Inhaltsverzeichnis zur Anleitung zum TOOL3
****

Allgemeines zum TOOL3 . . . . . Seite 3-2

CLR . . . . . Seite 3-3

DEC . . . . . Seite 3-5

ERL . . . . . Seite 3-7

ERN . . . . . Seite 3-7

FRC . . . . . Seite 3-9

FRE . . . . . Seite 3-11

HEX$ . . . . . Seite 3-13

INSTR . . . . . Seite 3-15

PGM . . . . . Seite 3-17

PSIZE . . . . . Seite 3-18

PURGE . . . . . Seite 3-19

REDIM . . . . . Seite 3-21

RESUME . . . . . Seite 3-24

STRING$ . . . . . Seite 3-26

SWAP . . . . . Seite 3-28

VKEEP . . . . . Seite 3-30

VLIST . . . . . Seite 3-32

Liste der Funktionen und Instruktionen . . . . . Seite 3-34

Allgemeines zum TOOL3
****

TOOL3 stellt vorallem weitere Funktionen zur Verarbeitung von Variablen zur Verfügung.

Bei der Auswahl der Funktionen wurde auf folgendes Wert gelegt:

1. Die Funktion lässt sich nicht durch ein BASIC-Programm realisieren (z.B. REDIM, PURGE)

2. Die Funktion lässt sich zwar durch ein BASIC-Programm realisieren, braucht aber viel Aufwand (z.B. INSTR, CLR) oder, bei häufiger Anwendung, viel Rechenzeit (z.B. FRC, STRING$).

Variablen-Bezeichnungen in dieser Anleitung:

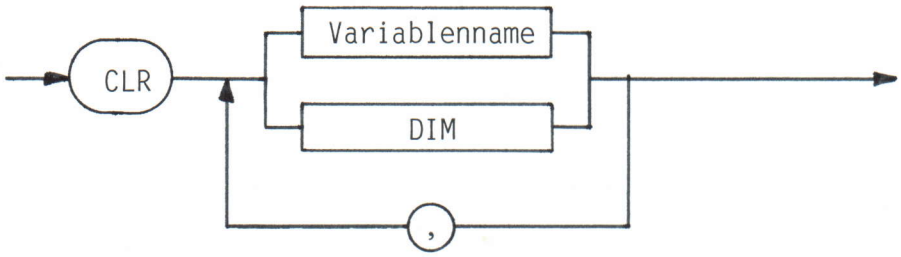
- Festvariablen:	Variablen die nicht im Programmspeicher angelegt sind, sondern die durch das System immer zur Verfügung stehen. Festvariablen heissen A ... Z oder A$ ... Z$ und haben keinen Index.
- 2-Zeichen-Variablen:	Variablen, deren Namen aus 2 Zeichen bestehen und die nicht durch DIM angelegt werden, sondern durch eine Zuweisung. 2-Zeichen-Variablen haben keinen Index.
- DIM-Felder:	DIM-Felder werden durch die Anweisung DIM angelegt, haben einen Namen aus einem oder zwei Zeichen und haben einen Index.

Anmerkung: Das $-Zeichen gehört nicht zum Namen einer Variablen oder eines DIM-Feldes.

CLR

```
*************************************************************
*                                                           *
*  Setzt den Inhalt der angegebenen Variablen, bzw.         *
*  DIM-Felder null.                                         *
*                                                           *
*************************************************************
```

Syntax



Parameter Beschreibung:

- Variablenname: Festvariable A ... Z, A$ ... Z$ oder 2-Zeichen-Variable die null gesetzt werden soll, bzw. DIM-Feld, das null gesetzt werden soll.

- DIM: Wird DIM angegeben, so werden alle 2-Zeichen-Variablen und DIM-Felder null gesetzt.

Hinweise:

- Mit @(*) werden A ... Z, mit @$(*) A$ ... Z$ null gesetzt.

- Es können nur ganze DIM-Felder null gesetzt werden, aber keine einzelnen Elemente von Feldern.

- Es können nur bestehende 2-Zeichen-Variablen, bzw. DIM-Felder null gesetzt werden !

- TIME kann mit CLR nicht null gesetzt werden.

- CLR kann manuell oder programmkontrolliert ausgeführt werden.

CLR (Beispiele)

Beispiel 1

Die Anweisung

```
CLR DIM,@(*),@$(*)
```

setzt alle Festspeicher, 2-Zeichen-Variablen und DIM-Felder null.

Beispiel 2

Die Programmanweisung

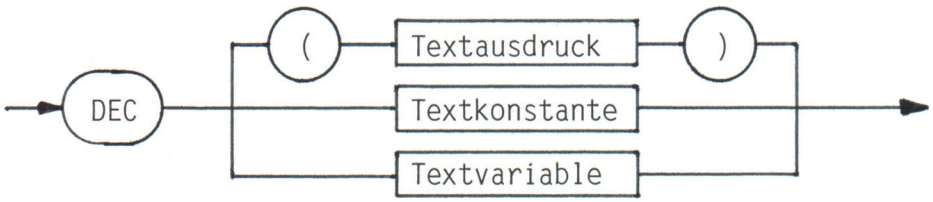
```
100: CLR A,B,XX$,T(*),IN$(*),AA,X,Z$
```

setzt die Festspeicher A,B,X und Z$, die 2-Zeichen-Variablen AA und XX$, sowie die DIM-Felder IN$(*) und T(*) null.

DEC

```
*************************************************************
*                                                           *
*  Die Funktion DEC berechnet den Dezimalwert einer als     *
*  Zeichenfolge angegebenen hexadezimalen Zahl.             *
*                                                           *
*************************************************************
```

Syntax



Ergebnis: numerischer Wert

Funktionsbeschreibung:

- Die Funktion berechnet den Dezimalwert einer hexadezimalen Zahl, ähnlich wie die Funktion VAL (siehe Handbuch zum PC-1500, Seite 92).
- Die Zeichenfolge wird aber als Hexadezimalzahl aufgefasst wie nach &.
  Im Gegensatz zu & kann die Zeichenfolge aber auch in einer Textvariablen gespeichert sein oder einen Textausdruck darstellen.
- DEC ist die Umkehrfunktion zu HEX$ (siehe Seite 3-13).
- Die Konvertierung beginnt mit dem ersten Zeichen des Textausdruckes und wird abgebrochen, wenn ein Zeichen auftritt, das nicht in der Liste (0...9,A...F) enthalten ist.

Funktionsbereich:

- Die Funktion verarbeitet nur vorzeichenlose Textausdrücke von 0...FFFF (hexadezimal).
- Vornullen sind zugelassen, Zwischenräume (SPACE) nicht.

Hinweis:

- DEC kann manuell oder programmkontrolliert ausgeführt werden.

DEC (Beispiele)

Manuelles Rechnen (Protokoll-Ausdruck):

```
DEC "3E8"
                1000

DEC "00000F1"
                 241

DEC ("5A"+"C3")
               23235
```

Programm-Beispiel:

```
100:"Z"REM DEC
110:A$="5AC3"
120:A=DEC (MID$ ("
    * 5DC *",3,3))
130:LPRINT A
140:LPRINT DEC A$
150:END
```

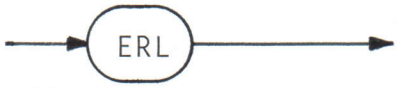
Resultat des obigen Programmablaufs:

```
                1500
               23235
```

ERL

```
*************************************************************
*                                                           *
*  Liefert die Zeilennummer, in welcher zuletzt ein ERROR   *
*  auftrat.                                                 *
*                                                           *
*************************************************************
```

Syntax

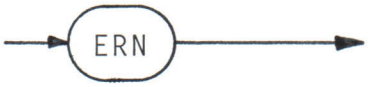


Ergebnis: numerischer Wert

ERN

```
*************************************************************
*                                                           *
*  Liefert die Nummer des zuletzt aufgetretenen ERRORs.     *
*                                                           *
*************************************************************
```

Syntax



Ergebnis: numerischer Wert

Hinweise:
- Die Funktionen ERL und ERN des TOOL3 sind vollständig kompatibel mit denjenigen der Schnittstelle SHARP CE-158.
- Die Werte für ERL und ERN werden nur mit Clear All (CA) oder bei Programmstart mit RUN null gesetzt. Bei Programmstart mit DEF Markenname bleiben die letzten Werte erhalten, bis ein neuer ERROR auftritt !
- ERL und ERN können manuell oder programmkontrolliert ausgeführt werden.

ERL / ERN (Beispiele)

Manuelle Abfrage (Protokoll-Ausdruck):

```
SIN (10,20)
ERROR 18

ERL
                    0

ERN
                   18
```

Programm-Beispiel:

```
 10:"Z"REM ERL,ERN
 20:ON ERROR GOTO
    "ERR"
 30:DIM A(1)
 40:FOR I=0TO 5
 50:A(I)=I:NEXT I
 60:END
100:"ERR"LPRINT "*
     error";ERN ;"
     in";ERL ;" *"
    :END
```

Resultat des obigen Programm-Ablaufs:

1. Ausführung:

```
* error 9 in 50 *
```

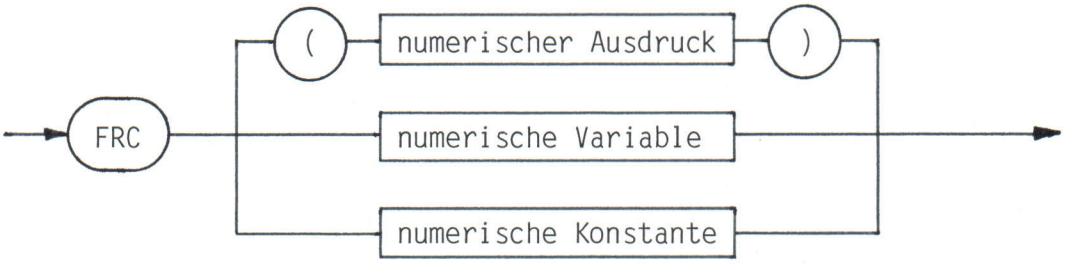
2. Ausführung:

```
* error 5 in 30 *
```

FRC

```
*************************************************************
*                                                           *
*  Liefert den Nachkommateil eines numerischen Ausdrucks.   *
*                                                           *
*************************************************************
```

Syntax



Ergebnis: numerischer Wert

Hinweise: - FRC ist ein Ergänzung zur Funktion INT: FRC X = X - INT X.

- FRC kann manuell oder programmkontrolliert ausgeführt werden.

FRC (Beispiele)

Manuelles Rechnen (Protokoll-Ausdruck):

```
FRC π
   1.415926536E-01


1E6*FRC (TIME /100
)
                225057
```

Programm-Beispiel:

```
10:"Z"REM FRC
20:FOR I=0TO 9
30:IF FRC (I/2)>0
   THEN 50
40:LPRINT USING "
   ###";I;
50:NEXT I:LPRINT
   :END
```

Resultat des obigen Programm-Ablaufs:

```
0  2  4  6  8
```

FRE

```
*************************************************************
*                                                           *
*  Liefert die Anzahl freier Bytes unter Berücksichtigung   *
*  der 2-Zeichen-Variablen und der DIM-Felder.              *
*                                                           *
*************************************************************
```

Syntax

Ergebnis: numerischer Wert

Hinweise:
- FRE hat grundsätzlich die selbe Funktion wie MEM, berücksichtigt aber im gegensatz zu MEM auch die im Programmspeicher angelegten Variablen.
- FRE kann manuell oder programmkontrolliert ausgeführt werden.

FRE (Beispiele)

Manuelle Abfrage (Protokoll-Ausdruck):

```
FRE
                    1850

DIM X(5)

FRE
                    1795
```

```
MEM
                    1850

DIM X(5)

MEM
                    1850
```

Programm-Beispiel:

```
10:"A"REM FRE
20:DIM XX$(0)*1
30:USING "#####"
40:LPRINT FRE ;"
   Bytes frei"
50:N=FRE /10
60:REDIM XX$(N-1,
   9)*1
70:LPRINT FRE ;"
   Bytes frei"
80:END
```

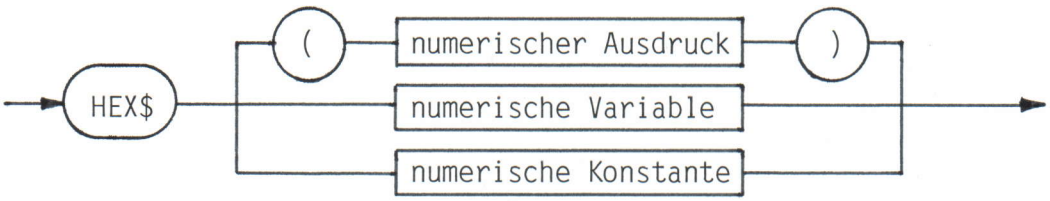
Resultat des obigen Programm-Ablaufs:

```
1724 Bytes frei
   5 Bytes frei
```

HEX$

****
* Die Funktion HEX$ wandelt den Wert eines numerischen *
* Ausdrucks in die hexadezimale Zeichenfolge um. *
****

Syntax



Ergebnis: Text

Funktionsbeschreibung:

- Die Funktion wandelt das Ergebnis des numerischen Ausdrucks in die hexadezimale Zeichenfolge um, ähnlich wie STR$ (siehe Handbuch zum PC-1500, Seite 91).
  Die Zeichenfolge ist aber eine Hexadezimalzahl.
- HEX$ ist die Umkehrfunktion zu DEC (siehe Seite 3-5).

Funktionsbereich:

- Die Funktion verarbeitet nur Ergebnisse von 0 ... 65535 (dezimal).
- Die Funktion verarbeitet nur den ganzzahligen Anteil des Ergebnisses.

Hinweise:

- Die resultierende Zeichenfolge besteht immer aus zwei oder vier Hexadezimal-Ziffern.
- HEX$ kann manuell oder programmkontrolliert ausgeführt werden.

HEX$ (Beispiele)

Manuelles Rechnen (Protokoll-Ausdruck):

```
HEX$ 1000
03E8


HEX$ 241
F1


HEX$ (20*√10)
3F
```

Programm-Beispiel:

```
10:"="REM HEX$
20:FOR I=0TO 100
   STEP 10
30:LPRINT USING "
   ####";I;"  -->
     &";HEX$ I
40:NEXT I
50:END
```

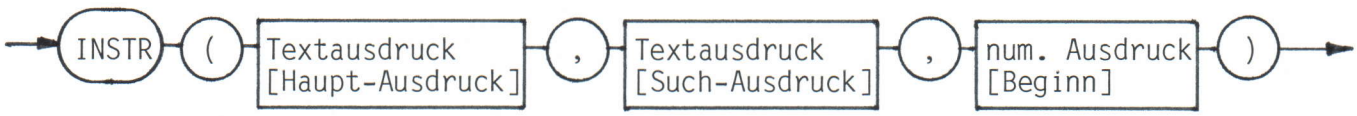
Resultat des obigen Programmablaufs:

```
  0  -->  &00
 10  -->  &0A
 20  -->  &14
 30  -->  &1E
 40  -->  &28
 50  -->  &32
 60  -->  &3C
 70  -->  &46
 80  -->  &50
 90  -->  &5A
100  -->  &64
```

INSTR

```
*************************************************************
*                                                           *
*  Liefert die Position des Beginns des Such-Ausdruckes     *
*  im Haupt-Ausdruck. Gesucht wird ab der angegebenen       *
*  Position im Haupt-Ausdruck.                              *
*                                                           *
*************************************************************
```

Syntax



Ergebnis: numerischer Wert

Parameter Beschreibung:

- [Haupt-Ausdruck]: Textausdruck, in welchem gesucht werden soll.
- [Such-Ausdruck]: Textausdruck, der im Haupt-Ausdruck gesucht werden soll.
- [Beginn]: Position, ab welcher der Such-Ausdruck im Haupt-Ausdruck gesucht werden soll.

Funktionsbeschreibung:

- Wird der Such-Ausdruck im Haupt-Ausdruck gefunden, so ist das Ergebnis die Position des ersten Zeichens des Such-Ausdrucks im Haupt-Ausdruck (1 ... 79).
- Wird der Such-Ausdruck nicht gefunden, so ist das Ergebnis 0.
- Ist der Haupt-Ausdruck ein Leer-String (""), so ist das Ergebnis immer 0.
- Ist der Such-Ausdruck ein Leer-String (""), und Beginn kleiner als LEN (Haupt-Ausdruck), so ist das Ergebnis gleich Beginn.
  Andernfalls ist das Ergebnis 0.

Hinweis:

- INSTR kann manuell oder programmkontrolliert ausgeführt werden.

<u>Einschränkung:</u>

- Haupt- und Such-Ausdruck dürfen zusammen maximal 80 Zeichen umfassen.

INSTR (Beispiele)

Ergebnisse von INSTR:

```
T$
*** ABCDACAB ***

INSTR (T$,"AB",1)
                5

INSTR (T$,"AB",6)
               11

INSTR (T$,"AB",12)
                0

INSTR (T$,"XYZ",1)
                0

INSTR (T$,"",3)
                3

INSTR (T$,"",18)
                0
```

Programm-Beispiel:

```
 10:"="REM INSTR
 20:INPUT "HEX-ZAH
    L = ";H$
 30:F=0
 40:FOR I=1TO LEN
    H$
 50:IF INSTR ("012
    3456789ABCDEF"
    ,MID$ (H$,I,1)
    ,1)=0LET F=1
 60:NEXT I
 70:LPRINT H$
 80:IF FLPRINT " S
    yntax-Fehler":
    GOTO 100
 90:LPRINT " Einga
    be gut"
100:LF 1:GOTO 10
```

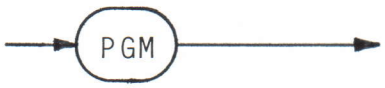
Resultat des obigen Programmablaufs:

```
1238CB
 Eingabe gut

01*325
 Syntax-Fehler

FFFFBC1
 Eingabe gut

23654$0001&12
 Syntax-Fehler
```

PGM

```
*************************************************************
*                                                           *
*  Liefert die Nummer des aktiven Programm-Moduls.          *
*                                                           *
*************************************************************
```

Syntax



Ergebnis: numerischer Wert

Hinweise:
- Befindet sich kein Programm im Speicher, so ist das Ergebnis 0.
- PGM kann manuell oder programmkontrolliert ausgeführt werden.
- Mit ERASE PGM wird das aktive Programm-Modul gelöscht (Nur möglich, wenn Sie auch im Besitz des TOOL1 sind, siehe Anleitung Seite 1-11).

PSIZE

```
*************************************************************
*                                                           *
*  Liefert die Anzahl Bytes, die das aktive Programm-       *
*  Modul im Speicher belegt.                                *
*                                                           *
*************************************************************
```

Syntax

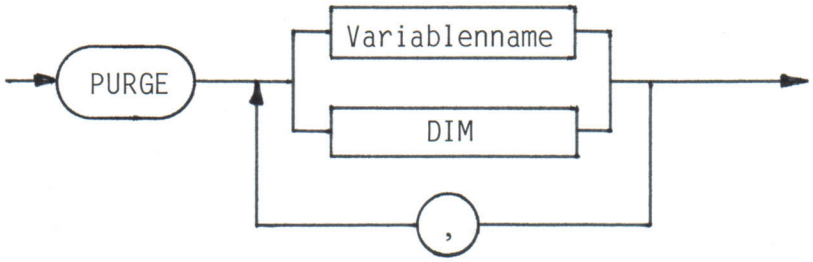
Ergebnis: numerischer Wert

Hinweise:
- Befindet sich kein Programm im Speicher, so ist das Ergebnis 0.
- PSIZE kann manuell oder programmkontrolliert ausgeführt werden.

PURGE

```
*************************************************************
*                                                           *
*  Löscht die angegebenen 2-Zeichen-Variablen, bzw.         *
*  DIM-Felder aus dem Speicher.                             *
*                                                           *
*************************************************************
```

Syntax



Parameter Beschreibung:

- Variablenname: 2-Zeichen-Variable die, bzw. DIM-Feld, das aus dem Speicher gelöscht werden soll.

- DIM: Wird DIM angegeben, so werden alle 2-Zeichen-Variablen und DIM-Felder aus dem Speicher gelöscht.
  PURGE DIM wirkt ähnlich wie CLEAR, lässt aber im Gegensatz zu CLEAR die Festvariablen unverändert.

Hinweise:

- Es können nur ganze DIM-Felder aus dem Speicher gelöscht werden.

- Es können nur bestehende 2-Zeichen-Variablen, bzw. DIM-Felder gelöscht werden !

- PURGE kann manuell oder programmkontrolliert ausgeführt werden.

PURGE (Beispiele)

Beispiel 1

Mit der Anweisung

PURGE DIM

werden alle 2-Zeichen-Variablen und DIM-Felder aus dem Speicher gelöscht, die Festvariablen A ... Z und A$ ... Z$ dagegen bleiben erhalten:

```
A=100
                    100

PURGE DIM


A
                    100
```

Beispiel 2

Programm-Beispiel

```
10:"="REM PURGE
20:DIM TX$(4,3)*5
   ,G(3,2),Z(9)
30:XX=2:AB=π:IN$=
   "TEST"
40:LPRINT FRE ;"
   Bytes frei"
50:PURGE AB,TX$(*
   ),IN$,Z(*)
60:LPRINT FRE ;"
   Bytes frei"
70:END
```

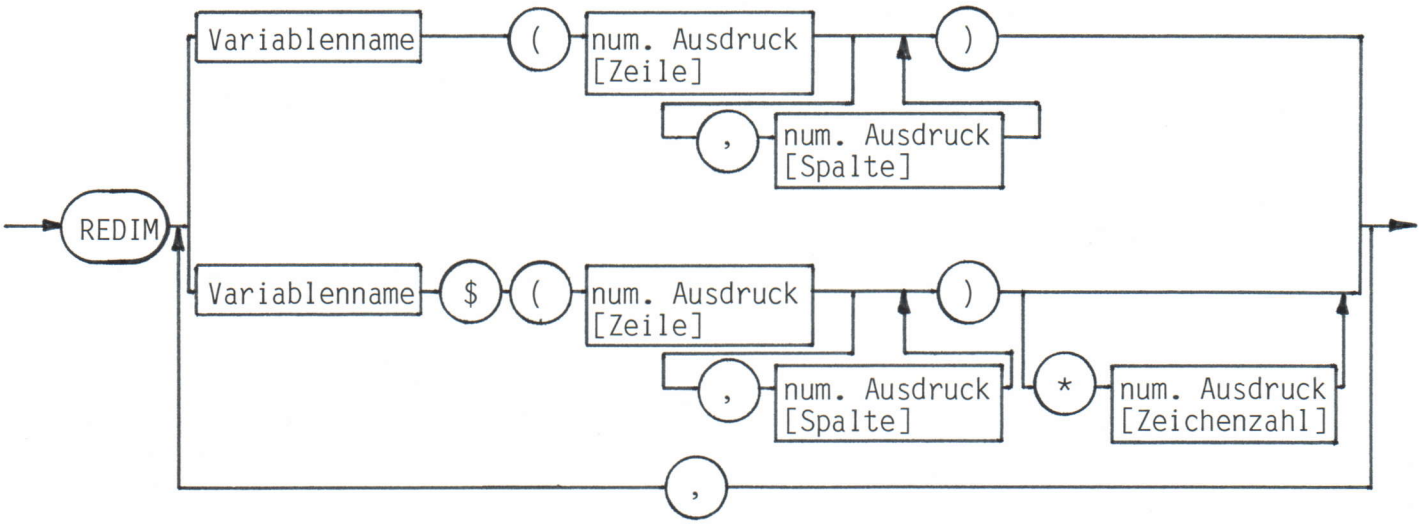
Resultat des obigen Programmablaufs:

```
1359 Bytes frei
1591 Bytes frei
```

REDIM

```
*************************************************************
*                                                           *
*  Dimensioniert bereits bestehende DIM-Felder neu, wobei   *
*  die Felderinhalte soweit als möglich erhalten bleiben.   *
*                                                           *
*************************************************************
```

Syntax



Parameter Beschreibung:

- Variablenname: Name des zu ändernden DIM-Feldes.
- [Zeile]: Anzahl Elemente minus 1 in der einen Ausdehnung.
- [Spalte]: Anzahl Elemente minus 1 in der andern Ausdehnung.
  Wird nichts angegeben, so wird das Feld eindimensional.
- [Zeichenzahl]: Anzahl Zeichen, die ein Element eines Textfeldes maximal aufnehmen kann.
  Wird nichts angegeben, so bietet ein Element 16 Zeichen Platz.

Hinweise:

- Es gelten grundsätzlich die gleichen Regeln wie bei DIM, siehe Anleitung zum PC-1500, Seiten 60 ... 63
  Es können aber nur bestehende DIM-Felder verändert werden !
- Es können aus eindimensionalen Feldern zweidimensionale Felder gemacht werden und umgekehrt.
- REDIM kann manuell oder programmkontrolliert ausgeführt werden.

Beschreibung der Wirkung von REDIM:

a) Vergrössern einer Ausdehnung (Zeile oder Spalte):

Die neuen Elemente werden mit höheren Indizes angefügt und null gesetzt, d.h. die neuen Elemente werden gleich initialisiert wie bei DIM, während bei den bereits bestehenden Elementen der Inhalt unverändert bleibt.

b) Verkleinern einer Ausdehnung (Zeile oder Spalte):

Die Elemente mit den höheren Indizes gehen verloren, während bei den bestehenden Elementen der Inhalt erhalten bleibt.

c) Vergrössern der Elemente (nur Textfelder):

Werden die einzelnen Elemente eines Textfeldes vergrössert, so bleiben die Inhalte der einzelnen Elemente unverändert, ein Element kann nun aber mehr Zeichen aufnehmen. Die Elemente werden mit dem ASCII-Zeichen Null aufgefüllt, so dass die tatsächliche Länge der Elemente unverändert bleibt.

d) Verkleinern der Elemente (nur Textfelder):

Werden die Elemente eines Textfeldes verkleinert, so werden sie rechts abgeschnitten.

Die oben genannten Veränderungen können - soweit als möglich - beliebig kombiniert werden !
Soll nur ein Teil der DIM-Spezifikationen eines Feldes geändert werden, so müssen trotzdem alle Angaben gemacht werden.

Speziallfall 2-Zeichen-Textvariablen:

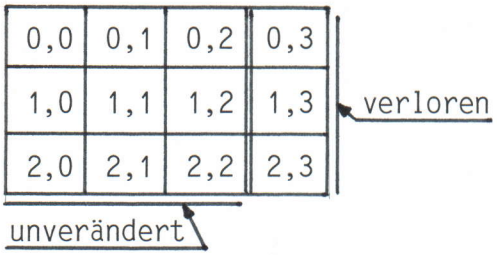
Bei 2-Zeichen-Textvariablen kann die Länge der Variablen ebenfalls verändert werden, d.h auch diese Variablen können nun in einer Länge von 1 ... 80 Zeichen angelegt werden !

ACHTUNG: Wenn bei sehr grossen Textfeldern die Länge der Elemente verändert wird, können Verarbeitungszeiten bis zu einer Minute und mehr entstehen !

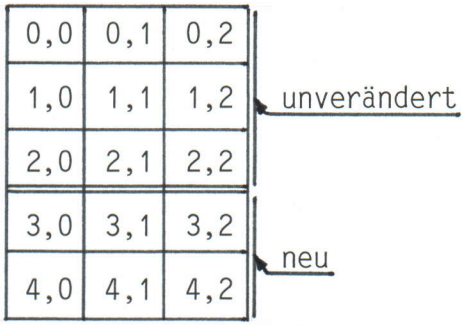
REDIM (Beispiele)

Beispiel 1

DIM AB(2,3)

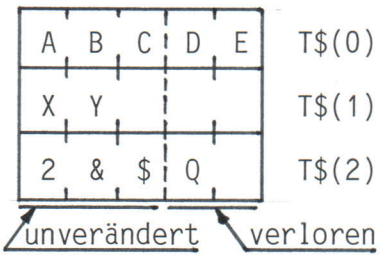


REDIM AB(4,2)



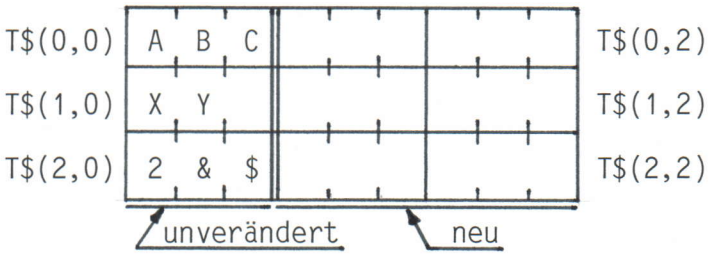
Beispiel 2

DIM T$(2)*5



LEN T$(0) = 5
LEN T$(1) = 2
LEN T$(2) = 4

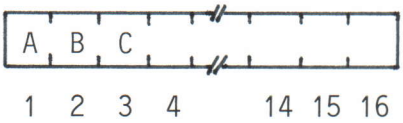
REDIM T$(2,2)*3



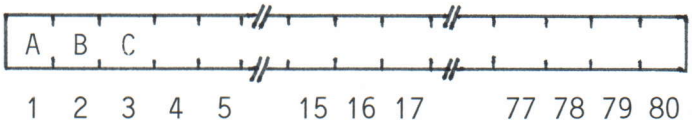
LEN T$(0,0) = 3
LEN T$(1,0) = 2
LEN T$(2,0) = 3
LEN T$(1,0) ... T$(2,2) = 0

Beispiel 3

IN$ "ABC"



REDIM IN$*80



RESUME

```
*************************************************************
*                                                           *
*  Ermöglicht, zusammen mit ON ERROR, die Fortsetzung       *
*  des Programmablaufs nach einem ERROR mit der nächst-     *
*  folgenden Programm-Anweisung.                            *
*                                                           *
*************************************************************
```

Syntax

Hinweise: - RESUME ist eine Programm-Anweisung und kann nicht manuell ausgeführt werden.

- Als Ende einer Anweisung gilt das Zeilenende oder ein Doppelpunkt (:).
  Komma (,) gilt nicht als Anweisungsende !
  Dies bedeutet, dass in einer Reihe von Zuweisungen, die durch Kommas getrennt sind, im Falle eines Fehlers die restlichen Zuweisungen übersprungen werden.

RESUME (Beispiele)

Beispiel 1

```
 10:"="REM RESUME
 20:ON ERROR GOTO
    100
 30:FOR X=-4TO 4
 40:LPRINT USING "
    ###.###";X;TAB
    10;(3*X-4)/(1-
    X^2)
 50:NEXT X:END
100:IF ERN <>38END
110:LPRINT TAB 10;
    "***.***":
    RESUME
```

Resultat des obigen Programm-Ablaufs:

```
-4.000      1.066
-3.000      1.625
-2.000      3.333
-1.000    ***.***
 0.000     -4.000
 1.000    ***.***
 2.000     -0.666
 3.000     -0.625
 4.000     -0.533
```

Beispiel 2

```
 10:"="REM RESUME
 20:ON ERROR GOTO
    "ERR"
 30:CLEAR
 40:A=1,B="A",C=3,
    D=4
 50:LPRINT USING "
    ###";A;B;C;D
 60:END
100:"ERR"LPRINT "e
    rror";ERN ;" i
    n";ERL :RESUME
```

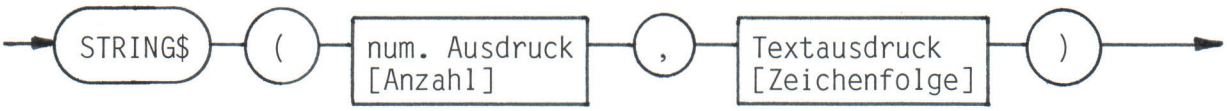
Resultat des obigen Programm-Ablaufs:

```
error  7 in 40
  1  0  0  0
```

STRING$

```
*************************************************************
*                                                           *
*  Liefert ein beliebiges Zeichen oder eine beliebige       *
*  Zeichenfolge so oft wie angegeben.                       *
*                                                           *
*************************************************************
```

Syntax



Ergebnis: Text

Parameter-Beschreibung:
- [Anzahl]: Gibt an, wie oft die Zeichenfolge ausgegeben werden soll.
- [Zeichenfolge]: Einzel-Zeichen oder Zeichenfolge, die ausgegeben werden soll.

Hinweise:
- Ist Anzahl 0, so ist das Ergebnis ein Nullstring ("").
- Das Ergebnis darf maximal 80 Zeichen umfassen.
- STRING$ kann manuell oder programmkontrolliert ausgeführt werden.

STRING$ (Beispiele)

Beispiel 1

```
 10:"="REM STRING$
 20:INPUT "Titel =
     ";T$
 30:LPRINT STRING$
    (18,"*")
 40:N=INT ((16-LEN
    T$)/2)
 50:LPRINT "*";
    STRING$ (N," "
    );
 60:LPRINT T$;
    STRING$ (16-
    LEN T$-N," ");
    "*"
 70:LPRINT STRING$
    (18,"*")
 80:END
```

Resultat des obigen Programm-Ablaufs:

```
******************
*    TRAMsoft    *
******************

******************
*    PC-1500     *
******************
```

Beispiel 2

```
 10:"="REM STRING$
 20:FOR I=0TO 8
 30:IF FRC (I/2)=0
    GOSUB 100:GOTO
    50
 40:GOSUB 200
 50:NEXT I:END
100:LPRINT STRING$
    (5,"+--");"+"
110:RETURN
200:LPRINT STRING$
    (5,"!  ");"!"
210:RETURN
```

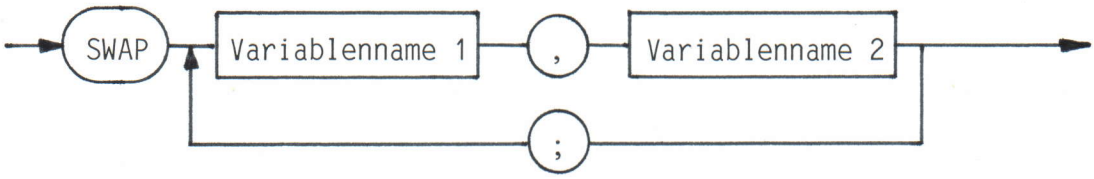
Resultat des obigen Programm-Ablaufs:

```
+--+--+--+--+--+
!  !  !  !  !  !
+--+--+--+--+--+
!  !  !  !  !  !
+--+--+--+--+--+
!  !  !  !  !  !
+--+--+--+--+--+
!  !  !  !  !  !
+--+--+--+--+--+
```

SWAP

```
*************************************************************
*                                                           *
*  Vertauscht die Inhalte der angegebenen Variablen.        *
*                                                           *
*************************************************************
```

Syntax



Parameter-Beschreibung: - Variablenname 1 und Variablenname 2: Bezeichnen die Variablen, deren Inhalte vertauscht werden sollen.
Anstelle einer Festvariablen kann auch eine 2-Zeichen-Variable oder ein Element eines DIM-Feldes stehen.
Es können aber nur einzelne Elemente eines DIM-Feldes vertauscht werden und nicht ganze DIM-Felder.

Hinweise:
- Es können nur Variablen der gleichen Datenart vertauscht werden.
- Es können nur Inhalte von bereits bestehenden Variablen ausgetauscht werden !
- Werden Textvariablen unterschiedlicher Länge miteinander vertauscht, so wird der Inhalt der grösseren Variablen wenn nötig rechts abgeschnitten.
- Der Inhalt von TIME kann mit SWAP nicht ausgetauscht werden.
- SWAP kann manuell oder programmkontrolliert ausgeführt werden.

SWAP (Beispiele)

Manuelle Ausführung (Protokoll-Ausdruck):

```
A=1, B=2, C=3, D=4
                    1
                    2
                    3
                    4
SWAP A, B; C, D

A
                    2

B
                    1

C
                    4

D
                    3
```

Programm-Beispiel:

```
 10:"="REM SWAP
 20:E=3:LF 1:
    LPRINT "**  Ra
    ngliste:  **":
    LF 1
 30:FOR I=0TO 3
 40:FOR J=0TO E
 50:IF PZ(J)<PZ(J+
    1)SWAP PZ(J),P
    Z(J+1);NA$(J),
    NA$(J+1)
 60:NEXT J
 70:E=E-1:NEXT I
 80:FOR I=0TO 4
 90:LPRINT USING "
    ###";I+1;". ";
    NA$(I);TAB 15;
    PZ(I)
100:NEXT I:END
```

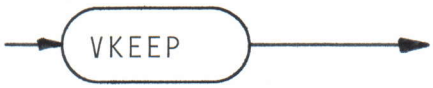
Resultat des obigen Programm-Ablaufs:

```
Peter            14
Walter           20
Willi             9
Urs              19
Martin           12

**  Rangliste:   **

  1. Walter      20
  2. Urs         19
  3. Peter       14
  4. Martin      12
  5. Willi        9
```

VKEEP

```
*************************************************************
*                                                           *
*  Ermöglicht die Adressierung von 2-Zeichen-Variablen      *
*  und DIM-Feldern nach CLEAR, NEW, PURGE DIM oder RUN.     *
*                                                           *
*************************************************************
```

Syntax



Hinweise:

- Konnte eine 2-Zeichen-Varable oder ein DIM-Feld adressierbar gemacht werden, so wird der Name, die Datenart, bei DIM-Feldern die Dimension(en) und, falls es sich um eine Textvariable handelt, die Länge eines Elementes, angezeigt. Andernfalls erscheint das BASIC-Bereitschafts-Zeichen.

- Ist der PRINT-Schalter am CE-150 in der Stellung P, so werden die oben genannten Angaben vom Plotter ausgegeben.

- Mit VKEEP wird jeweils nur eine Variable adressierbar gemacht. Sollen mehrere Variablen adressierbar gemacht werden, so muss so oft VKEEP gegeben werden, wie Variablen adressierbar gemacht werden sollen.

- Der Inhalt der adressierbar gemachten Variablen bleibt unverändert !

- Es ist möglich, dass nach VKEEP ein Variablenname angezeigt wird, der gar nicht angelegt wurde. Durch erneutes VKEEP kann die vermisste Variable aber immer gefunden werden. Niemals mit PURGE diese Variable löschen, da sonst alle weiteren Variablen verloren sind !

- Die Variablen werden in der selben Reihenfolge adressierbar gemacht, wie sie angelegt wurden.

- Variablen, die mit PURGE gelöscht wurden, können mit VKEEP nicht wieder adressierbar gemacht werden, ausgenommen die zuletzt angelegte Variable und nach PURGE DIM.

- VKEEP kann nur manuell ausgeführt werden.

VKEEP (Beispiele)

Es werden die folgenden Variablen angelegt:

```
DIM A(3)

XX=π
        3.141592654

DIM Z$(5,8)*2

TI=TIME
         61813.5131
```

Danach werden die folgenden Zuweisungen gemacht:

```
Z$(2,7)="XP"
XP

Z$(2,8)="Q"
Q

Z$(3,0)="DE"
DE

Z$(3,1)="FG"
FG
```

Mit CLEAR werden die Variablen gelöscht.
Mit VKEEP können sie wieder adressierbar gemacht werden:

```
VKEEP
A(3)

VKEEP
XX

VKEEP
PQ*71

VKEEP
Z$(5,8)*2

VKEEP
TI

VKEEP
>
```

Dieser Variablenname wird angezeigt auf Grund der Inhalte des Feldes Z$(*)

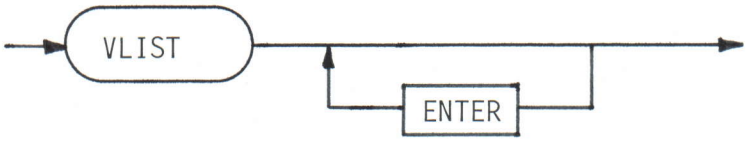
Die Variablen stehen wieder mit unverändertem Inhalt zur Verfügung:

```
XX
        3.141592654

TI
         61813.5131

Z$(2,7)
XP
```

VLIST

```
*************************************************************
*                                                           *
*  Listet 2-Zeichen-Variablen und DIM-Felder mit Namen,     *
*  Datenart, Dimension(en) und Länge wahlweise in die       *
*  Anzeige oder auf dem Plotter CE-150.                     *
*                                                           *
*************************************************************
```

Syntax



Hinweise:

- VLIST bringt den Namen, die Datenart, bei DIM-Felder die Dimension(en) und, falls es sich um eine Textvariable handelt, die Länge eines Elementes, in die Anzeige.

- Befinden sich keine Variablen im Speicher oder wurden alle Variablen gelistet, so erscheint das BASIC-Bereitschafts-Zeichen in der Anzeige.

- Ist der PRINT-Schalter des CE-150 in der Stellung P, so werden alle oben genannten Angaben vom Plotter ausgelistet.
  Ist CSIZE grösser als 2, so wird CSIZE auf 2 reduziert.
  Der Schreibvorgang kann mit BREAK sofort abgebrochen werden.
  Wird der PRINT-Schalter während dem Schreiben zurückgestellt, so wird die Zeile fertig geschrieben und die nächste Variable in die Anzeige gebracht.
  Wird der PRINT-Schalter in die Stellung P gebracht, während dem eine Variable in der Anzeige steht, so werden die Angaben ab der nächsten Variablen vom Plotter ausgegeben, sobald ENTER gedrückt wird.

- Werden die Angaben in die Anzeige gelistet, so kann die nächste Variable durch drücken der ENTER-Taste angezeigt werden.

- Durch drücken einer anderen Taste als ENTER wird der Listvorgang beendet.
  Die 4 Tasten zur Cursor-Bewegung haben keine direkte Auswirkung, beenden den Listvorgang aber ebenfalls.

- Die Variablen werden in der umgekehrten Reihenfolge angezeigt, als sie angelegt wurden, d.h. die zuletzt angelegte Variable wird zuerst angezeigt.

- VLIST kann nur manuell ausgeführt werden.

VLIST (Beispiele)

Nach anlegen der Variablen

```
DIM DT$(10,20)*1

PQ=π
        3.141592654

DIM TA(24)

DIM C$(5)

NN$="Test"
Test
```

ermöglicht VLIST die folgenden Listen:

```
CSIZE 7

VLIST

NN$*16
C$(5)*16
TA(24)
PQ
DT$(10,20)*1
```

```
CSIZE 1

VLIST

NN$*16
C$(5)*16
TA(24)
PQ
DT$(10,20)*1
```

Liste der Funktionen und Instruktionen
****

TOOL3 bietet zusätzlich folgende Funktionen und Instruktionen an:

BEFEHL	Abkürzung	Kurzbeschreibung	Seite
CLR	--	Setzt den Inhalt der angegebenen Variablen, bzw. DIM-Felder null.	3-3
DEC	--	Die Funktion DEC berechnet den Dezimalwert einer als Zeichenfolge angegebenen hexadezimalen Zahl.	3-5
ERL	--	Liefert die Zeilennummer, in welcher zuletzt ein ERROR auftrat.	3-7
ERN	--	Liefert die Nummer des zuletzt aufgetretenen ERRORs.	3-7
FRC	FR.	Liefert den Nachkommateil eines numerischen Ausdruckes.	3-9
FRE	--	Liefert die Anzahl freier Bytes unter Berücksichtigung der 2-Zeichen-Variablen und der DIM-Felder.	3-11
HEX$	H.	Die Funktion HEX$ wandelt den Wert eines numerischen Ausdrucks in die hexadezimale Zeichenfolge um.	3-13
INSTR	INS.	Liefert die Position des Beginns des Such-Ausdruckes im Haupt-Ausdruck. Gesucht wird ab der angegebenen Position im Haupt-Ausdruck.	3-15
PGM	PG.	Liefert die Nummer des aktiven Programm-Moduls.	3-17
PSIZE	PS.	Liefert die Anzahl Bytes, die das aktive Programm-Modul im Speicher belegt.	3-18
PURGE	PU.	Löscht die angegebenen 2-Zeichen-Variablen, bzw. DIM-Felder aus dem Speicher.	3-19

BEFEHL	Abkürzung	Kurzbeschreibung	Seite
REDIM	RED.	Dimensioniert bereits bestehende DIM-Felder neu, wobei die Felderinhalte soweit als möglich erhalten bleiben.	3-21
RESUME	RESU.	Ermöglicht, zusammen mit ON ERROR, die Fortsetzung des Programmablaufs nach einem ERROR mit der nächstfolgenden Programm-Anweisung.	3-24
STRING$	STRI.	Liefert ein beliebiges Zeichen oder eine beliebige Zeichenfolge so oft wie angegeben.	3-26
SWAP	SW.	Vertauscht die Inhalte der angegebenen Variablen.	3-28
VKEEP	VK.	Ermöglicht die Adressierung von 2-Zeichen Variablen und DIM-Felder nach CLEAR, NEW, PURGE DIM, oder RUN.	3-30
VLIST	VL.	Listet 2-Zeichen-Variablen und DIM-Felder mit Namen, Datenart, Dimension(en) und Länge wahlweise in die Anzeige oder auf dem Plotter CE-150.	3-32