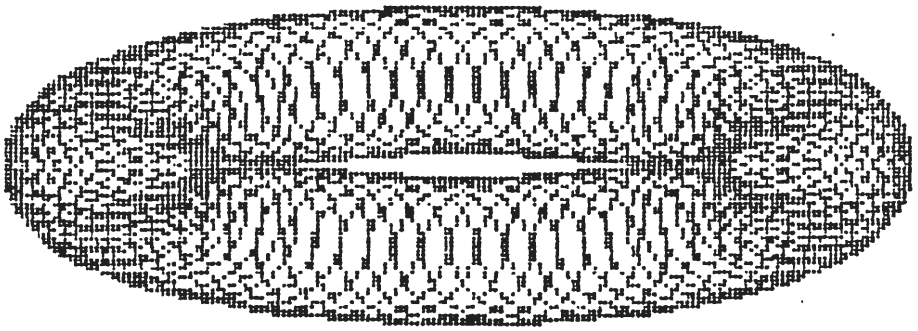
# Manual
# EXTENDED BASIC II PLUS

TI – 99/4 A
EXPANDED
GRAFIC BASIC

HIGH
RESOLUTION GRAFIC
BY
© APESOFT



MECHATRONIC

# MECHATRONIC

44 K BYTES
OF PROGRAMMED MEMORY
PLUS
HIGH RESOLUTION
GRAFIC BASIC

# EXTENDED BASIC II PLUS

MANUFACTURED UNDER LICENSES
BY TEXAS INSTRUMENTS
AND APESOFT

# E X T E N D E D  B A S I C  I I  P L U S

Allgemeine Befehlserweiterungen:

CALL BHCOPY("Dateiname", "ESC-Sequenz")

Dieses Unterprogramm erstellt eine Hardcopy des Bildschirms im sogenannten Bit Image Mode (8-Bit Einzelnadeldruck, 8-Bit Einzelnadelgrafik) für Matrixdrucker. Geeignet ist diese Routine z. B. für Epson-Matrixdrucker. Für Drucker, die eine andere Graphikgenerierung benutzen, ist dieser Befehl nicht geeignet.

Innerhalb des Expanded Grafic Basic ist dies eine Hardcopy des Graphik-Bildschirms, sonst erfolgt eine Hardcopy des normalen Bildschirms. Beide Male werden Sprites nicht mit abgebildet. Als Dateinamen können Sie alle gültigen Dateinamen der angeschlossenen Peripheriegeräte verwenden, sinnvoll ist aber nur eine Schnittstelle, an die ein Drucker angeschlossen ist, also z. B. "RS232.BA=9600.DA=8.CR" oder "PIO.CR". Bitte beachten Sie, daß das .CR wichtig ist für das einwandfreie Funktionieren der Routine. Ebenfalls müssen Sie bei der seriellen RS 232-Schnittstelle Datenbits einstellen. Weiter muß der Drucker auf "Autolinefeed" eingestellt werden (nähere Hinweise dazu finden Sie in dem Handbuch Ihres Druckers).

Hinsichtlich der ESC-Sequenz müssen Sie das Handbuch zu Ihrem Drucker beachten. BHCOPY sendet zu Beginn jeder Zeile ein CHR$(27) (ESC), dann den String, den Sie einsetzen (darf bis zu 10 Zeichen lang sein) und danach die Zeichen CHR$(0), CHR$(1) (für 256 folgende Bytes) und nun die 256 Bytes, wie sie der Hardcopy einer Zeile entsprechend und abschließend ein "CR" (CHR$(13). Bei vielen Matrixdruckern kommen Sie mit "L" und "K" für die ESC-Sequenz zu guten Ergebnissen. Der Drucker muß, um ein richtiges Druckbild zu erhalten, vor einer Hardcopy noch auf den richtigen Zeilenabstand eingestellt werden.

<u>Beispiele:</u>

CALL BHCOPY("RS232.BA=9600.DA=8.CR","L")
gibt eine Hardcopy über die serielle Schnittstelle aus
und
CALL BHCOPY("PIO.CR","K")
über die parallele. Der Drucker muß dabei auf Autolinefeed und vorher auf den richtigen Zeilenabstand eingestellt werden.

CALL VPEEK(Adresse, numerische Variablenliste)

Mit dem VPEEK Unterprogramm können die Speicherstellen im VDP-RAM des Computers direkt gelesen werden. Es gelten sinngemäß die Angaben zu CALL PEEK im Handbuch zum Extended Basic. Bitte beachten Sie, daß der Adressenbereich des VDP-RAM von 0 bis 16383 reicht. Geben Sie größere Werte ein, kann es zu Fehlfunktionen des Computers kommen.

Beispiel:

100 CALL VPEEK(100,A,B)
Diese Programmzeile ordnet den numerischen Variablen A und B die Bytewerte in den Speicherstellen 100 und 200 des Video-Ram's zu.

CALL VPOKE(Adresse, numerische Daten)

Mit dem VPOKE Unterprogramm kann direkt in einzelne Speicherstellen des VDP-RAM's geschrieben werden. Hinsichtlich der Adressenangabe gilt das bei VPEEK gesagte. Bei unbedachter Benutzung des VPOKE Unterprogramms kann es ebenfalls zu Fehlfunktionen des Computers kommen. Eventuell kann er sich auch "aufhängen", so daß nur noch das Ab- und Wiedereinschalten hilft.

Beispiel:

CALL VPOKE(100,161)
schreibt ein A auf den Bildschirm.

Das Video-Ram wird unter Extended Basic wie folgt benutzt:
0-767: Bildschirm (mit Offset 96)
768-879: Sprite-Definitionen
880-1919: Charakter-Definitionen
1920-2047: Sprite Geschwindigkeitstabellen
2048-2079: Farbtafeln
2080-16383: Platz für Strings, Variablen, Symboltabellen und Programm, sofern keine Speichererweiterung angeschlossen ist. Ist ein Disk-Controller angeschlossen, so ist hier ab 13784 ein Bereich für diesen reserviert. In diesem Bereich sollten auf keinen Fall Werte mit CALL VPOKE geschrieben werden, da sonst die Informationen auf den im Laufwerk befindlichen Disketten zerstört werden können.

CALL GPEEK(Adresse, numerische Variablenliste)

DAS GPEEK Unterprogramm dient zum Auslesen einzelner Speicherstellen in den GROM's des Computers. Die Funktion ist analog CALL VPEEK. Dieser Befehl kann dem Interessierten zum Auslesen der GROM's dienen.

CALL ALLSET

Das ALLSET Unterprogramm arbeitet wie das CALL CHARSET im Extended Basic, nur daß die Zeichen mit dem ASCII-Wert 32 bis 126, also auch die Kleinbuchstaben, auf die Grunddefinitionen zurückgesetzt werden.

CALL WAIT(Dauer)

Das WAIT Unterprogramm wartet eine gewisse Zeit. Die Zeitdauer ist dabei ein numerischer Ausdruck zwischen 0 und 16382. Die Wartedauer entspricht dabei dem Wert geteilt durch 50, d. h. die Angabe von 1000 entspricht ca. 20 Sekunden. CALL WAIT wird aber sofort beendet, wenn eine beliebige Taste gedrückt wird. Für die Dauer können Werte von 1 bis 16000 eingegeben werden. Andere Werte ergeben keine nützlichen Zeilen bzw. einen Bad Value Error. Dieser Befehl kann die sonst üblichen Warteschleifen ersetzen.

Beispiel:

```
100 CALL WAIT(200)
```

läßt den Computer 4 Sekunden bzw. bis eine Taste innerhalb dieser Zeitspanne gedrückt wird warten, bevor er mit der Ausführung des Programmes fortfährt.

CALL MOVE(Modus, Startadresse, Zieladresse, Anzahl)

Mit CALL MOVE kann der Inhalt von Speicherblöcken innerhalb des Computers verschoben werden. Es stehen vier Modi zur Verfügung:

1 = Start VDP-RAM, Ziel VDP-RAM
2 = Start VDP-RAM, Ziel CPU-RAM
3 = Start CPU-RAM, Ziel VDP-RAM
4 = Start CPU-RAM, Ziel CPU-RAM

Bei Moduswerten von kleiner als 1 oder größer als 4 erfolgt ein Numeric Overflow Error. Mit CALL MOVE kann es, wenn es nicht bedacht eingesetzt wird, zu Fehlfunktionen des Computers kommen. Ebenfalls können leicht Teile des Programmes oder der Variablen usw. überschrieben werden, so daß Sie sich genau im Klaren darüber sein sollten, was Sie machen, bevor Sie CALL MOVE einsetzen.

Beispiel:

Das MOVE Unterprogramm kann unter anderem zur Bildschirmspeicherung verwendet werden. Wenn keine Maschinenprogramme benutzt oder geladen sind (also darf dieses Beispiel auch nicht innerhalb des Expanded Grafic Basic angewendet werden), kann mit:

CALL MOVE(2,0,8192,768)
der Bildschirminhalt in den unteren Teil der Speichererweiterung der von 8192 bis 16383 reicht, gesichert werden und mit:

CALL MOVE(3,8192,0,768)
wieder auf den Bildschirn geholt werden.

Ein weiteres Beispiel ist eine Laufschrift auf dem Bildschirm:

```
100 CALL CLEAR
110 DISPLAY AT(20,2): "DAS IST EINE DEMONSTRATION"
120 CALL MOVE(1,608,448,160)
130 FOR X=1 TO 450
140 CALL MOVE(1,164,163,445)
150 NEXT X
```

CALL MSAVE("Dateiname", Startadresse, Anzahl Bytes)

Das MSAVE Unterprogramm speichert Abschnitte des CPU-RAM-Inhaltes im sogenannten Programm-Format ab. Dieses Unterprogramm kann sinnvoll nur mit Speichererweiterung eingesetzt werden. Damit können nun auch mit dem Kassetten-Rekorder Maschinensprache-Programme abgespeichert werden, und nur dazu darf dieser Befehl eingesetzt werden.

Mit
CALL MSAVE("C51",8192,8192)
wird das ganze zur Verfügung stehende Maschinensprache-RAM (Low Memory der Speichererweiterung) abgespeichert. Es können maximal 8192 Bytes gespeichert werden. Selbstverständlich ist als Dateiname auch jede andere gültige Datei, z. B. "DSK1.MASCHINE" zu verwenden. Da der normale Loader des Extended Basic für Maschinenprogramme recht langsam arbeitet, ist dieser Befehl durchaus auch in Zusammenhang mit einem Diskettenlaufwerk interessant.

CALL MLOAD("Dateiname", Modus)

MLOAD ist das Gegenteil vom MSAVE. Es lädt den mit MSAVE abgespeicherten CPU-RAM Inhalt. Als Dateiname kann wieder jede gültige Datei verwendet werden. Beim Laden von mit CALL MSAVE gespeicherten Maschinenprogrammem für das Extended Basic sollte der Modus weggelassen oder 0 eingesetzt werden. Wird als Modus ein numerischer Wert größer als Null angegeben, so erfolgt ein Autostart eines mit der Save-Utility des Editor/Assembler im Programm-File Format erstellten Maschinenprogramms. Vor dem Start erfolgt auch die Umschaltung des VDP-RAM's auf die Werte des Editor/Assemblers.

Bitte beachten Sie, daß mit CALL MLOAD alle Dateien im sogenannten Programm-Format (auch Basic-Programme) geladen werden können, ohne daß es zu einem ERROR kommt, die einwandfreie Funktion ist aber nur bei Program-Files des Editor-Assembler oder bei mit MSAVE abgespeicherten RAM-Inhalten gegeben. Weiter benötigen die Unterprogramme MSAVE und MLOAD einen ziemlich großen Bereich im VDP-RAM, so daß es sehr schnell zu einem Memory Full Error kommen kann.

CALL BYE

Das BYE Unterprogramm arbeitet wie das BYE im Extended Basic, nur kann CALL BYE auch im Programmablauf verwendet werden.

Beispiel:

```
100 REM PROGRAMMBEGINN
       '
       '
3000 PRINT "NOCH EINMAL J/N?"
3010 CALL KEY(0,K,M) :: IF M>1 THEN 3010
3020 IF K=74 OR K=106 THEN 100 ELSE CALL BYE
```

CALL NEW

Das NEW Unterprogramm entspricht dem NEW im Direktmodus, nur kann CALL NEW auch im Programm verwendet werden.

CALL RESTORE(Variable)

CALL RESTORE ist genau gleich wie Restore im Extended Basic, nur kann bei CALL RESTORE auch eine Variable angegeben werden. Werden im CALL RESTORE feste Zeilennummern eingegeben, so erfolgt bei einem RES keine Änderung dieser Nummern. CALL RESTORE kann nicht in Zusammenhang mit Dateien eingesetzt werden.

Beispiel:

```
100 DATA "FARBE"
110 DATA "FORM"
120 DATA "GEWICHT"
130 A=100+INT(RND*2.9)*10
140 CALL RESTORE(A)
150 READ A$ :: PRINT A$
160 CALL WAIT(30) :: GOTO 130
```

CALL QUITOF
CALL QUITON

CALL QUITOF schaltet die Funktion der Quit-Taste aus, d. h. ein unbeabsichtigtes Drücken hat keine unangenehmen Folgen mehr. CALL QUITON schaltet die Funktion wieder an, d. h. es ist mit Drücken der Quit-Taste wieder der Titel zu erreichen.

CALL SPRON
CALL SPROF

CALL SPROF schaltet die Bewegung aller Sprites ab. Mit CALL SPRON können alle Sprites gleichzeitig wieder in Bewegung gesetzt werden. Beide Unterprogramme werden hauptsächlich bei aus mehreren Sprites zusammengesetzten Gebilden verwendet, die gleichförmig über den Bildschirm bewegt werden sollen. Dazu müssen natürlich die Bewegungen mit CALL MOTION oder CALL SPRITE gesetzt sein.

Beispiel:

```
100 CALL CLEAR
110 CALL CHAR(126,"FFFFFFFFFFFFFFFF")
120 FOR X=1 TO 4
130 CALL SPRITE(#X,126,2,124,124)
135 NEXT X
140 CALL MAGNIFY(2)
150 CALL COLOR(#1,5,#2,7,#3,7,#4,5)
160 CALL LOCATE(#2,124,140,#3,140,124,#4,140,140)
170 CALL WAIT(200)
180 CALL SPROF
190 CALL MOTION(#4,0,124,#3,0,124,#2,0,124,#1,0,1  24)
200 CALL SPRON
210 CALL WAIT(1000)
220 END
```

Bitte beachten Sie, daß CALL QUITOF und CALL SPROF nicht vom Betriebssystem zurückgesetzt werden, d. h. auch nach einem BYE oder NEW noch erhalten sind. Sie können nur mit QUITON und SPRON wieder aufgehoben werden, oder durch Ab- bzw. Wiedereinschalten der Konsole.

CALL SCREENON
CALL SCREENOF

Mit SCREENOF kann der Bildschirm abgeschaltet und mit SCREENON wieder eingeschaltet werden. Bei einer Programmunterbrechung wird der Bildschirm automatisch vom Betriebssystem wieder eingeschaltet.

Beispiel:

```
100 CALL SCREENOF
110 FOR X=1 TO 30
120 PRINT "DEMONSTRATION SCREENOF"
130 NEXT X
140 CALL SCREENON
150 CALL WAIT (700)
```

CALL FIND("Suchstring", "Stringarray"(), Rückgabevariable)

Das FIND Unterprogramm sucht in einem eindimensionalen Stringarray nach dem als "Suchstring" eingegebenen Begriff. Die Rückgabevariable ist numerisch und nimmt den Wert des gesuchten Gliedes im Array ein. Wird der Suchstring nicht gefunden, so ist der Wert -1.

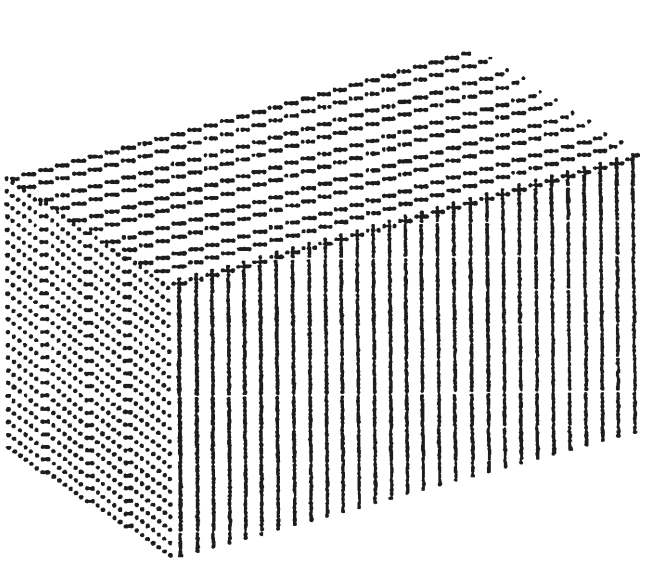
Beispiel:

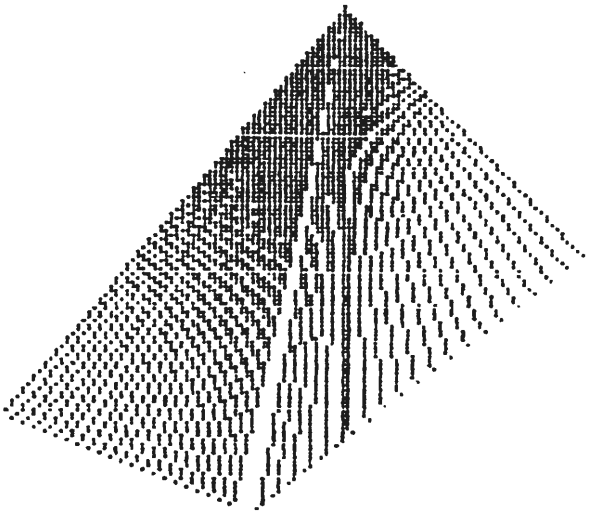
```
100 DIM C$(100)
110 FOR X=0 TO 100
120 C$(X)="TEST"
130 NEXT X
140 C$(70)="NAME"
150 INPUT "SUCHSTRING:":A$
160 CALL FIND(A$,C$(),B)
170 IF B=-1 THEN PRINT "NICHT GEFUNDEN"
    ELSE PRINT "GEFUNDEN IM ELEMENT";B
180 GOTO 150
```

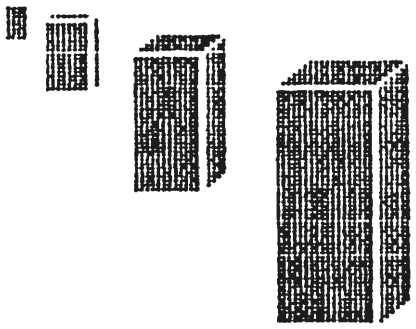
MICROCOMPUTER

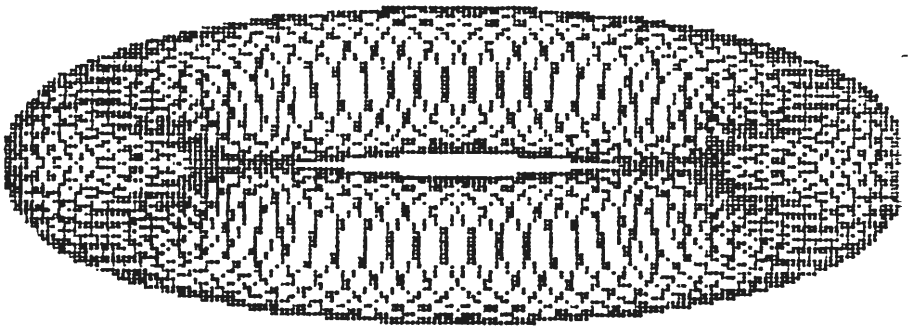
# Opsoft

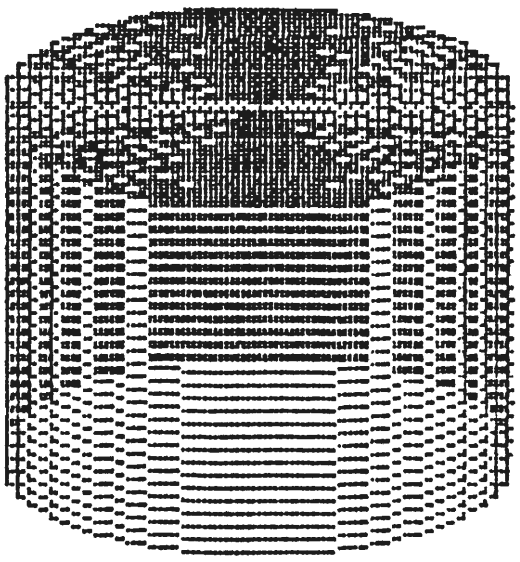
SOFTWARE

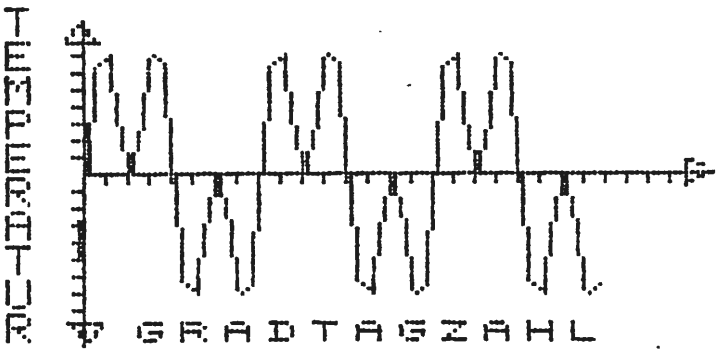












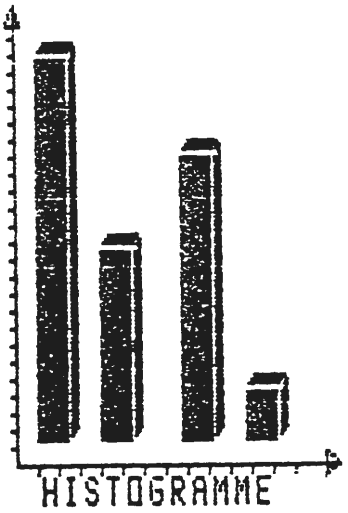

MITTLERE
ANALYSENWERTE

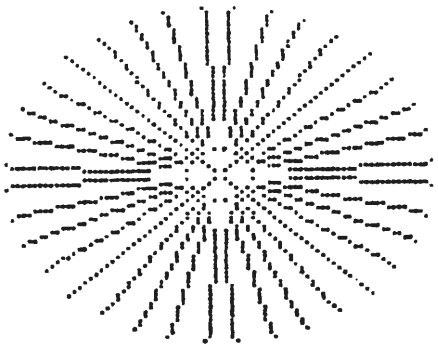
# I N H A L T S V E R Z E I C H N I S

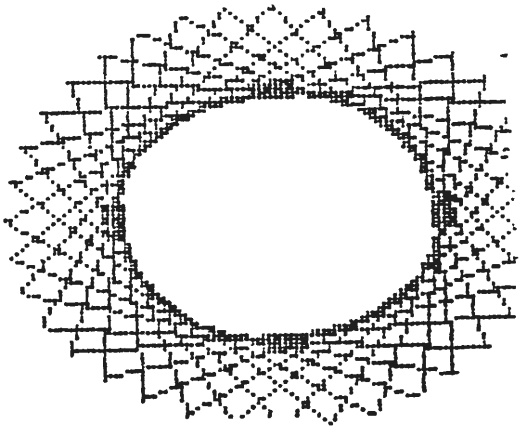
TI-99/4A

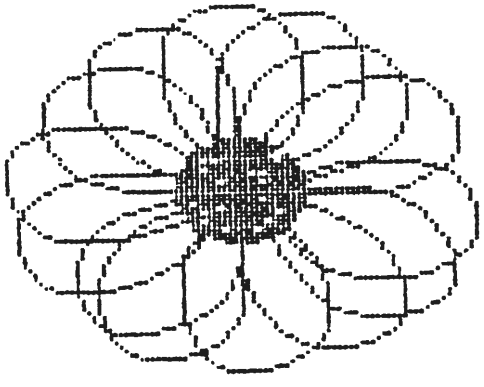
EXPANDED
GRAFIC BASIC

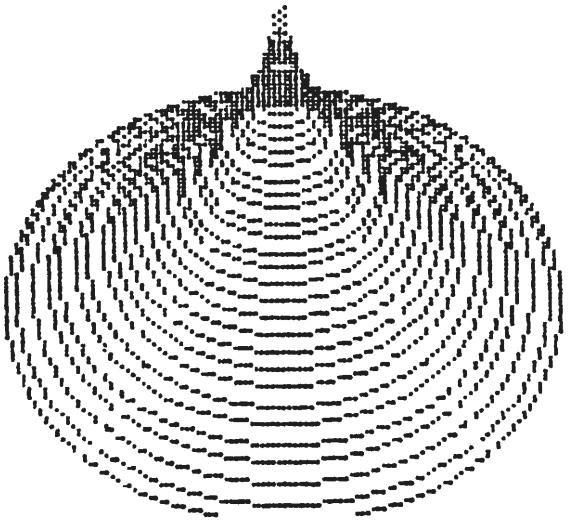
HIGH
RESOLUTION GRAFIC
BY
@APESOFT

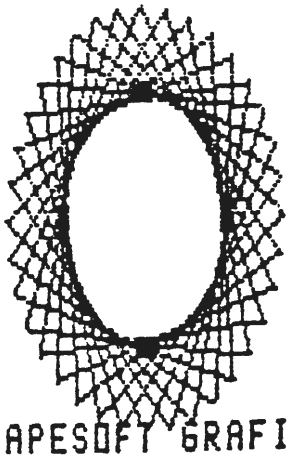










# 1. E I N L E I T U N G

APESOFT EXPANDED GRAFIC BASIC erfüllt den Wunsch des TI-99/4A Besitzers nach High Resolution Graphik! Mit APESOFT EXPANDED GRAFIC BASIC, kurz APESOFT BASIC, entfalten sich die wunderbaren graphischen Fähigkeiten des Computers.

Die Adressierung des Bildschirms für jeden Einzelpunkt wird möglich, und das in 16 Vordergrund- und Hintergrundfarben! 40 mächtige Graphikbefehle stehen bereit und können von TI-BASIC oder TI-EXTENDED BASIC aufgerufen werden.

Auch die Graphikausgabe über einen Matrixdrucker ist softwaremäßig implementiert.

Die Graphikgenerierung arbeitet superschnell. Sämtliche Routinen sind in der 16 Bit-Maschinensprache TMS 9900 ASSEMBLER geschrieben. APESOFT XGBASIC bei der Arbeit zu beobachten, ist kreativ und macht Spaß.

Das MECHATRONIC Extended Basic II plus bietet, vorausgesetzt eine 32K-Byte Speichererweiterung ist angeschlossen, die ganze APESOFT GRAFIC.

# 2. PRINZIP VON APESOFT XGBASIC

## 2.1 Arbeitsweise von APESOFT BASIC

Das Prinzip von APESOFT BASIC ist sehr einfach.
APESOFT BASIC ist eine Cursor- oder Plotter-Graphik.

Der Bildschirm ist das Zeichenblatt, und bei Initialisation des "GRAFIC MODUS" steht der Cursor oder Zeichenstift auf Pixelposition (1,120) im Zeichenfenster (linke untere Ecke) bereit, unter dem Winkel 0 Grad zur horizontalen Bildschirmachse loszufahren. Dies ist gleichzeitig der 0-Punkt des benutzerdefinierten Koordinatensystems.

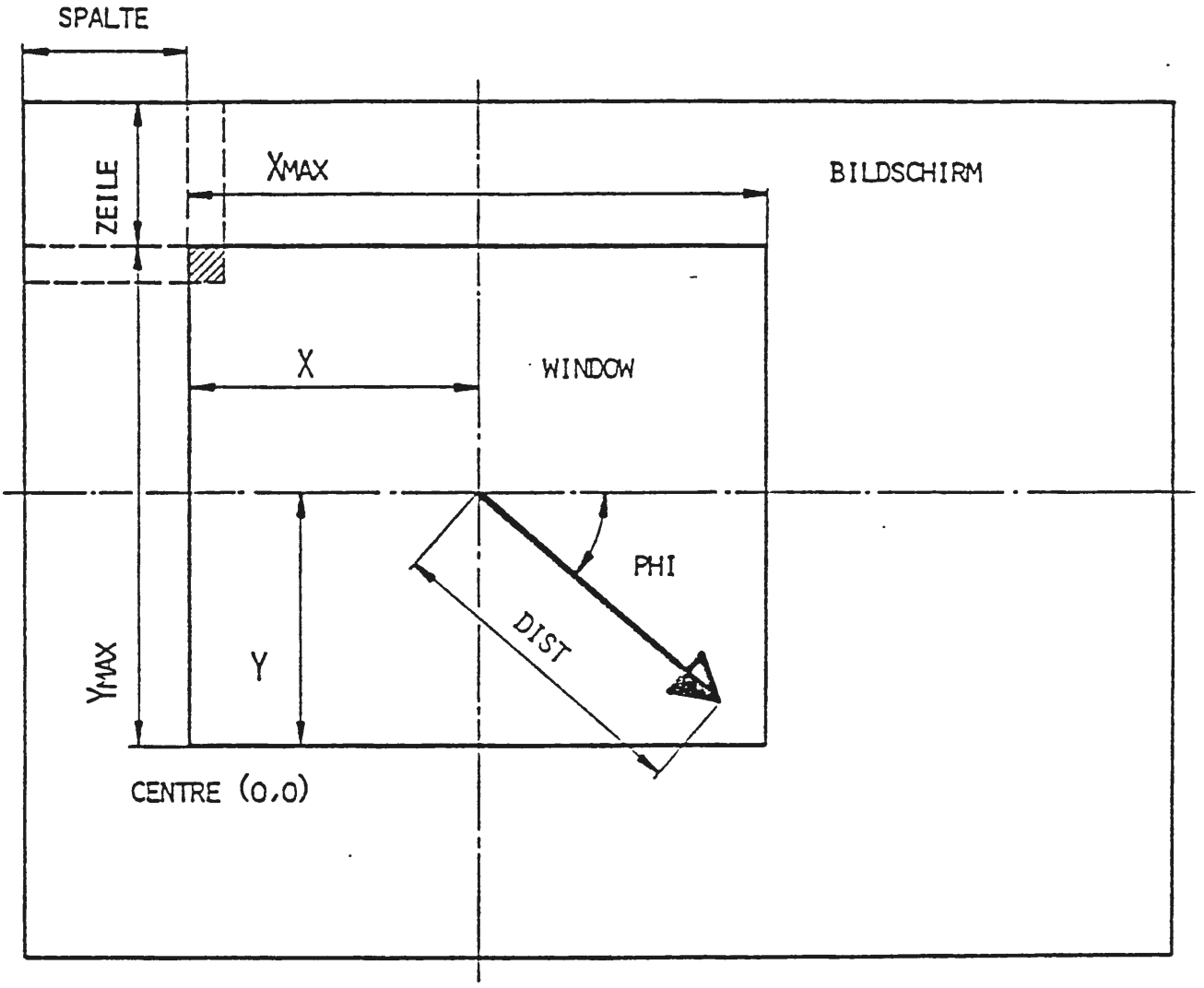
Durch einfache Graphikbefehle wird der Cursor über den Bildschirm gelenkt (siehe Abb. 1). Dabei sind alle Koordinaten wie in der Mathematik definiert.

Es werden Linien gezogen oder gelöscht, Kreise, Rechtecke, Ellipsen, Bögen und vieles mehr gezeichnet. Es entstehen auf superschnelle Weise sehr komplexe geometrische Figuren, Diagramme usw.

Und das in einer Auflösung von 256 mal 192 Punkten in allen den TI-99/4A zur Verfügung stehenden Farben.

Diese Graphiken können für einen späteren Abruf auf Diskette gespeichert werden.

Mit APESOFT EXPANDED GRAFIC BASIC haben Sie ein sehr mächtiges Programmierwerkzeug zur interessanten Gestaltung Ihrer Programme erworben.




```
= CALL LINK("WINDOW",ZEILE,SPALTE)
= CALL LINK("SETTO",X,Y)
= CALL LINK("TURNTO",PHI)
= CALL LINK("MOVE",DIST)
```

Abbildung. 1: Prinzip von APESOFT XGBASIC

# 3. CALL APESOFT

## 3.1 Das Laden von XGBASIC

CALL APESOFT

Mit dem APESOFT Unterprogramm kann das APESOFT EXPANDED GRAFIC BASIC aus dem Modul in die Speichererweiterung ausgelagert werden, und diese ist dann gemäß den nachfolgenden Erläuterungen mittels CALL LINK aufrufbar. Ausgenommen sind die Befehle "GSAVE", "GLOAD" und "BHCOPY". Diese stehen als direkte Unterprogramme zur Verfügung. Bezüglich "BHCOPY" beachten Sie bitte die schon oben angeführten Erklärungen.

Bitte beachten Sie, daß CALL APESOFT nur im Direktmodus ausgeführt werden kann. Ebenfalls ist die Speichererweiterung notwendig. CALL APESOFT schließt alle noch offenen Dateien, lagert die Graficroutinen in die Speichererweiterung aus dem Modul aus, sichert den Bereich im VDP-RAM, den die Graphik benötigt und führt anschließend ein NEW durch, so daß alle Programme im Speicher gelöscht sind. Nach CALL APESOFT sollten Sie kein CALL FILES mehr durchführen, dieser Befehl arbeitet nun nicht mehr richtig.

CALL CLRAPE

Mit diesem Unterprogramm wird wieder der Ausgangszustand hergestellt, d. h. die Graficroutinen werden in der Speichererweiterung gelöscht. Es kann nur nach einem CALL APESOFT durchgeführt werden, sonst erfolgt ein Syntax Error. CALL CLRAPE führt ein CALL INIT durch, setzt das VDP-RAM zurück, schließt alle noch offenen Dateien und führt ein NEW durch, so daß auch hier alle Programme im Speicher gelöscht werden.

Die Anzahl der offenen Dateien in XGBASIC ist durch XGBASIC auf maximal 2 Dateien beschränkt.

### 3.1.1 Fehlmeldungen

Die Fehlermeldungen von XGBASIC bei Exekution von XGBASIC Befehlen sind nicht immer korrekt, da EXTENDED BASIC nicht den in der Konsole integrierten Fehlerreport benutzt.

Fallweise kann es geschehen, daß bei Programmabbruch mit "FCTN CLEAR" oder Anwahl ener Subroutine der TI-99/4A abstürzt und nur durch Aus- oder Wiedereinschalten zur Funktion gebracht werden kann.

# 4. B E F E H L E

APESOFT EXPANDED GRAFIC BASIC besteht aus einer Reihe sehr mächtiger Befehle. Sie erleichtern die Entwicklung von komplizierten Graphiken ungemein.

Es sind BASIC Support Befehle, die alle folgendermaßen aufgebaut sind:

= CALL LINK("PROZEDURNAME",PARAMETER(,PARAMETER...).)

"CALL LINK" schafft die Verbindung zwischen BASIC und ASSEMBLER Programm, der Prozedurname spricht eine bestimmte TMS 9900 Routine an, PARAMETER sind optimal. Ausdrucke in *()* können beliebig oft wiederholt werden. Ein Satz von maximal 15 Parametern kann auf einmal übergeben werden.

Abhängig von den Erfordernissen sind folgende Parameter zulässig:

- numerische Konstante
- numerische Variable
- numerische Arrayelemente
- numerische Ausdrucke
- Stringkonstante
- Stringvariable
- Stringarrayelemente
- Stringausdrucke

Alle konventionellen Fehler werden vom TI-BASIC Interpreter richtig aufgefunden und signalisiert.

Probleme gibt es manchmal bei EXTENDED BASIC, da hier die Fehlermeldungen nicht immer im richtigen Wortlaut erscheinen.

## 4.1 CALL LINK("GRAFIC",MODUS)

Dieser Befehl signalisiert dem Computer den Grafic Modus und initialisiert alle Computerregister dafür.

Es wird abhängig vom Modus ein Graphikfeld (Table) mit maximal 128 vertikalen und 120 horizontalen Linien definiert. Dieser Table-Ausschnitt steht für APESOFT EXPANDED GRAFIC zur Einzeladressierung der Pixel zur Verfügung.

Die Beschränkung auf 128 * 120 = 15.360 Punkte ist notwendig, da sonst im VDP-RAM der Konsole nicht genügend Speicherplatz für BASIC-Programme, Stringvariable, Variable usw. bleibt.

Außerdem benötigt die direkte Adressierung von 192 * 256 = 49.152 Pixel 12 K VDP-RAM Speicher; dabei überschreiben Zeichen und Farbentabellen Pufferadressen des BASIC Interpreters.

Da jedoch der Grafic-Table beliebig auf den Bildschirm gebracht werden kann, sind dennoch 256 * 192 = 49.152 Pixel einzeln zu adressieren.

CALL LINK("GRAFIC",MODUS)
legt folgende interne Parameter fest:

| | |
|---|---|
| PHI-0 | Startwinkel des Cursors (Pixel) |
| TABLEBREITE | 16 |
| VORDERGRUNDFARBE | grün |
| HINTERGRUNDFARBE | schwarz |

Vom Modus hängen weitere interne Parameter ab:

| | |
|---|---|
| MODUS = 0 : | Graficmodus (255 Zeichen für Graphik)<br>15 * 16 Tablezeilen und -spalten |
| X=1, Y=120 : | Startposition des Cursors (Pixel)<br>= 0,0)-Punkt |
| MODUS () 0 : | Textmodus (192 Zeilen für Graphik)<br>12 * 16 Tablezeilen und -spalten<br>Die Befehle "DSPLAY" und "ACCEPT"<br>stehen für Ein/Ausgabeoperationen<br>zur Verfügung |
| X=1, Y=88 : | Startposition des Cursors<br>- (0,0)-Punkt |

CALL LINK("GRAFIC",MODUS) muß der erste Befehl vor einer XGBASIC-Anweisung sein. Sobald dieser Befehl exekutiert ist, geht der Standardcharakterzeichensatz verloren. Die üblichen Bildschirm I/O-Befehle führen nicht mehr zu den gewünschten Resultaten.

CALL LINK("BYEBYE") (siehe Punkt 4.21) hebt jedoch den XGBASIC-Status auf, und die I/O-Befehle arbeiten wie gewohnt.

Mit "BREAK"("FCTN CLEAR") wird das Programm abgebrochen und der Standardstatus des Computers wieder hergestellt.

= CONTINUE

führt aber nicht mehr zur Rückkehr in den XGBASIC-Modus, es sei denn, die erste Anweisung auf = CON ist CALL LINK ("GRAFIC",MODUS)!

Beispiel:

```
100 REM SPIRAL
110 REM ******
120 CALL LINK("GRAFIC",0)
125 CALL LINK("WINDOW",3,8)
130 CALL LINK("SETTO",64,60)
140 FOR DIST=5 TO 50 STEP 5
150 CALL LINK("MOVE",DIST)
160 CALL LINK("TURN",90)
170 NEXT DIST
180 GOTO 180
```

- zeichnet eine hellgrüne rechteckige Spirale auf schwarzem Grund.

Achtung:

MODUS () 0 bei der MINI MEMORY Version von XGBASIC ist sinnlos, da es nur den Grafictable begrenzt, aber die Befehle "DSPLAY" und "ACCEPT" nicht verfügbar sind!

## 4.2 WINDOW

Dieser Befehl bringt Ausschnitte des Grafictables oder den gesamten Grafictable auf den Bildschirm und besitzt zwei Formate.

### 4.2.1 CALL LINK("WINDOW",ZEILE,SPALTE)

Der gesamte Grafictable (16*8 Spalten, 15*8 Zeilen oder 11*8 Zeilen im Textmodus) wird auf Bildschirmposition Spalte (0 bis +32) und Zeile (0 bis +24) gesetzt. Das Graphikfeld kann dabei zur Gänze oder teilweise außerhalb des Bildschirms (24 Zeilen, 32 Spalten) liegen.

Alle vor Ausführung des Befehles

= CALL LINK("WINDOW",SPALTE,ZEILE)

definierten Graphikfelder werden gelöscht, wenn einer der beiden Parameter negativ ist. Ausgewertet werden nur die Absolutbeträge der Parameter.

### 4.2.2 CALL LINK("WINDOW",Z,S,ZA,SA,DZ,DS)

Dieser Befehl bringt Ausschnitte des Graphikfeldes auf den Bildschirm. Dabei bedeutet:

Z,S ..... Bildschirmzeile (Z) und -spalte (S), auf die der linke obere Eckpunkt des Graphikfeldes projeziert wird

ZA,SA ..... linker oberer Eckpunkt des Graphikfeldes, wo mit der Projektion begonnen wird

DZ ..... Anzahl der Grafictable-Zeichen in Zeilenrichtung

DS ..... Anzahl der Grafictable-Zeichen in Spaltenrichtung

Sind Z oder S oder beide negativ, werden alle Graphikfelder, die sich auf dem Bildschirm befinden, gelöscht, bevor der neue Ausschnitt übertragen wird.

Es können beliebig viele Fenster des Tables auf den Bildschirm übertragen werden.

Beispiel:

```
100 REM CIRCLES
110 REM *******
120 CALL LINK("GRAFIC",0)
130 CALL LINK("WINDOW",1,1)
140 FOR R=2 TO 42 STEP 2
150 CALL LINK("CIRCLE",64,60,R)
160 NEXT R
170 CALL LINK("WINDOW",12,18)
175 CALL LINK("WINDOW",1,19)
180 FOR I=1 TO 1000
190 NEXT I
200 CALL LINK("SETCOL",16,5)
210 FOR I=1 TO 500
220 NEXT I
230 CALL LINK("INVERT",1,1,128,120)
240 CALL LINK("WINDOW",4,-8)
250 GOTO 250
```

Zuerst entstehen links oben im Bildschirm eine Reihe konzentrischer Kreise.

Diese werden dann durch die Zeilen 170 und 180 rechts und links unten kopiert (verdreifacht). Zeile 250 rückt die Kreise in Bildschirmmitte zurück, und alle anderen Kreise werden wieder gelöscht. Auf den Weg dahin ändert sich auch die Farbe der Graphik, sie wird inventiert.

Beispiel:

```
100 REM PYRAMIDES
110 REM *********
120 CALL LINK("GRAFIC",0)
130 CALL LINK("WINDOW",1,1,1,1,10,10)
140 CALL LINK("WINDOW",17,1,1,13,17,13)
142 CALL LINK("WINDOW",13,1)
144 CALL LINK("WINDOW",13,17)
150 CALL LINK("TURNTO",45)
160 CALL LINK("SETTO",64,48)
170 FOR A=1 TO 36 STEP 2
180 CALL LINK("RECT",A,A,-A,A,-A,-A,A-A)
190 NEXT A
200 GOTO 200
```

Dieses Beispiel zeichnet gleichzeitig 4 Pyramidenansichten von oben. Dafür sind die Statements in den Zeilen 130 bis 160 verantwortlich.

Das linke obere Fenster ist jedoch nur ein Fragment, da das Statement 130 nur einen Ausschnitt des Graphikfeldes übertragt.

## 4.3 SETBLE/CLTBLE/TABLE

Dies sind Anweisungen für die Generierung des Graphikfeldes.

### 4.3.1 CALL LINK("SETBLE",BREITE)

Diese Anweisung dimensioniert das Graphikfeld.

BREITE ..... Spaltenanzahl des Graphikfeldes; diese kann von 1 bis 32 variiert werden

Es stehen im Textmodus 191 und im Graphikmodus 255 Zeichen für den Grafictable bereit.

Die von der Breite abhängige Höhe des Grafictables ergibt sich aus:

```
HÖHE = INT (255/BREITE)      für Graphikmodus
HÖHE = INT (191/BREITE)      für Textmodus
```

Auf diese Weise können höhere oder breitere Graphiken generiert werden.

Wichtig:

Auf jeden "SETBLE"-Befehl muß ein "WINDOW"-Befehl folgen, der den Bildschirm neu ordnet, sonst erfolgt die Graphikgenerierung nicht wie gewünscht.

"SETBLE" legt gleichzeitig den Mittelpunkt des benutzerdefinierten Koordinatensystems neu fest, und zwar auf Pixelposition:

```
CENTRX = 1
CENTRY = HÖHE x 8 = YMMAX
```

### 4.3.2 CALL LINK("CLTBLE")

Dieser Befehl löscht das Graphikfeld und damit die Graphik. Die durch "WINDOW"-Anweisungen auf den Bildschirm gebrachten Feldausschnitte bleiben jedoch für die Aufnahme von Graphiken erhalten.

### 4.3.3 CALL LINK("TABLE",Z,S,XMAX,YMAX,BYTES)

Diese Anweisung liefert die momentanen Parameter des Grafictables (-feldes) in folgende Variable zurück:

| | |
|---|---|
| Z ..... | Zeilenanzahl des Feldes |
| S ..... | Spaltenanzahl des Feldes |
| IMAX ..... | maximale Pixelspalten des Feldes |
| YMAX ..... | maximale Pixelzeilen des Feldes |
| BYTES ..... | Anzahl der für die Graphik zur Verfügung stehenden Bytes |

Die Charakterbytes im Graphikfeld sind beginnend mit Zeile 1 und Spalte 1 immer aufsteigend angeordnet.

Die Bytenummer berechnet sich folgendermaßen:

CHAR# = (ZEILE - 1) x BREITE + SPALTE - 1

| | |
|---|---|
| ZEILE ..... | Zeile des Graphikfeldes |
| SPALTE ..... | Spalte des Graphikfeldes |
| BREITE ..... | absolute Breite des Graphikfeldes |
| CHAR# ..... | Charakterbytenummer des Tables |

## 4.4 SETCOL

Dieser Befehl besitzt zwei Formate und legt die Vordergrund- und Hintergrundfarbe der Graphik fest.

Es können alle von BASIC her bekannten 16 Farben jeweils als Vordergrund- oder Hintergrundfarben verwendet werden. Mehrere verschiedene Vordergrund- und Hintergrundfarben können gleichzeitig in einer Graphik eingesetzt werden.

### 4.4.1 CALL LINK("SETCOL",VORDERGRUND-,HINTERGRUNDFARBE)

Sind nur 2 Parameter in der Parameterliste vorhanden, so ändern sich Vorder- und Hintergrundfarbe gleichzeitig für die gesamte Graphik.

### 4.4.2 CALL LINK("SETCOL",N,FG,BG*(N1,FG1,BG1....)*)

Sind mehr als 2 Parameter vorhanden, so legt "SETCOL" die Vordergrundfarbe (FG) und Hintergrundfarbe (BG) für den mit N spezifizierten Zeichensatz fest.

Dabei sind die Zeichensätze für die Zeilen des Grafictables wie folgt definiert:

8 aufeinanderfolgende Bytes bilden einen Zeichensatz (0-7, 8-15, ... usw.).
Farben für Zeilen und Spalten des Graphikfeldes (Breite = 16) =

| ZEILEN | SPALTEN 1-7 | SPALTEN 8-16 |
|---|---|---|
| 1 | 1 | 2 |
| 2 | 3 | 4 |
| 3 | 5 | 6 |
| 4 | 7 | 8 |
| 5 | 9 | 10 |
| 6 | 11 | 12 |
| 7 | 13 | 14 |
| 8 | 15 | 16 |
| 9 | 17 | 18 |
| 10 | 19 | 20 |
| 11 | 21 | 22 |
| 12 | 23 | 24 |
| 13 | 25 | 26 |
| 14 | 27 | 28 |
| 15 | 29 | 30 |

Innerhalb der in der vorhergehenden Tabelle angeführten Spezifikationen können die Vordergrund- und Hintergrundfarben völlig frei definiert werden.

Dadurch wird eine Vielzahl von Farbkombinationen möglich. Mit einer Parameterliste können gleichzeitig bis zu 5 Farbensätze übergeben werden.

Beispiel:

```
100 REM LEAF
110 REM ****
120 RANDOMIZE
130 CALL LINK("GRAFIC",0)
135 CALL LINK("WINDOW",3,8)
140 FOR PHI=0 TO 90 STEP 5
150 CALL LINK("SETTO",64,119)
160 CALL LINK("TURNTO".PHI)
170 CALL LINK("MOVE",1.2*PHI)
180 CALL LINK("TURNTO".180-PHI)
190 CALL LINK("SETTO",64,119)
200 CALL LINK("MOVE",1.2*PHI)
210 NEXT PHI
220 FOR DELAY=1 TO 500
230 NEXT DELAY
240 FG=2+14*RND
250 BG=2+14*RND
260 CALL LINK("SETCOL",FG,BG)
270 FOR DELAY=1 TO 500
280 NEXT DELAY
290 CALL LINK("INVERT",1,1,128,120)
300 GOTO 220
```

Zeile 220 legt Vordergrund- und Hintergrundfarbe der Graphik fest, Zeile 290 inventiert die Graphik.

## 4.5 CALL LINK("INVERT",X,Y,DX,DY)

Mit "INVERT" werden Ausschnitte der Graphik invertiert. Folgende Parameter sind erforderlich:

X,Y ..... Pixelposition des linken oberen Ecks des Graphikausschnittes, welcher invertiert wird

DX ..... Pixelspaltenposition des Ausschnittes

DY ..... Pixelzeilenposition des Ausschnittes

Dabei werden in den Ausschnitt Pixel, die gesetzt sind, gelöscht und umgekehrt.

## 4.6 CALL LINK("CLSCRN")

Dieser Befehl wirkt ähnlich dem von BASIC bekannten Befehl "CALL CLEAR".

Er löscht den Bildschirm; die Graphik im Table und alle internen Cursor Parameter bleiben jedoch erhalten.

Beispiel:

```
100 REM RANDOM STRAIGHT LINES
110 REM **********************
120 CALL LINK("GRAFIC",0)
125 CALL LINK("WINDOW",-3,8)
130 VG =Z+13*RND
140 IF VG<=3 THEN 130
150 CALL LINK("SETCOL",VG,2)
160 FOR I=1 TO 20
170 X=124*RND
180 Y=120*RND
190 CALL LINK("MOVETO",X,Y)
200 NEXT I
210 FOR J=1 TO 250
210 NEXT J
220 CALL LINK("CLSCRN")
230 FOR I=1 TO 500
240 NEXT I
250 CALL LINK("WINDOW",1,1)
260 GOTO 125
```

Dieses Programmbeispiel zeichnet ausgehend von Position (1.120) in zufälliger Richtung aufeinanderfolgende Linien (Zeile 100) und löscht die Graphik (Zeile 220).

Nach kurzer Pause zeigt der "WINDOW"-Befehl (250), daß nur der Bildschirm mit Zeile 220 gelöscht wurde, die Graphik jedoch im Table erhalten geblieben ist.

"WINDOW" mit einem negativen Zeilenparameter (Zeile 125) übt zunächst ein "CLSCRN" aus, bevor der Grafictable-Ausschnitt auf den Bildschirm gebracht wird!

## 4.7 CALL LINK("CENTRE",X,Y)

Diese XGBASIC-Anweisung legt das benutzerdefinierte Koordinatensystem fest:

X ..... X-Koordinate des 0-Punktes im Graphikfeld
Y ..... Y-Koordinate des 0-Punktes im Graphikfeld

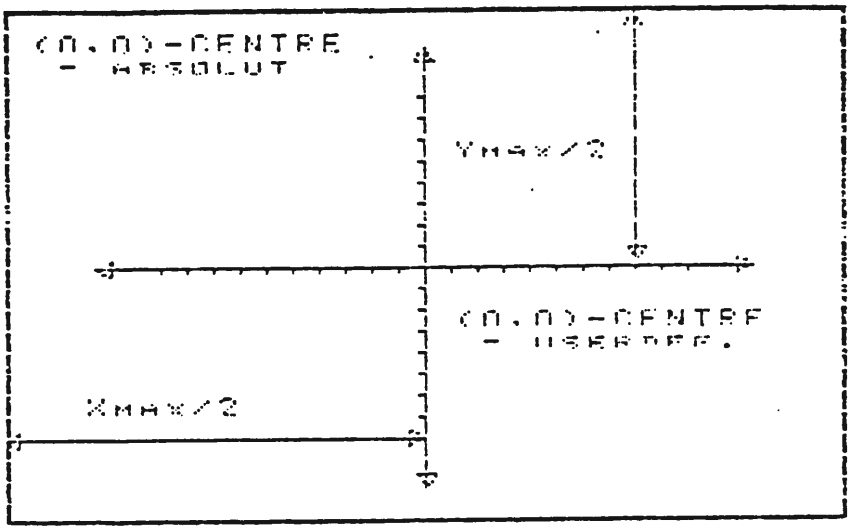


Abbildung 4.7.1: Benutzerdefiniertes Koordinatensystem

Nach einem "GRAFIC"-Befehl befindet sich das Zentrum exakt auf Position (1,120) (linke untere Ecke des Tables).

Mit "CENTRE" kann dieser 0-Punkt beliebig verschoben werden, auch außerhalb des Tables.

Beispiel:

```
100 REM SYSTEM OF CO-ORDINATES
110 REM ************************
120 CALL LINK("GRAFIC",1)
130 CALL LINK("WINDOW",7,8)
132 CALL LINK("SETTO",1,1)
134 CALL LINK("RECT",127,-87)
140 CALL LINK("CENTRE",64,44)
150 CALL LINK("AXIS",0,60,60,4,0,40,40,2)
160 CALL LINK("WRITE",8,10,"(0,0)")
170 CALL LINK("DSPLAY",1,5,26,">CALL LINK(""CENTRE"".
64,44)")
180 CALL LINK("DSPLAY",5,5,22,"(-64,+44) (64,+44)")
190 CALL LINK("DSPLAY",20,5,22,"(-64,-44) (64,-44)")
200 CALL LINK("BHCOPY",7,"RS232.BA=9600.DA=8.CR",
CHR$(27)&"A"&CHR$(8))
210 STOP
```

Dieses Programmbeispiel fertigt die Zeichnung für Abbildung 4.7.1.

## 4.8 SETTO/RESET/IFSET

### 4.8.1 CALL LINK("SETTO",X,Y*(,X1,Y1....)*)

"SETTO" setzt Pixel mit den Koordinaten Zeile Y, Spalte X. Spalte 1-128 und Zeile 1-120 befinden sich auf dem Bildschirm.

Hinsichtlich des Bereiches dieser Werte bestehen keine Einschränkungen, die Koordinaten können positiv oder negativ, beliebig groß und auch Gleitkommazahlen sein. Sie werden intern auf die nächste Integerzahl gerundet. Zahlen größer als 32.768 und kleiner als -32.767 werden falsch dargestellt. Eine Fehlermeldung erfolgt bei dieser Bereichsüberschreitung nicht! Mit der Parameterliste können maximal 7 Punkte gleichzeitig difiniert werden. Der interne Winkel PHI des Cursors bleibt durch "SETTO" unverändert.

### 4.8.2 CALL LINK("RESET",X,Y*(,X1,Y1....)*)

"RESET" löscht Pixel mit den Koordinaten X,Y.
Es gelten alle für "SETTO" angegebenen Feststellungen gleichermaßen.

### 4.8.3 CALL LINK("IFSET",X,Y,VAR*(,X1,Y1,VAR1....)*)

Dieses Statement prüft, ob ein Pixel mit den Koordinaten X,Y gesetzt ist und liefert in den Variablen VAR folgende Werte zurück:

| | | |
|---|---|---|
| Pixel (X,Y) | gesetzt | VAR=-1 |
| Pixel (X,Y) | gelöscht | VAR= 0 |
| Pixel (X,Y) | außerhalb des Fensters | VAR=+1 |

Bis zu 5 Pixel können in einer Parameterliste gleichzeitig abgefragt werden.

Ansonsten gelten die für "SETTO" und "RESET" aufgestellten Bedingungen.

Beispiel:

```
100 REM SINE
110 REM ****
120 CALL LINK("GRAFIC",0)
125 CALL LINK("WINDOW",3,8)
127 CALL LINK("CENTRE",4,60
130 PI180=4*ATN(1)/180
140 CALL LINK("AXIS",0,0,118,5,0,50,50,5)
150 REM
160 FOR PHI=0 TO 360 STEP 2
170 X=PHI/3+20
180 Y=20*SIN(PHI*PI180)
190 CALL LINK("SETTO",X,Y,X,Y*1.5,X,Y*2)
200 NEXT PHI
210 FOR PHI=0 TO 360 STEP 2
220 X=PHI/3+20
230 Y=20*SIN(PHI*PI180)
240 CALL LINK("RESET",X,Y*1.5,X,Y*2)
250 NEXT PHI
260 X=INT(128*RND)+1
270 Y=INT(120*RND)+1
280 CALL LINK("IFSET",X,Y,A)
290 IF A=0 THEN 260
300 CALL LINK("BYEBYE")
310 CALL CLEAR
320 PRINT "PUNKT X=";X;"Y=";Y;":IST GESETZT"
330 STOP
```

Dieses Programmbeispiel zeichnet 3 Sinuslinien (160-200), löscht sie wieder (210-250), und bleibt dann solange in einer Warteschleife, bis es einen gesetzten Punkt gefunden hat (280-290).

## 4.9 MOVE/REMOVE

### 4.9.1 CALL LINK("MOVE",DIST)

Zeichnet eine Linie der Länge DIST, ausgehend von der momentanen Position X,Y des Cursors mit dem momentanen internen Winkel PHI. Die Lage DIST entspricht waagrecht und senkrecht exakt der Anzahl der Pixel, dazwischen hängt die Pixelanzahl vom entsprechenden Winkel ab. Nach Ausführung von DIST nimmt der Cursor die Endkoordinaten (letztes gespeichertes Pixel) an. PHI bleibt unverändert.

Positive Werte von DIST gehen in die augenblickliche Richtung des Cursors, negative 180 Grad entgegengesetzt. DIST kann jeden Wert annehmen, wobei alle unter Punkt 4.8.1 angegebenen Bereichseinschränkungen gelten.

### 4.9.2 CALL LINK("REMOVE",DIST)

"REMOVE" wirkt wie "MOVE", jedoch werden hier alle Pixel ab Position X,Z bis zur DIST entfernten neuen Position des Cursors gelöscht.

Beispiel:

```
100 REM STAR
110 REM ****
120 CALL LINK("GRAFIC",0)
125 CALL LINK("WINDOW",3,8)
130 CALL LINK("WRITE",15,3,"@APESOFTGRAFIC")
140 FG=3+13*RND
150 CALL LINK("SETCOL",FG,2)
160 CALL LINK("SETTO",2,60)
170 CALL LINK("TURNTO",36)
180 FOR I=1 TO 10
190 CALL LINK("TURN",108)
200 CALL LINK("MOVE",60)
210 NEXT I
220 CALL LINK("SETTO",2,60)
230 CALL LINK("TURNTO",36)
240 FOR I=1 TO 10
250 CALL LINK("TURN",108)
260 CALL LINK("REMOVE",60)
270 NEXT I
280 GOTO 140
```

Durch geschickte Anwendung weniger Befehle wird ein Stern auf den Bildschirm gezaubert, wieder gelöscht und das Spiel beginnt mit neuer Farbe von vorne.

## 4.10 MOVETO/REMVTO

### 4.10.1 CALL LINK("MOVETO",X,Y,*(,X1,Y1....)*)

"MOVETO" zeichnet eine Linie von der augenblicklichen internen Position des Cursors bis zu der nächsten durch das Parameterpaar X,Y festgelegten Position.

Die Parameterlinie kann maximal 7 Positionen festhalten. Sind mehr als 2 Parameter angegeben, so wird die Linie immer von der vorhergehenden Position zu nächsten weitergezogen.

Nach Ausführung von "MOVETO" nimmt der Cursor die letzte Position der Parmeterliste an.

Der interne Winkel und die Farben bleiben durch "MOVETO" unverändert.

Linien können auch außerhalb des Graphikfensters hinausgezogen werden.

### 4.10.2 CALL LINK("REMVTO",X,Y*(,X1,Y1....)*)

"REMVTO" arbeitet wie "MOVETO" mit dem Unterschied, daß hier die Linien gelöscht werden.

Es gelten alle für "MOVETO" aufgestellten Bedingungen.

Beispiel:

```
100 REM THREAD GRAPHIC
110 REM ***************
120 CALL LINK("GRAFIC",0)
125 CALL LINK("WINDOW",3,8)
130 D=2.6
140 Z=125
150 S1=128
160 FOR S=1 TO 126 STEP 5
170 Z=Z-5
180 S1=S1-D
190 CALL LINK("SETTO",S,120)
200 CALL LINK("MOVETO",S1,Z)
210 NEXT S
220 Z=125
230 S1=1
240 FOR S=126 TO 1 STEP -5
250 Z=Z-5
260 S1=S1+D
270 CALL LINK("SETTO",S,120)
280 CALL LINK("MOVETO",S1,Z)
290 NEXT S
300 GOTO 300
```

## 4.11 TURN/TURNTO

### 4.11.1 CALL LINK("TURN",PHI)

Dieser Befehl addiert zum augenblicklichen internen Winkel des Cursors den Winkel PHI in Dezimalgraden.

Die Grenzen für den Winkel sind +/-2047 Grad. Der Winkel wird intern auf 0-360 Grad moduliert.

Die Winkelfunktionen werden intern im Computer über Interpolationstabellen gebildet. Da insbesondere beim MINI MEMORY Modul der Speicherplatz sehr beschränkt ist, wird hier der Winkel im 5 Grad Bereich interpoliert. Dies führt bei Zwischenwerten nicht immer zu exakten Resultaten.

### 4.11.2 CALL LINK("TURNTO",PHI)

Stellt den internen Winkel unbedingt auf den Winkel PHI (Grad) ein. Es gelten alle für "TURN" aufgestellten Einschränkungen.

Beispiel:

```
100 REM OKTAGONS
110 REM ********
120 CALL LINK("GRAFIC",0
130 CALL LINK("WINDOW",4,5)
140 CALL LINK("WINDOW",6,19)
150 DIST=2
160 FOR S=28 TO 108 STEP 4
170 CALL LINK("SETTO",S,42)
180 CALL LINK("TURNTO",90)
190 DIST=DIST+2
200 FOR I=1 TO 8
210 CALL LINK("TURN",45)
220 CALL LINK("MOVE",DIST)
230 NEXT I
240 NEXT S
250 GOTO 250
```

## 4.12 RECT/CLRECT

### 4.12.1 CALL LINK("RECT",A,B*(,A1,B1....)*)

Ausgehend von der augenblicklichen Position und den internen Winkel des Cursors zeichnet "RECT" Rechtecke, und zwar in der Folge

A -) B-) A -) B

Das Rechteck dreht im Uhrzeigersinn, falls B positiv ist. Wie die Vorzeichen der Seitenlängen die endgültige Position des Rechteckes beeinflussen, zeigt nachfolgendes Beispiel.

Der interne Winkel und die Position des Cursors werden durch "RECT" nicht beeinflußt. Die Werte für die Seitenlängen des Rechtecks können beliebig groß sein.

Es können maximal 7 Rechtecke mit einer Parameterliste übergeben werden.

Sie beginnen aber alle am selben Ausgangspunkt und enden auch wieder dort.

### 4.12.2 CALL LINK("CLRECT",A,B*(,A1,B1....)*)

Arbeitet völlig identisch mit "RECT", nur mit dem Unterschied, daß "CLRECT" die Rechtecke löscht.

Es gelten alle für "RECT" aufgestellten Bedingungen.

Beispiel:

```
100 REM RECTANGLES
110 REM **********
120 CALL LINK("GRAFIC",0)
125 CALL LINK("WINDOW",3,8)
130 CALL LINK("SETTO",64,60)
140 A=40
150 B=20
160 CALL LINK("RECT",A,B,A,-B,-A,B,-A,-B)
170 FOR I=1 TO 250
180 NEXT I
190 CALL LINK("CLRECT",A,B,A,-B,-A,B,-A,-B)
200 CALL LINK("TURN",45)
210 GOTO 160
```

Zeile 160 zeichnet mit einem einzigen Befehl 4 Rechtecke,
Zeile 190 löscht diese wieder.
Zeile 200 dreht den internen Winkel um 45 Grad weiter.

## 4.13 CIRCLE/CLCRCL

### 4.13.1 CALL LINK("CIRCLE",X,Y,R*(,X1,Y1,R1....)*)

"CIRCLE" zeichnet einen Kreis mit dem Mittelpunkt X,Y und dem Radius R.

Es können maximal 5 unterschiedliche Kreise mit einer Parameterliste gezeichnet werden.

Die Parameter können alle Werte annehmen. Für den Radius wird automatisch der Absolutbetrag ausgewertet. Ist der Radius 0, setzt "CIRCLE" einen Punkt. Nach Ausführung der Prozedur "CIRCLE" hat der Cursor die Position des letzten Kreismittelpunktes angenommen.

Der interne Winkel PHI bleibt unverändert.

Durch interne Rundungsfehler kann es vorkommen, daß die Kreisbögen nicht ganz glatt aussehen.

### 4.13.2 CALL LINK("CLCRCL",X,Y,R*(,X1,Y1,R1....)*)

"CLCRCL" arbeitet völlig identisch mit "CIRCLE", nur werden hier die Kreise gelöscht.

Es gelten alle für "CIRCLE" aufgestellten Bedingungen.

Beispiel:

```
100 REM CIRCLES
110 REM *******
120 PI=4*ATN(1)
130 CALL LINK("GRAFIC",0)
135 CALL LINK("WINDOW",3,8)
137 CALL LINK("CENTRE",64,60)
140 CALL LINK("CIRCLE",0,0,30)
150 FOR PHI=0 TO 2*PI STEP PI/16
160 CALL LINK("CIRCLE",30*COS(PHI),30*SIN(PHI),30)
170 NEXT PHI
180 GOTO 180
```

## 4.14 ARCUS/CLARCS

Diese Befehle gelten nicht für MINI MEMORY Modul.

### 4.14.1 CALL LINK("ARCUS",X,Y,R,PHI,DPHI* (,X1,Y1,R1,PHI1,DPHI1....)*)

"ARCUS" zeichnet Kreisbögen mit folgenden Parametern:

X,Y ..... Mittelpunkt des Kreisbogens

R ..... Radius des Kreisbogens

PHI ..... Startwinkel (absolut) des Kreisbogens

DPHI .... Bogenwinkel des Kreisbogens

Bis zu 3 Bögen gleichzeitig können mit einem Befehl generiert werden. Infolge interner Rundungsfehler und Interpolationsbestimmung der Winkelfunktionen sind die Resultate nicht immer befriedigend.

Die Koordinaten des Cursors beschreiben nach Exekution von "ARCUS" den letzten gezeichneten Bogenpunkt.

### 4.14.2 CALL LINK("CLARCS",X,Y,R,PHI,DPHI* (,X1,Y1,R1,PHI1,DPHI....)*)

"CLARCS" arbeitet identisch wie "ARCUS", nur werden hier alle Bogenpunkte gelöscht.

## 4.15 ELLIPS/CLLIPS

### 4.15.1 CALL LINK("ELLIPS",X,Y,A,B*(,X1,Y1,A1,B....)*)

"ELLIPS" zeichnet Ellipsen mit dem Achsenmittelpunkt X,Y, großer Halbachse A, kleiner Halbachse B und Neigung PHI der großen Halbachse.

Es können maximal 3 verschiedene Ellipsen mit einer Parameterliste generiert werden. Die Parameter können beliebige Werte annehmen, ausgenommen 0. Für die große und kleine Halbachse wird automatisch der Absolutbetrag eingesetzt.

Nach Ausführung von "ELLIPS" hat der Cursor die Werte der Halbachsenschnittpunkte angenommen. Der interne Winkel PHI des Cursors bleibt unverändert.

Durch interne Rundungsfehler kann es vorkommen, daß die Ellipsenbogen nicht ganz glatt aussehen. Dies tritt insbesondere bei Neigungen der Hauptachsen zur Waagrechten und Senkrechten auf, da die Koordinatentransformationen mit interpolierten Winkelfunktionen erfolgen.

### 4.15.2 CALL LINK("CLLIPS",X,Y,A,B*(,X1,Y1,A1,B1....)*)

"CLLIPS" arbeitet wie "ELLIPS", mit dem Unterschied, daß hier Ellipsen gelöscht werden.

Es gelten alle für "ELLIPS" aufgestellten Bedingungen sinngemäß.

Beispiel:

```
100 REM CONE
110 REM ****
120 CALL LINK("GRAFIC",0)
125 CALL LINK("WINDOW",3,8)
130 CALL LINK("SETCOL",15,2)
140 M=1
150 A=42
160 B=22
170 FOR Y=81 TO 1 STEP -8
180 IF M=0 THEN 220
190 M$="ELLIPS"
200 REM
210 GOTO 230
220 M$="CLLIPS"
230 CALL LINK(M$,64,Y,A,B)
240 A=A-4
250 B=B-2
260 NEXT Y
270 IF M=1 THEN 140
280 M=0
290 GOTO 150
```

Durch Steuerung über die Statements 270-290 wird immer, wenn M = 1, ein Kegel gezeichnet. Man beachte, daß als PROZEDURNAME auch eine Stringvariable übergeben werden kann!

Durch "FCIN CLEAR" wird das Programm abgebrochen.

## 4.16 CALL LINK("VALUES",X,Y,PHI,FG,BG)

"VALUES" liefert die momentanen internen Parameter in die Variablenliste zurück.

X ..... Cursorspalte

Y ..... Cursorzeile

PHI ..... Cursorwinkel

FG ..... Vordergrundfarbe

BG ..... Hintergrundfarbe

Da der Winkel intern moduliert wird, liegt er immer zwischen 0-360 Grad, unabhängig von der vorhergehenden Eingabe!

Beispiel:

```
100 REM EXAMPLE
110 REM *******
120 CALL LINK("GRAFIC",0)
125 CALL LINK("WINDOW",3,8)
130 FOR I=1 TO 11
140 CALL LINK("SETTO",64,60)
150 CALL LINK("MOVE",30)
160 CALL LINK("VALUES",X,Y,PHI,FG,BG)
170 CALL LINK("CIRCLE",X,Y,20)
180 CALL LINK("TURN",30)
190 NEXT I
200 GOTO 200
```

Zeile 150 zieht vom Mittelpunkt des Graphikfensters aus eine Linie der Länge 30.

Zeile 160 ermittelt die letzte Cursorposition,
Zeile 170 nimmt diese als Mittelpunkt für einen Kreis mit dem Radius 20.

## 4.17 AXIS/HSTDIA/CRCDIA

APESOFT stellt mit EXPANDED GRAFIC BASIC 3 Hilfsprogramme für besondere graphische Applikationen zur Verfügung.

Diese Programme heißen "AXIS", "HSTDIA" und "CRCDIA".

### 4.17.1 CALL LINK("AXIS",X,LENIR,LENXL,DELTAX, Y,LENYU,LENYD,DELTAY)

"AXIS" zeichnet ein Achsenkreuz mit folgenden Parametern:

X,Y ..... Mittelpunkt des Achsenkreuzes

LENXL ... Linke X-Halbachse (Länge)

LENXR ... Rechte X-Halbachse (Länge)

DELTAX .. Schrittweite des X-Rasters

LENYU ... Obere Y-Halbachse (Länge)

LENYD ... Untere Y-Halbachse (Länge)

DELTAY .. Schrittweite des Y-Rasters

Von sämtlichen Werten wird nur der Absolutbetrag genommen. Ist für eine Halbachsenlänge ein Wert von 0 angegeben, so wird diese Halbachse nicht gezeichnet.

Ist für den Raster ein Wert von 0 oder ein größerer Wert als die Lage der betreffenden Halbachse gegeben, so wird kein Raster gezeichnet.

Die Koordinaten des Cursors nehmen nach Ausführung von "AXIS" den Mittelpunkt des Achsenkreuzes an.

Der interne Winkel PHI wird verändert. Das Koordinatenkreuz muß nicht zur Gänze innerhalb des Graphikfensters liegen.

Beispiel:

```
100 REM ZYKLOM
110 REM ******
120 CALL LINK("GRAFIC",0)
125 CALL LINK("WINDOW",3,8)
130 PI2=8*ATN(1)
140 CALL LINK("AXIS",8,0,116,4,60,60,59,4)
150 X=7
160 CALL LINK("SETTO",X,40)
170 FOR PHI=0 TO 3*PI2 STEP PI2/16
180 X=X+2
190 Y=20*(SIN(2*PHI)+2*COS(PHI))+60
200 CALL LINK("MOVETO",X,Y)
210 NEXT PHI
220 GOTO 220
```

### 4.17.2 CALL LINK(HSTDIA,X,Y,BREITE,HÖHE,TIEFE)

"HSTDIA" zeichnet ein Blockdiagramm mit folgenden Parametern:

X,Y ..... Koordinaten der linken unteren Ecke des Blocks

BREITE .. Breite des Blockdiagramms

HÖHE .... Höhe des Blockdiagramms

TIEFE ... Tiefe des Blockdiagramms

Von sämtlichen Werten wird nur der Absolutbetrag genommen.

Beispiel:

```
100 REM HISTOGRAMS
110 REM **********
120 CALL LINK("GRAFIC",0)
125 CALL LINK("WINDOW",3,8)
130 FOR N=2 TO 20
140 CALL LINK("SETCOL",N,14,2)
150 NEXT N
160 CALL LINK("AXIS",8,10,120,8,20,90,0,4)
170 CALL LINK("HSTDIA",16,22,12,80,6)
180 CALL LINK("HSTDIA",40,22,12,45,6)
190 CALL LINK("HSTDIA",70,22,12,67,6)
200 CALL LINK("HSTDIA",94,22,16,12,6)
210 CALL LINK("WRITE",15,3,"HISTOGRAMS")
220 GOTO 220
```

### 4.17.3 CALL LINK("CRDIA",X,Y,RADIUS,PHI,DPHI* (,X1,Y1,PHI1,DPHI1...)*)

"CRCDIA" zeichnet ein Kreisdiagramm mit folgenden Parametern:

X,Y ..... Koordinaten des Kreissegmentmittelpunktes

RADIUS .. Radius des Kreissegmentes

PHI1 .... Startwinkel des Kreissegmentes (absolut)

DPHI .... Endwinkel des Kreissegmentes (absolut)

Von sämtlichen Werten wird nur der Absolutwert genommen.

Beispiel:

```
100 REM CIRCULAR DIAGRAMS
110 REM ******************
120 CALL LINK("GRAFIC",0)
125 CALL LINK("WINDOW",3,8)
130 FOR N=1 TO 13 STEP 2
140 CALL LINK("SETCOL",N,16,2,N+1,14,2,N+14,9,2,N+15,
13,2)
150 NEXT N
160 CALL LINK("CRCDIA",64,60,38,180,276)
170 CALL LINK("CRCDIA",68,60,38,276,450)
180 CALL LINK("CRCDIA",68,64,38,90,180)
190 CALL LINK("WRITE",14,2,"CIRC-DIAGRAMS")
200 GOTO 200
```

## 4.18 WRITE/DSPLAY/ACCEPT

Diese Befehle erlauben das Mischen von Graphiken und Text und erweitern die I/O-Operationen. Sie können auch im normalen BASIC Mode des Computers eingesetzt werden.

### 4.18.1 CALL LINK("WRITE",Z,S,STRING*(Z1,S1,STRING1....)*)

"WRITE" erlaubt das Mischen von Graphik und Text unmittelbar im Graphikfeld.

An Position Zeile (Z) und Spalte (S) des Graphikfensters schreibt "WRITE" einen String (STRING).

Grenzen: Z ..... 1 bis Tablehöhe (max. 24)
S ..... 1 bis Tablebreite (max. 32)

Geht der String nicht in die betreffende Zeile, wird der Rest abgetrennt.

Es können maximal 5 Strings mit einer Parameterliste übergeben werden.

Es stehen die ASCII-Codes zur Verfügung (Groß- und Kleinbuchstaben, Ziffern und Sonderzeichen).

### 4.18.2 CALL LINK("DSPLAY",Z,S,SIZE,VAR$)

Dieser Befehl entspricht dem bekannten Befehl "DISPLAY AT" von EXTENDED BASIC und benützt folgende Parameter:

| | | |
|---|---|---|
| Z ..... | Bildschirmzeile | (1-24) |
| S ..... | Bildschirmspalte | (1-32) |
| SIZE .... | Stringlänge | (max 32) |
| VAR$ .... | Stringvariable | (max 32 Zeichen) |

Durch dieses Statement werden SIZE Zeichen des Strings VAR$ an die Bildschirmposition (Z,S) gebracht.

Dabei werden unter dem String liegende Graphikwerte gelöscht (Unterschied zu "WRITE"!).

Ist SIZE negativ, werden SIZE Positionen vor der Ausgabe des Strings nicht gelöscht!
(Dieser Befehl gilt nicht für MINI MEMORY Modul!)

"DSPLAY" steht im "GRAFIC MODUS" nur zur Verfügung, wenn MODUS () 0 gesetzt wurde.

### 4.18.3 CALL LINK("ACCEPT",Z,S,SIZE,VAR$)

Dieser Befehl entspricht dem bekannten Befehl "ACCEPT AT" von EXTENDED BASIC und benutzt folgende Parameter:

| | | |
|---|---|---|
| Z ..... | Bildschirmzeile | (1-24) |
| S ..... | Bildschirmspalte | (1-32) |
| SIZE .... | Stringlänge | (max 32) |
| VAR$ .... | Stringvariable | (max 32 Zeichen) |

"ACCEPT" übernimmt SIZE Zeichen eines Strings an der Bildschirmposition (Z,S).

Ist SIZE positiv, werden vorher SIZE Positionen des Bildschirmes gelöscht.

Während der Eingabe sind folgende Tasten aktiv:

UALPHA ...ASCII-Codes 32 - 96

(= ..... Cursor links

=) ..... Cursor rechts

ERAGE ... löscht die Eingabe

ENTER ... übernimmt String

REPEAT .. Repeatfunktion

"ACCEPT" steht im "GRAFIC MODUS" nur zur Verfügung, wenn MODUS () 0 gesetzt wurde.

Achtung:

"FCTN QUIT" und "FCTN CLEAR" sind während dieses Befehles wirkungslos.

Während der Eingabe können alle Tastencodes angesprochen werden, für Textzwecke sind aber nur UALPHAs sinnvoll (Großbuchstaben).

Beispiel:

```
100 REM STAR1
110 REM *****
120 CALL LINK("GRAFIC",1)
125 CALL LINK("WINDOW",3,8)
130 CALL LINK ("SETCOL",12,2)
140 CALL LINK("SETTO",5,60)
150 CALL LINK("MOVE",120)
160 CALL LINK("SETTO",64,1)
170 CALL LINK("TURN",90)
180 CALL LINK("MOVE",120)
190 S1=5
200 Z1=60
210 Z2=36
220 S2=64
230 S3=124
240 Z3=60
250 Z4=84
260 S4=64
270 FOR I=1 TO 10
280 CALL LINK("SETTO",S1,Z1)
290 CALL LINK(MOVETO",S2,Z2,S3,Z3,S4,Z4,S1,Z1)
300 S1=S1+4
310 Z2=Z2-4
320 S3=S3-4
330 Z4=Z4+4
340 NEXT I
350 CALL LINK("WRITE",15,1,"@APESOFT GRAFIC")
360 CALL LINK("DSPLAY",17,3,27,"BY ENTERING OF STOP")
370 CALL LINK("DSPLAY",18.3,"THE GRAFIC IS STOPPED!")
380 CALL LINK("ACCEPT",22,3,4,M$)
390 IF M$<>STOP THEN 120
400 STOP
```

## 4.19 CALL LINK("SHIFT",DELTAX,DELTAY)

Dieser Befehl übt eine lineare Transformation auf die Graphik aus.

DELTAX....Verschiebung in X-Richtung (Spalten)
(Positive Werte nach rechts)

DELTAY .. Verschiebung in Y-Richtung (Zeilen)
(Positive Werte nach unten)

Achtung:

Die Graphikfenster auf dem Bildschirm werden nicht transformiert!
(Dieser Befehl steht für MINI MEMORY Modul nicht zur Verfügung!)

## 4.20 GSAVE/GLOAD

### 4.20.1 CALL GSAVE("DATEINAME")

Graphiken können nur auf Diskette gespeichert werden. Jeder gültige Dateiname kann verwendet werden. Die Speicherung erfolgt im "MEMORY-IMAGE"-Format. Die Farbe der Graphik, die Position des Graphikfensters und die Graphikfeldparameter werden mitgespeichert.

### 4.20.2 CALL GLOAD("DATEINAME")

Dieses Statement lädt eine Graphik namens "DATEINAME" von Diskette ins VDP-RAM. Das Ansprechen von "GLOAD" ohne vorhergehendes "GRAFIC" führt zu Fehlermeldung und Programmabbruch.

Beispiel:

```
100 REM STORE GRAPHIC
110 REM **************
120 CALL LINK("GRAFIC",0)
125 CALL LINK ("WINDOW",3,8)
130 FOR R=2 TO 40
140 CALL LINK("CIRCLE",64,60,R)
150 NEXT R
160 CALL LINK("GSAVE","DSK2.SAVETEST")
170 STOP

> NEW

100 REM LOAD GRAPHIC
110 REM *************
120 CALL LINK("GRAFIC",0)
130 CALL LINK("GLOAD","DSK2.SAVETEST")
140 GOTO 140
```

Zu beachten ist dabei, daß der Dateiname nicht länger als 9 Buchstaben sein darf.

Das vorangegangene Beispiel führt zur Bildung folgender Dateien auf DSK2.:

"SAVETEST" ... enthält Graphikfeld und Parameter

"SAVETEST1" .. enthält Bildschirmauszug

"SAVETEST2" .. enthält Farben der Graphik

Beim Speichern und Laden der Graphik darf immer nur der erste Dateiname, in unserem Fall "SAVETEST", angegeben werden.
Die Eingabe des zweiten und dritten Dateinamens führt zu Fehlfunktionen.

Der Befehl "WINDOW" kann beim Laden der Graphik entfallen, da alle Fenster auf dem Bildschirm von der geladenen Graphik überschrieben werden.

"GSAVE/GLOAD" stehen für MINI MEMORY Modul nicht zur Verfügung.

## 4.21 HARDCOPY

Beispiel:

Abschließend ist noch ein Programmbeispiel namens "HCOPYTEST" eingefügt, das zeigt, wie die Graphik des Titelblattes des Manuals hergestellt wird.

```
100 REM   ***********************
110 REM  *************************
120 REM  **                     **
130 REM  **  H C O P Y D E M O  **
140 REM  **                     **
150 REM  *************************
160 REM   ***********************
170 REM
180 REM by §APESOFT
190 REM
200 REM 19830717
210 REM
220 REM
230 POPT$="PIO.CR"
240 SEC$="K"
250 REM
260 REM SONDERZEICHEN FUER DAS "§APESOFT" COPYRIGHT
270 REM ******************************************
280 REM
290 DATA 96,00000107@F1F1F3F,97,3F3F7F7F7F7F7F7F,98,003FFFFFFFFFFFF,99,E3C180E
80C1E3FF
300 DATA 100,030F1F3F3F7F7F7F,101,7F7F7F7F7F7F7F7F,102,60787C7E7E7E7F7F,103,7F7
7E7E7E7C786@
310 DATA 104,7F7F7F7F7F7F7F7F
320 DATA 106,030F1F3F3F3F7F7F,107,7F7F3F3F3F1F0F03,108,60787C7E7E7E7F7F,109,00@
7E7E7E7C786@
330 DATA 110,030F1F3F3F3F7F7F,111,7F00003F3F1F0F03,112,E0F8FCFEFEFE8080,113,FFF
FEFEFEFCF8E0
340 DATA 114,030F1F3F3F3F7F7F,115,7F7F7F3F3F1F0F03,116,60787C7E7E7E7F7F,117,7F7
7F7E7E7C786@
350 DATA 118,00000003@F1F3F3F,119,0000001C3E7F7F7F,120,3F7F7F7F7F7F7F7F,121,7F7
7F7F7F7F7F7F
360 DATA 122,3E1C000000006060,123,6060606
370 DATA 124,0000007F7F7F7F7F0,125,0000000000000006,126,7F7F7F7F7F7F7F7F,127,7F7
7F3F3F1F0F03
380 DATA 128,6060606060000000,129,001C3E7F7F7F3E1C
390 DATA 130,00FCFFFFFFFFFFFF,131,9F8783818180808@,132,000080E0F0F8F8FC,133,FCF
FEFEFEFEFEFE
400 DATA 134,7F7F7F7F7F7F7F3F,135,3F3F1F1F0F070100,136,8080C0C0C0E0F0FC,137,FFF
FFFFFFFFF3F
410 DATA 138,808080808080808@,139,FFFFFFFFFFC0FFFF,140,FEFEFEFEFEFEFCFC,141,FCF
F8F0E000FEFE
420 REM
430 REM APECHARACTERS UEBERTRAGEN
440 CALL CLEAR
450 CALL SCREEN(2)
460 READ I,N$
470 CALL CHAR(I,N$)
480 IF I<141 THEN 460
490 PRINT TAB(9);"MICROCOMPUTER": :
500 PRINT TAB(7);"`b"&CHR$(130)&CHR$(132)&"         vwöü"
510 PRINT TAB(7);"ac"&CHR$(131)&CHR$(133)&"dfjlnprtxzß"&CHR$(128)
520 PRINT TAB(7);CHR$(134)&CHR$(136)&CHR$(138)&CHR$(140)&"egkmoqsuyä"&CHR$(127)&
CHR$(129)
530 PRINT TAB(7);CHR$(135)&CHR$(137)&CHR$(139)&CHR$(141)&"h"
540 PRINT :TAB(11);"SOFTWARE": : : : : : : : : : : : : : :
550 CALL SCREEN(4)
560 CALL BHCOPY(POPT$,SEC$)
570 REM
580 REM QUADER
590 REM ******
600 CALL LINK("GRAFIC",0)
```

```
610 CALL LINK("CENTRE",1,-2)
620 CALL LINK("WINDOW",1,1)
630 CALL LINK("TURNTO",41.8103)
640 FOR Y=-30 TO -95 STEP -3
650 CALL LINK("SETTO",X,Y)
660 CALL LINK("MOVE",40)
670 NEXT Y
680 Y=-55
690 CALL LINK("TURNTO",90)
700 FOR X=32 TO 117 STEP 3
710 Y=Y+1
720 CALL LINK("SETTO",X,Y)
730 CALL LINK("MOVE",65)
740 NEXT X
750 CALL LINK("TURNTO",-18.4349)
760 Y=-30
770 FOR X=1 TO 33 STEP 3
780 CALL LINK("SETTO",X,Y)
790 CALL LINK("MOVE",92)
800 Y=Y-2.5
810 NEXT X
820 CALL BHCOPY(POPT$,SEC$)
830 REM
840 REM PYRAMIDE
850 REM ********
860 CALL LINK("CLTBLE")
870 CALL LINK("CENTRE",64,1)
880 YG=-96
890 FOR XG=-63 TO -20 STEP 2
900 CALL LINK("SETTO",0,0)
910 CALL LINK("MOVETO",XG,YG)
920 YG=YG-1
930 NEXT XG
940 CALL LINK("SETTO",0,0)
950 CALL LINK("REMVTO",XG,YG)
960 FOR XG=XG+3 TO 46 STEP 3
970 CALL LINK("SETTO",0,0)
980 CALL LINK("MOVETO",XG,YG)
990 YG=YG+3
1000 NEXT XG
1010 CALL BHCOPY(POPT$,SEC$)
1020 REM
1030 REM ZYLINDER
1040 REM ********
1050 CALL LINK("CLTBLE")
1060 CALL LINK("SETBLE",12)
1070 CALL LINK("WINDOW",-1,4)
1080 CALL LINK("CENTRE",48,1)
1090 CALL LINK("TURNTO",0)
1100 FOR Y=-120 TO -40 STEP 3
1110 CALL LINK("ELLIPS",0,Y,48,24)
1120 NEXT Y
1130 FOR R=48 TO 2 STEP -2
1140 CALL LINK("ELLIPS",0,Y,R,R/2)
1150 NEXT R
1160 CALL BHCOPY(POPT$,SEC$)
1170 REM
1180 REM ZYKLOM
1190 REM ******
1200 PI2=8*ATN(1)
```

```
1210 CALL LINK("GRAFIC",1)
1220 CALL LINK("WINDOW",5,8)
1230 CALL LINK("CENTRE",1,40)
1240 CALL LINK("AXIS",8,116,0,4,0,36,40,4)
1250 X=7
1260 CALL LINK("SETTO",X,-20)
1270 FOR PHI=0 TO 3*PI2 STEP PI2/16
1280 X=X+2
1290 Y=20*(SIN(2*PHI)*2*COS(PHI))
1300 CALL LINK("MOVETO",X,Y)
1310 NEXT PHI
1320 CALL LINK("DSPLAY",16,9,8,"MITTLERE")
1330 CALL LINK("DSPLAY",17,9,13,"ANALYSENWERTE")
1340 M$="TEMPERATUR"
1350 FOR Z=5 TO 14
1360 CALL LINK("DSPLAY",Z,7,1,SEG$(M$,Z-4,1))
1370 NEXT Z
1380 CALL LINK("WRITE",10,3,"gradtagzahl")
1390 CALL BHCOPY(POPT$,SEC$)
1400 REM
1410 REM HISTROGRAMME
1420 REM ************
1430 CALL LINK("GRAFIC",0)
1440 CALL LINK("SETBLE",24)
1450 CALL LINK("WINDOW",1,1)
1460 CALL LINK("HSTDIA",1,68,4,8,2,8,56,8,16,4,24,32,12,32,8)
1470 CALL LINK("HSTDIA",50,0,18,56,11)
1480 CALL BHCOPY(POPT$,SEC$)
1490 REM
1500 REM TORUS
1510 REM *****
1520 CALL LINK("CLTBLE")
1530 CALL LINK("CENTRE",96,40)
1540 PIARC=4*ATN(1)
1550 FOR PHI=0 TO 2*PIARC STEP PIARC/40
1560 CALL LINK("CIRCLE",66*SIN(PHI),20*COS(PHI),17)
1570 NEXT PHI
1580 CALL BHCOPY(POPT$,SEC$)
1590 REM
1600 REM BLUMEN
1610 REM ******
1620 CALL LINK("GRAFIC",0)
1630 CALL LINK("WINDOW",1,4)
1640 CALL LINK("CENTRE",54,48)
1650 CALL LINK("SETTO",0,0)
1660 FOR PHI=0 TO 22.5 STEP 22.5
1670 CALL LINK("TURNTO",PHI)
1680 FOR X=1 TO 7
1690 FOR W=0 TO 15
1700 CALL LINK("MOVE",4)
1710 CALL LINK("TURN",W)
1720 NEXT W
1730 FOR W=16 TO 4 STEP -1
1740 CALL LINK("MOVE",4)
1750 CALL LINK("TURN",W)
1760 NEXT W
1770 CALL LINK("MOVETO",0,0)
1780 CALL LINK("TURN",6)
1790 NEXT X
1800 NEXT PHI
```

```
1810 FOR R=1 TO 12
1820 CALL LINK("CIRCLE",0,0,R)
1830 NEXT R
1840 CALL BHCOPY(POPT$,SEC$)
1850 REM
1860 CALL LINK("BYEBYE")
1870 END
```

## 4.22 CALL LINK("BYEBYE")

"BYEBYE" hebt den GRAFIC-Modus auf. Es lädt die Standard Character Zeichensätze und lenkt wieder in den BASIC oder XBASIC-Modus ein.

Der Computer funktioniert wie gewohnt. SOUND und SPRITES können wieder verwendet werden.

Der Computer funktioniert wie gewohnt. Vor Ausführung eines neuen APESOFT EXPANDED GRAFIC BASIC-Befehles muß mehrmals ein CALL LINK("GRAFIC",MODUS) stehen, sonst bricht das Programm mit einer Fehlermeldung ab.

# 5. K U R Z R E F E R E N Z

CALL LINK("GRAFIC",MODUS)

signalisiert den Grafic-Modus, initialisiert alle Computerregister dafür und definiert abhängig vom MODUS (Graphik- oder Textmodus) ein Graphikfeld mit maximal 128 vertikalen und 120 horizontalen Linien.

CALL LINK("WINDOW",ZEILE,SPALTE)

setzt das gesamte Graphikfeld (16*8 Spalten, 15*8 Zeilen oder 11*8 Zeilen im Textmodus) auf Bildschirmposition Spalte (1-32) und Zeile (1-24).

CALL LINK("WINDOW",Z,S,ZA,SA,DZ,DS)

bringt Ausschnitte des Graphikfeldes auf den Bildschirm, wobei Z die Bildschirmzeile, S die Bildschirmspalte, ZA, SA den linken oberen Eckpunkt des Graphikfeldes, DZ die Anzahl der Grafictable-Zeichen in Zeilenrichtung und DS die Anzahl der Grafictable-Zeichen in Spaltenrichtung bedeuten.

CALL LINK("SETBLE",BREITE)

dimensioniert das Graphikfeld.

CALL LINK("CLTBLE")

löscht das Graphikfeld und somit die Graphik.

CALL LINK("TABLE",Z,S,XMAX,YMAX,BYTES)

liefert die momentanen Parameter des Graphikfeldes in die angegebenen Variablen zurück, wobei Z die Zeilenanzahl, S die Spaltenanzahl des Feldes, XMAX die maximalen Pixelspalten, YMAX die maximalen Pixelzeilen des Feldes und BYTES die Anzahl der zur Verfügung stehenden Bytes bedeuten.

CALL LINK("SETCOL",VORDERGRUND-,HINTERGRUNDFARBE)

ändert die Vordergrund- und Hintergrundfarbe gleichzeitig für die gesamte Graphik.

CALL LINK("SETCOL",N,FG,BG*(N1,FG1,BG1....)*)

legt die Vordergrundfarbe (FG) und Hintergrundfarbe (BG) für den mit N spezifizierten Zeichensatz fest.

CALL LINK("INVERT",X,Y,DX,DY)

invertiert Ausschnitte der Graphik; mit X,Y wird die Pixelposition des linken oberen Ecks, mit DX die Pixelspaltenposition und mit DY die Pixelzeilenposition des Graphikausschnittes, der invertiert wird, festgelegt.

CALL LINK("CLSCRN")

löscht den Bildschirm, die Graphik im Table und alle anderen internen Parameter bleiben erhalten.

CALL LINK("CENTRE",X,Y)

legt das benutzerdefinierte Koordinatensystem mit dem (0,0)-Punkt auf Position (X,Y) des Graphikfeldes.

CALL LINK("SETTO",X,Y*(,X1,Y1....)*)

setzt Pixel mit den Koordinaten Zeile Y und Spalte X.

CALL LINK("RESET",X,Y*(,X1,Y1....)*)

löscht Pixel mit den Koordinaten X,Y.

CALL LINK("IFSET",X,Y,VAR*(,X1,Y1,VAR1....)*)

prüft, ob ein Pixel mit den Koordinaten X,Y gesetzt ist und liefert das Ergebnis in der Variablen VAR zurück.

CALL LINK("MOVE",DIST)

zeichnet eine Linie der Länge DIST, ausgehend von der momentanen Position X,Y des Cursors mit dem momentanen internen Winkel PHI.

CALL LINK("REMOVE",DIST)

löscht alle Pixel ab Position X,Y bis zur DIST entfernten neuen Position des Cursors.

CALL LINK("MOVETO",X,Y,*(,X1,Y1....)*)

zeichnet eine Linie von der augenblicklichen internen Position des Cursors bis zur nächsten, durch das Parameterpaar X,Y festgelegten Position.

CALL LINK("REMVTO",X,Y*(,X1,Y1....)*)

löscht alle Linien.

CALL LINK("TURN",PHI)

addiert zum momentanen internen Winkel des Cursors den Winkel PHI in Dezimalgraden. PHI dreht im Uhrzeigersinn.

CALL LINK("TURNTO",PHI)

stellt den internen Winkel unbedingt auf den Winkel PHI (Grad) ein.

CALL LINK("RECT",A,B*(,A1,B1....)*)

zeichnet Rechtecke ausgehend von den augenblicklichen Cursorparametern in der Folge A -) B -) A -) B.

CALL LINK("CLRECT",A,B*(,A1,B1....)*)

löscht die Rechecke.

CALL LINK("CIRCLE",X,Y,R*(,X1,Y1,R1....)*)

zeichnet einen Kreis mit dem Mittelpunkt X,Y und dem Radius R.

CALL LINK("CLCRCL",X,Y,R*(,X1,Y1,R1....)*)

löscht die Kreise.

CALL LINK
("ARCUS",X,Y,R,PHI,DPHI*(,X1,Y1,R1,PHI1,DPHI1....)*)

zeichnet Kreisbögen mit dem Mittelpunkt X,Y, dem Radius R, dem Startwinkel PHI und dem Bogenwinkel des Kreisbogens DPHI.

CALL LINK
("CLARCS",X,Y,R,PHI,DPHI*(,X1,Y1,R1,PHI1,DPHI....)*)

löscht alle Bogenpunkte.

CALL LINK("ELLIPS",X,Y,A,B*(,X1,Y1,A1,B....)*)

zeichnet Ellipsen mit dem Achsmittelpunkt X,Y, großer Halbachse A, kleiner Halbachse B und Neigung PHI der großen Halbachse.

CALL LINK("CLLIPS",X,Y,A,B*(,X1,Y1,A1,B1....)*)

löscht alle Ellipsen.

```
CALL LINK("VALUES",X,Y,PHI,FG,BG)
```

liefert die momentanen internen Parameter in die Variablenliste zurück, wobei X der Cursorspalte, Y der Cursorzeile, PHI dem Cursorwinkel, FG der Vordergrund- und BG der Hintergrundfarbe entsprechen.

```
CALL LINK
("AXIS",X,LENIR,LENXL,DELTAX,Y,LENYU,LENYD,DELTAY)
```

zeichnet ein Achsenkreuz mit dem Mittelpunkt X,Y, der linken X-Halbachse LENXL, der rechten X-Halbachse LENXR, der Schrittweite des X-Rasters DELTAX, der oberen Y-Halbachse LENYU, der unteren Halbachse LENYD und der Schrittweite des Y-Rasters DELTAY.

```
CALL LINK("HSTDIA",X,Y,BREITE,HÖHE,TIEFE)
```

zeichnet ein Blockdiagramm mit den Koordinaten des linken unteren Endes des Blockes X,Y, der Breite BREITE, der Höhe HÖHE und der Tiefe TIEFE des Blockdiagrammes.

```
CALL LINK
("CRDIA",X,Y,RADIUS,PHI,DPHI*(,X1,Y1,PHI1,DPHI1...)*)
```

zeichnet ein Kreisdiagramm mit den Koordinaten des Kreissegmentmittelpunktes X,Y, dem Radius RADIUS, dem Startwinkel PHI und dem Endwinkel des Kreissegmentes DPHI.

```
CALL LINK("WRITE",Z,S,STRING*(Z1,S1,STRING1....)*)
```

erlaubt das Mischen von Graphik und Text unmittelbar im Graphikfeld und schreibt einen String (STRING) an Position Zeile (Z) und Spalte (S) des Graphikfeldes.

```
CALL LINK("DSPLAY",Z,S,SIZE,VAR*)
```

bringt SIZE Zeichen des Stringes VAR$ an die Bildschirmposition (Z,S).

```
CALL LINK("ACCEPT",Z,S,SIZE,VAR$)
```

übernimmt SIZE Zeichen des Stringes VAR$ an der Bildschirmposition (Z,S).

```
CALL LINK("SHIFT",DELTAX,DELTAY)
```

transformiert eine Graphik linear um die Werte DELTAX in Spalten- und DELTAY in Zeilenrichtung.

CALL GSAVE("DATEINAME")

speichert die Farbe der Graphik, die Position des Graphikfeldes und die Graphikparameter im "MEMORY-IMAGE"-Format auf Diskette.

CALL GLOAD("DATEINAME")

lädt eine Graphik namens "DATEINAME" von Diskette ins VDP-RAM.

CALL LINK("BYEBYE")

hebt den Grafic-Modus auf, lädt die Standard Charakter Zeichensätze und lenkt wieder in den BASIC oder XBASIC-Modus ein.

```
100 REM  GRAPHIK-DEMO
110 REM  ************
120 REM
130 REM  GRAFDEMO
140 REM
150 REM   TI-BASIC
160 REM    19830725
170 REM   BY §APESOFT
180 REM
190 RANDOMIZE
200 PI180=4*ATN(1)/180
210 REM
220 REM TITELBILD
230 REM *********
240 CALL LINK("GRAFIC",1)
250 CALL LINK("WINDOW",1,1)
260 CALL LINK("WINDOW",12,1)
270 CALL LINK("WINDOW",1,16)
280 CALL LINK("WINDOW",12,16)
290 FOR Z=1 TO 90 STEP 4
300 CALL LINK("SETTO",1,Z)
310 CALL LINK("MOVE",128)
320 NEXT Z
330 CALL LINK("TURN",-90)
340 FOR S=1 TO 125 STEP 4
350 CALL LINK("SETTO",S,1)
360 CALL LINK("MOVE",120)
370 NEXT S
380 FOR N=1 TO 30
390 FG=3+13*RND
400 CALL LINK("SETCOL",N,FG,2)
410 NEXT N
420 CALL WAIT(150)
430 M$="                    "
440 FOR Z=3 TO 21
450 CALL LINK("DSPLAY",Z,6,20,M$)
460 NEXT Z
470 CALL LINK("DSPLAY",5,12,8,"TI-99/4A")
480 CALL LINK("DSPLAY",8,12,8,"EXPANDED")
490 CALL LINK("DSPLAY",10,10,12,"GRAFIC BASIC")
500 CALL LINK("DSPLAY",14,14,4,"HIGH")
510 CALL LINK("DSPLAY",16,8,17,"RESOLUTION GRAFIC")
520 CALL LINK("DSPLAY",18,15,2,"BY")
530 CALL LINK("DSPLAY",20,12,8,"§APESOFT")
540 CALL WAIT(300)
560 REM
570 REM STRAHLENKRANZ
580 REM  -------------
590 CALL LINK("GRAFIC",0)
600 CALL LINK("WINDOW",-3,8)
610 CALL LINK("CLTBLE")
620 FOR PHI=0 TO 360 STEP 5
630 CALL LINK("SETTO",64,50)
640 CALL LINK("TURNTO",PHI)
650 CALL LINK("MOVE",40)
660 NEXT PHI
670 CALL LINK("WRITE",15,2,"STRAHLENKRANZ")
680 CALL WAIT(150)
690 CALL LINK("SETCOL",4,2)
700 FOR I=1 TO 10
```

```
710 Z=6*RND+1
720 S=15*RND+1
730 FG=5*RND+10
740 CALL LINK("WINDOW",Z,S)
750 CALL LINK("SETCOL",FG,2)
760 NEXT I
770 CALL LINK("WINDOW",-3,8)
780 FOR PHI=0 TO 360 STEP 10
790 CALL LINK("RESET",64,50)
800 CALL LINK("TURNTO",PHI)
810 CALL LINK("REMOVE",40)
820 NEXT PHI
830 CALL WAIT(150)
840 REM
850 REM RECHTECKE
860 REM ---------
870 CALL LINK("CLTBLE")
880 CALL LINK("SETCOL",15,2)
890 CALL LINK("CENTRE",1,1)
900 FOR X=1 TO 30
910 CALL LINK("TURNTO",(X-1)*2)
920 CALL LINK("SETTO",X*2,-X)
930 CALL LINK("RECT",60+X,30+X)
940 NEXT X
950 CALL WAIT(150)
960 REM
970 REM  KREISE
980 REM  ------
990 CALL LINK("GRAFIC",0)
1000 CALL LINK("WINDOW",-1,1)
1010 CALL LINK("WINDOW",9,17)
1020 CX=10
1030 CY=13
1040 FOR I=0 TO 35
1050 CX=CX+2
1060 CY=CY+SQR(I)/3
1070 CALL LINK("CIRCLE",CX,CY,10+I)
1080 NEXT I
1090 CALL WAIT(150)
1100 CALL LINK("CLTBLE")
1110 FOR I=1 TO 71 STEP 2
1120 RADIUS=1+I*I/100
1130 CALL LINK("CIRCLE",60,I,RADIUS)
1140 NEXT I
1150 CALL LINK("SETCOL",4,2)
1160 CALL WAIT(150)
1170 FOR I=71 TO 1 STEP -2
1180 RADIUS=1+I*I/100
1190 CALL LINK("CLCRCL",60,I,RADIUS)
1200 NEXT I
1210 CALL WAIT(150)
1220 REM
1230 REM STERNE
1240 REM ------
1250 CALL LINK("CLTBLE")
1260 FOR N=1 TO 28
1270 FG=3+13*RND
1280 CALL LINK("SETCOL",N,FG,2)
1290 NEXT N
1300 CALL LINK("SETTO",10,55)
```

```
1310 CALL LINK("WRITE",15,1,"§APESOFT GRAFIC")
1320 FOR PHI=0 TO 3960 STEP 110
1330 PHIARC=PHI*PI180
1340 X=60-50*COS(PHIARC)
1350 Y=55+50*SIN(PHIARC)
1360 CALL LINK("MOVETO",X,Y)
1370 NEXT PHI
1380 CALL WAIT(150)
1390 REM
1400 REM  MIXED
1410 REM  *****
1420 CALL LINK("WINDOW",-3,1,3,1,12,16)
1430 CALL LINK("WINDOW",1,17,3,1,12,16)
1440 CALL LINK("WINDOW",13,1)
1450 CALL LINK("WINDOW",13,17)
1460 CALL LINK("TURNTO",0)
1470 CALL LINK("SETTO",64,60)
1480 FOR A=2 TO 30 STEP 2
1490 B=A+5
1500 CALL LINK("RECT",A,B,-A,B,A,-B,-A,-B)
1510 NEXT A
1520 CALL WAIT(150)
1530 CALL LINK("CLTBLE")
1540 CALL LINK("WINDOW",1,-1,2,1,14,16)
1550 CALL LINK("WINDOW",1,17,2,1,14,16)
1560 CALL LINK("WINDOW",12,17)
1570 CALL LINK("WINDOW",12,1)
1580 CALL LINK("SETCOL",6,2)
1590 A=48
1600 B=20
1610 FOR PHI=0 TO 355 STEP 5
1620 PHIARC=PHI*PI180
1630 MX=64+A*COS(PHIARC)
1640 MY=60+B*SIN(PHIARC)
1650 CALL LINK("CIRCLE",MX,MY,15)
1660 NEXT PHI
1670 CALL WAIT(150)
1680 CALL LINK("CLTBLE")
1690 CALL LINK("WINDOW",-3,8)
1700 CALL LINK("SETCOL",12,2)
1710 CALL LINK("SETTO",64,60)
1720 CALL LINK("TURNTO",0)
1730 X=0
1740 FOR I=1 TO 40
1750 X=X+2
1760 CALL LINK("MOVE",X)
1770 CALL LINK("TURN",90)
1780 NEXT I
1790 CALL WAIT(150)
1800 CALL LINK("CLTBLE")
1810 A=40
1820 B=20
1830 FOR PHI=0 TO 350 STEP 10
1840 CALL LINK("SETTO",64,99)
1850 CALL LINK("TURNTO",PHI)
1860 PHIARC=PHI*PI180
1870 MX=64+A*COS(PHIARC)
1880 MY=99+B*SIN(PHIARC)
1890 CALL LINK("MOVETO",MX,MY,64,1)
1900 NEXT PHI
```

```
1910 CALL WAIT(150)
1920 REM
1930 REM BLUMEN
1940 REM ******
1950 CALL LINK("CLTBLE")
1960 CALL LINK("WINDOW",-13,1)
1970 CALL LINK("WINDOW",13,17)
1980 CALL LINK("WINDOW",1,1,2,1,14,16)
1990 CALL LINK("WINDOW",1,17,2,1,14,16)
2000 CALL LINK("SETTO",64,60)
2010 FOR PHI=0 TO 22.5 STEP 22.5
2020 CALL LINK("TURNTO",PHI)
2030 FOR X=1 TO 7
2040 FOR W=0 TO 15
2050 CALL LINK("MOVE",4)
2060 CALL LINK("TURN",W)
2070 NEXT W
2080 FOR W=16 TO 4 STEP -1
2090 CALL LINK("MOVE",4)
2100 CALL LINK("TURN",W)
2110 NEXT W
2120 CALL LINK("MOVETO",64,60)
2130 CALL LINK("TURN",6)
2140 NEXT X
2150 NEXT PHI
2160 FOR R=1 TO 12
2170 CALL LINK("CIRCLE",64,60,R)
2180 NEXT R
2190 CALL WAIT(150)
2200 FOR I=1 TO 5
2210 FG=3+13*RND
2220 CALL LINK("SETCOL",FG,2)
2230 CALL WAIT(150)
2240 NEXT I
2250 REM
2260 REM Y=2*SIN(X)+COS(2*X)
2270 REM ********************
2280 CALL LINK("GRAFIC",0)
2290 CALL LINK("WINDOW",3,8)
2300 CALL LINK("CENTRE",4,60)
2310 CALL LINK("AXIS",0,120,0,3,0,59,50,3)
2320 CALL LINK("TURNTO",270)
2330 X=2
2340 FOR PHI=0 TO 8*ATN(1)STEP ATN(1)/7
2350 X=X+2
2360 DIST=20*(2*SIN(PHI)+COS(2*PHI))
2370 CALL LINK("SETTO",X,0)
2380 CALL LINK("MOVE",DIST)
2390 NEXT PHI
2400 CALL WAIT(150)
2410 REM
2420 REM HISTOGRAMME
2430 REM ************
2440 CALL LINK("CLTBLE")
2450 FOR N=2 TO 26 STEP 2
2460 CALL LINK("SETCOL",N,14,2)
2470 NEXT N
2480 CALL LINK("CENTRE",1,110)
2490 CALL LINK("AXIS",008,120,000,008,0,108,000,004)
2500 CALL LINK("HSTDIA",016,6,012,090,006)
```

```
2510 CALL LINK("HSTDIA",040,06,012,045,006)
2520 CALL LINK("HSTDIA",070,06,012,067,006)
2530 CALL LINK("HSTDIA",094,6,012,012,006)
2540 CALL LINK("WRITE",15,3,"HISTOGRAMME")
2550 CALL WAIT(150)
2560 REM
2570 REM KREISDIAGRAMME
2580 REM **************
2590 CALL LINK("GRAFIC",0)
2600 CALL LINK("WINDOW",3,8)
2610 FOR N=1 TO 13 STEP 2
2620 CALL LINK("SETCOL",N,16,2,N+1,14,2,N+14,9,2,N+15,13,2)
2630 NEXT N
2640 CALL LINK("CENTRE",1,1)
2650 CALL LINK("CRCDIA",060,-55,038,180,96)
2660 CALL LINK("CRCDIA",068,-55,038,276,174)
2670 CALL LINK("CRCDIA",060,-65,038,90,90)
2680 CALL LINK("WRITE",15,2,"KREISDIAGRAMME")
2690 CALL WAIT(150)
2700 GOTO 240
```