] wurde nicht mehr gefunden.
FIND !!;120 (enter)	Cursor steht auf Position 2.
CL, FIND !LOG! (enter)	Cursor steht auf Position 4.
CL, FIND (enter)	BASIC-Bereitschaftszeichen in der Anzeige: [AUSDRUCK] wurde nicht mehr gefunden.
FIND "LOG" (enter)	Cursor steht auf Position 5.
CL, FIND (enter)	Cursor steht auf Position 6.
CL, A$="X" (enter) FIND A$ (enter)	A$="X" Cursor steht auf Position 7.
CL, FIND !TIME!;"Z" (enter)	ERROR 11: Der Markenname "Z" existiert im aktiven Programm-Modul nicht.
CL, PROGRAM 1 (enter)	Programm-Modul 1 ist nun aktives Modul.
FIND !!;"Z" (enter)	Cursor steht auf Position 8.

KEEP

```
*************************************************************
*                                                           *
*  Ermöglicht das Editieren des eingelesenen Programm-      *
*  teiles nach ERROR 44 bei CLOAD, MERGE oder FLOAD.        *
*  Ausserdem können aus Versehen mit NEW gelöschte          *
*  Programme wieder verfügbar gemacht werden.               *
*                                                           *
*************************************************************
```

Syntax



Hinweise:

- Tritt beim Rückladen eines Programms von einer Cassette mit CLOAD, MERGE oder FLOAD ein ERROR 44 auf, so hat der Rechner einen Fehler im Code festgestellt und den Einlesevorgang abgebrochen.
  Obwohl das BASIC-Programm bis zum Auftreten des Fehlers in den Speicher eingelesen wurde, ist es nicht verfügbar.
  Mit KEEP werden alle fehlerfreien Programmzeilen editier- und ausführbar gemacht.

- Wurde versehentlich der Kommand NEW gegeben, so kann mit KEEP das Programm wieder editier- und ausführbar gemacht werden.
  Wurden mehrere Programm-Module gelöscht, so muss so oft KEEP gemacht werden, wie Module gelöscht wurden.

- Nach KEEP ist das neu editierbar gemachte Programm-Modul aktives Modul.
  Konnte kein neues Modul editierbar gemacht werden, bleibt das vorher aktiv gewesene Modul weiterhin aktiv.

- Nach KEEP wird die erste Zeile des nun aktiven Programm-Moduls angezeigt.

- KEEP wirkt, unabhängig vom Betriebsmodus, ausschliesslich auf BASIC-Programme.

<u>ACHTUNG</u> IST DIE ERSTE ZEILENNUMMER IM ERSTEN PROGRAMM-MODUL GROESSER ALS 255, SO IST DIE ERSTE ZEILENNUMMER INKORREKT !

LINK

```
*************************************************************
*                                                           *
*  Verbindet das aktive Programm-Modul und das nachfolgende *
*  Modul zu einem Programm-Modul.                           *
*                                                           *
*************************************************************
```

Syntax

Hinweis: Das zusammengefügte Programm-Modul wird aktives Programm-Modul.

Besondere Fehlermeldungen:

- ERROR 11: Hinter dem aktiven Programm-Modul befindet sich kein weiteres Modul.
- ERROR 19: Die höchste Zeilennummer des aktiven Programm-Moduls ist höher als die niedrigste Zeilennummer des nachfolgenden Programm-Moduls. (Dies kann mit RENUMBER, siehe Seite 1-21, geändert werden)

Bemerkungen:

- Nach LINK belegt das BASIC-Programm ein Byte weniger im Speicher.
- LINK ist die Umkehrfunktion zu SPLIT (siehe Seite 1-26).

PLAST

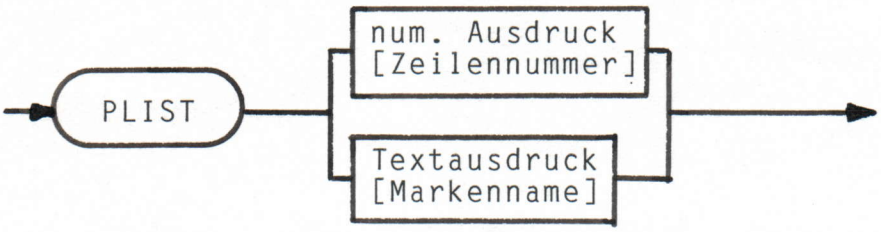
```
*************************************************************
*                                                           *
*  Listet die letzte Zeile des aktiven Programm-Moduls.     *
*                                                           *
*************************************************************
```

Syntax

PLIST

```
*************************************************************
*                                                           *
*  Listet die angegebene Zeile im aktiven Programm-Modul.   *
*                                                           *
*************************************************************
```

Syntax



Parameter-Beschreibung:

- [Zeilennummer]: Die angegebene Zeilennummer im aktiven Programm-Modul wird in die Anzeige gebracht.
- [Markenname]: Die Zeile mit dem angegebenen Markennamen wird in die Anzeige gebracht.

Hinweise:

- Wird nichts angegeben, wird die erste Zeile des aktiven Programm-Moduls in die Anzeige gebracht.
- Im übrigen gelten die gleichen Regeln wie bei LIST (Seite 67 im Sharp-Handbuch).

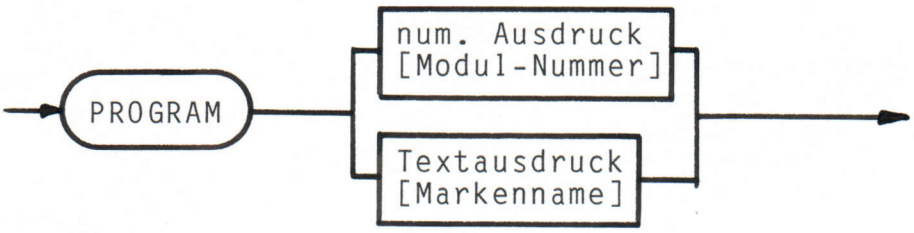
Besondere Fehlermeldung:

- ERROR 11: Es gibt im aktiven Programm-Modul keine Zeilennummer die gleich oder höher als [Zeilennummer] ist, oder der angegebene [Markenname] kommt im aktiven Programm-Modul nicht vor.

PROGRAM

```
*************************************************************
*                                                           *
*  Macht das angegebene Programm-Modul aktiv.               *
*                                                           *
*************************************************************
```

Syntax



Parameter-Beschreibung:

- [Modul-Nummer]: Nummer des zu aktivierenden Programm-Moduls.
- [Markenname]: Markenname der irgendwo im zu aktivierenden Programm-Modul steht.

Besondere Fehlermeldung:

- ERROR 11: Die angegebene [Modul-Nummer] ist zu hoch, das heisst es existieren nicht soviele Module.

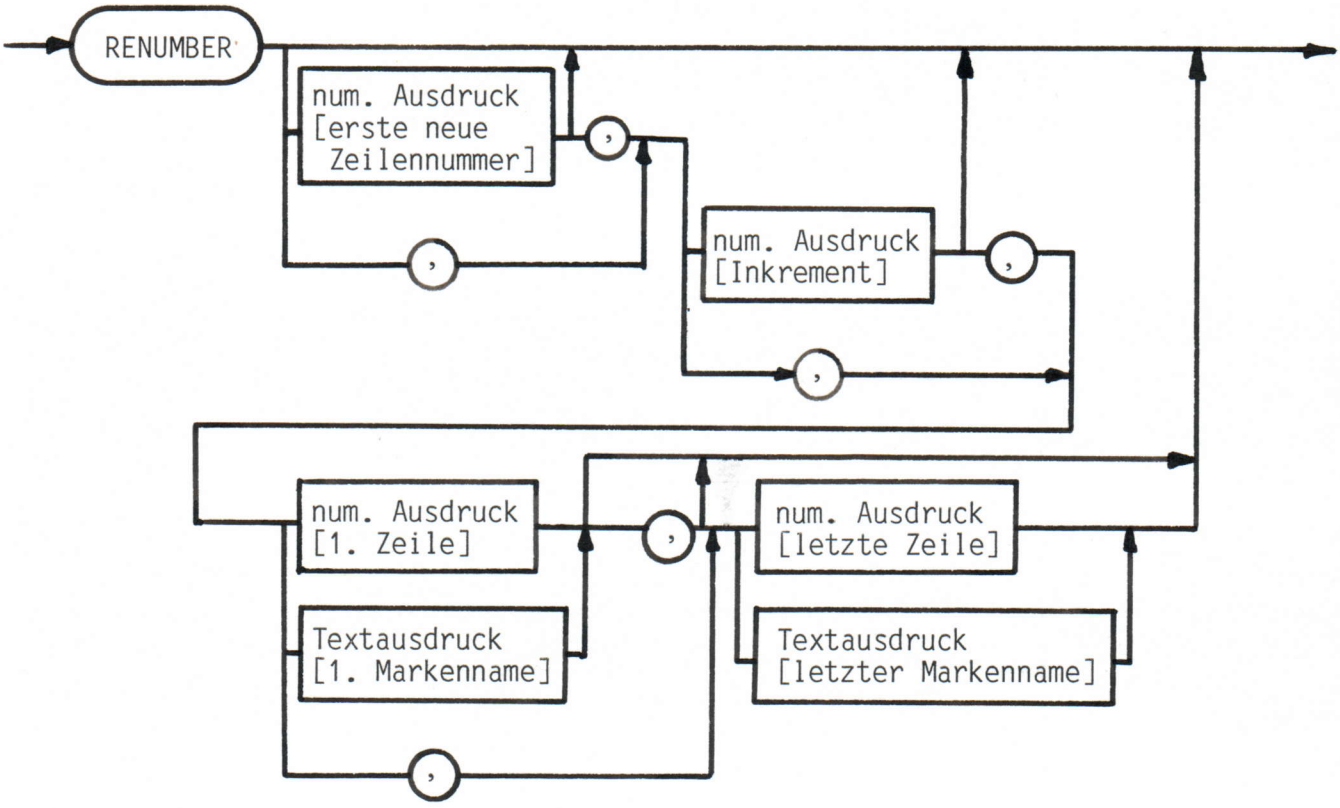
Einschränkung: Es können maximal 255 Programm-Module verwaltet werden.

Bemerkung: Beachten Sie bitte im Anhang die Erläuterungen zur modularen Programmierung (Seite A-6).

RENUMBER

```
*************************************************************
*                                                           *
*  Nummeriert das aktive Programm-Modul neu.                *
*  Die erste neue Zeilennummer, die Erhöhung (Inkrement)    *
*  und der Bereich können angegeben werden.                 *
*                                                           *
*************************************************************
```

Syntax



Parameter-Beschreibung:

- [erste neue Zeilennummer]: Mit diesem Parameter kann die erste neue Zeilennummer angegeben werden.
  Wird nichts angegeben, ist die erste neue Zeilennummer 100.

- [Inkrement]: Um den mit diesem Parameter angegebenen numerischen Ausdruck wird die jeweils folgende Zeilennummer erhöht.
  Wird nichts angegeben, wird die jeweils folgende Zeilennummer um 10 erhöht.

- [1.Zeile] oder [1.Markenname]: Mit diesem Parameter kann angegeben werden, ab welcher Zeile das aktive Programm-Modul neu nummeriert werden soll.
  Wird nichts angegeben, beginnt die Neunummerierung bei der ersten Zeile des aktiven Moduls.

- [letzte Zeile] oder [letzter Markenname]: Mit diesem Parameter kann angegeben werden, bei welcher Zeile oder bei welchem Markennamen die Neunummerierung enden soll.
  Wird nichts angegeben, so endet die Neunummerierung bei der letzten Zeile des aktiven Moduls.

Hinweis

- RENUMBER passt alle Sprungargumente nach GOSUB, GOTO, RESTORE und THEN im gesamten aktiven Programm-Modul, unabhängig von einem allfällig angegebenen Bereich, an die neue Zeilennummerierung an.

- Sprungargumente, die eine Zeile referenzieren, die im aktiven Programm-Modul nicht existiert, aber innerhalb des angegebenen Bereichs liegen müsste, werden 0 gesetzt.

ACHTUNG: SPRUNGARGUMENTE, DIE FUNKTIONEN, VARIABLEN ODER OPERATIONEN ENTHALTEN, WERDEN NICHT VERAENDERT !!!

Besondere Fehlermeldungen:

-	ERROR 17	[erste neue Zeilennummer] oder [Inkrement] nicht numerisch.
-	ERROR 19	Die letzte neue Zeilennummer würde grösser als die folgende bestehende Zeilennummer.
-	ERROR 30	Die neue Zeilennummer würde den Wert 65279 übersteigen.
-	ERROR 39	Inkrement ist 0
-	ERROR 67 IN xx:	Die Zeile xx würde beim Ersetzen des Sprungargumentes die maximal zulässige Länge übersteigen.

- ERROR 177 — Das BASIC-Programm ist nicht in Ordnung.

- ERROR 178 IN xx: — Beim Ersetzen des Sprungargumentes in Zeile xx würden Daten überschrieben, d.h. für das Ersetzen des Sprungargumentes steht nicht mehr genügend Speicherplatz zur Verfügung.

RENUMBER (Beispiel)

Ausgangssituation:

Die nebenstehenden drei Programm-Module befinden sich im Speicher:

*** Modul 2 ist aktives Modul. ***

```
100:"A"REM MODUL 1
110:IF ATHEN 130
120:A=TIME
130:END
100:"B"REM MODUL 2
110:DATA 5,100,120
120:RESTORE 110
130:IF A<0THEN 160
140:ON ERROR GOTO
    200
150:GOSUB 250
160:ON XGOTO 100,B
    ,130,115,170
170:GOTO 120*Z
180:END
200:PRINT "Error"
210:END
250:RETURN
260:END
100:"C"REM MODUL 3
110:GOSUB 200
120:END
200:RETURN
210:END
```

Nach der Anweisung

RENUMBER 10,,,150 (enter)

ist folgendes im Speicher:

```
100:"A"REM MODUL 1
110:IF ATHEN 130
120:A=TIME
130:END
 10:"B"REM MODUL 2
 20:DATA 5,100,120
 30:RESTORE 20
 40:IF A<0THEN 160
 50:ON ERROR GOTO
    200
 60:GOSUB 250
160:ON XGOTO 10,B,
    40,0,170
170:GOTO 120*Z
180:END
200:PRINT "Error"
210:END
250:RETURN
260:END
100:"C"REM MODUL 3
110:GOSUB 200
120:END
200:RETURN
210:END
```

Nach der weiteren Anweisung:

RENUMBER 1E4,100*10,200, (enter)

ist folgendes im Speicher:

```
100:"A"REM MODUL 1
110:IF ATHEN 130
120:A=TIME
130:END
 10:"B"REM MODUL 2
 20:DATA 5,100,120
 30:RESTORE 20
 40:IF A<0THEN 160
 50:ON ERROR GOTO
    10000
 60:GOSUB 12000
160:ON XGOTO 10,B,
    40,0,170
170:GOTO 120*Z
180:END
10000:PRINT "Error
      "
11000:END
12000:RETURN
13000:END
100:"C"REM MODUL 3
110:GOSUB 200
120:END
200:RETURN
210:END
```

Die weitere Anweisung:

RENUMBER 10000,,160,180 (enter)

führt zu einem ERROR 19, weil Zeile 180 zur Zeile 10020 würde und somit grösser wäre als die nächst folgende Zeile.

Nach der weiteren Anweisung:

RENUMBER (enter)

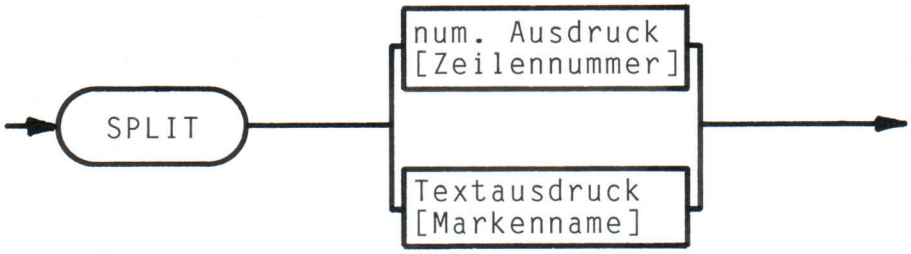
befindet sich folgendes im Speicher:

```
100:"A"REM MODUL 1
110:IF ATHEN 130
120:A=TIME
130:END
100:"B"REM MODUL 2
110:DATA 5,100,120
120:RESTORE 110
130:IF A<0THEN 160
140:ON ERROR GOTO
    190
150:GOSUB 210
160:ON XGOTO 100,B
    ,130,0,170
170:GOTO 120*Z
180:END
190:PRINT "Error"
200:END
210:RETURN
220:END
100:"C"REM MODUL 3
110:GOSUB 200
120:END
200:RETURN
210:END
```

SPLIT

```
*****************************************************************
*                                                               *
*  Unterteilt das aktive Programm-Modul bei der angegebenen     *
*  Zeile in zwei unabhängige Programm-Module.                   *
*                                                               *
*****************************************************************
```

Syntax



Parameter-Beschreibung:
- [Zeilennummer]: Die angegebene Zeilennummer wird zur ersten Zeile im hinteren Programm-Modul.
- [Markenname]: Die Zeile mit dem angegebenen Markennamen wird zur ersten Zeile im hinteren Programm-Modul.

Hinweise:
- Der vordere Teil des unterteilten Programm-Moduls wird zum aktiven Programm-Modul.
- Vorderes Programm-Modul = Teil des unterteilten Moduls mit den niedrigeren Zeilennummern.
- Hinteres Programm-Modul = Teil des unterteilten Moduls mit den höheren Zeilennummern.

Besondere Fehlermeldungen:
- ERROR 11: Es gibt im aktiven Programm-Modul keine Zeilennummer die gleich oder höher als [Zeilennummer] ist, oder der angegebene [Markenname] kommt im aktiven Programm-Modul nicht vor.
- ERROR 22: Zur Unterteilung des Programm-Moduls steht kein Speicherplatz mehr zur Verfügung.

Hinweise:
- Nach SPLIT belegt das BASIC-Programm ein Byte mehr im Speicher.
- SPLIT ist die Umkehrfunktion zu LINK (siehe Seite 1-17).

Liste der Instruktionen
****

TOOL1 bietet zusätzlich folgende Instruktionen an:

BEFEHL	Abkürzung	Kurzbeschreibung	Seite
APPEND	AP.	Eröffnet ein neues Programm-Modul.	1-2
CHANGE	CHAN.	Ersetzt [AUSDRUCK 1] durch [AUSDRUCK 2] im aktiven Programm-Modul.	1-4
DELETE	DEL.	Löscht im aktiven Programm-Modul die angegebene Zeile, bzw. die angegebenen Zeilen.	1-9
ERASE	ERA.	Löscht das angegebene Programm-Modul.	1-11
FIND	FI.	Sucht [AUSDRUCK] im aktiven Programm-Modul.	1-13
KEEP	K.	Ermöglicht das Editieren eines unvollständig eingelesenen oder gelöschten Programm-Moduls.	1-16
LINK	---	Verbindet das aktive Programm-Modul und das nachfolgende Modul zu einem Programm-Modul.	1-17
PLAST	PLA.	Listet die letzte Zeile des aktiven Programm-Moduls.	1-18
PLIST	PL.	Listet die angegebene Zeile im aktiven Programm-Modul.	1-19
PROGRAM	PRO.	Macht das angegebene Programm-Modul aktiv.	1-20
RENUMBER	REN.	Nummeriert das aktive Programm-Modul neu.	1-21
SPLIT	SP.	Unterteilt das aktive Programm-Modul bei der angegebenen Zeile in zwei unabhängige Programm-Module.	1-26