Klaus

A T A R I - L O G O

Allgemeine Hinweise

LOGO ist eine höhere Programmiersprache. Sie wurde entwickelt, um Kindern das Arbeiten mit dem Computer zu erleichtern.
LOGO wurde stark auf Grafikanwendungen ausgerichtet. Aber auch Routinen zum Arbeiten mit Buchstaben und Zahlen fehlen nicht. Bekannt ist vor allem die Schildkröte (Turtle). Sie kann durch einfache Befehle dazu gebracht werden sich zu bewegen, sich zu drehen und dabei die verschiedensten Figuren zu zeichnen.
Im Folgenden sollen die einzelnen LOGO-Befehle erläutert werden. Dabei ist folgendes zu beachten:

1. Zu manchen Befehlen gibt es Abkürzungen, die jeweils mit angegeben werden. Einige Befehle bestehen nur aus Abkürzungen, da das Wort dafür zu lang war. Abkürzungen und Wörter stehen in ( ) nach dem Befehl.
2. Viele Befehle erfordern zusätzliche Werte (Parameter). In diesem Fall befindet sich hinter dem Befehl eine Parameterbezeichnung in der Form < >.
3. Für manche Befehle werden eckige Klammern benötigt. Durch sie werden in LOGO Listen (d.h. Sammlungen von Worten und Zahlen) eingeschlossen.
4. Leider rechnet LOGO sehr ungenau, wenn mit Bruchzahlen gearbeitet wird. Zum Teil werden Zahlen sogar ungenau gemacht, obwohl gar nicht gerechnet wurde.
5. Zwischen dem Befehl und den Parametern muß mindestens ein Leerzeichen stehen, zwischen den Parametern ebenfalls.

Über die Funktion einiger Befehle im LOGO gibt es z.Z. noch keine Klarheit. Eine Auflistung dieser Befehle befindet sich am Schluß dieser Beschreibung.

S C H I L D K R ö T E N B E F E H L E

Nach dem Laden meldet sich LOGO im Textmodus. Wenn ein Schildkrötenbefehl eingegeben wird, springt LOGO automatisch in den Grafikmodus (mit Textfenster).
Der Bildschirm ist wie ein Koordinatensystem aufgeteilt. Die Turtle befindet sich im Koordinatenursprung und hat die Werte x=0, y=0.
Die Werte der x-Koordinate reichen von -159 bis 160, die der y-Koordinate von -119 bis 120.
Richtungen und Winkelangaben sind in LOGO wie auf einer Landkarte festgelegt. Der Bildschirm ist in 360 Grad eingeteilt.
Das Textfenster kann ausgeblendet werden, dann erscheint der darunterliegende Grafikbereich.

BACK <Strecke> (BK)
BK 90 bewegt die Turtle um 90 Pixel zurück (entgegen der Pfeilrichtung). Dabei zieht sie eine Linie.

FORWARD <Strecke> (FD)
FD 100 bewegt die Turtle um 100 Pixel vor (in Pfeilrichtung), wobei sie eine Linie zieht.

SETX <x-Koordinate>
Durch SETX 30 wird die Turtle auf eine Position gebracht, die sich 30 Pixel rechts neben dem Ursprung befindet. Sie verändert dabei weder Richtung noch vertikale Position und zieht auf ihrem Weg eine Linie.

Durch SETX -60 wird die Turtle auf eine Position 60 Pixel links vom Ursprung (Mittelpunkt) gebracht. Befindet sich die Turtle bereits auf dieser Position, so bewegt sie sich nicht.

SETY <y-Koordinate>
bewirkt dasselbe wie SETX, diesmal aber in vertikaler Richtung. SETY 10 setzt die Turtle auf einen Punkt 10 Pixel über dem Ursprung, SETY -10 auf einen 10 Pixel darunter. Dabei wird wieder eine Linie gezogen.

SETPOS [<x-Koordinate><y-Koordinate>]
ist die Zusammenfassung von SETX und SETY. SETPOS [10 23] setzt die Turtle auf einen Punkt mit den Koordinaten 10,23, d.h. 10 Pixel rechts und 23 Pixel über dem Ursprung. Auf ihrem Weg zieht die Turtle eine Linie.

POS
gibt die Koordinaten zurück, auf denen sich die Turtle vom Ursprung aus gesehen befindet.

XCOR
ist ein Unterbefehl zu POS und zeigt die x-Koordinate der Turtle (vom Ursprung aus gesehen) an.

YCOR
zeigt die y-Koordinate der Turtle an.

LEFT <Winkel> (LT)
LT 45 dreht die Turtle um 45 Grad nach links.

RIGHT <Winkel> (RT)
RT 58 dreht die Turtle um 58 Grad nach rechts.

SETH <Gradzahl> (SETHEADING)
dreht die Turtle in Richtung auf die angegebene absolute Gradzahl. Nach SETH 0 zeigt die Turtle nach Norden (oben); nach SETH 180 nach Süden (unten) usw.

HEADING
fordert die absolute Richtung an, in die die Turtle zeigt (in Grad).

HT (HIDETURTLE)
macht die Turtle unsichtbar. Alles was die Turtle macht (Linien usw.) bleibt sichtbar.

ST (SHOWTURTLE)
macht eine über HT unsichtbar gemachte Turtle wieder sichtbar.

SHOWNP
Abfrage, ob die Turtle sichtbar ist oder nicht.

SETPC <Stiftnummer><Farbe>
Zuordnen einer Farbe zum jeweiligen Stift

PC <Stiftnummer>
Frage nach der jeweiligen Stiftnummer-Farbe

PEN
Abfrage, ob der Stift oben oder unten ist

SETBG <Hintergrundfarbe>
Wählen der Hintergrundfarbe

BG
Abfrage der Hintergrundfarbe

SETCURSOR [Koordinaten]
setzt den Textcursor auf die angegebenen Koordinaten

CURSOR
fragt nach der Cursorposition

DOT [<x-Koordinate><y-Koordinate>]
plottet einen Punkt auf die angegebenen Koordinaten

SETSCR <Wert>
staucht oder dehnt einen Körper in Richtung der y-Achse; normal ist 0,8

SCRUNCH
Abfrage des Wertes von SETSCR

HOME
veranlaßt die Turtle, auf den Ursprung zurückzukehren. Dabei zieht sie wieder eine Linie (anders als beim ähnlichen Befehl SETPOS [0 0].

G R A F I K B I L D S C H I R M

WINDOW
Überschreitet die Turtle den Bildschirmrand, so ist sie zwar nicht mehr zu sehen, doch tritt keine Fehlermeldung auf.

WRAP
Hier taucht die Turtle, sobald sie den Bildschirmrand überschreitet, am gegenüberliegenden Rand wieder auf.

CS (CLEARSCREEN)
löscht den Grafikbildschirm und setzt die Turtle wie der HOME-Befehl auf die Ausgangsposition zurück.

CLEAN
löscht wie CS den Grafikbildschirm, ohne dabei die Turtle anzutasten

FS
schaltet das Textfenster aus

SS
schaltet das Textfenster ein

TS
schaltet von Grafik auf Text-Bildschirm (Bildinformation bleibt erhalten)

CT
löscht den Textbildschirm (CURSOR HOME)

SETSH <Nummer der selbstdefinierten Turtle>
Mit diesem Befehl kann man anstelle der Turtle einen selbstdefinierten Cursor erscheinen lassen. Werte von 0-15 sind möglich.
Mit SETSH <0> erscheint die Turtle wieder.

SHAPE
Abfrage, welche der 16 Möglichkeiten von SETSH eingeschaltet ist

PUTSH <Nummer des selbstdefinierten Cursors> [16 Werte]
Hiermit kann man sich selbst den Cursor definieren. Die 16 Werte bestimmen das Aussehen.
(Es gibt auch noch eine elegantere Lösung).

GETSH <Nummer des selbstdefin. Cursors>
zeigt den Wert des selbstdefinierten Cursors

SETC <Farbe> (SETCOLOR)
Einfärben der Turtle. Für die Farbe muß eine Zahl stehen.

COLOR
Abfrage, welche Farbe die Turtle gerade hat

TELL [Nummer der Turtle]
bestimmt, wieviele und welche Turtle die nächsten Befehle ausführen.
Es stehen 4 Turtle zur Verfügung (z.B. TELL [0 1 2 3])

WHO
Abfrage zu TELL

SETSP <Geschwindigkeit>
Hier läuft die Turtle(s) von allein; die Zahl gibt die Geschwindigkeit an.

SPEED
Abfrage der Geschwindigkeit

PENUP (PU)
Die Turtle nimmt ihren Zeichenstift hoch und zieht keine Linie bei ihrer Bewegung.

PENDOWN (PD)
Aufhebung des Befehls PU

PE (PENERASE)
Die Turtle tauscht ihren Zeichenstift gegen einen Radiergummi, sie kann keine Linie mehr ziehen und radiert beim überqueren andere Linien aus.

PX (PENREVERSE)
Jeder Punkt, den die Turtle berührt, wird in seine Gegenfarbe umgewandelt. Beim monochromen Monitor zieht die Turtle auf freien Flächen eine Linie, löscht aber beim Berühren bestehende Linien.

SETPN <Stiftnummer>
Auswahl des Stiftes (0,1,2 sind möglich)

PN
Abfrage nach der Stiftnummer

Z A H L E N V E R A R B E I T U N G

SUM <Zahlen>
Es gibt 2 Arten der Addition: 1. 1+2+3+n
2. (SUM 1 2 3 n)

\- (Minus)
Subtrahieren kann man wie gewohnt. Z.B. 10-2

PRODUKT <Zahlen>
Es gibt wieder 2 Möglichkeiten: 1. 1*2*3*n
2. (PRODUKT 1 2 3 n)

/ (Division)
z.B. 3/2

REMAINDER <Zahl><Zahl>
Es wird der Rest bei einer Division bestimmt. REMAINDER 5 2 ergit 1 (weil 5/2=2 Rest 1)

SQRT <Zahl>
ermittelt die Quadratwurzel aus einer Zahl. SQRT 4 ergibt 2.

INT <Zahl>
gibt den ganzzahligen Teil einer Zahl zurück. ?INT 3.5 ergibt 3. ?INT -3.5 ergibt -3.

ROUND <Zahl>
rundet eine Zahl echt: ?ROUND 3.4 ergibt 3; ?ROUND 3.5 ergibt 4.

SIN <Zahl>
ermittelt den Sinus eines Winkels. SIN 90 ergibt 1.

COS <Zahl>
ermittelt den Cosinus eines Winkels. COS 90 ergibt 0.

RANDOM <Zahl>
ergibt eine ganzzahlige Zufallszahl, die zwischen 0 und der hinter RANDOM angegebenen Zahl liegt. Die Zahl kann zwischen 32767 und -32768 liegen.

RERANDOM
bewirkt, daß diesselbe Zufallszahl wiederholt wird, die beim letzten RANDOM-Befehl erzeugt wurde.

W O R T E U N D L I S T E N

ASCII "<Buchstabe>
ermittelt den ASCII-Code des angegebenen Buchstabens bzw. des ersten Buchstabens eines Wortes. Z.B. ASCII "A ergibt 65.

CHAR <ASCII-Code>
gibt das zum jeweiligen ASCII-Code gehörende Zeichen aus. CHAR 65 = A

COUNT <Wort oder Liste>
bestimmt die Anzahl der Buchstaben in einem Wort bzw. die Anzahl der Elemente einer Liste. Z.B. COUNT "Wort ergibt 4; COUNT [a2y] ist 3.

FIRST <Wort oder Liste>
ermittelt den ersten Buchstaben eines Wortes bzw. das erste Element einer Liste. FIRST "Wort ergibt W; FIRST [1 2 3] ist 1.

LAST <Wort oder Liste>
ermittelt den letzten Buchstaben bzw. das letzte Element

BUTFIRST <Wort oder Liste> (BF)
ergibt alle Buchstaben eines Wortes bzw. alle Elemente einer Liste mit Ausnahme des ersten. BF "Wort ergibt ort, BF [1 2 3] ist [2 3]

BUTLAST <Wort oder Liste> (BL)
ergibt alle Buchstaben eines Wortes oder alle Elemente einer Liste mit Ausnahme des letzten. BL "Wort ergibt Wor

MEMBERP <Wort oder Liste>
ermittelt, ob ein Buchstabe oder eine Zahl Element eines Wortes oder einer Liste ist. MEMBERP "O"WORT ergibt TRUE (wahr), da der Buchstabe O in "WORT" enthalten ist. MEMBERP "4 [1 2 3] ergibt FALSE, da 4 nicht Element von [1 2 3] ist.

NUMBERP "<Objekt>
ergibt TRUE, wenn das Objekt eine Zahl ist (z.B. bei NUMBERP "1);
ergibt FALSE, wenn es keine Zahl ist (z.B. NUMBERP "Wort).

WORDP <Objekt>
ergibt TRUE, wenn das Objekt ein Wort oder eine Zahl ist (z.B. WORDP "Wort oder WORDP "2) bzw. FALSE, wenn das Objekt eine Liste ist (z.B. WORDP [1 2 3]).

LISTP <Objekt>
ergibt TRUE, wenn das Objekt eine Liste ist (z.B. LISTP [1 2]), sonst FALSE (z.B. LISTP "1).

EMPTYP <Objekt>
ergibt TRUE, wenn das Objekt ein leeres Wort (") oder eine leere Liste ([ ]) ist, sonst FALSE (z.B. EMPTYP "Wort).

EQUALP <Wort, Zahl, Liste><Wort, Zahl, Liste>
ergibt TRUE, wenn die beiden darauffolgenden Worte, Zahlen oder Listen identisch sind, sonst FALSE. Deutsche Sonderzeichen sind nicht zulässig.

WORD <Wort 1><Wort 2>
setzt 2 Worte zu einem neuen Wort zusammen (WORD "HAUS"BOOT ergibt HAUSBOOT).

SENTENCE <Worte, Zahlen, Listen> (SE)
erzeugt Listen aus mehreren Worten oder Zahlen, setzt Worte und Zahlen in Listen ein und fügt Listen zusammen.
z.B. ?SE "HAUS"BOOT ergibt [HAUSBOOT]
?SE "1[A B C] ergibt [1 A B C]
?SE [1 2][3 4] " [1 2 3 4]
?(SE"A"B"C 3) " [A B C 3]

LIST <Worte, Zahlen, Listen>
erzeugt ähnlich wie SENTENCE Listen, behält aber bei Angabe von Listen deren eigene Klammern:
?LIST "HAUS "BOOT ergibt [HAUS BOOT]; ?LIST "1 [A B] ergibt [1[A B]]
?LIST [1 2][3 4] " [[1 2][3 4]]
?(LIST "A "B "C 3) " [A B C 3]

FPUT <Wort, Zahl, Liste><Wort, Zahl, Liste>
setzt den ersten Ausdruck an den Anfang des zweiten.
?FPUT "TISCH "BEIN ergibt TISCHBEIN; FPUT 1 [2 3] ergibt [1 2 3]

T E X T D A R S T E L L U N G

PRINT <Ausgabeliste> (PR)
gibt Zahlen, Worte oder Listen auf den Textbildschirm aus. Nach jedem PRINT-Befehl wird ein Zeilenvorschub vorgenommen (auch durch PR allein). Bei der Ausgabe von Listen werden die äußeren Klammern entfernt.

SHOW <Wort, Zahl, Liste>
gibt ähnlich wie PRINT auf dem Textbildschirm aus. Nach jedem SHOW-Befehl erfolgt ein Zeilenvorschub. Bei Listen bleiben die äußeren Klammern erhalten. Außerdem kann auf SHOW nur ein Objekt und nicht eine ganze Ausgabeliste folgen.

TYPE <Ausgabeliste>
dient wie PRINT zur Ausgabe und entfernt bei Listen die äußeren Klammern. Im Unterschied zu PRINT wird im ersten Objekt kein Zeilenvorschub vorgenommen.

V A R I A B L E N

MAKE "<Variablenname><Inhalt>
ordnet die Variablen einem Inhalt zu:
z.B. ?MAKE "Auto 10
?MAKE "Auto "Rennwagen
?MAKE "Auto [1 2 Wagen Haus]
Wenn Sie den Inhalt der Variablen auslesen oder für andere Zwecke benutzen wollen, muß dem Variablennamen ein Doppelpunkt vorangestellt werden, z.B. ? :Auto.

NAMEP "<Variablenname>
gibt TRUE aus, wenn die angegebene Variable einen Inhalt hat.
FALSE bedeutet, daß sie nicht definiert wurde.

THING "<Variablenname>
gibt den Inhalt einer Variablen aus.

S T R U K T U R B E F E H L E

REPEAT <Wiederholungszahl><Befehlsliste>
gibt an, wieviele Male die Befehlsliste durchlaufen werden soll.
z.B. REPEAT 4 [FD 100 RT 90]

RUN [<Befehlsliste>]
dient der Abarbeitung einer Befehlsliste.
z.B. MAKE "QUADRAT [REPEAT 4 [FD 100 RT 90]]
RUN :QUADRAT

STOP
dient ähnlich wie die Tasten CONTROL und G zur Unterbrechung eines Programmes, z.B. zur Korrektur oder zur Sicherung auf Kassette.

OUTPUT "<Wort, Zahl, Liste> (OP)
bricht wie STOP die Ausführung einer Prozedur ab, zeigt aber zusätzlich den angegebenen Ausdruck an. OP "FEHLER bricht die Prozedur mit der Meldung FEHLER ab. Anders als STOP darf man OP nur innerhalb einer Prozedur und nicht direkt eingeben.

WAIT <Länge>
Hier läuft der Rechner in einer Warteschleife, dessen Länge festgelegt wurde.

BREAK
unterbricht die Programmausführung. Vorsicht! Dabei stürzt der Rechner oft ab!

KEYP
dient der Tastaturabfrage. Wurde eine Taste gedrückt, ergibt der Befehl eine wahre Aussage.
z.B. IF KEYP= "TRUE [SPRUNG]

B E D I N G U N G E N

LOGO ist in der Lage zu überprüfen, ob bestimmte Bedingungen wahr oder falsch sind:

gleich	1=1	TRUE;	1=2	FALSE
größer	2>1	TRUE;	1>1	FALSE
kleiner	1<2	TRUE;	1<1	FALSE

AND <logische Ausdrücke>
verknüpft Ausdrücke und gibt TRUE zurück, wenn beide Aussagen wahr sind; sonst FALSE.
z.B. AND 1=1 2=2 ist TRUE, (AND 1=1 3=Ø) ist FALSE

OR <logischer Ausdruck>
gibt schon TRUE zurück, wenn eine der Aussagen wahr ist. Sind alle falsch, so meldet sich FALSE.

NOT
kehrt den Sinn einer Bedingung um. Trifft die Bedingung nicht zu, so meldet LOGO TRUE.
z.B. NOT 1=1 ergibt FALSE

IF <logischer Ausdruck>[<Befehlsliste>]
macht die Ausführung der folgenden Befehlsliste von der Erfüllung einer Bedingung (logischer Ausdruck ergibt TRUE) abhängig.
z.B. IF :A > :B [QUADRAT]
Ist die Bedingung im Beispiel erfüllt (Wert von A > von B), so springt LOGO zur Prozedur QUADRAT. Wenn nicht, geht es zur nächsten Programmzeile.

S P E I C H E R - U N D S Y S T E M - P R I M I T I V E S

NODES
gibt Auskunft über den noch freien Arbeitsspeicher

RECYCLER
reorganisiert den Arbeitsspeicher und entfernt alle überflüssigen Daten

.EXAMINE <Speicheradresse>
gibt den Byte-Inhalt der angegebenen Adresse aus (wie "PEEK" im BASIC)

.DEPOSIT <Speicheradresse><Wert>
ordnet der angegebenen Speicheradresse den folgenden Wert zu (wie "POKE" im BASIC); z.B. .DEPOSIT 9809 23 ordnet der Adresse 9809 den Inhalt 23 zu

.CALL <Sprungadresse>
Sprung ins Maschinenprogramm

.PRIMITIVES
zeigt alle systemeigenen Befehle

E I N - U N D A U S G A B E O P E R A T I O N E N

.DOS
Srung zum DOS

LOAD "C"
lädt LOGO-Datei von Kassette

LOAD "D:"
lädt LOGO-Datei von Diskette

SAVE "C"
speichert LOGO-Datei auf Kassette

SAVE "D:"
speichert LOGO-Datei auf Diskette

CATALOG "<D1-8:><Name>
zeigt die Directory der Diskette im jeweiligen Laufwerk. Steht hinter dem Doppelpunkt ein Name, wird nur dieses File mit seiner Anzahl der Sektoren angezeigt.

SETWRITE "<Buchstabe>"
öffnet einen Ausgabekanal für das angegebene Gerät zum Schreiben (analog BASIC)

SETREAD "<Buchstabe>"
öffnet einen Eingabekanal für das angegebene Gerät zum Lesen

RL (READLIST)
liest eine Zeile von Tastatur oder Datenfile, die mit RETURN abgeschlossen wird und wertet den Inhalt als Inhalt einer Liste.
z.B. ?MAKE "LISTE RL (Liste einlesen) ?LISTE

RC (REDCHAR)
liest ein Zeichen von der Tastatur oder einem Datenfile

JOY <Nummer>
Abfrage des Joystick. Nummer 0-3 sind möglich. Beim XL ist aber nur 0 und 3 sinnvoll.

JOYB <Nummer>
Abfrage des Feuerknopfes. Wenn er gedrückt ist, gibt es bei LOGO eine wahre Aussage.

PADDEL <Nummer>
Abfrage des PADDEL-Wertes. Für Nummer sind Werte von 0-7 möglich; beim XL aber nur 0-3 sinnvoll.

PADDELB <Nummer>
Abfrage des Feuerknopfes am Paddel analog zu JOYB

TOOT <Kanal><Tonhöhe><Lautstärke><Länge>
dient zur Tonerzeugung. Es gibt 2 Tonkanäle 0 und 1. Tonwerte können ab 62 aufwärts eingegeben werden. Lautstärke minimal =0, max. =255. Achtung: Kanal 0 und 1 haben bei gleichen Tonwerten unterschiedliche Frequenzen.

SETVENY <Kanalnummer><Verzerrung>
gibt den Verzerrungswert für die Tonkanäle 0 und 1 an. Werte 0-127

P R O G R A M M I E R E N I N L O G O

Es gibt bei LOGO keine Zeilennummern wie im BASIC. In LOGO ist es möglich, daß man nicht nur die Grundbefehle ("Primitives") benutzt, sondern selbst neue Befehle definiert. Jedes Unterprogramm eines LOGO-Programms erhält einen Namen und wird dann wie fest eingebaute Befehls-Funktion aufgerufen. Diese Unterprogramme heißen Prozeduren.

TO <Prozedurname> und END
Mittels TO werden alle Prozeduren benannt und ihre Eingabe wird ermöglicht. Während der Eingabe einer Prozedur zeigt LOGO zu Beginn jeder neuen Zeile nicht das übliche Fragezeichen, sondern einen Pfeil (>). Die Eingabe wird durch END abgeschlossen.
Da nie 2 Prozeduren mit dem selben Namen existieren können, wird ggf. eine gleichnamige Prozedur überschrieben.

Beispiele:

(1)
```
?TO QUADRAT
>CS FD 100 RT 90
>FD 100 RT 90
>FD 100 RT 90
>FD 100 RT 90
>END
```

(2)
```
?TO QUADRAT :L
>CS FD :L RT 90
>FD :L RT 90
>FD :L RT 90
>FD :L RT 90
>END
```

Beispiel 1 definiert eine Prozedur, die ein Quadrat mit der Seitenlänge 100 zeichnet. Die Prozedur wird einfach durch Eingabe des Wortes QUADRAT aufgerufen.
Beispiel 2 zeigt, daß auch Prozeduren von Variablen abhängen können. Hier wird die Seitenlänge durch :L festgelegt. Geben Sie QUADRAT 10 ein, so wird ein Quadrat mit der Seitenlänge 10 gezeichnet u.s.w. Wird nur QUADRAT eingegeben, so meldet LOGO einen Fehler.

Im folgenden Beispiel wird gezeigt, daß eine Prozedur auch andere Prozeduren (hier QUADRAT) oder sogar sich selbst (hier FIGUR) aufrufen kann. Wenn die Prozedur sich selbst aufruft, dreht sie sich solange im Kreise, bis sie durch die BREAK-Taste abgebrochen wird.

```
?TO FIGUR
>FD 20 RT 90
>QUADRAT
>FIGUR
>END
```

PO "<Prozedur>
gibt die Befehlsfolge der angegebenen Prozedur aus

POALL
gibt die Befehlsfolgen aller Prozeduren und Variablen aus

POPS
gibt die Befehlsfolgen aller Prozeduren aus

PONS
gibt alle Variablen und ihre Werte aus

POD <Nummer>
zeigt, was in WHEN <Nummer> definiert wurde

PODS
zeigt alles, was in WHEN definiert worden ist

ERN [<Variablenname>]
löscht die angegebenen Variablen

ERNS
löscht alle Variablen

ERALL
löscht alle Prozeduren und Variablen

ERPS
löscht alle Prozeduren

ERASE [<Prozedurname>] (ER)
löscht die angegebene Prozedur

ERF <Filename>
löscht das angegebene File auf der Diskette. Achtung! Keine Rückfrage!

E D I T O R F U N K T I O N E N

EDIT [<Prozedurname>] (ED)
bringt die angegebene Prozedur ins Editierfeld

EDNS
bringt alle Variablen ins Editierfeld

EDSH <Nummer des selbstdefinierten Cursors>
zum Umdefinieren des eigenen Cursors. Das schwarze Feld stellt die maximale Größe des Cursors dar. Bewegen Sie den Editorcursor mit Hilfe der Cursortasten ohne <CONTROL>. Wenn Sie einen Pixel aktivieren wollen, drücken Sie die Insert-Taste ohne <CONTROL>. Wollen Sie einen Pixel löschen, so benutzen Sie <CLEAR>. Ist Ihr Gebilde fertig, drücken Sie die ESC-Taste und der Cursor ist definiert. Sie befinden sich jetzt im Textmodus. Vergessen Sie nicht, die Cursornummer mit SETSH auf Ihre selbstdefinierte Cursornummer umzuändern.

STEUERUNG DES CURSORS IM EDITORFELD:

<DELETE>
löscht ein Zeichen links vom Cursor

<DELETE> + <CONTROL>
löscht das Zeichen unter dem Cursor

<DELETE> + <SHIFT>
löscht alle Zeichen rechts vom Cursor, einschließlich des Zeichens unter dem Cursor

<RETURN>
schiebt alle Zeichen rechts vom Cursor, einschließlich des Zeichens unter dem Cursor, in die nächste Zeile

Cursorsteuerung genau wie in BASIC mit <CONTROL>

<ESC>
Verlassen des Editorfeldes

B E F E H L E , D E R E N F U N K T I O N U N K L A R

ASK <Turtlenummer> [<Befehl>]

EACH: [Befehl]

WHEN: <Nummer (0-23)> [Liste]

COND: <Nummer (0-23)>

B E I S P I E L E

```
TO START
CT CS
PR [+++++DEMO+++++]
REPEAT 4 [PR" ]
PR [ B LAUE TURTLE]
PR [ D REHENDE QUADRATE]
PR [ W ÜRFEL]
PR "PR"
PR [WÄHLEN SIE DURCH TASTENDRUCK]
MAKE "W RC
IF :W = "B [BLAU]
IF :W = "D [DRQ]
IF :W = "W [WÜRFEL]
START
END

TO BLAU
CT CS
PR [Abbrechen durch Tastendruck]
SPRUNG
IF KEYP = "TRUE [CS START]
FD 20 RANDOM 45
SPRUNG
END

TO DRQ
CS FS
REPEAT 36 [REPEAT [FD 75 RT 90] RT 10]
WAIT 250
CS
START
END
```

```
TO WÜRFEL
FS
PU SETPOS [ -50 -50]
PD REPEAT 4 [FD 100 RT 90]
PU SETPOS [ -10 -10]
PD REPEAT 4 [FD 10 RT 90]
TELL [0 1 2 3] PU
ASK 1 [SETPOS [90 -10]]
ASK 2 [SETPOS [ -10 90]]
ASK 3 [SETPOS [90 90]]
PD LT 135 FD 35
WAIT 100
HT
WAIT 250
CS ST
START
END
```