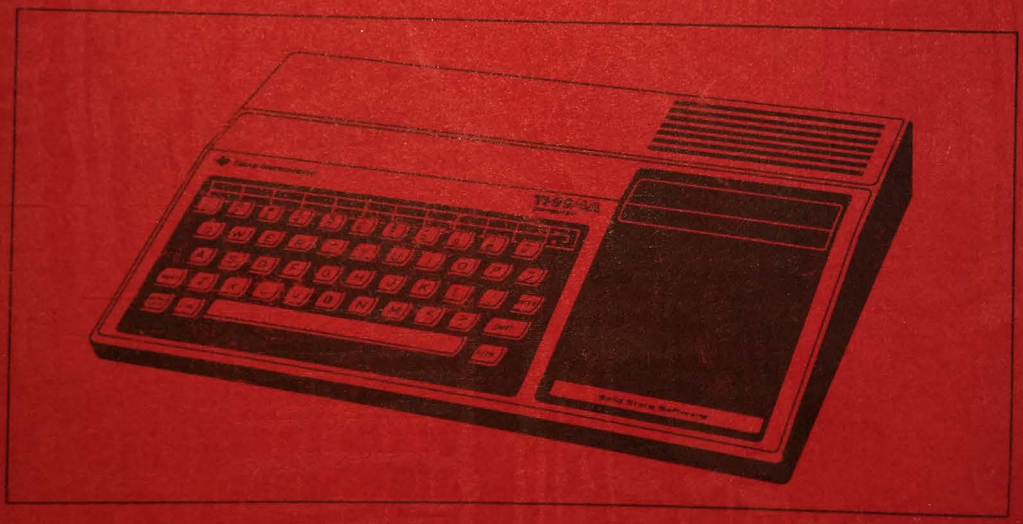
TORPEDO BASIC®



TI 99/4a
Manual

```
**********************************************************************
**********************************************************************
**                                                                  **
**                                                                  **
**                                                                  **
**   H A G E R A  (R)                                  (c) 1985     **
**                                                                  **
**                                                                  **
**   NEUE IDEEN FÜR DEN TI-99/4A HOMECOMPUTER                       **
**                                                                  **
**                                                                  **
**                                                                  **
**   ================================================================ **
**                                                                  **
**   T O R P E D O  B A S I C  E X P A N S I O N  E/A-XB            **
**                                                                  **
**   ================================================================ **
**                                                                  **
**                                                                  **
**                                                                  **
**   Alle Rechte bei:                                               **
**                                                                  **
**   Rausch + Haub                                                  **
**   Vertriebsgesellschaft                                          **
**   Postfach 320313                                                **
**                                                                  **
**   5300  Bonn  3                                                  **
**                                                                  **
**                                                                  **
**                                                                  **
**   HAGERA(R) ist ein rechtlich geschütztes Warenzeichen.          **
**                                                                  **
**                                                                  **
**                                                                  **
**********************************************************************
**********************************************************************
```


TITELENTWURF geschützt durch HAGERA(R)

VORWORT

Die TORPEDO BASIC EXPANSION ist eine wertvolle Hilfe für alle Basic und Extended Basic Programmierer auf dem TI-99/4a. Die Programmdiskette erweitert die Möglichkeiten des Computers um ein vielfaches. Zur Verfügung gestellt werden 24 neue Befehle, die mit LINK aus dem Basic/Extended Basic heraus ansprechbar sind.

TORPEDO BASIC ermöglicht die Definition von bis zu 32 voneinander unabhängigen 'Fenstern' - das sind Bildschirmausschnitte, die jeder für sich wie ein eigenständiger Screen behandelt werden können.

TORPEDO BASIC stellt außerdem einen Buffer zur Verfügung, mit dessen Hilfe es erlaubt ist, ganze Bildschirmteile mit einem 'versteckten' Screen zu tauschen. Es sind Befehle zum übertrag von Strings in den Buffer, vom Bildschirm in den Buffer und vom Bildschirm in Stringvariable vorhanden, was eine komfortable Benutzung gestattet.

TORPEDO BASIC bietet die Möglichkeit, den Text-Modus des TI-99/4a zu benutzen. Darin stehen 24x40 Zeichen auf dem Screen zur Verfügung. Natürlich sind alle TORPEDO BASIC Befehle auch im Text-Modus verwendbar.

TORPEDO BASIC erlaubt darüberhinaus Extended Basic Besitzern das direkte Peeken und Poken ins VDP-Ram. Editor/Assembler-Benutzer haben diese Befehle bereits in ihrem Modul eingebaut.

TORPEDO BASIC ermöglicht die Anfertigung von HARDCOPYs auf einem Epson FX-80 Drucker (oder einem vergleichbaren) sowie die Anzeige eines Disketten-Directorys auf dem Bildschirm ohne Benutzung des Disk Manager Moduls.

TORPEDO BASIC gestattet Basic-Verzweigungen zu Adressen-Namen, das schnelle auffinden von Strings auf dem Screen, das Scollen des Bildschirms oder Teilen daraus in alle Richtungen und die Benutzung von 9 internationalen (und einem wissenschaftlichen) Zeichensätzen.

TORPEDO BASIC enthält Interruptroutinen für verschiedene Aufgaben, die Sie aus dem Basic heraus aktivieren können; so zum Beispiel die Benutzung beliebiger Bildschirmfarben auch während der Programmierung und das Löschen des Bildschirms auf Tastendruck.

TORPEDO BASIC stellt in einem definierten Bildschirmfenster einen voll-screenorientierten Cursor mit vielen Funktionen zur Verfügung.

TORPEDO BASIC - damit macht Programmieren noch mehr Spaß!

Bevor wir die Funktionen der einzelnen neuen Befehle erläutern, sollten Sie jedoch unsere kurze Einleitung lesen. Darin erhalten Sie verschiedene Hinweise zur Arbeitsweise der Programmdiskette.

Wer tiefer in die Assembler-Programmierung einsteigen möchte, dem empfehlen wir unseren:

ASSEMBLER KURS II für Einsteiger 344 Seiten + Diskette DM 79.90

ASSEMBLER KURS III für Fortgeschrittene 348 Seiten DM 79.90

URHEBERRECHTLICHER HINWEIS

GARANTIE

Auf den Datenträger gewähren wir die gesetzliche Garantie von 6 Monaten. Sollten während dieses Zeitraumes Mängel auftreten, die nicht durch unsachgemäße Behandlung, fehlerhafte Anwendung, falsche oder fehlerhafte Interpretation der Bedienungsanleitung oder ähnlichen Umständen, die nicht in der Verantwortung unseres Vertriebes liegen, so wird der Artikel nach unserem Ermessen repariert, zurückgenommen oder durch ein mangelfreies gleiches Exemplar ersetzt.

Die Garantie erlischt mit der unerlaubten oder unsachgemäßen Behandlung oder Verwendung. Darunter fällt auch die Entfernung des Sicherheitsstreifens über der Diskettenaussparung sowie das Überkleben von Cassettenschutz-Kerben.

HAGERA SERVICE

Einsender der HAGERA-Servicekarte können bei einer Beschädigung der Originaldiskette diese an RAUSCH & HAUB einsenden und erhalten gegen Selbstkosten wie in den Servicebestimmungen beschrieben ein unbeschädigtes Exemplar zurück; auch nach Ablauf der gesetztlichen Garantiefrist innerhalb von 12 Monaten. Dies gilt ausschließlich für mechanische Beschädigungen, die nicht durch unerlaubte oder unsachgemäße Behandlung entstanden sind.

Sollte Ihrer Auslieferung keine Servicekarte zur Einsendung an uns beigelegen haben, wenden Sie sich bitte umgehend an unseren Vertrieb.

HAFTUNG

Unser Vertrieb haftet nicht für Schäden an Datenträgern und Geräten, die durch falsche oder fehlerhafte Behandlung oder unrichtige Interpretation von Bedienungsanleitungen entstanden sind, auch nicht für Schäden an anderen Geräten als jene, welche im Lieferumfang dieses Artikels enthalten sind. Dies gilt auch für mangelnde oder fehlerhafte Information aus Werbeträgern. Ausschlaggebend für eine richtige Bedienung ist ausschließlich der Inhalt dieses Buches.

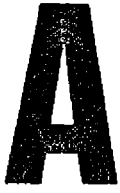
Eventuelle Schadenersatzansprüche beschränken sich in jedem Fall auf den Bezugspreis dieses Produktes. Die Kaufrechnung ist vom Käufer vorzulegen - der Anspruch beschränkt sich in jedem Fall im Maximum auf den von RAUSCH & HAUB empfohlenen Verkaufspreis.

SONSTIGE BESTIMMUNGEN

Mit dem Erwerb dieses Artikels erkennt der Käufer unsere allgemeinen Geschäftsbedingungen an. Zuwiderhandlungen werden rechtlich verfolgt und können Gerichtskosten von über DM 1000.-, abgesehen von im Einzelfall festzusetzenden Schadenersatzansprüchen unsererseits, nach sich ziehen.

I N H A L T

I N H A L T



BEFEHLSÜBERSICHT

Die nachfolgenden Befehle und Funktionen werden von TORPEDO BASIC zur Verfügung gestellt. Vor Inbetriebnahme sind die Befehle zum Laden des Maschinenprogramms, wie im Anschluß an diese Übersicht beschrieben, durchzuführen.

BRANCH Zeilennummer-Ausdruck
Verzweigung im Basic zu einer Zeile, die sich aus dem Ausdruck berechnet.

CHRSET Internationaler Charactersatz
Auswahl eines internationalen Zeichensatzes und Initialisierung der entsprechenden Interruptroutine.

CLEAN Zeile,Spalte<,Stringlänge<,Ascii-Wert<,Richtung>>>
Löschen eines Bildschirmraumes.

CLTEXT Fensternummer<,Asciiwert>
Löschen eines Bildschirmfensters.

COPY <Fensternummer,>"Gerätename.Optionen"
Hardcopy auf Epson FX-80 oder vergleichbarem Drucker.

DIR Laufwerknummer<,Startzeile,Endzeile>
Ausgabe des Disketten-Directorys in einem bestimmten Bildschirmteil.

GET Zeile,Spalte<,Format<,Typ>>,Variable
Eingaben über die Tastatur auf den Bildschirm.

GETSTR Zeile,Spalte,Format,Stringvariable
Übernahme eines Strings vom Bildschirm.

HIDE Windownummer<,Windownummer,...>
Kopieren des Bildschirminhaltes in den Buffer.

INSTR Zeile,Spalte,Stringausdruck
Schreiben von Strings direkt in den Buffer.

MODE Modusflag
Modusauswahl Grafik-Mode / Text-Mode.

OUTSTR Zeile,Spalte,Format,Stringvariable
übernahme eines Strings vom Buffer in die Variable.

QUIT
Aufruf des TI-Titelbildes.

SCROLL <Windownummer<,String><,Richtung>>
Bewegen des Bildschirms oder Teilen daraus in eine beliebige Richtung.

SEARCH Windownummer,String oder ASCII-Ausdruck,Zeile,Spalte
Suchen nach dem String/ASCII auf dem Screen und Wiedergabe der gefundenen Position in Zeile und Spalte.

SWAP Windownummer<,Windownummer>
Austausch von Screenfenstern mit dem korrespondierendem Bufferbereich.

TABLE Windownummer<,Vordergrund-,Hintergrundfarbe>
Bildschirmorientierter Cursor für das angegebene Fenster unter Umschaltung der Farben im Interruptbereich.

TAKE Windownummer<,Windownummer>
Füllen eines Fensters mit dem korrespondierendem Bereich im Buffer.

TORPED
Auswahl der Torpedo-Basic Standartfarben und Bereitschaftsanzeige.

USECOL <Vordergrund-,Hintergrundfarbe>
Auswahl der Bildschirmfarben, Initialisierung und Start aller initialisierten Interruptroutinen. Ohne Farbangabe werden alle Interrupt-Routinen abgeschaltet.

VDPEEK* Adresse,numerische Variable<,numerische Variable...>
Lesen von Bytes aus dem VDP-Ram.

VDPOKE** Adresse,ASCII-Wert<,ASCII-Wert...>
Schreiben von Bytes ins VDP-Ram.

WINDOW Fensternummer,Startzeile,-Spalte,Endzeile,-Spalte
Definieren eines Bildschirmfensters.

WRITE Zeile,Spalte,Zahl oder Stringausdruck/Variable
Schreiben auf den Bildschirm.

---

* Im E/A nicht vorhanden. Benutzen Sie dort PEEKV!
** Im E/A nicht vorhanden. Benutzen Sie dort POKEV!

Funktionen	Bedeutung		Zugriff
<CTRL>+<1>	CLEAR Screen	Bildschirm löschen	I
<FCTN>+<1>	DELETE Character	Zeichen löschen	LW
<FCTN>+<2>	INSERT Character	Zeichen einfügen	LW
<FCTN>+<3>	ERASE Line	Zeile löschen	LW
<FCTN>+<4>	CLEAR Program Exec.	Programmablauf stoppen	E
<FCTN>+<5>	BEGIN Window-C. Home	Cursor linke obere Ecke	W
<FCTN>+<6>	PROC'D Clear Window	Cursor Home + Clear Screen	W
<FCTN>+<9>	BACK to Program	Verlassen des Fensters	W
<FCTN>+<S>	ARROW Left/Backspace	Rückwärtsschritt	LW
<FCTN>+<D>	ARROW Right	Vorwärtsschritt	LW
<FCTN>+<X>	ARROW Down	Cursor 1 Zeile nach unten	W
<FCTN>+<E>	ARROW Up	Cursor 1 Zeile nach oben	W
<ENTER>	ENTER Input Line	Eingabe bestätigen/beenden	LW

I Interruptgesteuert immer nutzbar, wenn Routine eingeschaltet.
L Line-Cursor im GET-Befehl.
W Window-Cursor im TABLE-Befehl.
E Extern, nur wenn kein Maschinenprogramm läuft.

LADEN DES PROGRAMMS

Beiliegend zu diesem Buch erhalten Sie die Programmdiskette oder Cassette mit dem lauffähigen Programmpaket TORPEDO BASIC. Zur Inbetriebnahme benötigen Sie eine der folgenden Konfigurationen:

- TI-99/4a Homecomputer (Grundausstattung + Bildschirmgerät)
- Extended Basic Modul
- Speicherweiterung 32Kbyte
- Peripherer Speicher (Cs,Dsk)

oder

- TI-99/4a Homecomputer (Grundausstattung + Bildschirmgerät)
- Editor/Assembler Modul
- BSCSUP-Utilities von E/A Diskette, Part A
- Diskettenlaufwerk

Nach dem Einschalten aller Geräte geben Sie folgende Befehlssequenz ein, wenn Sie die Extended Basic Version auf Diskette besitzen.

```
CALL INIT
CALL LOAD("DSK1.OTORPEDOX")
```

Bei der E/A-Version legen Sie zunächst die E/A-Diskette, Part A ein (ist Bestandteil Ihres Editor/Assembler Pakets und nicht im Lieferumfang von TORPEDO BASIC enthalten).

```
CALL INIT
CALL LOAD("DSK1.BSCSUP")
```

- Jetzt die TORPEDO BASIC Diskette einlegen.

```
CALL LOAD("DSK1.OTORPEDO")
```

Wer mehr als ein Laufwerk besitzt, kann die Disketten natürlich auch schon vorher entsprechend einlegen.

Besitzer der Cassettenversion legen diese in den Recorder und starten das Programm mit

```
 .
 .
* Ready
 .
 .
> RUN "CS1"
 .
 .
* Ready
 .
 .
> NEW
 .
 .
* Ready
 .
 .
> CALL LINK("TORPED")
 .
 .
```

Beachten Sie, daß die Cassettenversion sehr lange zum Laden braucht. Bei Einlesefehlern müssen Sie das Band erst zurückspulen, bevor Sie den Lesevorgang wiederholen können. Gegebenenfalls muß der Tonkopf Ihres Recorders korrigiert werden, um ein ordnungsgemäßes Einlesen zu ermöglichen. Wenden Sie sich in solchen Fällen an Ihren Händler.

Bei den Diskettenversionen wird TORPED automatisch durchgeführt. Der Bildschirm wird gelöscht, und die interruptgesteuerte Farbroutine mit den Farben grün/schwarz wird aktiviert. Wenn das Programm ornungsgemäß geladen wurde, meldet sich der Computer mit

```
* TORPEDO BASIC EXPANSION
  (C) 1985 BY HAGERA

  RAUSCH & HAUB
  Vertriebsgesellschaft
  Postfach 320313
  5300 Bonn 3

  Version 1.1 ...

* Ready

>
```

Sie können nun Ihre eigenen Basic/XBasic-Programme einlesen oder solche programmieren. Welche Befehle und Funktionen Ihnen zusätzlich zur Verfügung stehen, erfahren Sie auf den folgenden Seiten.

MODE CONTROL II

Was ist das eigentlich?

Mode Control ist eine Befehlserweiterung für TI-Basic bzw. TI-Extended Basic. Durch Mode Control erhalten Sie Zutritt zu den vier verschiedenen Arbeitsmodi des TI-99/4a:

Grafik-Modus: In diesem Modus befinden Sie sich nach Auswahl von TI-Basic bzw. Extended Basic. Es stehen 24x28 Schrift- oder 24x32 Grafikpositionen auf dem Screen zur Verfügung. In diesem Modus schreiben Sie auch Ihre Basic-Programme.

Text-Modus: Für Textverarbeitung, Verwaltung und Organisation ist der Grafik-Modus nicht geeignet. Deshalb können Sie mittels Mode Control für die Schriftausgabe den Bildschirm auf 24x40 Bildschirmpositionen erweitern. In diesem Modus wird Textbearbeitung zum reinen Vergnügen.

Multicolor-Modus: Ideal zum Beispiel für Balkendiagramme, Grundrisszeichnungen etc.! Es stehen 48x64 Bildschirmpositionen zur Verfügung, die Sie beliebig färben können.

Bit-Map-Modus: Hochauflösende Grafik ist auch für den TI-99/4a kein Problem. Jeder einzelne der 192x256 Bildschirmpunkte ist einzeln ansprechbar. Sie zeichnen mittels eines 'Zeichenstiftes' durch ein Basicprogramm und geben die so generierte Grafik danach an den Video-Speicher. Leichter kann man gar nicht Zeichnen.

Gemessen an den Leistungen benötigt Mode Control II sehr wenig Speicherplatz.

Für wen?

Jeder, der mehr aus seinem TI-99/4a herausholen möchte, sollte Mode Control II besitzen. Selbstverständlich sind alle neuen Befehle reine Maschinenprogramme, die über das Format 'CALL LINK("Befehl"(,Parameter))' aus den Basic aufgerufen werden. Einfach in die Basicprogramme einbinden!

MODE CONTROL II

DIE NEUEN BEFEHLE

BINIT - Initialisiert den Bit Map Modus

BITMAP - Übergibt die Plottergrafik ans VDP-Ram

* CLEAN - löscht einen Bildschirmteil im Text- und Grafikmodus

* CLTEXT - löscht den gesamten Text-Mode Screen

* GET - Input/Accept at über 32 bzw. 40 Spalten

MCLEAR - Löscht den Multicolor-Screen

MINIT - Initialisiert die Multicolor-Befehle

MEXIT - Kehrt vom Multicolor-Screen zum Basic zurück (ohne Zerstörung des Grafikbildschirms)

* MODE - Zum Hin- und Herschalten zwischen Text/Grafik

MTAKE - Feststellen der Farbe eines Multicolorpunktes

MULCOL - ruft den Multicolor-Screen auf

PLOT - Zeichnet eine Bit-Map-Linie oder einen Punkt

QUIT - Rückkehr zum TI-Titelbild als Programmbefehl

* ROLLUP - Scrollen des Bildschirms und Ausschnitten daraus (im Text- und im Grafik-Mode!)

SET - Färbt einen Multicolor-Punkt nach Wahl

* TABLE - Farbbestimmung im Textmodus

UNPLOT - Bewegen des Bit-Map-Zeichenstiftes ohne zu zeichnen

* WRITE - Print/Display at über 32 bzw. 40 Spalten

*) Die mit diesem Zeichen versehenen Programme sind auch zusammen als Text-Utilities erhältlich (siehe letzte Seite).
Auf der Diskette Mode Control II sind alle Befehle enthalten.

MODE CONTROL II

BENÖTIGTE KONFIGURATION:

Mode Control II läuft auf folgenden unterschiedlichen Konfigurationen:

1.) TI-99/4a Grundausstattung;
Editor/Assembler Modul;
BSCSUP-Utilities von Part-A Diskette;
Speichererweiterung;
Diskettenlaufwerk.

2.) TI-99/4a Grundausstattung;
Extended Basic Modul
Speichererweiterung;
Diskettenlaufwerk.

BEZUGSQUELLE:

RAUSCH & HAUB
Vertriebsgesellschaft dbR.
Postfach 320313
5300 Bonn 3

LIEFERUMFANG UND PREISE:

DISKETTE mit Programmpaket;
ausführliches Bedienungshandbuch;
komplett in Kunststoffkladde.

Für Editor/Assembler Modul:

04036	Mode Control II	komplett	DM 39.90
04038	Text Mode Util.	allein	DM 29.90

Für Extended Basic Modul:

04031	Mode Control II	komplett	DM 39.90
04037	Text Mode Util.	allein	DM 29.90

Für beide Versionen auf einer Diskette:

04033	Mode Control II	komplett	DM 49.90
04039	Text Mode Util.	allein	DM 36.90

Handbuch-Vorablieferung:

08101	Mode Control II	(24 S.)	DM 10.--

Handbücher werden beim Kauf des Originalprogramms innerhalb von 30 Tagen nach Bezug des Handbuchs mit 75% vergütet!

B

ABKÜRZUNGEN

Aus Gründen des Platzes und der Übersicht verwenden wir in den Syntax-Erläuterungen und Beschreibungen der TORPEDO BASIC Befehle auf den nachfolgenden Seiten eine Reihe von Abkürzungen und Zeichen. Diese bedeuten:

<...> Die in Klammern gesetzten Parameter sind Optional.

<...<,...>> Wie vor. wenn der zweite Parameter gesetzt wird, muß auch der erste gesetzt werden.

<..>,<..> Reihenfolge und Vorhandensein sind optional.

<../..> Verschiedene Möglichkeiten, eine kann gegeben sein.

../.. Verschiedene Möglichkeiten, eine muß gegeben sein.

nva	numerische Variable
sva	Stringvariable
num	numerischer Ausdruck
str	Stringausdruck
dig	Ziffer 0-9

zln	numerischer Ausdruck = Zeilennummer
asc	Ascii-Zeichen
zei	Wert oder Ausdruck 1-24
spa	Wert oder Ausdruck im Grafik-Modus 1-32 im Text-Modus 1-40
adr	numerischer Ausdruck = VDP-Adresse
for	String-Format 1-255
dir	Richtung 1-4
typ	Eingabe-Begrenzer (siehe GET-Befehl)

Im Anschluß an diese Kürzel können Ziffern zur Angabe einer Anzahl stehen.

nva1,nva2 steht für die Übergabe zweier numerischer Variablen.

Unterhalb der Syntax finden Sie horizontale Klammern mit den Angaben:

in	Parameter wird an das Maschinenprogramm übergeben.
out	Parameter wird vom Maschinenprogramm an das Basic-Programm zurückgegeben.

Für alle Befehle gilt folgende allgemeine Syntax:

CALL LINK("Programmname"<,parameter>)

Nähere Erläuterungen zur Arbeitsweise der Programme sowie einige Tabellen zum Finden gültiger Parameterwerte finden Sie im Anhang.

Zu jedem Befehl erhalten Sie die Bezeichnung, die Beschreibung und ein Beispiel, soweit erforderlich ein Beispielprogramm. Die Beispiele setzen voraus, daß TORPEDO BASIC geladen ist!

B R A N C H

Format:

```
CALL LINK("BRANCH",zln)
                  <in>
```

Beschreibung:

BRANCH verzweigt in Ihrem Basic-Programm zu einer Zeilennummer, die durch den Parameter-Ausdruck bestimmt wird. Ist die Zeile nicht vorhanden, erscheint die Nachricht BAD LINE NUMBER und die Programmausführung wird unterbrochen. Beachten Sie, daß bei RES (Umnummerierung der Zeilen) der numerische Ausdruck im BRANCH Befehl nicht korrigiert wird.

zln kann eine Zahl zwischen 1 und 32767 sein. Andere Werte erzeugen eine Fehlermeldung.

Beispiel:

```
100 INPUT A
110 IF A<1 THEN 100
120 CALL LINK("BRANCH",1000+A)
130 GOTO 100

1000 PRINT "GOTO 1000"
1010 GOTO 100
1020 PRINT "GOTO 1020"
1030 GOTO 100
1040 PRINT "GOTO 1040"
1050 GOTO 100
```

B R A N C H

Das Programm verzweigt, je nach eingegebenem Wert A, zu einer der Zeilen 1000,1020 oder 1040 und bringt eine entsprechende Nachricht auf den Bildschirm. Bei A>3 bricht das Programm mit einer Fehlermeldung ab.

C H R S E T

Format:

```
CALL LINK("CHRSET",dig)
                  <in>
```

Beschreibung:

CHRSET wählt, solange Sie den BUFFER nicht verändert haben, einen der folgenden Zeichensätze aus, welche mit den Codes des Epson FX-80 Druckers übereinstimmen. Der Zeichensatz wird interruptgesteuert immer wieder generiert, wenn mit USECOL die Routine aktiviert wird. Das erlaubt es, die Sonderzeichensätze auch im Direktmodus zu verwenden.

Wenn Sie den BUFFER verändern, stehen die Zeichensätze nicht mehr zur Verfügung. Stattdessen wird der Inhalt des Buffers zur Zeichendefinition herangezogen, was zu seltsamen Effekten führen kann - bei sachgemäßer Anwendung aber auch zu gewollten besonderen Zeichen. Nähere Hinweise dazu finden Sie im Anhang.

dig wählt den Zeichensatz aus:

0 USA
1 Frankreich
2 Deutschland
3 England
4 Dänemark
5 Schweden
6 Italien
7 Spanien
8 Japan
9 Wissenschaftlich (nicht auf Epson FX-80).

C H R S E T

Welche Character jeweils betroffen sind, zeigen die Tabellen im Anhang.

Beispiel:

```
100 CALL LINK("CHRSET",2)
```

stellt den deutschen Zeichensatz bzw. den Inhalt des Buffers in den Bytes 192 bis 287 als Zeichensatz zur Verfügung.

Hinweis:

Der ausgewählte Zeichensatz kann nach einer Interruptabschaltung durch USECOL wieder reaktiviert werden, auch wenn zwischenzeitlich der BUFFER verändert wurde.

Nach einer Bufferveränderung gehen jedoch die nicht aktivierten Zeichensätze verloren, da der Buffer diesen Bereich aus Speicherplatzgründen auch für 'vertsteckte Bildschirme' benutzt.

C L E A N

Format:

```
CALL LINK("CLEAN",zei,spa<,for<,asc<,dir>>>)
                  <_________in_____________>
```

Beschreibung:

CLEAN löscht einen 255 Character langen bereich, der bei der angegebenen Zeilen- und Spaltenposition beginnt, durch schreiben des Characters 32 (Space). Mit Angabe eines Formats (1-255) können Sie den zu löschenden Raum begrenzen.

Sind Zeile, Spalte und Format gegeben, ist außerdem eine Angabe des zu schreibenden ASCII-Characters möglich, wodurch CLEAR wie eine RPT$-Funktion arbeitet, allerdings auf dem ganzen definierten Bildschirmbereich des Grafik- oder Textmodes.

Sind Zeile, Spalte, Format und ASCII-Code gegeben, so können Sie weiterhin die Wiederholungsrichtung des Befehls bestimmen, was eine Funktion ähnlich HCHAR oder VCHAR zulässt, allerdings in alle vier Richtungen. Zeilen/Spaltenumbruch sind durch Fenster Nr. 0 definiert (siehe WINDOW).

Beispiel:

```
100 CALL LINK("CLEAN",10,10,40,64,2)
```

schreibt vierzig mal den Buchstaben A, beginnend an Position 10/10, untereinander, wobei 'nach rechts' geschrieben wird.

C L T E X T

Format:

```
CALL LINK("CLTEXT"<,win<,asc>>)
                  <____in____>
```

Beschreibung:

CLTEXT löscht den Bildschirm im Text- und Grafik-Modus. Da CLEAR im Text-Modus nicht richtig funktioniert, sollten Sie diesen Befehl stattdessen benutzen.

Bei Angabe eines Fensters wird nur das entsprechende Fenster gelöscht.

Löschen meint hierbei die Überschreibung mit ASCII 32 (Space).

Wenn eine Fensternummer angegeben ist, können Sie außerdem den zu schreibenden ASCII-Character bestimmen, der dann anstelle des SPACE-Characters benutzt wird.

Beispiel:

```
100 CALL LINK("CLTEXT",4,120)
```

schreibt in den als Fenster Nr. 4 definierten Bildschirmbereich den ASCII-Character 120, normalerweise 'x'.

C O P Y

Format:

CALL LINK("COPY",<Fensternr.,>"Gerätebezeichnung"<.Optionen>)
<_____in_____>

Beschreibung:

COPY fertigt eine Bildschirmkopie - auch von selbstdefinierten Sonderzeichen - auf einem Epson FX-80 oder einem vergleichbaren Drucker an. Es kann sowohl eine Parallel- als auch eine serielle Schnittstelle verwendet werden. Sie finden dazu im Manual zu Ihrer Schnittstelle weitere Informationen.

Optionen können zum Beispiel Übertragungsgeschwindigkeiten sein, wie etwa BA=300.

Während der Hardcopy-Anfertigung ist die Interrupt-Routine angeschaltet, was aber keine Auswirkungen hat, da eine Rückdefinitionen von Farben und Character-Sätzen nur bei Unterbrechungen im Direkt-Modus stattfindet.

Bei der Hardcopy werden im entsprechenden Teil des VDP-RAM gesetzte Punkte gedruckt und nicht gesetzte Punkte nicht gedruckt. Farben finden keine Berücksichtigung.

Mit der optionalen Fensternummer können Sie den auszudruckenden Bildschirmausschnitt begrenzen.

Beispiel:

```
100 CALL LINK("COPY","PIO.LF")
```

erzeugt eine Hardcopy über die Parallelschnittstelle. LF schaltet den automatischen Zeilenvorschub ab, da dieser durch das Maschinenprogramm bestimmt wird.

Hinweis:

Dieser Befehl ist im Text-Modus nicht verfügbar!

D I R

Format:

CALL LINK("DIR",Laufwerknummer<,zei1,zei2>)
<_____in_____>

Beschreibung:

DIR gibt das Diskettendirectory, also eine Auflistung aller auf einer Diskette enthaltenen Programme und Dateien, auf dem Bildschirm aus, einschließlich einer Angabe über den noch zur Verfügung stehenden Diskettenplatz. Mit der Laufwerknummer bezeichnen Sie die Laufwerke 1-3.

Zusätzlich können Sie eine Zeilenbegrenzung angeben, sodaß die Ausgabe des Directorys nur in einem festgelegten Bereich auf dem Bildschirm erscheint. Andernfalls erfolgt die Ausgabe auf dem gesamten Bildschirm.

Der Bildschirm bzw. der Inhalt des definierten Zeilenbereichs scrollt während der Ausgabe des Directorys nach oben. Durch Niederhalten einer Taste können Sie die Ausgabe unterbrechen. Wenn Sie BACK (FCTN-<9>) drücken, wird die Directory-Ausgabe abgebrochen.

DIR löscht den gewählten Bildschirmteil nicht. Sie sollten jedoch vorher CLTEXT durchführen. Während DIR läuft, ist die Interrupt-Routinen-Ausführung unterbrochen.

Beispiel:

```
100 CALL LINK("DIR",1,18,24)
```

gibt den Diskettenkatalog (Directory) von der Diskette in Laufwerk 1 auf dem Bildschirm in den Zeilen 18 bis 24 aus.

G E T

Format:

```
CALL LINK("GET",zei,spa<,for<,typ>>,nva/sva)
                  <_____in_____> <_out_>
```

Beschreibung:

GET ermöglicht Eingaben über die Tastatur. Der Befehl arbeitet ähnlich dem Ihnen bekannten DISPLAY AT im Extended Basic.

zei und spa definieren die erste Eingabeposition. Die obere Grenze von spa richtet sich nach dem Modus, in dem Sie sich befinden.

for ist optional und bestimmt die Länge des Eingabbereichs. Ist das Eingabeformat nicht spezifiziert, ist der Bereich auf 255 Character festgelegt.

typ kann nur gegeben werden, wenn auch for spezifiziert ist. Sie können damit, ähnlich wie bei VALIDATE im DISPLAY AT Befehl des Extended Basic die Eingabe auf Bestimmte Tasten Beschränken. Enthält die Eingabe andere als vorgesehene Zeichen, wird sie ignoriert. Die Eingabe muß dann, ohne daß eine Fehlermeldung gezeigt wird, wiederholt werden. Die folgenden Werte bestimmen die Art des Eingabe-Datentyps:

0 Keine Beschränkung
1 Ziffern
2 Ziffern und mathematische Zeichen
3 Hexadezimal-Zeichen
4 Großbuchstaben
5 Kleinbuchstaben
6 Groß- und Kleinbuchstaben
7 Großbuchstaben und Satzzeichen
8 Groß-, Kleinbuchstaben und Satzzeichen
9 Groß-, Kleinbuchstaben und Zeichen der Internationalen Zeichensätze

Anstelle einer Typen-Ziffer kann auch ein String spezifiziert werden.

"String" = Zeichen, die in Anführungszeichen angegeben werden.

In einem GET-Befehl kann immer ein typ oder die Möglichkeit des selbstdefinierten Strings benutzt werden.

Wenn Hexadezimalzeichen als typ spezifiziert werden, und die Ausgabevariable numerisch ist, enthält diese anschließend die Zahl in Dezimalschreibweise.

Möglich sind jedoch nur Hexadezimalzahlen zwischen 0 und FFFF, wobei jene ab >8000 als negativ interpretiert werden.

G E T

Der Eingabestring oder die Zahl wird in der Variablen ans Basic übergeben, wenn ENTER gedrückt wird und eine ggf. festgelegte Datentypbegrenzung nicht überschritten wurde. Wenn Sie einen String eingeben, obwohl die Variable numerisch ist, erscheint nach ENTER eine Fehlermeldung.

Beispiele:

```
100 CALL LINK("GET",10,1,A$)
110 CALL LINK("GET",11,1,S,B)
120 CALL LINK("GET",12,1,10,4,M$)
130 CALL LINK("GET",13,1,4,2,DEZ)
```

Zeile 100 erwartet die Eingabe eines bis zu 255 Zeichen langen Strings. Zeile 200 akzeptiert eine bis zu 5 Stellen lange Zahl. Zeile 120 gestattet die Eingabe von bis zu 10 Großbuchstaben, während Zeile 130 vier Hexadezimalziffern erwartet. Nehmen wir an, die Hexadezimalziffer ist ACB4, dann enthält die Variable DEZ nach ENTER den Wert -11445.

G E T S T R

Format:

```
CALL LINK("GETSTR",zei,spa,for,sva)
                   <___in____><out>
```

Beschreibung:

GETSTR übernimmt einen String von bestimmter Länge for, beginnend bei zei/spa, vom Bildschirm und übergibt diesen in der Stringvariablen ans Basic.

Beispiel:

```
100 CALL LINK("GETSTR",10,10,24,A$)
```

Angenommen, auf dem Bildschirm befindet sich beginnend an Position 10/6 der String

"Donaudampfschiffahrtskapitaensmuetze"

dann enthält A$ nach Durchführung des Befehls den Teilstring

"Dampfschiffahrtskapitaen".

H I D E

Format:

```
CALL LINK("HIDE",win1<,win2,...,win16>)
                  <_______in_______>
```

Beschreibung:

HIDE kopiert den Inhalt eines Bildschirmfensters in den dazu korrespondierenden Bufferbereich. Sie finden im Anhang mehr Informationen über die Benutzung des Buffers.

Beispiel:

```
100 CALL LINK("HIDE",1,3,12)
```

kopiert die definierten Fenster 1,3 und 12 in die mit ihnen jeweils korrespondierenden Bufferbereiche.

Hinweis:

Beachten Sie hierzu auch den WINDOW-Befehl.

I N S T R

Format:

CALL LINK("INSTR",zei,spa,str/sva)
<____in____>

Beschreibung:

INSTR übergibt einen String direkt in den BUFFER, und zwar an jene Buffer-Position, welche mit der gegebenen Zeile und Spalte des Bildschirms korrespondiert, die mit zei und spa gegeben wird.

Beispiel:

```
100 F$="TORPEDO BASIC"
110 CALL LINK("INSTR",5,9,F$)
```

schreibt die ASCII-Werte des Strings TORPEDO-BASIC an die Bufferposition, welche zu den Bildschirmpositionen ab 5/9 korresponsiert.

Hinweis:

Beachten Sie hierzu die Hinweise im Anhang.

MODE

Format:

```
CALL LINK("MODE",Modusflag)
                 <___in__>
```

Beschreibung:

Mit diesem Befehl können Sie zwischen dem Text-Modus und dem Grafik-Modus hin- und herschalten. Sie finden im Anhang mehr Informationen zu den Modi.

Modusflag=1 schaltet den Grafik-Modus ein.
Modusflag=2 schaltet den Text-Modus ein.

Beispiel:

```
100 CALL LINK("MODE",2)
```

aktiviert den Text-Modus. Im Text Modus stehen 24 Zeilen zu je 40 Zeichen zur Verfügung.

Hinweis:

Dieser Befehl darf nicht im Direkt-Modus verwendet werden. Bevor Sie vom Text-Modus aus ein Programm unterbrechen oder beenden, muß undbedingt in den Grafik-Modus zurückgeschaltet werden.

OUTSTR

Format:

```
CALL LINK("OUTSTR",zei,spa,for,sva)
                   <___in___><out>
```

Beschreibung:

OUTSTR übernimmt einen String von bestimmter Länge in die Stringvariable, welches sich an der Bufferposition befindet, die zur Bildschirmposition zei/spa korrespondiert.

Beispiel:

```
100 CALL LINK("OUTSTR",8,8,1,CH$)
```

übernimmt ein Zeichen aus dem Buffer, welches sich an der Position befindet, die zu den Bildschirmkoordinaten 8/8 korrespondiert. Das Zeichen wird in CH$ gespeichert.

Hinweis:

Beachten Sie dazu die Hinweise im Anhang.

Q U I T

Format:

```
CALL LINK("QUIT")
```

Beschreibung:

Aufruf des TI-Titelbildes. Alle Programme im RAM gehen dadurch verloren, und das Monitor-Testbild wird angezeigt. Sie können so zum Beispiel Ihre Programme gegen unbefugte Benutzung absichern.

Hinweis:

Dieser Befehl schließt keine Dateien, da er nicht BYE, sondern dem Drücken der QUIT-Taste entspricht.

S C R O L L

Format:

```
CALL LINK("SCROLL"<,win<,str/sva<,dir>>>)
                  <_________in_________>
```

Beschreibung:

SCROLL ermöglicht es, den Bildschirm in eine beliebige Richtung zu bewegen, wobei die jeweils freiwerdende Spalte oder Zeile durch Leerzeichen ersetzt wird.

Wenn eine Fensternummer angegeben wird, scrollt nur der mit dieser Nummer definierte Bildschirmbereich.

Wenn win spezifiziert ist, können Sie einen String angeben, mit dessen Inhalt die freiwerdenden Stellen ausgefüllt werden. Zu kurze Strings werden mit Leerzeichen ergänzt, bei zu langen wird vom ersten Zeichen an nur der benötigte Teil verwendet. Beim Scrollen nach Oben und Unten erfolgt die Stringausgabe von links nach rechts, beim Scrollen nach links oder rechts erfolgt diese von Oben nach Unten.

Wenn sowohl ein Fenster als auch ein String definiert ist, können Sie eine Scrollrichtung (1-4) angeben. Ohne diese Angabe scrollt der Screen(ausschnitt) nach oben.

Folgende Werte geben die Richtung an:

1 = nach oben
2 = nach rechts
3 = nach unten
4 = nach links

S C R O L L

Beispiel:

```
100 X$="xxxxxxxxxx"
110 CALL LINK("SCROLL",4,X$,2)
```

Der mit Window Nr. 4 festgelegte Bildschirmbereich scrollt nach rechts. Die äußerste rechte Spalte geht verloren. Die links freiwerdende Spalte wird mit 'x' gefüllt. Ist der String zu kurz, werden die restlichen Positionen mit CHR$(32) (Space) gefüllt.

Hinweis:

Beachten Sie hierzu bitte den WINDOW-Befehl.

S E A R C H

Format:

```
CALL LINK("SEARCH",win,str/asc,zei,spa)
                   <___in____> <_out_>
```

Beschreibung:

SEARCH sucht nach einem String oder nach einem ASCII-Zeichen in einem Bildschirmfenster. Die Position des ersten Auftretens des ASCII-Zeichens oder des ersten Zeichens des Strings wird als absolute Koordinate in zei und spa an das Basic-Programm zurückgegeben.

Der Befehl kann zum Beispiel bei Spielen schnell eine Figur finden, auch wenn die Koordinaten nicht gespeichert sind, wobei die Maschinenroutine um etliches schneller ist als eine gleichwertige GCHAR-Schleife.

Beispiele:

```
100 CALL LINK("SEARCH",2,"HAGERA",X,Y)
110 CALL LINK("SEARCH",5,66,Z,S)
```

Zeile 100 gibt in X/Y das erste Auftreten des Strings "HAGERA" an (X/Y=Position des 'H' im String). Zeile 110 findet das erste Auftreten des Buchstaben 'B' und gibt dieses in Z/S zurück.

S W A P

Format:

```
CALL LINK("SWAP",win1<,win2,...,win16>)
                 <_____in_____>
```

Beschreibung:

SWAP tauscht den Inhalt des Bildschirmfensters mit dem korrespondierendem Bufferbereich aus.

Beispiel:

```
100 CALL LINK("SWAP",1,9,12,14)
```

tauscht die Inhalte der Fenster 1,9,12 und 14 mit den jeweils korrespondierenden Bufferbereichen aus.

Hinweis:

Beachten Sie hierzu die Hinweise zur BUFFER-Benutzung im Anhang.

T A B L E

Format:

```
CALL LINK("TABLE",win<,far1,far2>)
                  <_____in_____>
```

Beschreibung:

TABLE stellt einen bildschirmorientierten Cursor im angegebenen Fenster zur Verfügung. Gleichzeitig kann wahlweise eine neue Farbkombination gewählt werden. Die Beschreibung dazu finden Sie bei USECOL, allerdings schaltet TABLE keinen Interrupt ein.

Wenn Sie jedoch eine Farbe wählen, bezieht diese sich auch auf das Interruptprogramm; ein ggf. aktiver Interrupt ist während der Farbumschaltung blockiert, was jedoch keine Auswirkung auf den Programmablauf hat.

Welche Cursor-Funktionen zur Verfügung stehen, ist in der Übersichtstabelle zm Anfang dieses Buches erläutert.

Die Übergabe von Texten ans Basic, die mit TABLE entwickelt wurden, kann zum Beispiel mit GETSTR erfolgen.

TABLE

Beispiele:

```
100 CALL LINK("TABLE",5)
110 CALL LINK("TABLE",0,1,16)
```

Zeile 100 stellt in Fenster 5 einen bildschirmorientierten Cursor zur Verfügung. Zeile 100 bewirkt gleiches in Fenster 0, was in der Regel der gesamte Bildschirm ist (siehe dazu die Erläuterung zum WINDOW-Befehl). Gleichzeitig werden für den gesamten Bildschirm (auch wenn das Fenster kleiner ist) die Farben schwarz/weiß gesetzt.

TAKE

Format:

```
CALL LINK("TAKE",win1<,win2,...,win16>)
                  <_______in_________>
```

Beschreibung:

TAKE kopiert den zum angegebenen Bildschirmfenster korrespondierenden Bufferbereich in das Fenster, wobei der alte Bildschirminhalt verloren geht.

Beispiel:

```
100 CALL LINK("TAKE",2,5)
```

überschreibt den Bildschirminhalt an den Stellen, der durch die Fenster 2 und 5 definiert ist, mit dem jeweiligen korrespondierenden Bufferbereich.

Hinweis:

Beachten Sie hierzu die Erläuterungen des Window-Befehls und die Buffer-Beschreibungen im Anhang.

T O R P E D

Format:

```
CALL LINK("TORPED")
```

Beschreibung:

Dieser Befehl schaltet die Standartfarben ein und gibt eine Bereitschaftsmeldung für TORPEDO BASIC. Der Befehl wird von den Diskettenprogrammen automatisch nach dem Laden der Systemerweiterung durchgeführt. Cassettenbenutzer müssen dieses selbständig tun.

Hinweis:

Dieser Befehl kann nur im Grafik-Modus benutzt werden. Der Bildschirminhalt wird beim Aufruf von TORPED gelöscht. Dies kann Ihren Programmablauf beeinträchtigen; deshalb sollte TORPED nur im Direkt-Modus aufgerufen werden. Für die Dauer der Durchführung sind die Interrupt-Routinen gesperrt.

U S E C O L

Format:

```
CALL LINK("USECOL"<,vf,hf>)
                  <__in__>
```

Beschreibung:

USECOL schaltet alle aktivierten Interruptroutinen von TORPEDO BASIC ab, wenn keine Farben angegeben sind.

Wenn Farben angegeben sind, so werden diese gesetzt und alle Interrruptroutinen aktiviert.

Beispiele:

```
100 CALL LINK("USECOL",16,1)
110 CALL LINK("USECOL")
```

In Zeile 100 werden die Farben weiß/schwarz gesetzt und alle Interrupt-Routinen aktiviert. In Zeile 110 werden alle Interrupts wieder abgeschaltet, was bedeutet, daß die Basic Standartfarben gesetzt und ein Sonderzeichensatz gelöscht wird, sobald das Programm unterbricht oder endet. Außerdem ist die Löschfunktion CTRL-1 (CLEAR SCREEN) dann desaktiviert.

U S E C O L

Hinweis:

Während der Durchführung wird der Interrupt blockiert, was jedoch keine Auswirkungen auf den Programmablauf hat.

VDPEEK

Format:

```
CALL LINK("VDPEEK",adr,nval<,nval2,...nval15>)
                    <in><_______out_______>
```

Beschreibung:

VDPEEK ermöglicht das Auslesen von Bytes aus dem VDP-Ram mit dem Extended Basic Modul. Mit einer Anweisung können durch Angabe einer Adresse des VDP-Rams (>0000->3FFF) und einer entsprechenden Anzahl numerischer Variablen bis zu 15 Bytewerte gleichzeitig gelesen werden.

Beispiel:

```
100 CALL LINK("VDPEEK",0,A,B,C,D)
```

gibt an A den Inhalt von VDP-Adresse >0000, an B den Inhalt von >0001, an C den Inhalt von >0002 und an D den Inhalt von >0004. Beim Versuch, Werte aus einem nicht vorhandenden Bereich des VDP-Ram zu lesen, erscheint eine Fehlermeldung.

Hinweise:

Für das Editor/Assembler-Modul kann der bereits dort eingebaute Befehl PEEKV verwendet werden. VDPEEK ist nur auf dem Extended Basic Modul vorrätig.

V D P E E K

ACHTUNG:

In allen Versionen muß der ASCII-Offset berücksichtigt werden, wenn Sie zum Beispiel Zeichen auf dem Bildschirm darstellen wollen. Dieser Offset ist für die Benutzung von maschinenprogrammen aus dem Basic heraus erforderlich, da sich im VDP-Ram verschiedene Tabellen überlagern. Sie finden dazu mehr Informationen im ASSEMBLER KURS III von HAGERA(R) sowie im Handbuch zum Editor/Assembler.

V D P O K E

Format:

```
CALL LINK("VDPOKE",adr,asc1<,asc2,...asc15>)
                  <_____in_____>
```

Beschreibung:

VDPOKE ermöglicht das Schreiben von Bytes in das VDP-Ram mit dem Extended Basic Modul. Mit einer Anweisung können durch Angabe einer Adresse des VDP-Rams (>0000->3FFF) und einer entsprechenden Anzahl ASCII-Codes (0-255) bis zu 15 Bytewerte gleichzeitig geschrieben werden.

Beispiel:

```
100 CALL LINK("VDPOKE",0,72,65,76,76,79)
```

schreibt, beginnend an der oberen linken Bildschirmecke, das Wort 'HALLO' auf den Bildschirm, da ab Adresse >0000 im VDP-Ram die Bildschirmdarstellungstabelle beginnt.
Beim Versuch, in eine Adresse zu schreiben, die im VDP-Ram nicht existiert, erscheint eine Fehlermeldung.

Hinweise:

Für das Editor/Assembler-Modul kann der bereits dort eingebaute Befehl PEEKV verwendet werden. VDPEEK ist nur auf dem Extended Basic Modul vorrätig.
Beachten Sie hierzu unsere Hinweise auf den Aufbau des VDP-Ram im Anhang.

V D P O K E

ACHTUNG: Das 'poken' an verschiedene Adressen kann einen Programmabsturz herbeiführen. Speichern Sie ein Programm, welches diesen Befehl enthält, immer erst ab, bevor Sie es laufen lassen!

Ferner beachten Sie bitte die unter VDPEEK genannten Hinweise.

W I N D O W

Format:

```
CALL LINK("WINDOW",win,zei1,spa1,zei2,spa2)
                   <________in__________>
```

Beschreibung:

WINDOW definiert ein Bildschirmecke durch Angabe der beiden Zeilen und Spalten, die es begrenzen. zei1/spa1 definieren die obere linke, zei2/spa2 die untere rechte Ecke.

Sie können insgesamt 32 Bildschirmfenster frei definieren, unabhängig jeweils 16 im Grafik-Modus und 16 im Text-Modus. Die Fenster, die für den einen Modus definiert sind, haben auf jene des anderen Modus keine Auswirkung, obwohl die Bezeichnung die Gleiche ist.

win=1 bezeichnet sowohl ein Fenster im Grafik-Modus als auch eines im Text-Modus, woraus folgt, daß win zwischen 0 und 15 sein kann. Wenn Sie WINDOW benutzen, solange sich der Computer im Direkt-Modus befindet oder während der Programmdurchführung der Grafik-Modus aktiv ist, können mit WINDOW die 16 Fenster des Grafik-Modus definiert werden. Ist hingegen, was nur während eines Programmablaufs möglich istm der Text-Modus aktiv, so verändert WINDOW die Fenster des Text-Modus, nicht aber die des Grafik-Modus.

Wenn TORPEDO BASIC geladen wird, werden die 32 Fenster auf eine bestimmte Art und Weise vordefiniert, um Basic-Ram-Platz für die Definition häufig gebrauchter Fenster zu sparen. Fenster 0 ist dabei der gesamte Bildschirm. Welche Begrenzungen die übrigen Fenster haben, zeigt eine Grafik im Anhang.

WINDOW

Fenster 0	Grafik-Modus	Text-Modus
zei1	1	1
spa1	1	1
zei2	24	24
spa2	32	40

Durch Umschalten der Modi gehen weder Fensterdefinitionen verloren, noch sind die Definitionen in irgendeiner Weise voneinander abhängig.

Die Fenster können danach in verschiedenen Befehlen durch Angabe der Nummer, mit der sie definiert sind, benutzt werden. Solche Window-Befehle sind:

CLTEXT, HIDE, SCROLL, SEARCH, SWAP, TABLE und TAKE.

Fenster Nummer 0 sollte, auch wenn dies möglich ist, nicht umdefiniert werden, da sonst zum Beispiel bei CLTEXT nicht mehr der gesamte Bildschirm, sondern eben nur noch der Bereich von Fenster 0 gelöscht wird.

Durch Kombination der WINDOW- und BUFFER-Befehle sind sehr schöne Programmiermethoden möglich.

Fenster können sich gegenseitig überlappen, sie können ineinander verschachtelt sein oder aneinander angrenzen. Bereiche, die sich überschneiden, sind dann entsprechend durch verschiedene Fensternummern ansprechbar.

W I N D O W

Die Befehle GET und WRITE beziehen sich beim Zeilenumbruch auf Fenster 0, unabhängig von der Start-Koordinate. Beachten Sie die Hinweise bei diesen Befehlen. Andere Fenstergrenzen werden von ihnen nicht erkannt. Benutzen Sie stattdessen den TABLE-befehl oder Beschreiben Sie Fenster durch Kopieren von Buffer-Arrays. Nähere Informationen hierzu finden Sie im Anhang.

Mit dieser neuen Window-Technologie steht Ihr TI-99/4a anderen Computertypen, die solche Befehle bereits implementiert haben, in nichts mehr nach, zumal jetzt endlich auch die volle Bildschirmbreite von 40 Zeichen/Zeile zur Verfügung steht.

Die WINDOW-Befehle eignen sich sowohl für verschiedene Verwaltungprogramme (Datei, Textverarbeitung) als auch für Spiele (durch den schnellen Scroll-Befehl).

Beispiel:

```
100 FOR I=2 TO 1 STEP -1
110 CALL LINK("MODE",I)
120 FOR J=0 TO 15
130 READ Z1,S1,Z2,S2
140 CALL LINK("WINDOW",J,Z1,S1,Z2,S2)
150 NEXT J
160 NEXT I

1000 DATA ...(128 Werte für Beginn/Endkoordinaten)
```

Das Programm initialisiert die 32 Fenster der beiden Modi mit den Werten, welche Sie in einer entsprechenden Datentabelle abgelegt haben. Mit dieser Schleife erreichen Sie eine blitzschnelle Definition der Fensterbereiche.

W I N D O W

Hinweis:

Wenn sowohl Startzeile und Endzeile als auch Startspalte. und Endspalte identisch sind, ist das Fenster eine Bildschirmposition groß. Ein Fenster mit 0 Bildschirmpositio- nen ist nicht möglich.

W R I T E

Format:

```
CALL LINK("WRITE",zei,spa,str/sva/num/nva)
                  <__in__><____out____>
```

Beschreibung:

WRITE schreibt einen String oder eine Zahl, beginnend bei Position zei/spa, auf den Bildschirm. Überschreitet die Länge der Ausgabe den Bildschirmbereich, unterbricht das Programm mit einer Fehlermeldung.

Beispiele:

```
100 A=15
110 F$="HAGERA"
120 CALL LINK("WRITE",10,10,A+5)
130 CALL LINK("WRITE",12,10,F$)
140 CALL LINK("WRITE",14,10,"HALLO")
150 CALL LINK("WRITE",16,10,100)
```

W R I T E

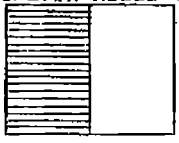
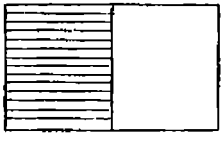
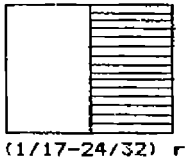
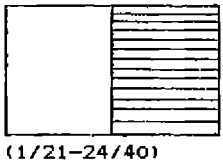
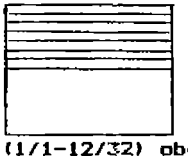
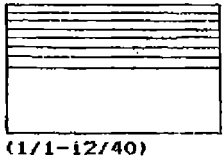
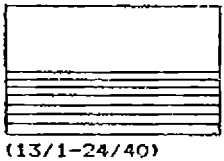
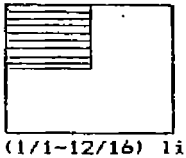
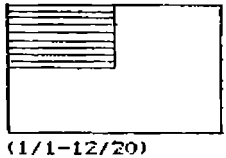
Zeile 120 führt zur Ausgabe von '20' an 10/10.
Zeile 130 schreibt den String "HAGERA", beginnend an Position 12/10, auf den Bildschirm.
Zeile 140 schreibt den String "HALLO" auf den Bildschirm, der an Position 14/10 beginnt.
Zeile 150 gibt die Zahl '100' an Position 16/10 aus.

Hinweise:

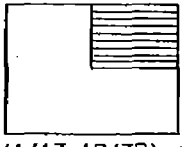
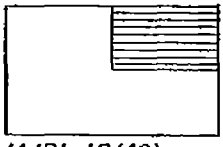
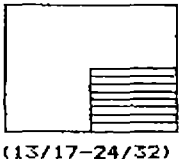
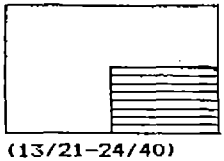
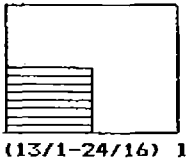
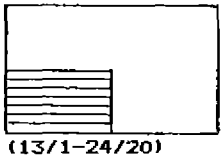
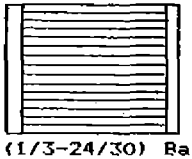
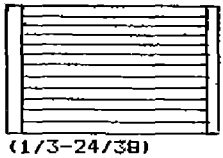
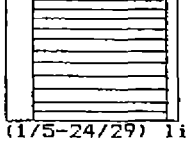
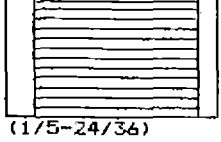
Die Zeichenzahl pro Zeile richtet sich nach dem ausgewählten Modus. Ausschlaggebend ist der für den entsprechenden Modus festgelegte Bereich des Fensters 0.

Wenn Sie als Startpunkt für eine Ausgabe einen Bereich außerhalb des fensters 0 angeben oder versuchen, über die Begrenzungen des Fensters hinauszuschreiben, erscheint eine Fehlermeldung.

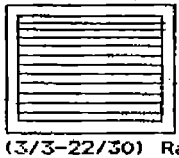
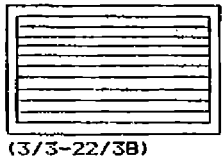
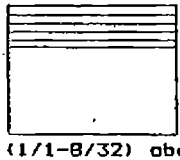
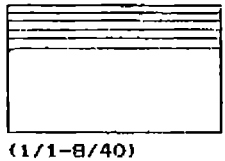
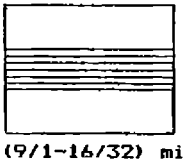
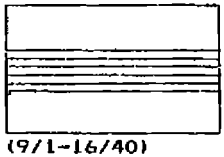
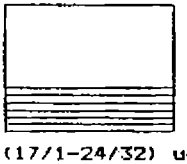
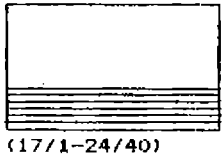
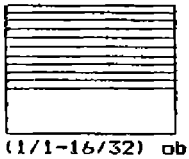
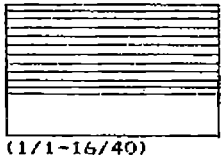
VORDEFINIERTE FENSTERBEREICHE

Nr.	Grafik-Modus und Direkteingabe	Text-Modus
1	 (1/1-24/16) linke Hälfte	 (1/1-24/20)
2	 (1/17-24/32) rechte Hälfte	 (1/21-24/40)
3	 (1/1-12/32) obere Hälfte	 (1/1-12/40)
4	 (13/1-24/32) untere Hälfte	 (13/1-24/40)
5	 (1/1-12/16) linkes oberes Viertel	 (1/1-12/20)

VORDEFINIERTE FENSTERBEREICHE

Nr.	Grafik-Modus und Direkteingabe	Text-Modus
6	 (1/17-12/32) rechtes oberes Viertel	 (1/21-12/40)
7	 (13/17-24/32) rechtes unteres Viertel	 (13/21-24/40)
8	 (13/1-24/16) linkes unteres Viertel	 (13/1-24/20)
9	 (1/3-24/30) Basic Standart / kl. Rand	 (1/3-24/38)
10	 (1/5-24/29) linker und rechter Rand	 (1/5-24/36)

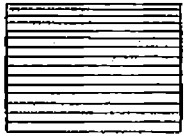
VORDEFINIERTE FENSTERBEREICHE

Nr.	Grafik-Modus und Direkteingabe	Text-Modus
11	 (3/3-22/30) Rahmen	 (3/3-22/38)
12	 (1/1-8/32) oberes Drittel	 (1/1-8/40)
13	 (9/1-16/32) mittleres Drittel	 (9/1-16/40)
14	 (17/1-24/32) unteres Drittel	 (17/1-24/40)
15	 (1/1-16/32) obere Zweidrittel	 (1/1-16/40)

VORDEFINIERTE FENSTERBEREICHE

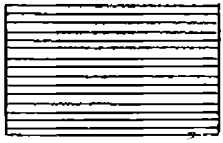
Nr. Grafik-Modus und Direkteingabe

0



(1/1-24/32) Ganzer Bildschirm

Text-Modus



(1/1-24/40)

Fenster 0 sollte nicht ohne zwingenden Grund verändert werden!

Jede Fensterdefinition benötigt vier Bytes in der Fensterdefinitionstabelle. Diese Tabelle ist 128 Bytes lang. Die Daten sind in den RAM-Adressen folgendermaßen enthalten, vorausgesetzt, das Maschinenprogramm startet bei >A000 (Sie können die Fensterbereiche auch mit der LOAD-Instruktion verändern, was jedoch länger dauert und auch mehr Basic-Speicherplatz in Anspruch nimmt.):

Modus 1	Beginnzeile	-Spalte	Endzeile	-Spalte
0	A3C0	A3C1	A3C2	A3C3
1	A3C4	A3C5	A3C6	A3C7
2	A3C8	A3C9	A3CA	A3CB
3	A3CC	A3CD	A3CE	A3CF
4	A3D0	A3D1	A3D2	A3D3
5	A3D4	A3D5	A3D6	A3D7
6	A3D8	A3D9	A3DA	A3DB
7	A3DC	A3DD	A3DE	A3DF
8	A3E0	A3E1	A3E2	A3E3
9	A3E4	A3E5	A3E6	A3E7
10	A3E8	A3E9	A3EA	A3EB
11	A3EC	A3ED	A3EE	A3EF
12	A3F0	A3F1	A3F2	A3D3
13	A3F4	A3F5	A3F6	A3F7
14	A3F8	A3F9	A3FA	A3FB
15	A3FC	A3FD	A3FE	A3FF

VORDEFINIERTE FENSTERBEREICHE

Modus 1	Beginnzeile	-Spalte	Endzeile	-Spalte
0	A400	A401	A402	A403
1	A404	A405	A406	A407
2	A408	A409	A40A	A40B
3	A40C	A40D	A40E	A40F
4	A410	A411	A412	A413
5	A414	A415	A416	A417
6	A418	A419	A41A	A41B
7	A41C	A41D	A41E	A41F
8	A420	A421	A422	A423
9	A424	A425	A426	A427
10	A428	A429	A42A	A42B
11	A42C	A42D	A42E	A42F
12	A430	A431	A432	A433
13	A434	A435	A436	A437
14	A438	A439	A43A	A43B
15	A43C	A43D	A43E	A43F

Das jeweils in einem Befehl aktivierte Fenster wird in einem anderen Speicherbereich übernommen. Dies reicht jedoch zur Benutzung eines Fensters nicht aus, weshalb Sie hier nur die TORPEDO BASIC Befehle und nicht die LOAD-Instruktion verwenden sollten. Aus diesem Grund haben wir die einzelnen Zutritts-Adressen nicht angegeben.

Eine Übersicht, wie die Speichererweiterung von TORPEDO BASIC genutzt wird, finden Sie in diesem Buchteil.

BUFFER-BENUTZUNG

TORPEDO BASIC stellt einen Speicherbereich von 960 Bytes zur Verfügung, den Sie auf verschiedenste Art und Weise benutzen können. Der Buffer-Bereich beginnt an der Adresse, an die die Systemerweiterung geladen wird, normalerweise bei >A000.

Wenn Sie TORPEDO BASIC geladen haben, enthält der Buffer die Charactermuster der Sonderzeichen, welche Sie mit CHRSET aktivieren können. Jeder Sonderzeichenblock besteht aus 96 Bytes, von denen jeweils 8 aufeinanderfolgende das Muster eines Zeichens bilden.

Definiert werden mit CHRSET die ASCII-Codes 35, 36, 64, 91, 92, 93, 94, 96, 123, 124, 125 und 126, wie sie in der Mustertabelle in diesem Anhang beschrieben sind.

Jeder Block enthält einen Sonderzeichensatz wie folgt:

Byte-Bereich	Zeichensatz
1 - 96	USA
97 - 192	Frankreich
193 - 288	Deutschland
289 - 384	England
385 - 480	Dänemark
481 - 576	Schweden
577 - 672	Italien
673 - 768	Spanien
769 - 864	Japan
865 - 960	Scientific

BUFFER-BENUTZUNG

Die Sonderzeichensätze entsprechen - mit Ausnahme des wissenschaftlichen Satzes (Scientific), denen nach der ASCII-Tabelle zum Epson FX-80 Drucker.

Durch CHRSET wird ein solcher Zeichensatz in den Zugriffsblock für das Interrupt-Programm kopiert und dann ständig ins VDP-Ram kopiert, weshalb der Zeichensatz auch dann noch zur Verfügung steht, wenn der Buffer verändert oder das Programm abgebrochen wird.

Sie können den Inhalt dieses Buffers verändern und dazu benutzen, ihren Bildschirm zu bearbeiten. Dabei stellt der Buffer exakt dieselbe Anzahl Bytes zur Verfügung wie der Bildschirm. Im Grafik-Modus werden allerdings nur die ersten 768 Byte-Eingänge benutzt. Sie können sich den Buffer als Vektor vorstellen, der exakt dieselbe Ausdehnung hat wie ihr Screen.

Mit bestimmten Befehlen ist es nun möglich, den Inhalt des Bildschirms oder teile daraus in den entsprechenden Bufferteil zu kopieren, den Buffer in den Bildschirm zu kopieren oder beide Inhalte auszutauschen. Außerdem können Sie in diesen 'versteckten Bildschirm', den Buffer, direkt etwas hineinschreiben, ohne daß dies auf dem sichtbaren Bildschirm angezeigt wird.

Der Buffer kann durch LOAD-Befehle oder kopieren eines leeren Bilschirms in den Buffer gelöscht werden. Bei einem Buffer-übertrg werden keine Farben verändert.

BUFFER-BENUTZUNG

ACHTUNG: Die unsachgemäße Verkettung von Bufferbefehlen und dem Befehl CHRSET kann unangenehme Folgen haben. Entscheiden Sie sich dabei unbedingt, ob Sie die implementierten Zeichensätze oder die Bufferbefehle verwenden möchten. Sie können auch erst mit CHRSET einen Zeichensatz wählen, um später die Buffer-Befehle zur Verfügung zu haben.

SONDERZEICHENSÄTZE

Folgende Sonderzeichen werden von TORPEDO BASIC nach dem Laden zur Verfügung gestellt:

USA

35 #	91 [	94 ^	124 |
36 $	92 \	96 `	125 }
64 @	93]	123 {	126 ~

Frankreich

35 #	91 °	94 ^	124 ù
36 $	92 ç	96 `	125 è
64 à	93 §	123 é	126 ¨

Deutschland

35 #	91 Ä	94 ^	124 ö
36 $	92 ö	96 `	125 ü
64 §	93 Ü	123 ä	126 ß

Großbritannien

35 £	91 [	94 ^	124 |
36 $	92 \	96 `	125 }
64 @	93]	123 {	126 ~

SONDERZEICHENSÄTZE

Dänemark

35 #	91 Æ	94 ^	124 æ
36 $	92 Ø	96 `	125 å
64 @	93 Å	123 æ	126 ~

Schweden

35 #	91 Ä	94 Ü	124 ö
36 ¤	92 Ö	96 é	125 à
64 É	93 Å	123 ä	126 ü

Italien

35 #	91 °	94 ^	124 ò
36 $	92 \	96 ù	125 è
64 @	93 é	123 à	126 ì

Spanien

35 ₧	91 ;	94 ^	124 ñ
36 $	92 Ñ	96 `	125 }
64 @	93 ¿	123 ¨	126 ~

SONDERZEICHENSÄTZE

Japan

35 #	91 Ä	94 ^	124 ö
36 $	92 ö	96 `	125 ü
64 §	93 ü	123 ä	126 ß

Scientific

35 ≠	91 [	94 ^	124 ¦
36 PI	92 \	96 v	125 }
64 @	93]	123 Sum	126 ~

DER GRAFIK MODUS

Im Grafik-Modus befinden Sie sich, wenn Sie den TI-99/4a Computer einschalten. Es stehen 24 Zeilen zu je 32 Zeichen zur Verfügung. Die Basic-Befehle PRINT, INPUT, DISPLAY und ACCEPT benutzen von diesen 32 Zeichen aber nur die mittleren 28; es bleiben am rechten und linken Rand zwei Spalten ungenutzt. Sie können Sprites und deren automatisches Bewegungsmoment benutzen.

Die Befehle des TORPEDO BASIC, GET und WRITE, erlauben es, den benutzbaren Spaltenausschnitt zu verändern. Nach dem Laden der Systemerweiterung steht die volle Bildschirmbreite mit 24x32 Zeichen zur Verfügung.

Die mit GET, WRITE und anderen Zeilenbefehlen benutzten Umbruchstellen können durch Veränderung des Fensters Nr. 0 nach Ihren Wünschen verändert werden. Dieses Fenster definiert die absolute Bildschirmgröße, wenn auch andere Fenster wiederum über diesen Bereich hinausreichen können.

Nach dem Laden der Erweiterung umfasst Fenster Nr. 0 den gesamten Bildschirmbereich. Die Darstellung von WRITE-Ausgaben erfolgt folgendermaßen:

```
--------------------------------
Hier sehen Sie, wie Fenster Nr.0
definiert ist, wenn Sie die Sys-
temerweiterung geladen haben.
--------------------------------
```

DER GRAFIK MODUS

Wenn Sie Fenster 0 mit den Spalten auf 1/27 begrenzen, erfolgt eine Darstellung folgendermaßen:

```
---------------------------
Hier sehen Sie, wie Fenster
Nr. 0 den Zeilenumbruch be-
wirkt, wenn Sie dafür ande-
re Spalten festlegen.
---------------------------
```

Im Grafik-Modus sind alle Befehle benutzbar. Die Befehle PRINT, DISPLAY, INPUT und ACCEPT halten sich allerdings nicht an die gesetzten Fenstergrenzen wie GET und WRITE. Die Basic-Befehle arbeiten wie bisher im 28-Spalten-Bereich, sodaß Sie diese ohne Windowdefinition weiterhin normal benutzen können.

Nur im Grafik-Modus können Sie ein Basic-Programm erstellen. Der Basic-Interpreter ist nicht darauf ausgelegt, auch im Text-Modus zu arbeiten, welchen Sie mit dieser Systemerweiterung ansprechen können.

Sie sollten ein Programm nur an Stellen unterbrechen oder beenden, an denen Sie sich im Grafik-Modus befinden. Eine Unterbrechung, wenn Sie sich nicht im Grafik-Modus befinden, kann unliebsame Veränderungen des VDP-RAM nach sich ziehen, wodurch ein Aus- und Wiedereinschalten des Computers erforderlich wird.

DER GRAFIK MODUS

Im Grafik-Modus stehen unterschiedliche Befehle zur Farbgebung zur Verfügung. Benutzt werden können außer den Ihnen vom Basic her bekannten auch die der Systemerweiterung. Die Farbbefehle der Systemerweiterung lassen allerdings immer nur eine Vorder- und eine Hintergrundfarbe zu.

Torpedo-Basic wechselt nach dem Laden die Bildschirmfarbe auf grün/schwarz, was angenehmer für das Auge ist als die in den Computer eingebaute standartfarbe. Diese Farbgebung ist interruptgesteuert und wird immer wieder eingeschaltet, auch dann, wenn Sie im Programm die Farben mit SCREEN oder COLOR ändern. Bevor Sie die Farben ändern können, muß mit USECOL der Interrupt abgeschaltet werden, da dieser keine Definitionen mit COLOR oder SCREEN zulässt. Die neuen Farben bleiben auch bei einem Programmabbruch im Grafik-Modus erhalten.

DER TEXT MODUS

Der TEXT-Modus des TI-99/4a ist normalerweise vom Basic her nicht zugänglich. Gerade aber für Verwaltungsprogramme hat es sich als sehr nützlich erwiesen, 40 und nicht nur 32 Zeichen pro Zeile zur Verfügung zu haben. Der Editor/Assembler und der TI-Writer (TEXAS INSTRUMENTS SOLID STATE CARDRIDGE(R)) arbeiten in diesem Modus. Es ist darin immer eine Farbe für Character und eine Farbe für den Bildschirm möglich. Sprites können in diesem Modus nicht verwendet werden. Ebenso sind einige andere Befehle im Text-Modus nicht oder nur bedingt nutzbar, da sie als Basic-Befehle auf den Grafik-Modus ausgelegt sind. Die Befehle dieser Systemerweiterung, TORPEDO BASIC, arbeiten jedoch fast alle sowohl im Grafik- als auch im Text-Modus.

Für die Benutzung der Befehle gelten im Prinzip dieselben Regeln wie im Grafik-Modus. Für die Angabe einer Spaltenposition sind in allen dafür vorgesehenen Befehlen jetzt Werte zwischen 1 und 40 möglich. Im Text-Modus wird der gesamte 960 Byte große Buffer genutzt, da auch der Bildschirm 960 Positionen aufweist.

Farben sollten im Text-Modus nur mit TABLE und USECOL festgelegt werden. COLOR und SCREEN reichen nicht aus, um die Farben dieses Modus zu ändern. Sie finden dazu im Handbuch zum EDITOR/ASSEMBLER sowie im ASSEMBLER KURS III (erhältlich über unseren Vertrieb) mehr Informationen.

Nachfolgend nun eine Liste der Basic/Extended Basic Befehle, welche im Text-Modus nicht oder nicht richtig funktionieren. Die Liste erhebt keinen Anspruch auf Vollständigkeit. Zur Benutzung des Text-Modus sollten hauptsächlich TORPEDO BASIC Befehle verwendet werden.

DER TEXT MODUS

Folgende Befehle arbeiten fehlerhaft:

HCHAR, VCHAR, GCHAR
INPUT,PRINT
DISPLAY, ACCEPT

Folgende Befehle arbeiten gar nicht:

SPRITE
und alle anderen Sprite-Befehle des X-Basic,
wie COINC, DELSPRITE, MOTION, PATTERN, etc.

Alle Direktbefehle sind nicht benutzbar (s.u.)

Ausserdem: COPY und TORPED von Torpedo-Basic.

Folgende Befehle führen einen Systemabsturz herbei:

STOP
END
unmarkiertes Programmende

DAS VDP-RAM

Der TI-99/4a Homecomputer verfügt über einen speziellen Speicherbereich, welcher zur Bildschirm-, Farb-, Sprite- und Charactermusterdarstellung verwendet wird. Je nach Modus wird dieser Bereich unterschiedlich genutzt. Mit den Befehlen VDPEEK und VDPOKE. (im Editor/Assembler PEEKV und POKEV) haben Sie nun die Möglichkeit, direkt auf diesen Bereich zuzugreifen.

Im Grafik-Modus ist das VDP-Ram wie folgt definiert:

>0000 - >02FF Bildschirmdarstellungstabelle

>0300 - >037F Sprite-Attributen-Liste

>0380 - >03FF Farbtabelle und Freiraum

>0400 - >077F Spritemuster-Beschreibungstabelle

>0780 - >07FF Sprite-Bewegungs-Tabelle

>0800 - >09FF Charactermuster-beschreibungstabelle

>1000 - >37D6 Freier Platz für PABs und Buffer

>37D7 - >3FFF Reservierte Blocks (Disketten-DSR)

Mehr über diesen Speicherbereich finden Sie im Handbuch zum EDITOR ASSEMBLER und im ASSEMBLER KURS III (erhältlich durch unseren Vertrieb).

DAS VDP-RAM

Im Text-Modus ist die Benutzung etwas anders. Im Vergleich zum Grafik-Modus wird der Bereich folgendermaßen benutzt (eine Farbtabelle ist nicht erforderlich, da nur eine Vorder- und eine Hintergrundfarbe existiert, die in einem VDP-Register* festgelegt werden):

>0000 - >03BF Bildschirmdarstellungstabelle

>03C0 - >037F unbenutzt

>0380 - >03FF unbenutzt

>0400 - >077F unbenutzt

>0780 - >07FF unbenutzt

>0800 - >09FF Charactermuster-beschreibungstabelle

>1000 - >37D6 Freier Platz für PABs und Buffer

>37D7 - >3FFF Reservierte Blocks (Disketten-DSR)

Beachten Sie die besonderen Hinweise im Zusammenhang mit dem PEEKV und POKEV-Befehl bei der Veränderung des VDP-Ram

---

*) VDP-REGISTER: Besonderer Registerbereich, indem die Startadressen der Tabellen, die Modi, Bildschirmfarben und andere Besonderheiten festgelegt werden. Es ist nicht möglich, diese Register auszulesen. Es ist auch nicht möglich, diese Register ohne besondere Software zu verändern. Einige Basic- und TORPEDO BASIC Befehle greifen jedoch auf diese Register zu.

VERWANDTE BEFEHLE

Die Befehle der TORPEDO BASIC Systemerweiterung lassen sich in 8 Gruppen einteilen. Diese Befehle sind miteinander 'verwandt'.

Zeilenorientierte Befehle:

CLEAN, GET, GETSTR, WRITE

Windoworientierte Befehle:

CLTEXT, SCROLL, SEARCH, TABLE, WINDOW

Window/Buffer-Kommunikationsbefehle:

HIDE, SWAP, TAKE

Basic/Buffer-Kommunikationsbefehle:

INSTR, OUTSTR

Farb-Veränderung

USECOL (,TABLE)

VERWANDTE BEFEHLE

Peripherie-Zugriffsbefehle

COPY, DIR

VDP-Zugriffsbefehle

VDPEEK, VDPOKE

Sonstige Befehle

BRANCH, CHRSET, MODE, QUIT, TORPED

Mit Ausnahme von COPY sind alle Befehle sowohl im Grafik- als auch im Text-Modus verfügbar.

VDPEEK und VDPOKE sind nur mit EXTENDED BASIC verfügbar. Mit dem Editor/Assembler Modul benutzen Sie bitte unter Beachtung der Hinweise im E/A-Handbuch die Befehle PEEKV und POKEV.

ASCII-CODE TABELLE

30 Cursor	63 ?	96 `
31	64 @	97 a
32 Space	65 A	98 b
33 !	66 B	99 c
34 "	67 C	100 d
35 #	68 D	101 e
36 $	69 E	102 f
37 %	70 F	103 g
38 &	71 G	104 h
39 '	72 H	105 i
40 (	73 I	106 j
41)	74 J	107 k
42 *	75 K	108 l
43 +	76 L	109 m
44 ,	77 M	110 n
45 -	78 N	111 o
46 .	79 O	112 p
47 /	80 P	113 q
48 0	81 Q	114 r
49 1	82 R	115 s
50 2	83 S	116 t
51 3	84 T	117 u
52 4	85 U	118 v
53 5	86 V	119 w
54 6	87 W	120 x
55 7	88 X	121 y
56 8	89 Y	122 z
57 9	90 Z	123 {
58 :	91 [	124 |
59 ;	92 \	125 }
60 <	93]	126 ~
61 =	94 ^	127 DEL
62 >	95 _	

STEUERCODE-TABELLE

1	AID
2	CLEAR
3	DELETE
4	INSERT
5	QUIT
6	REDO
7	ERASE
8	LINKSSCHRITT
9	RECHTSSCHRITT
10	CURSOR DOWN
11	CURSOR UP
12	PROC'D
13	ENTER
14	BEGIN
15	BACK

129-159 Für TORPEDO BASIC ohne Bedeutung.

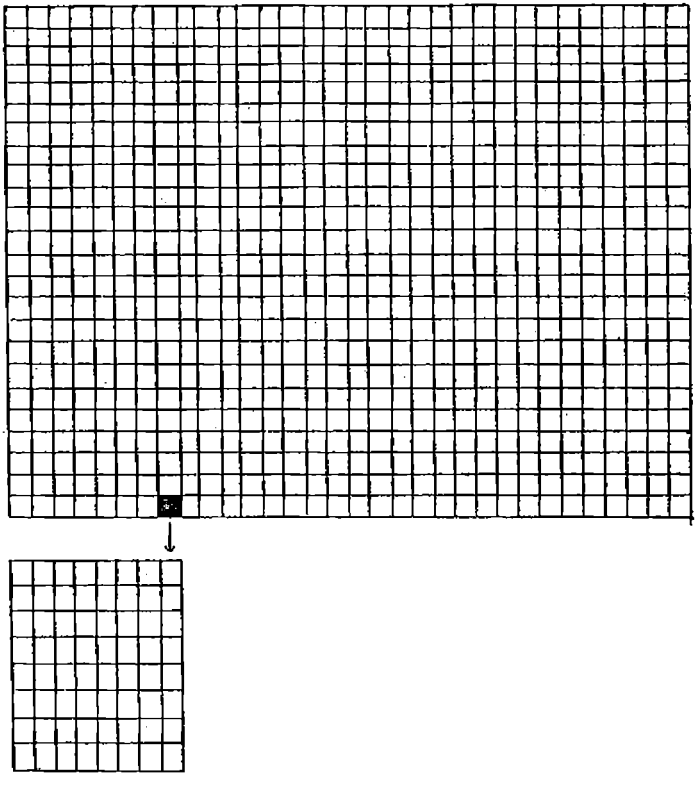
Die Cursorfunktionen zum TABLE-Befehl finden Sie am Anfang des Buches unmittelbar vor den Befehlserläuterungen.

FARB-TABELLE

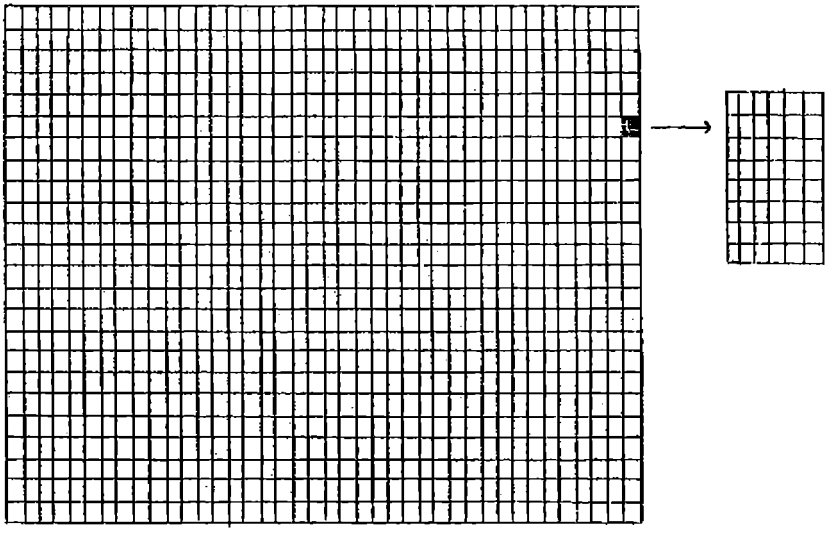
1	transparent
2	schwarz
3	mittelgrün
4	hellgrün
5	dunkelblau
6	hellblau
7	dunkelrot
8	kornblumenblau
9	mittelrot
10	hellrot
11	dunkelgelb
12	hellgelb
13	dunkelgrün
14	magentarot
15	grau
16	weiss

In ASSEMBLER startet die Farb-Codierung mit '0' und endet mit '>F' (15). Beachten Sie dies, wenn Sie mit dem LOAD-Befehl den Programmablauf von TORPEDO BASIC manipulieren wollen.

BILDSCHIRMAUFTEILUNG IM GRAFIK-MODUS



BILDSCHIRMAUFTEILUNG IM TEXT-MODUS



Im Text-Modus werden von jeder Character-Darstellung die letzten beiden Bit-Eingänge ignoriert. Deshalb kommt der Bildschirm mit 192x256 Pixeln aus, obwohl 40 Zeichen pro Zeile dargestellt werden. Jeder Character wird als 8x6 Pixel großes Muster dargestellt.

SPEICHERBENUTZUNG DURCH TORPEDO BASIC

Voraussetzung für die Gültigkeit ist es, daß das Maschinenprogramm an Adresse >A000 der Speichererweiterung geladen wird. Bei anderen Startadressen sind die Bereiche entsprechend verschoben.

>A000 - >A3BF Bufferbereich

>A3C0 - >A3FF Windowtabelle des Grafik-Modus

>A400 - >A43F Windowtabelle des Text-Modus

>A440 - >A443 Window-Benutzer-Tabelle

>A444 - >AF9F Interrupt-Charactersatz-Tabelle

>A4A0 - >A5FF wird intern benutzt (Reserviert)

>A600 ... Eigener Registerbereich
Initialisierungskonstante
Daten
Programmbereich

Der benutzte Bereich ab >A600 ist je nach verwendeter Konfiguration unterschiedlich lang und wird auch unterschiedlich benutzt.

neu

Assembler Kurs III

von Hagera

D

Computer Kontakt 10/85

Assembler Kurs II

Nach wie vor besteht bei TI-Usern ein großes Interesse an Büchern, die sich mit der Assemblerprogrammierung befassen. Allerdings gab es bislang noch kein Buch, das sich an absolute Neulinge wendet. Hier bietet nun die Firma Rausch & Haub einen Assemblerkurs an, der dem Anfänger einen Einstieg in das schwierige Gebiet der Assemblerprogrammierung zeigt.

Im ersten Kapitel wird zunächst der Umgang mit den Möglichkeiten des Editors und des Assemblers beschrieben und zwar mit Beispielen, so daß man direkt am Bildschirm alles nachvollziehen kann. Schritt für Schritt werden nun wieder mit Beispielen die einzelnen Befehle und Utility-Unterprogramme erläutert. Am Ende eines jeden Kapitels werden dann Aufgaben gestellt, an denen man seine neu erworbenen Kenntnisse erproben kann. Die Lösungen der meisten Aufgaben sind außerdem auf der mitgelieferten Diskette enthalten.

Weiter geht es im Kurs mit Bildschirmausgaben und der Programmierung von Tönen. Als krönender Abschluß wird dann die Programmierung eines Spiels erläutert, das sich ebenfalls auf der Diskette befindet. Im Anhang sind dann noch die Beschreibung aller verwendeten Assembler-Befehle sowie Tabellen mit wichtigen Systemadressen aufgeführt.

Dieser über 300 Seiten umfassende Assemblerkurs führt Anfänger sehr gut in die Materie ein, ohne dabei stark in die doch teilweise recht trockene Theorie abzuschweifen. Daher wird derjenige, der sich durch diesen Kurs durcharbeitet, zwar kein perfekter Programmierer sein, doch ist die Basis für eine vertiefende Beschäftigung auf diesem Gebiet gelegt. Mit einem Preis von 80.– DM (inkl. Diskette) ist dieses Buch sehr empfehlenswert.

Bezugsquelle:
Räusch & Haub
3300 Bonn 3
Postfach 32 03 13

Info 4

ASSEMBLER KURS I, II und III

WAS IST DAS EIGENTLICH?

Seit fast zwei Jahren wird der TI-99/4a nicht mehr produziert. Trotzdem hält sich dieses Gerät angesichts der auf den Markt drängenden Superrechner mit hoher Leistungsfähigkeit und enormen Speicherplatz beständig - nie gab es mehr Software, Peripherie und Zubehör als im Augenblick. Doch wer hätte sich daß träumen lassen?

Trotz dieser 'Softwareschwämme' lässt es sich nicht leugnen, daß auch alle Importlager in naher Zukunft erschöpft sein werden, wenn sich auch immer wieder neue Quellen, vor allem im südeuropäischen Ausland, auftun. Was soll man also mit seinem liebgewonnenem Computer machen? Wegwerfen? - undenkbar! Verkaufen? - Bringt nicht viel ein!

Die Alternative: Weiter Einsteigen! Die Devise "Selbst ist der Mann (oder die Frau)" gilt nirgendwo so wie beim Programmieren von Computern. Die alten Ausreden "Kann ich nicht", "Verstehe ich nicht" oder "Blackout" ziehen nicht mehr! Computern macht Spaß; auch oder gerade mit einem 'alten Eisen' wie dem TI-99/4a.

Gemeint ist das selbständige Programmieren! Natürlich sind wirklich gute Programme in Basic, auch in Extended basic, kaum denkbar. Besser ist da schon Pascal, leider aber beim TI-99/4a zu wenig verbreitet. Aber wozu gibt es schließlich die Maschinensprache und einen hervorragenden Assembler von Texas Instruments, der seinesgleichen sucht?

Sicher haben Sie schon versucht, mit dem Assembler etwas aus Ihrem Rechner herauszubekommen. Vielleicht ist es Ihnen sogar gelungen, obwohl literarische Unterstützungen bisher fehlten. Doch dann kam (in aller Bescheidenheit) ein neues Buch auf den Markt: Der ASSEMBLER KURS II von Rausch & Haub; mit allem ausgerüstet, was der Anfänger für den Einstieg braucht. Nicht Theorie, sondern Praxis ist Trumpf - und die Kritiken belegen die gelungene Einführung in die Assembler-Sprache.

ASSEMBLER KURS I, II und III

KRITIKEN ZUM ASSEMBLER KURS II:

COMPUTER KONTAKT, die bekannte Zeitschrift für alle Anwender, schieb in Ausgabe 10/85: Der Kurs führt Anfänger sehr gut in die Materie ein, ohne in die teilweise doch recht trockene Theorie abzuschweifen...

USER-STIMMEN belegen den Erfolg: Sehr Empfehlenswert, Für Anfänger sehr gut geeignet, hervorragend, sehr gut... dies waren nur einige der 'Lobeshymnen'. Nicht zuletzt wurde immer wieder gewünscht, diesen Kurs doch zu erweitern, was wir nun endlich auch getan haben.

Immer wieder wird in Computerzeischriften und Club-Infos positives über den Kurs II berichtet - und das von Anwendern, die es ja schließlich wissen müssen.

Nicht ganz unschuldig an dem positiven Echo ist sicher auch die beiliegende Diskette. Angefangen bei einer einfachen Routine für die Bildschirmausgabe bis hin zum kompletten Spiel ist alles enthalten. Damit ist es möglich, alle erlernten Schritte sofort am Computer nachzuvollziehen. Wir beginnen auch nicht mit grauer Theorie - Ergebnisse sind vielmehr sofort am Bildschirm sichtbar. Zu jedem Kapitel gibt es kleine Übungsaufgaben, die eine Vertiefung des Erlernten Stoffes ohne Probleme ermöglichen, und natürlich sind auch Musterlösungen vorhanden.

Wir haben uns dem Ruf nach einer Fortsetzung gebeugt und können Ihnen nun ASSEMBLER KURS III vorstellen. Auf den nächsten beiden Seiten finden Sie eine detaillierte Beschreibung des Inhaltes beider Bücher.

Oft erreichen uns Anfragen, ob es nicht einen KURS I gäbe. Dies ist nicht der Fall. Kurs I war die 'Ur-Fassung' des heutigen Kurses II mit nur 228 Seiten, der aber nicht mehr angeboten wird. Eventuell bringen wir aber einen neuen Kurs I mit Tabellen und allgemeinen Informationen heraus; allerdings frühestens Mitte nächsten Jahres.

ASSEMBLER KURS I, II und III

ASSEMBLER KURS II FÜR EINSTEIGER

Umfang:

- 344 Seiten und eine Programmdiskette.

Inhalt:

- Umgang mit dem Editor-Assembler
- Bildschirmausgabe
- Zeichen vom Bildschirm lesen
- Definieren von Sonderzeichen
- Farben Definition
- Abfrage der Tastatur
- Töne programmieren
- Mathematik und Programmtechnik
- Schleifen, Verzweigungen und Sprünge
- Verschieben und Vergleichen
- Arithmetische und logische Befehle
- Problemstellungen
- Software: Spiele-Listing
- Übungsaufgaben und Lösungen
- Assembler-Mnemonics (Befehle)
- Erläuterungen fremder Begriffe
- Anhang mit Liste der Diskettenfiles

ASSEMBLER KURS I, II und III

ASSEMBLER KURS III FÜR FORTGESCHRITTENE

Umfang:

348 Seiten weiterführende Informationen

Inhalt:

- Logische Befehle und Verknüpfungen
- Bitverschiebungen in Registern
- Sprites in Maschinensprache
- Der Grafik-Modus des TI-99/4a
- Der Text-Modus des TI-99/4a
- Der Multicolor-Modus des TI-99/4a
- Der Bit-Map-Modus des TI-99/4a
- Basic-Zugriff auf Maschinenprogramme
- Parameterübergabe vom Basic an Assembler-Maschinenprogramme, String und numerisch
- MC-Programmverkettung mit Extended Basic
- Erstellen und Möglichkeiten des Speicher-Auszuges
- Die CRU-Instruktionen
- Übungsaufgaben und Lösungen
- Anwendungsvorschläge
- Auszug aus HAGERA(R) 'MODE CONTROL
- VDPEEK und VDPOKE von HAGERA(R) 'TORPEDO BASIC'
- Tabellen und Listen
- Übersichten
- Hinweise auf empfehlenswerte Programme

Dieser KURS III baut voll auf dem Inhalt von KURS II auf. Sie können selbstverständlich aber auch beide Bücher getrennt verwenden, da sie ebensogut als Nachschlagewerk wie als Lehrbuch benutzbar sind.

ASSEMBLER KURS I, II und III

ERFORDERLICHE KONFIGURATION

Für die Benutzung der Programmdiskette aus Kurs II ist folgende Konfiguration erforderlich:

TI-99/4a Grundausstattung;
Editor/Assembler Solid State Modul (TI);
BSCSUP-Utilities von E/A-Diskette, Part A;
Speichererweiterung (Cardridge oder Extern);
Diskettenlaufwerk.

Die im Kurs III angegebenen Disketten-Empfehlungen sind in der Regel sowohl auf obiger als auch auf der nachfolgend genannten Konfiguration lauffähig:

TI-99/4a Grundausstattung;
Extended Basic Solid State Modul (TI) oder
deutscher Nachbau;
Speichererweiterung (Cardridge oder Extern);
Diskettenlaufwerk oder Cassettenrecorder.

Zu den Diskettenempfehlungen aus den Büchern sind ebenfalls kostenlose Informationsbroschüren und Prospekte erhältlich, die wir Ihnen auf Wunsch gerne zusenden.

BEZUGSQUELLE:

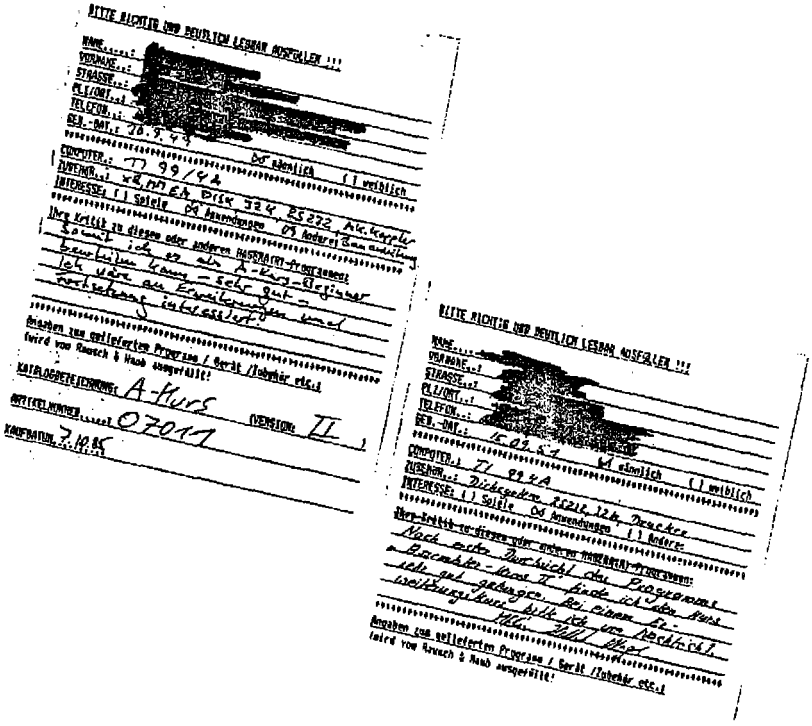
RAUSCH & HAUB
Vertriebsgesellschaft dbR.
Postfach 320313

5300 BONN 3

ASSEMBLER KURS I, II und III

UNSERE KRITIKEN SIND NICHT GELOGEN!

Hier einige Kritiken, die uns auf den HAGERA(R)-Servicekarten mitgeteilt wurden (Namen und Anschriften aus Gründen des Datenschutzes geschwärzt!).



ASSEMBLER KURS I, II und III

LIEFERUMFANG UND PREISE

ASSEMBLER KURS II:

1 gebundenes Buch, 344 Seiten Assemblersprachen-Einführung;
1 Diskette mit Musterlösungen und Spielprogramm im Quellen- und Objekt-Code

ASSEMBLER KURS III:

1 gebundenes Buch, 348 Seiten mit weiterführenden Informationen zum Editor/Assembler des TI-99/4a und verschiedenen interessanten Diskettenempfehlungen für Anwender.

Lieferbar sind die folgenden Ausstattungen:

Art.-Nr. 07011 ASSEMBLER KURS II Einführung in Assembler
Diskette und Handbuch komplett DM 79.90

Art.-Nr. 07021 ASSEMBLER KURS III für 'Fortgeschrittene'
Handbuch mit 348 Seiten DM 79.90

Diskettenempfehlungen: (Preise auf Anfrage)

TORPEDO BASIC	Betriebssystem-Erweiterung
MODE CONTROL	Alle 4 Modi des TI-99/4a fest im Griff
TEXT UTILITIES	Zugriff auf den Text Modus
PARTISAN VILLAGE	Das Spiel zum KURS II

ACHTUNG: CLUBS, SAMMELBESTELLER UND WIEDERVERKÄUFER

Für CLUBS gibt es ab sofort ermäßigte CLUB-Preise, wenn mehr als ein Exemplar zur gleichen Zeit an die gleiche Adresse angefordert wird. Verlangen Sie die Aktuelle Sonderpreisliste für Sammelbesteller und Clubs. Gewerbsmäßige Händler fordern bitte unsere Händlerpreisliste an!

Thanks to 99'er:
Karl Rüttger
for the scan of this document.

- Scanned and Reworked by:
TI99 Italian User Club in the year 2014.
(info@ti99iuc.it)

Hagera®
Rausch & Haub
Vertriebsgesellschaft dbR
Postfach 32 03 13
5300 BONN 3