M. Heilfort, R. Tannert

Turbo-FORTH

Einführung

An der Sektion Physik der MLU Halle entstand ein FORTH-System für Kleincomputer KC 87 und Äquivalenztypen, das sowohl für die Experimentautomatisierung als auch für die Informatikausbildung von Studenten dienen soll.

Dabei war zu berücksichtigen, daß die potentiellen Nutzer Nicht-Informatiker sind. Folglich mußte auf eine möglichst einfache Programmierumgebung Wert gelegt werden.
Besondere Aufmerksamkeit galt dem Zusammenspiel von Testlauf und Editieren des Quelltextes. Hier wirkte Turbo-Pascal als Vorbild.

Neben den FORTH-typischen Eigenschaften wie **Umgekehrt Polnische Notation, maschinennahe Befehle zur Steuerung der Peripherie, Integerarithmetik** sind folgende Hauptparameter für 'Turbo-FORTH' kennzeichnend:

*** Dateikonzept**
Abweichend von anderen FORTH-Versionen für Kleincomputer wurde ein Dateikonzept verwirklicht.
Der Quelltext wird dabei im Computer in einer automatisch angelegten RAM-Disk verwaltet. Hier lassen sich in maximal sechzehn verschiedenen Files Programme, Systemerweiterungen, Hilfs- und Testroutinen speichern.
Die Verwaltung der Screens übernimmt dabei 'Turbo-FORTH', so daß sich der Nutzer nicht merken muß, welcher Screen zu welchem File gehört.

*** gut handhabbare Programmierumgebung**
Ein Editor, der ähnlich Wordstar arbeitet, Klartextfehlermeldungen und ein schneller Compilerlauf erleichtern das Erstellen und Testen des Quelltextes wesentlich.
Das Umsortieren der Screens und ein beliebiger Transport von Screens zwischen mehreren Files sind durch das Kommando SMOVE möglich.

*** frei definierbare Funktionstasten**
Das Testen des Quelltextes wird durch eine weitere Besonderheit von 'Turbo-FORTH' unterstützt: die 'RUN'- und die 'ESC'-Taste sind mit Folgen aus 8 bzw. 24 Zeichen frei belegbar.

*** geringer Speicherbedarf**
Turbo-FORTH belegt einschließlich Editor und Klartextfehlermeldungen nur 10 kByte. Da Turbo-FORTH auch als EPROM-Version existiert, können bei Verwendung von zwei RAM-Modulen bis zu 57 Screens (28,5 KByte) Quelltext im ganzen bearbeitet werden.

*** schneller Programmlauf**
Über die Hälfte aller Befehle sind Primitives, d.h. in Maschinencode geschrieben. Dadurch kommt die Abarbeitungszeit von Turbo-FORTH in die Größenordnung von Assemblerprogrammen.
Weiterhin besteht die Möglichkeit, für extrem zeitkritische Prozesse direkt Maschinencode einzubinden.

*** Dialogbefehle auf hohem Niveau**
Allein für den Aufbau von Menü- und Dialogarbeit über Bildschirm und Tastatur stehen mehr als zwanzig zum Teil komplexe Befehle zur Verfügung.

*** allgemeine Massenspeicherbefehle**
Fuenf leistungsfähige Befehle ermöglichen es dem Nutzer, beliebige eigene Massenspeicherroutinen z.B. zur Meßwertspeicherung zu realisieren.

Kurzbedienungsanleitung Turbo-FORTH

Nach Eingabe von **TF** im Betriebssystem des Computers ist das Programm zu laden. Es startet sich selbst und führt eine automatische Einteilung des zur Verfügung stehenden Speicherplatzes in freies Wörterbuch, Systemzellen und RAM-Disk durch. Maschinenprogramme (z.B. Erweiterungen des Betriebssystemes), die am oberen Ende des Hauptspeichers stehen, werden nicht zerstört, wenn sie sich durch Verändern des 'Ramtop' (Speicherzelle 0036H/37H) geschützt haben.

Turbo-FORTH meldet sich mit einer Anzeige der noch verfügbaren Wörterbuchgröße und der freien Screens. Diese Anzeige wird bei jedem Warmstart von Turbo-FORTH (Befehl **WARM**) auf dem Bildschirm dargestellt.

Das Programm befindet sich im Kommandomodus. Jede Eingabe wird sofort nach Betätigen der 'ENTER'-Taste ausgeführt.
Falsche Eingaben sind nur dadurch korrigierbar, daß man den Cursor mit Hilfe der Backspace-Taste '<--' bis vor den Fehler setzt, und von da ab eine neue Eingabe vornimmt.

Turbo-FORTH kann durch den Befehl **BYE** verlassen werden. Eine Rückkehr ist aus dem Betriebssystem mit **WFORTH** fehlerfrei möglich, solange das Programm nicht zerstört wurde. Das Wörterbuch und die Screens bleiben erhalten.

Zur effektiven Arbeit mit Quelltext verfügt Turbo-FORTH über eine RAM-Disk, d.h. einen Hauptspeicherbereich des Computers, der wie eine Diskette verwaltet wird. Diesen Bereich legt Turbo-FORTH bei Neustart automatisch an.
In der RAM-Disk lassen sich maximal 16 Files speichern, die durch verschiedene Namen aus höchstens 8 Zeichen gekennzeichnet sind.
Die Namen der vorhandenen Files und die Anzahl der Screens pro File werden durch den Befehl **FILES** angezeigt.

Jedes File muß eröffnet werden, bevor man es in der RAM-Disk speichern kann. Dazu existieren prinzipiell zwei Möglichkeiten:

- Ein neues File ist durch **USE filename** (z.B. USE TEST) anzulegen. Es wird gleichzeitig eröffnet und enthält zunächst keine Screens.
- Um ein bereits bearbeitetes File von Kassette zu lesen, ist der Befehl **GET filename** einzugeben. Dabei wird das File automatisch vor dem Lesevorgang eröffnet.
  Existiert das File bereits, so werden die gelesenen Screens an das Ende angefügt.

Das Schließen eines Files geschieht mit dem Befehl **CLOSE filename.** Die zum File gehörenden Screens werden gelöscht.
In der Regel möchte man die Screens vorher auf Kassette speichern. Dazu dient der Befehl **SAVE filename.** Er kann beliebig oft angewandt werden.

Verschiedene Befehle (z.B. **LOAD**, **LIST**, **INDEX**, **EDIT**, **NAME**) beziehen sich stets auf das aktuelle File. Wurde mit CLOSE das aktuelle File gelöscht, muß erst durch **USE filename** ein anderes File zum aktuellen erklärt werden, bevor diese Befehle benutzt werden können.
Mit **NAME filename** kann der Name des aktuellen Files geändert werden. Generell gilt, daß das zuletzt eröffnete oder bearbeitete File aktuelles ist.

Bevor man die in FORTH geschriebenen Programme abarbeiten kann, müssen sie

mit **LOAD** übersetzt werden. Der Befehl wirkt auf das aktuelle File und erwartet die Screennummer des ersten zu übersetzenden Screens auf dem Datenstapel. Es werden automatisch auch alle Folgescreens compiliert.

Kenndaten Turbo-FORTH

```
Speichergröße:      10  KByte für Turbo-FORTH
         mindestens  4  KByte für Wörterbuch, Systemzellen und RAM-Disk

Grundgerät           : ca.  2200  {5500}  Bytes Wörterbuch,   4 {17} Screens
- " - + 1 RAM-Modul  : ca.  8300 {11650}      -- " --     ,  24 {37} Screens
- " - + 2 RAM-Module : ca. 14400 {17750}      -- " --     ,  44 {57} Screens

(Die Zahlenangaben in geschweiften Klammern gelten für die EPROM-Versionen.)

Compilerlaufzeit              Version 2.3

normal  gefüllter Screen :     ca. 0,2 s
maximal    --  "  --     :     ca. 0,5 s
```

Es stehen 275 Befehle zur Verfügung, darunter 132 Primitives, 23 Variablen und 14 Konstanten.

Um Fehler zu vermeiden, sollte auch der versierte Programmierer zur Erweiterung des Wörterbuches nur die in Turbo-FORTH vorhandenen Definitionsbefehle benutzen. Sie basieren auf **CREATE** und legen den Header richtig an.

Folgende Befehle stellen für die Arbeit mit Turbo-FORTH einen Mindestwortschatz dar:

```
GET       SAVE      CLOSE     USE       NAME      FILES
EDIT      SMOVE     LOAD      LIST      INDEX
:         ;         FORGET    (         WORDS
DUP       SWAP      DROP      ROT       OVER      NIP
+         -         *         /         MOD
/MOD      */        */MOD     MAX       MIN       ABS       NEGATE
2+        2-        2*        2/        1+        1-
AND       OR        XOR
DECIMAL   HEX       _.         .R        U.        .HEX
KEY?      KEY       EMIT      CR        ."        IN-NUM    J?N
PAGE      SCROFF    INK       PAPER     BORDER
SPACE     SPACES    LOC
DO        LOOP      +LOOP     LEAVE     I         J         K
BEGIN     UNTIL     WHILE     REPEAT    AGAIN     IF        ELSE      ENDIF
<         >         =         0<        0>        0=        U<        ?DUP
CASE      ENDCASE   NOOP
VARIABLE  @         !         ?         +!
COLD      WARM      ABORT     QUIT      RE-INIT   BYE
```

Da FORTH auch maschinennahe Befehle enthält, sollten nur Befehle aufgerufen werden, deren Wirkung man kennt. Andernfalls kann ein Systemabsturz die Folge sein, dessen Ursache in der Regel nur schwer erkennbar ist.

Generell gilt: ein neu erstelltes oder geändertes Programm **erst auf Kassette speichern, dann compilieren und testen !**

Der Screeneditor

Um den in den Files abgelegten Quelltext bearbeiten zu können, wurde in Turbo-FORTH ein leistungsfähiger Screeneditor implementiert. Er ist für das aktuelle File durch **EDIT**, für jedes andere File mit **USE filename EDIT** aufzurufen. Dabei wird automatisch der zuletzt bearbeitete oder der erste Screen des Files aufgerufen. Ist das File leer, so legt der Editor einen Leerscreen mit der Nummer 1 an und ruft diesen auf.
Der Screen gelangt zum Bearbeiten in den Screenpuffer (ab FIRST).

Die Benutzung des Editors erfolgt ähnlich einem Textverarbeitungssystem über Steuerbefehle (mit 'CONTR'-Taste aufrufbar; zur Benutzung dieser Taste siehe Systemhandbuch des Computers). Die möglichen Funktionen sind in einer Übersicht auf der letzten Seite zusammengestellt worden.

Dabei sollten Sie beachten:

- Der Editor verweigert das Einfügen von Zeichen oder Zeilen, wenn dadurch andere Zeichen aus dem Screen geschoben werden.
- Das Einfügen und Löschen von Zeichen erfolgt normalerweise zeilenweise. Sie können sich mit dem Kommando **<CONTR> E** an beliebiger Stelle eine Marke setzen, die bewirkt, daß nun alle Zeichen zwischen dem Kursor und dieser Marke verschoben werden, falls der Kursor in einer Zeile oberhalb der Marke oder vor ihr steht. Wenn sich ein Zeichen an der Markenposition befindet, verweigert der Editor das Einfügen von Zeichen.
  Die Marke wird gelöscht bei Verlassen des Screens oder bei erneuter Eingabe von <CONTR> E, wenn sich der Cursor an der Markenposition befindet.
- Falls Sie einzelne Zeilen duplizieren möchten, können Sie die betreffenden Zeilen mit **<CONTR> A** auf einen Zeilenstapel legen und von dort mit **<CONTR> O** (wirkt überschreibend !) wieder holen.
  Die Größe des Zeilenstapels hängt vom freien Wörterbuch ab. Reicht der Platz nicht mehr aus, wird <CONTR> A verweigert. Der Zeilenstapel bleibt erhalten, solange das Wörterbuch nicht verändert wird. Dadurch ist auch ein Transport von Zeilen von einem File in ein anderes möglich.
- Haben Sie aus Versehen Zeichen gelöscht, können Sie mit **<CONTR> R** den alten Screeninhalt rückladen, der bei Aufruf des Screens vorhanden war.
- **<CONTR> T** (**<COLOR>**) ermöglicht das Suchen einer Zeichenfolge.
  Es kann mit **<CONTR> W** fortgesetzt werden.
- Die Funktionen **<CONTR> V** (**<LIST>**), **<CONTR> N** (**<STOP>**), **<CONTR> Q** (**<RUN>**) und **<ESC>** beenden die Arbeit im angezeigten Screen, der dabei in die RAM-Disk übertragen wird.

Tritt während der Compilation mit **LOAD** ein Fehler auf, so steht beim nächsten Aufruf des Editors der Kursor unmittelbar hinter der Fehlerstelle.

Nach Starten von Turbo-FORTH ist in der Kommandoebene die 'RUN'-Taste mit dem Befehl **EDIT** belegt und als weitere Testhilfe die 'ESC'-Taste mit der Befehlsfolge: **EMPTY 1 LOAD .** (vgl. **RUN"** und **ESC"**)

Die Programmerstellung als Wechsel von Editieren und Testlauf des Compilers ist dadurch denkbar einfach:
Den Compilerlauf löst man durch Drücken von <ESC> aus.
Bei Fehler einfach <RUN> drücken und nach Korrektur zweimal <ESC>. (Verlassen des Editors, Säubern des Wörterbuches und Start des Compilers)
War der Compilerlauf fehlerfrei, gelangt man durch <RUN> (Befehl **EDIT**) in den zuletzt compilierten Screen.

Der Befehl SORT

Der Befehl SORT ermöglicht den Transport von Screens innerhalb eines Files bzw. zwischen zwei Files, wobei er die Zeiger im Vektorfeld (hinter LIMIT) umsetzt. Die zu verschiebenden Screens werden folglich nicht physisch transportiert, sondern nur neu zugeordnet.
Sämtliche Parameter werden abgefragt. Die Vorgaben kann man auch nur mit der 'ENTER'-Taste quittieren.
Bei Angabe der Schrittweite, die auch negativ sein kann, ist darauf zu achten, daß die neu zugewiesenen Screennummern innerhalb des gültigen Bereiches (1 bis 127) bleiben.
Tritt in SORT ein Fehler auf, bleibt der ursprüngliche Zustand erhalten.

Die Arbeit von SORT soll an einem Beispiel gezeigt werden. Die Eingaben des Nutzers sind dabei fett gedruckt, die Bildschirmausschriften normal.
Kommentare stehen rechts in geschweiften Klammern.

Sie haben von Kassette ein File UNTER.INC (3 Screens) mit häufig benötigten FORTH-Definitionen geladen (Damit ist UNTER.INC aktuelles File !) und wollen die Screens 1 und 2 dieses Files in das bereits existierende File PROGRAMM (Screens 1 bis 5) hinter den Screen 2 einfügen. Dazu sind zuerst die Screens 3 bis 5 des Files PROGRAMM in 5 bis 7 umzubenennen:

```
SORT <ENTER>
von File UNTER   .INC ? : PROGRAMM <ENTER>
ab Screen 1 ? : 3 <ENTER>
bis Screen 3 ? : 5 <ENTER>
nach File PROGRAMM.(F) ? : <ENTER>     { Vorgabe bestätigen                }
ab Screen 5 ? : <ENTER>
Schrittweite 1 ? : <ENTER>
ueberschreiben ? (J/N): J              { da Screen 5 schon existiert !     }
ok                                     {    alt: 1 2 3 4 5                 }
                                       {    neu: 1 2 5 6 7                 }
```

Jetzt werden die Screens aus UNTER.INC eingefügt:

```
SORT <ENTER>
von File UNTER   .INC ? : <ENTER>
ab Screen 1 ? : <ENTER>
bis Screen 1 ? : 2 <ENTER>
nach File UNTER   .INC ? : PROGRAMM <ENTER>
ab Screen 2 ? : 3 <ENTER>
Schrittweite 1 ? : <ENTER>
ueberschreiben ? (J/N): N              { da es jetzt im File PROGRAMM die  }
ok                                     { Screens 3 und 4 nicht mehr gibt ! }
```

Nun wollen Sie das neue File PROGRAMM unter dem Namen PROG1 sichern.
Dazu ist der Befehl NAME zu benutzen:

```
USE PROGRAMM   NAME PROG1 <ENTER>
ok
```

Zur Kontrolle kann der Befehl FILES aufgerufen werden.

```
FILES <ENTER>
* File PROG1    .(F)       7 Screen(s)                { aktuelles File }
  File UNTER    .INC       1 Screen(s)
ok
```

Jetzt **SAVE PROG1** eingeben, den Kassettenrecorder starten und danach die 'ENTER'-Taste drücken.

Befehlsbeschreibung Turbo-FORTH

Es werden alle Befehle von "Turbo-FORTH" angeführt und knapp erläutert.
Der Anhang enthält eine Übersicht der Verzweigungs- und Schleifenbefehle (Typ **VZW**). Ihre Beschreibung beschränkt sich auf das Laufzeitverhalten !

Die Befehle sind entsprechend des ASCII-Codes sortiert:

```
  ! " # $ % & ' ( ) * + , - . / 0 1 2 3 4 5 6 7 8 9 : ; < = > ?
@ A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
```

Verwendete Symbole:

```
c    - ASCII-Zeichen               (             0 ...           127 )
b    - Byte                        (             0 ...           255 )
n    - 16-Bit-Wort                 (        -32 768 ...        32 767 )
un   - 16-Bit-Wort vorzeichenlos   (             0 ...        65 535 )
dd   - 32-Bit-Zahl                 ( -2 147 483 648 ... 2 147 483 647 )
udd  - 32-Bit-Zahl vorzeichenlos   (             0 ... 4 294 967 295 )
vadr - Adresse einer Variablen
lfa  - Linkfeldadresse              |  cfa  - Codefeldadresse
nfa  - Namensfeldadresse            |  pfa  - Parameterfeldadresse
```

Ein 'name' in der Spalte (*) bedeutet, daß der Befehl wie folgt benutzt wird: befehl name
z.B.: OPEN TEST - eröffnet das File TEST
TICK EMIT - übergibt die cfa des Befehls EMIT.

Ein Bezeichner in der Spalte (#) gibt das Vokabular an, wenn der Befehl nicht im Vokabular FORTH steht.

(Stack vor -- Stack nach Befehlsausführung)
Jeweils rechts steht der Inhalt der obersten Stackzelle TOS.

Ein 'I' vor der Stackbelegung zeigt an, daß der Befehl während der Compilation sofort ausgeführt wird. (Immediate-Befehl)
Ein 'P' nach der Stackbelegung zeigt an, daß der Befehl ein Primitive ist.

Befehl	**Stackbelegung**	**(#)**	**(*)**	**Typ**
!	(n adr --) P			MEM

Speichert das Wort n auf die Adresse adr.

!CSP	(--)			DIA

Legt den momentanen Stand des Datenstacks in der Uservariablen CSP ab.

#	(udd1 -- udd2)			CON

Erzeugt von udd1 das nächste Zeichen im Ausgabestring und übergibt den Quotienten udd2, der bei ganzzahliger Division von udd1 durch die aktuelle Zahlenbasis entsteht.

#>	(udd -- adr anzahl)			CON

Beendet die Konvertierung einer Zahl in einen Ziffernstring und übergibt die Adresse des ersten Zeichens und die Länge (passend zu TYPE).

#S (udd1 -- udd2) CON
Ruft den Befehl '#' solange auf, bis udd Null geworden ist.

' (-- cfa) name DIC
Übergibt die cfa des Befehls name. (vgl. ['])

(I (--) COM
Kommentar wird in Klammern eingeschlossen. Nach der öffnenden Klammer muß ein Leerzeichen stehen.

(') I (--) name COM
Compiliert die cfa des Befehles name als Literal. (Alias zu ['])

((I (--) P COM
Schaltet den Execute-Modus ein. (Gegenstück zu '))'; Alias zu [ )

(;CODE) (--) SYS
Überschreibt das Code-Feld des zuletzt definierten Befehles so, daß der im Anschluß an ;CODE stehende Maschinencode zur Ausführung gelangt.

(COMPILE) I (--) name COM
Erzwingt die Compilation des Immediate-Befehls name. (Alias zu [COMPILE])

(CONSOLE) (c --) I/O CIO
Gibt das Zeichen c nur an den Drucker aus.

(EMIT) (c --) I/O CIO
Gibt das Zeichen c an die Konsole aus und bei PRINTING ON auch an den Drucker.

(ERROR) (n --) DIA
Erzeugt eine Fehlermeldung entsprechend der Fehlernummer n und führt anschließend ABORT aus.

(KEY) (-- c) I/O CIO
Holt ein Zeichen von der Konsole.

(KEY?) (-- flag) I/O CIO
Das Flag ist 'wahr', falls eine Taste betätigt wurde.

(LINE) (znr scrnr -- adr anzahl) SCR
Ermittelt für das aktuelle File aus Zeilennummer znr und Screennummer scrnr die Adresse adr des ersten Zeichens der Zeile. (anzahl = 32)

(PRINT) (c --) I/O CIO
Das Zeichen c wird (nur) an den Drucker geschickt.

)) (--) P COM
Schaltet den Compile-Modus ein. (Alias zu])

***** (n1 n2 -- n3) ARI
Multiplikation. n3 = n1 * n2 (siehe auch M*)

***/** (n1 n2 n3 -- n4) ARI
Multiplikation mit anschließender Division und einem 32-Bit-Zwischenergebnis. n4=(n1*n2)/n3

***/MOD** (n1 n2 n3 -- n4 n5) ARI
Wie */, jedoch ist n4 der Teilerrest und n5 der Quotient.

+ (n1 n2 -- n3) P ARI
Addition. n3=n1+n2

+! (n adr --) P MEM
Addiert n zum Inhalt der Adresse adr. n kann auch negativ sein.

+LOOP I (n --) VZW
Erhöht den Schleifenindex der DO-+LOOP-Schleife um die Schrittweite n. Ist der Endwert noch nicht erreicht, wird zum Befehl nach DO zurückgesprungen.

, (n --) P DIC
Trägt n ins Wörterbuch ein.

- (n1 n2 -- n3) P ARI
Subtraktion. n3=n1-n2

--> (--) SCR
Das Übersetzen wird mit dem nächsten Screen fortgesetzt.

-1 0 1 2 3 8 (-- n) ARI
Konstanten

-TRAILING (adr n1 -- adr n2) P CIO
Unterdrückt Leerzeichen am Ende des Textes, der auf Adresse adr beginnt. n1 und n2 sind die Textlängen vor- und nachher.

. (n --) CIO
Gibt n auf die Console aus.

." (--) CIO
Benutzung in der Form: ." text" . Der Text text wird zunächst compiliert und beim späteren Programmlauf ausgegeben. Das Leerzeichen vor text muß stehen.

.(I (--) CIO
Benutzung in der Form: .(text) . Der Text text wird sofort ausgegeben. Das Leerzeichen vor text muß stehen.

.ID (nfa --) DIC
Gibt den Namen des Befehles mit der Namensfeldadresse nfa aus.

.HEX (un --) CIO
Gibt un als vierstellige Hexadezimalzahl mit nachgestelltem 'H' aus.

.LINE (znr n --) SCR
Gibt die Zeile znr aus dem Screen n des aktuellen Files zur Console aus.

.R (n1 n2 --) CIO
n1 wird rechtsbündig in ein Feld der Weite n2 ausgegeben.

/ (n1 n2 -- n3) ARI
Ganzzahlige Division. n3=n1 div n2

/MOD (n1 n2 -- n3 n4) ARI
Ganzzahlige Division mit Rest. n3 ist der Rest, n4 der Quotient.

0< (n -- flag) P ARI
Das Flag ist 'wahr', wenn n < 0 ist.

0= (n -- flag) P ARI
Das Flag ist 'wahr', wenn n = 0 ist.

0> (n -- flag) P ARI
Das Flag ist 'wahr', wenn n > 0 ist.

1+ (n -- n+1) P ARI

1- (n -- n-1) P ARI

2! (dd adr --) P MEM
Speichert die doppeltgenaue Zahl dd ab Adresse adr.

2* (n -- n*2) P ARI
Verdoppelt n. (Linksverschiebung von n um ein Bit)

2+ (n -- n+2) P ARI

2- (n -- n-2) P ARI

2/ (n -- n/2) P ARI
Halbiert n. (Rechtsverschiebung von n um ein Bit nur für positive n !)

2@ (adr -- dd) P MEM
Liest die doppeltgenaue Zahl dd von der Adresse adr.

2DUP (dd -- dd dd) P STK
Dupliziert die doppeltgenaue Zahl dd. (entspricht OVER OVER)

2DROP (dd --) P STK
Löscht die doppeltgenaue Zahl dd. (entspricht DROP DROP)

2OVER (dd1 dd2 -- dd1 dd2 dd1) P STK
Doppeltgenaues OVER.

2SWAP (dd1 dd2 -- dd2 dd1) P STK
Vertauscht die beiden doppeltgenauen Zahlen dd1 und dd2 auf dem Stack.

: I (--) name DEF
Eröffnet die Definition des Befehles 'name'.

; I (--) DEF
Beendet die Definition eines Befehles.

< (n1 n2 -- flag) P ARI
Das Flag ist 'wahr', wenn n1 < n2 ist.

<# (dd1 -- dd2) CON
Eröffnet die Umwandlung der doppeltgenauen Zahl dd1 in einen Ziffern-String.

= (n1 n2 -- flag) P ARI
Das Flag ist 'wahr', wenn n1 = n2 ist.

> (n1 n2 -- flag) P ARI
Das Flag ist 'wahr', wenn n1 > n2 ist.

>R (n --) P RST
Bringt die Zahl n zum Returnstack. Mit Vorsicht verwenden !

? (adr --) MEM
Zeigt den Inhalt von Adresse adr (vorzeichenbehaftet) an.

?COMP (--) P DIA
Erzeugt eine Fehlermeldung, falls nicht im Compile-Modus.

?CSP (--) P DIA
Erzeugt eine Fehlermeldung, falls der augenblickliche Stand des Datenstacks nicht mit dem in CSP gemerkten übereinstimmt. (vgl. !CSP)

?DNEGATE (dd1 dd2 -- dd3) ARI
Das Vorzeichen der Zahl dd1 wird gewechselt, wenn dd2 negativ ist.

?DO I (anfwert endwert --) VZW
Eröffnet Schleife nur, wenn Anfangswert ungleich Endwert ist. (siehe DO)

?DUP (n -- n n ö 0) STK
Dupliziert die Zahl n, falls sie ungleich Null ist.

?ERROR (flag n --) P DIA
Aufruf von ERROR mit der Fehlernummer n, falls das Flag 'wahr' ist.

?EXEC (--) P DIA
Erzeugt eine Fehlermeldung, falls nicht im Execute-Modus.

?LEAVE (flag --) VZw
Verlassen einer DO-LOOP-Schleife wenn das Flag 'wahr' ist. (siehe LEAVE)

?LINE (un --) CIO
Führt einen Zeilenvorschub aus, wenn un Zeichen nicht mehr auf die Zeile passen. Die Zeilenlänge steht in RMARGIN.

?LOADING (--) P SCR
Erzeugt eine Fehlermeldung, falls nicht geladen wird.

?NEGATE (n1 n2 -- n3) ARI
Das Vorzeichen der Zahl n1 wird gewechselt, wenn n2 negativ ist.

?PAIRS (n1 n2 --) P COM
Erzeugt eine Fehlermeldung, falls n1 und n2 ungleich sind.

?STACK (--) P STK
Prüft, ob Über- oder Unterlauf des Datenstacks.

@ (adr -- n) P MEM
Übergibt die Zahl n, die auf Adresse adr steht.

A: (--) I/O MAS
Wählt das Laufwerk A. (nur bei Massenspeicher Diskette)

ABORT (--) DIA
Reinitialisierung der Stacks und Übergang in die Kommandoebene.

ABS (n -- önö) P ARI

Bildet den Betrag von n.

AGAIN I (--) VZW
In Verbindung mit BEGIN zum Aufbau einer Endlosschleife. (siehe Anhang)

ALLOT (n --) P DIC
Fügt eine Lücke von n Bytes in das Wörterbuch ein.

ALSO (--) ONLY VOC
Angewandt: ALSO vokabular. Nimmt vokabular neu in die Suchreihenfolge auf.

AND (n1 n2 -- n3) P ARI
Bitweise UND-Verknüpfung von n1 und n2.

B/BUF (-- n) SCR
Systemkonstante. Anzahl der Bytes pro Buffer. (n = 512 oder 1024)

B: (--) I/O CIO
Wählt das Laufwerk B. (nur bei Massenspeicher Diskette)

BASE (-- vadr) CIO
Uservariable. Enthält die aktuelle Zahlenbasis.

BEGIN I (--) VZW
Beginn einer Programmschleife. (mit AGAIN, UNTIL oder WHILE und REPEAT)

BL (-- space) CIO
Systemkonstante. Übergibt den ASCII-Code für das Leerzeichen. (=32d)

BLANK (adr un --) P MEM
Schreibt ab Adresse adr un Leerzeichen, falls un > 0.

BLK (-- vadr) SCR
Uservariable. Enthält die aktuelle Blocknummer.

BLOCK (n -- adr) P SCR
Transportiert den Screen n in den Screenpuffer (falls er sich noch nicht dort befindet) und übergibt die Anfangsadresse des Screenpuffers (=FIRST).

BORDER (frbnr --) P SPO
Setzt die Bildschirmrandfarbe auf frbnr.

BROFF (--) P DIA
Schaltet die Abfrage der 'STOP'-Taste aus. Endlosschleifen lassen sich jetzt nur noch durch die 'RESET'-Taste abbrechen. (vgl. BRON)

BRON (--) P DIA
Ab sofort wird nach der Abarbeitung jedes Primitives die 'STOP'-Taste abgefragt und im Falle der Betätigung ein ABORT ausgelöst. Die Programmlaufzeit vergrößert sich geringfügig.

BYE (--) DIA
Beendet die Arbeit in Turbo-FORTH.

C! (b adr --) P MEM
Schreibt das Byte b auf Adresse adr.

C, (b --) P DIC

Trägt das Byte b ins Wörterbuch ein.

C/L (-- n) CIO
Systemkonstante. Anzahl der Zeichen pro Screenzeile. (n= 32 oder 64)

C@ (adr -- b) P MEM
Übergibt das Byte b aus Adresse adr.

CASE I (entscheidungswert --) VZW
Mehrfachverzweigung. (siehe Anhang)

CLOSE (--) name SCR
Schließt das File 'name'.

CMP (adr1 adr2 un1 -- true ö un2 false) P MEM
Vergleich zweier Bytefolgen auf adr1 und adr2 für un1 Bytes. Bei Gleichheit wird ein 'wahr'-Flag übergeben, sonst die Position der ersten Abweichung (un2 = 0...un1) und ein 'falsch'-Flag.

COLD (--) P DIA
Kaltstart des FORTH-Systems. Das Wörterbuch wird auf denselben Stand wie beim letzten COLD gebracht. Die RAM-Disk bleibt erhalten. Soll auch sie gelöscht werden, ist **COLD** und danach **0 RE-INIT** einzugeben.

COMPILE (--) name COM
Compiliert die cfa des Befehls 'name' ins Wörterbuch.

CONSTANT (n --) name DEF
Definition einer Konstanten 'name'. n ist ihr Wert.

CONTEXT (-- vadr) DIC
Zeigt zu dem Vokabular, in dem Suchläufe zuerst beginnen.

CONVERT (udd1 adr1 -- udd2 adr2) CIO
Die Zeichenkette auf Adresse adr1+1 wird mit Anfangswert dd1 in eine doppeltgenaue Zahl dd2 konvertiert. adr2 ist die Adresse des ersten nicht-konvertierbaren Zeichens.

COUNT (adr1 -- adr2 n) P CIO
Übergibt die Adresse adr2 des eigentlichen Textbeginns und die Länge n eines Textes im FORTH-Standard-Format ab adr1.

CR (--) CIO
Vektor. Ruft normalerweise CRLF auf und setzt OUT auf Null.

CRLF (--) I/O CIO
Setzt den Kursor auf den Anfang der nächsten Zeile.

CREATE (--) name DEF
Erzeugt einen Wörterbucheintrag für den Befehlskopf 'name' im Current-Vokabular. Der nächste freie Platz im Wörterbuch ist die pfa, welche bei Aufruf von name auf den Stack gelegt wird.

CSP (-- vadr) DIA
Uservariable. Merkzelle für Stackposition. (vgl. !CSP und ?CSP)

CURRENT (-- vadr) DIC
Zeigt zu dem Vokabular, das gerade erweitert wird.

D+ (dd1 dd2 -- dd3) P ARI
Addition zweier doppeltgenauer Zahlen. dd3=dd1+dd2

D- (dd1 dd2 -- dd3) P ARI
Subtraktion zweier doppeltgenauer Zahlen. dd3=dd1-dd2

D. (dd --) CIO
Ausgabe der doppeltgenauen Zahl dd.

D.R (dd n --) CIO
dd wird rechtsbündig in ein Feld der Weite n ausgegeben.

D< (dd1 dd2 -- flag) P ARI
Doppeltgenauer Vergleich. Das Flag ist 'wahr', wenn dd1 < dd2 ist.

D= (dd1 dd2 -- flag) P ARI
Doppeltgenauer Vergleich. Das Flag ist 'wahr', wenn dd1 = dd2 ist.

DABS (dd -- öddö) P ARI
Bildet den Betrag von dd.

DECIMAL (--) P CIO
Vereinbart die dezimale Zahlenbasis.

DELCU (--) P CIO
Löscht das Kursorsymbol auf dem Bildschirm.

DEFINITIONS (--) ONLY VOC
Anwendung in der Form: vokabular DEFINITIONS. Damit werden die nächsten Befehle dem Vokabular vokabular angefügt.

DLITERAL I (dd --) COM
Compiliert die 32-Bit-Zahl dd ins Wörterbuch.

DNEGATE (dd -- -dd) P ARI
Vorzeichenwechsel von dd.

DO I (endwert anfangswert --) VZW
In Verbindung mit LOOP oder +LOOP zur Realisierung von Programmschleifen. Der Schleifenindex läuft von anfangswert bis ausschließlich endwert.
(siehe auch ?DO, I, J, K, LEAVE, ?LEAVE)

DOES> I (--) DEF
Angewandt innerhalb von Colon-Definitionen in der Form:
: cccc ... <create> ..... DOES> ..... ;
Bei jedem Aufruf von cccc in der Form 'cccc name' wird das Wort name definiert. <create> ist CREATE oder ein Befehl, welcher CREATE enthält.
Der Befehl name arbeitet den High-Level-Code hinter DOES> ab, nachdem die Adresse des ersten Parameters auf den Stack gelegt wurde.

DP (-- vadr) DIC
Uservariable. Enthält den aktuellen Wörterbuchzeiger.

DPL (-- vadr) CIO
Uservariable. Enthält die Anzahl der Ziffern rechts des Dezimalpunktes bei Eingabe einer doppeltgenauen Zahl.

DROP (n --) P STK
Löscht die Zahl n auf dem Stack.

DUP (n -- n n) P STK
Dupliziert die Zahl n.

EDIT (--) EDI
Editieren des Quelltextes. (vgl. Beschreibung Screeneditor)

ELSE I (--) VZW
Eröffnet den 'falsch'-Zweig in der IF..ELSE..ENDIF-Struktur. (siehe Anhang)

EMIT (c --) CIO
Vektor. Ruft normalerweise (EMIT) auf und incrementiert die Variable OUT.

EMPTY (--) COM
Löscht im Wörterbuch alle nachfolgenden Befehle.

ENDCASE I (--) VZW
Beendet eine Mehrfachverzweigung. (siehe Anhang)

ENDIF I (--) VZW
Beendet eine mit IF eingeleitete Programmverzweigung. (siehe Anhang)

EOD (-- adr) MEM
Systemkonstante. Übergibt die Endadresse der RAM-Disk.

EOD! (adr --) MEM
Vereinbart das Ende der RAM-Disk neu. Der neue Wert muß kleiner als der alte sein, da Turbo-FORTH bei der Grundinitialisierung den gesamten freien Speicher verwendet. Danach ist unbedingt RE-INIT aufzurufen.

ERASE (adr un --) P MEM
Löscht un Bytes ab Adresse adr mit 000, falls un > 0.

ERROR (n --) DIA
Vektor. Ruft normalerweise (ERROR) auf. Dieser Befehl erzeugt eine Fehlermeldung entsprechend der Fehlernummer n (0...31) und führt anschließend ABORT aus.

ESC" (--) DIA
Verwendet in der Form : ESC" text". text kann bis 24 Zeichen lang sein und wird bei Betätigen der 'ESC'-Taste in EXPECT (also z.B. in der Kommando-Eingabeschleife oder in IN-STR) übergeben. Anschließend wird <ENTER> simuliert. Die 'ESC'-Taste ist mit "EMPTY 1 LOAD " vorbelegt.

EXECUTE (cfa --) P SYS
Führt den Befehl aus, dessen cfa auf dem Stack liegt.

EXPECT (adr n --) CIO
Erwartet die Eingabe von n Zeichen (vorzeitiger Abbruch mit der 'ENTER'-Taste) und legt den Text ab der Adresse adr ab. Anscließend wird die Anzahl der eingegebenen Zeichen in die Variable SPAN eingetragen.
Bei Betätigen der 'RUN'- oder der 'ESC'-Taste wird der mit RUN" bzw. ESC" (siehe dort) vereinbarte Text übergeben und <ENTER> simuliert.

FCB (-- adr) MAS
Systemkonstante. Beginn des Standard-File-Control-Blockes. (siehe Anhang)

FENCE (-- vadr) DIC
Variable. Enthält die Grenze für den Befehl FORGET.

FILL (adr un b --) P MEM
Füllen eines Speicherbereiches der Länge un ab Adresse adr mit dem Byte b.

FIRST (-- adr) SCR
Systemkonstante. Beginn des Screenpuffers.

FORGET (--) name DIC
Streicht 'name' und alle später definierten Befehle aus dem Wörterbuch.

FORTH (--) ONLY VOC
Name des FORTH-Sprachkernes, macht ihn zum Context-Vokabular.

FRE (-- n) P SCR
Ermittelt die Anzahl noch freier Screens n.

GET (->) name MAS
Eröffnet das File 'name' und lädt es vom Massenspeicher in die RAM-Disk.
Es können mehrere Befehle **GET name** auf einer Eingabezeile stehen.
Bei Kassette ignoriert der Befehl Files mit einem anderen als dem angegebenen Namen und läßt sich nur dann durch die 'STOP'-Taste abbrechen, wenn ein Geräuschpegel vom Recorder anliegt.

HERE (-- adr) P DIC
Übergibt die Adresse der nächsten freien Stelle im Wörterbuch.

HEX (--) P CIO
Vereinbart die hexadezimale Zahlenbasis.

HLD (-- vadr) CIO
Uservariable. Bei Konvertierung von Ziffern-Strings benutzt.

HOLD (c --) CIO
Fügt das Zeichen c in den Ziffern-String ein.

I (-- schleifenindex) P RST
Übergibt den Schleifenindex der DO-LOOP-Schleife, in der I aufgerufen wird.

IF I (flag --) VZW
Eröffnet den 'wahr'-Zweig in einer Programmverzweigung. (siehe Anhang)

IMMEDIATE (--) P DIC
Setzt den zuletzt definierten Befehl auf 'Immediate', d.h., er wird während des Compilerlaufes abgearbeitet.

IN-NUM (-- false ö n flag) CIO
Dient der Eingabe einer Zahl n während des Programmlaufes. Es gibt drei verschiedene Rückkehrmöglichkeiten:
(-- false) Es wurde nur ENTER betätigt.
(-- n 001) Die eingegebene Zeichenfolge enthält nichtkonvertierbare Zeichen. n repräsentiert den konvertierten Teil.
(-- n true) Die Konvertierung verlief erfolgreich.

IN-STR (-- b) CIO
Eingabe von Text (max. 32 Zeichen), welcher im FORTH-Standard-Format (also

mit Count-Byte) nach HERE gelangt. IN-STR übergibt das Count-Byte (Länge), welches Null ist, falls nur die 'ENTER'-Taste betätigt wurde.

INCLUDE (--) name MAS
Compilation des Files name. INCLUDE sucht das File in der RAM-Disk und liest es von dort, wenn es vorhanden ist. Anderenfalls wird eine Leseoperation vom Massenspeicher ausgeführt und das File unter Umgehung der RAM-Disk ins Wörterbuch compiliert. Der Befehl unterstützt eine Einteilung in Haupt- und Unterprogramme und damit die Erstellung von Bibliotheken.
INCLUDE darf nicht geschachtelt verwendet werden.
(Soll das File von Kassette gelesen werden, muß es mit SAVE# gespeichert sein.)

INDEX (--) SCR
Gibt die ersten Zeilen aller Screens des aktuellen Files aus.

INK (farbnr --) P SPO
Setzt die Vordergrundfarbe auf farbnr.

INTERPRET (--) DIA
Ist der Kern des äußeren Interpreters.

J (-- schleifenindex) P RST
Übergibt den Schleifenindex der äußeren Schleife bei zwei ineinander verschachtelten Schleifen. Der Returnstack darf innerhalb der Schleifen nicht verändert werden!

J?N (-- flag) CIO
Gibt " ? (J/N): " aus und wartet auf eine Tastenbetätigung. Das Flag ist 'wahr', falls "J" oder "j" gedrückt wurde, sonst 'falsch'.

K (-- schleifenindex) P RST
Übergibt den Schleifenindex der äußersten Schleife bei drei ineinander verschachtelten Schleifen. (vgl. I und J)

KEY (-- c) CIO
Vektor. Ruft normalerweise (KEY) auf. c ist der ASCII-Code der gedrückten Taste.

KEY? (-- flag) CIO
Vektor. Ruft normalerweise (KEY?) auf. flag ist 'wahr', wenn eine Taste gedrückt wurde.

LAST (-- vadr) DIC
Variable. Enthält die nfa des zuletzt definierten Befehlskopfes.

LEAVE (--) P RST
Erzwingt das sofortige Verlassen einer DO...LOOP- oder DO...+LOOP-Schleife.

LIMIT (-- adr) SCR
Systemkonstante. Enthält die dem Screenpuffer folgende Adresse.

LIST (n --) SCR
Der Screen n wird angezeigt.

LITERAL I (n --) COM
Compiliert LIT und die Zahl n ins Wörterbuch.

LOAD (n --) SCR
Startet die Übersetzung des aktuellen Files ab Screen n.

LOC (zeile spalte --) P CIO
Setzt den Kursor auf zeile, spalte.

LOOP I (--) VZW
Wie +LOOP mit der Schrittweite 1.

M* (n1 n2 -- dd) ARI
Gemischtgenaue Multiplikation von n1 mit n2 und einem 32-Bit-Ergebnis.

M/ (dd n1 -- n2 n3) ARI
Vorzeichenbehaftete gemischtgenaue Division. n3 = dd div n1 ; n2 = Rest

M/MOD (udd1 un1 -- un2 udd2) P ARI
Vorzeichenlose gemischtgenaue Division. udd2 = udd1 div un1 ; un2 = Rest

MAX (n1 n2 -- n3) ARI
n3 ist die größere der beiden Zahlen n1 und n2.

MESSAGE (n --) DIA
Ausgabe der in Turbo-FORTH installierten Fehlermeldung n.

MIN (n1 n2 -- n3) ARI
n3 ist die kleinere der beiden Zahlen n1 und n2.

MS (un --) CIO
Wartet un Millisekunden.

MOD (n1 n2 -- n3) ARI
Ermittelt den Teilerrest der Division n1 div n2.

MOVE (adr1 adr2 un --) MEM
Transport einer Bytefolge der Länge un von adr1 nach adr2 bei un > 0. Die Speicherbereiche können sich überlappen.

NEGATE (n -- -n) P ARI
Vorzeichenwechsel von n.

NEXTSCR (scrnr1 -- scrnr2) P SCR
Ermittelt in der RAM-Disk zum Screen scrnr1 den Folgescreen scrnr2. Existiert er nicht, so ist scrnr2 = 0.

NIP (n1 n2 -- n1) STK
Löscht die Zahl unter dem TOS (Top of Stack).

NOOP (--) P SYS
Leerbefehl. (z.B. in der CASE-Anweisung als Anweisung 0)

NUMBER? (adr -- dd flag) CON
Der Ziffern-String ab adr wird in die 32-Bit-Zahl dd umgesetzt. Das Flag ist 'falsch', wenn nichtkonvertierbare Zeichen auftraten.

ONLY (--) ONLY VOC
Angewandt: ONLY vokabular. Jetzt werden nur vokabular und ONLY durchsucht.

OR (n1 n2 -- n3) P ARI

Bitweise ODER-Verknüpfung von n1 und n2.

OUT (-- vadr) CIO
Uservariable. Wird bei jedem EMIT um eins erhöht.

OVER (n1 n2 -- n1 n2 n1) P STK
Dupliziert den Second auf den TOS (Top of Stack).

P! (b padr --) P CIO
Gibt b an den durch padr adressierten Peripherieschaltkreises aus.

P@ (padr -- b) P CIO
Liest b von dem durch padr adressierten Peripherieschaltkreis.

PAD (-- adr) P DIC
Übergibt die Adresse des Text-Ausgabepuffers.

PAGE (--) CIO
Gibt Control-L aus. (= Bildschirmloeschen bzw. Seitenvorschub)

PAPER (farbnr --) P SPO
Stellt die Hintergrundfarbe farbnr ein. (vgl. INK und BORDER)

QUERY (--) CIO
Holt eine Eingabezeile, bringt sie zum TIB (Terminal-Input-Buffer) und hängt ein Nullbyte an als Endekennung für INTERPRET.

QUIT (--) DIA
Schaltet in den Kommandomodus.

R@ (-- n) P RST
Kopiert das oberste Wort des Returnstack zum Datenstack.

R/W (adr n1 n2 --) SCR
Transport des Screens n1. Bei n2=1 aus der Ramdisk zur Adresse adr, bei n2=0 von adr in die Ramdisk. Standard: adr = FIRST.

R0 (-- vadr) RST
Uservariable. Enthält die Grundadresse des Returnstacks.

R> (-- n) P RST
Bringt das Wort n vom Returnstack zum Datenstack. (vgl. >R)

RD-FCB (FCB -- flag) MAS
Eröffnet Lesevorgang vom dem Massenspeicher. (siehe Anhang)

RDS (adr FCB -- adr' FCB flag) MAS
Liest ein Record vom Massenspeicher. (siehe Anhang)

RE-INIT (un --) P DIA
Dient der Neuaufteilung des Speichers (z.B. bei vollem Wörterbuch).
Die Files werden zerstört und das Wörterbuch 'eingefroren', d.h. es bleibt erhalten und wird gegen FORGET und COLD geschützt.
Bei un=0 reserviert Turbo-FORTH etwa 5/8 des noch freien Speichers für Screens, und bei un>0 werden (falls möglich) un Screens eingerichtet.
Der restliche Speicher wird für das Wörterbuch verwendet.

REPEAT I (--) VZW

Abschluß der BEGIN..WHILE..REPEAT-Schleife, Rücksprung zu BEGIN. (s. Anhang)

ROT (n1 n2 n3 -- n2 n3 n1) P STK

RUN" (--) DIA
Verwendet wie ESC", jedoch wird die 'RUN'-Taste mit max. 8 Zeichen belegt. Nach Starten von Turbo-FORTH ist "EDIT " voreingestellt.

RP@ (-- n) P RST
Liest den momentanen Stand des Returnstacks.

S>D (n -- dd) P ARI
Wandelt die einfachgenaue Zahl n in die doppeltgenaue dd.

S0 (-- vadr) STK
Uservariable. Enthält die Grundadresse des Datenstacks.

SAVE (->) name MAS
Speichert das File 'name' aus der RAM-Disk auf den Massenspeicher. Es können mehrere Befehle **SAVE name** hintereinander angegeben werden. Zu beachten ist, daß der Speichervorgang sofort nach dem Quittieren der Eingabe durch die 'ENTER'-Taste beginnt.

SAVE# (--) name KAS
Schreibt das File 'name' mit Aufzeichnungslücken von 2 Sekunden zwischen den Screens auf Kassette. (nötig für INCLUDE ; auch durch GET lesbar)
Der Befehl existiert nur bei Massenspeicher Kassette.

SCR (-- vadr) SCR
Variable. Enthält die aktuelle Screennummer.

SCROFF (--) CIO
Schnelle Grundinitialisierung des Bildschirms.

SIGN (n dd -- dd) CON
Fügt das Vorzeichen von n in den Ziffern-String ein. (innerhalb <# ... #>)

SP@ (-- n) P STK
Übergibt den Stand des Datenstacks vor Aufruf des Befehles.

SPACE (--) P CIO
Gibt ein Leerzeichen zur Console aus.

SPACES (un --) P CIO
Gibt un Leerzeichen zur Console aus. Bei un <= 0 wird der Befehl ignoriert.

STATE (-- vadr) COM
Uservariable. Der Inhalt ist ungleich Null, wenn das System compiliert.

SWAP (n1 n2 -- n2 n1) P STK
Vertauscht den Second und den TOS.

TIB (-- adr) CIO
Übergibt die Adresse des Terminal-Input-Buffers.

TOGGLE (adr b --) P MEM
XOR-Verknüpfung des Bytes in Adresse adr mit b.

TYPE (adr n --) P CIO
Gibt n Zeichen, die ab Adresse adr stehen, an die Console aus.

UM* (un1 un2 -- udd) P ARI
Vorzeichenlose Multiplikation. udd= un1*un2

U. (n --) CIO
Die Zahl n wird vorzeichenlos ausgegeben.

UM/MOD (udd un1 -- un2 un3) P ARI
Vorzeichenlose Division. un3 = udd div un1 ; un2 - Rest

U< (un1 un2 -- flag) P ARI
Vorzeichenloser Vergleich. Das Flag ist 'wahr' bei un1 < un2.

UNTIL I (flag --) VZW
Sprung zu BEGIN, wenn das Flag 'falsch' ist. (siehe Anhang)

USE (--) name SCR
Erklärt das File 'name' zum aktuellen File. Wenn es noch nicht existiert, wird es automatisch angelegt.

USER (n --) name DEF
Definiert eine Uservariable 'name'. n ist der Adreßoffset der Variablen im USER-Feld. Die gerade Zahl n kann für Neudefinitionen 68...110 betragen.

VARIABLE (--) name DEF
Definiert eine Variable 'name', deren Anfangswert Null ist.

VOCABULARY (--) ONLY name DEF
Erzeugt ein neues Vokabular 'name'.

WARM (--) DIA
Warmstart des FORTH-Systems. Ausgabe der Systemmeldung.

WHILE I (flag --) VZW
Sprung hinter REPEAT, wenn das Flag 'falsch' ist. (siehe Anhang)

WIDTH (-- vadr) CIO
Uservariable. Enthält die max. Länge eines Definitionsnamens.

WINDOW (ob.Z. un.Z. l.Sp. r.Sp. --) P CIO
Definition eines WINDOWs. (obere / untere Zeile linke / rechte Spalte)

WORD (c -- HERE) DIA
Liest die nächsten Zeichen im Eingabestrom, wobei c als Trennzeichen dient, und bringt sie nach HERE. Das auf HERE stehende Count-Byte ist Null, wenn der Eingabestrom erschöpft ist.

WORDS (--) ONLY DIC
Zeigt alle Befehle des CONTEXT-Vokabulares an.

WR-FCB (FCB -- flag) MAS
Eröffnet das Schreiben zum Massenspeicher. (siehe Anhang)

WR-LAST (adr FCB -- flag) MAS
Schreibt das letzte Record zum Massenspeicher. (siehe Anhang)

WRS (adr FCB -- adr' FCB flag) MAS
Schreibt ein Record zum Massenspeicher. (siehe Anhang)

XOR (n1 n2 -- n3) P ARI
n1 und n2 werden bitweise EXCLUSIV-ODER-verknüpft.

[I (--) P COM
Einschalten des ausführenden Modus.

['] I (--) name COM
Compiliert die cfa von name als Literal ins Wörterbuch.

[COMPILE] I (--) name COM
Erzwingt die Compilation des Immadiate-Wortes name.

] (--) P COM
Einschalten des compilierenden Modus.

Fehlermeldungen

Nr.	Ausschrift	Bedeutung
0	??	= unbekannter Befehl oder Filename
1	Stack leer	= Zugriff auf leeren Datenstack
2	Dictionary voll	= Wörterbuchüberlauf beim Compilieren
4	existiert bereits	= Befehl oder Filename bereits vorhanden
6	0/	= Division durch Null bzw. Ergebnisüberlauf
7	Stack voll	= Datenstacküberlauf
8	ungueltige Screennummer	= Screen existiert nicht oder Screennummer kleiner 1 oder größer 127 oder in MOVE bei Überschreiben ohne Erlaubnis
9	maximal 16 Files	= Fileverzeichnis ist voll
10	kein File	= es existiert kein aktuelles File
11	RAM-Disk voll	= Überlauf der RAM-Disk
17	nur compilierbar	= Befehl innerhalb Befehlsdefinition benutzen
18	nur ausfuehrbar	= Befehl nicht in Befehlsdefinition benutzen
19	falsche Struktur	= falscher Verzweigungsbefehl verwendet
20	Struktur beenden	= Verzweigungsbefehl vergessen oder Datenstapel während des Compilierens verändert
21	geschuetzt	= Befehl ist gegen FORGET geschützt
22	nur ladbar	= Befehl nur im Quelltext verwenden
24	Vokabular angeben	= Befehl wurde in einem anderen Vokab. definiert

Strukturierte Anweisungen

```
Stapel                      Struktur / Ausführung

                             Verzweigungen

( flag --  )                IF
                            { ... }  wenn Flag 'wahr'
                            ELSE
                            { ... }  wenn Flag 'falsch'
                            ENDIF


( flag --  )                IF
                            { ... }  wenn Flag 'wahr'
                            ENDIF
                                     wenn Flag 'falsch', weiter


                          Mehrfachverzweigung

( entscheidungswert -- )    CASE
                            anweisung 0
                            anweisung 1
                            ...
                            anweisung K   K=1,2,...125
                            ENDCASE
```

Zur Ausführung gelangt die Anweisung, deren Nummer dem Entscheidungsparameter entspricht. Anweisung 0 muß **immer** stehen ! Sie wird bei einem Entscheidungsparameter von kleiner 1 oder größer als der Maximalwert K abgearbeitet.

```
                          Programmschleifen

( endwert anfwert --  )     DO
                            { ... }  mindestens einmal ausgeführt
                            LOOP


( endwert anfwert --  )     DO
                            { ... }  mindestens einmal ausgeführt
( schrittweite --  )        +LOOP


                            BEGIN
                            { ... }
( flag --  )                UNTIL    wenn Flag 'falsch', Sprung zu BEGIN ;
                                     sonst weiter

                            BEGIN
                            { ... }
( flag --  )                WHILE    wenn Flag 'falsch', hinter REPEAT weiter
                            { ... }
                            REPEAT   Sprung zu BEGIN
```

```
BEGIN
{ ... }  Endlosschleife
AGAIN
```

Kassettenbefehle

Die Befehle **WFREC**, **RFREC**, **WRS** und **RDS** ermöglichen dem Nutzer den Aufbau eigener Kassettenroutinen z.B. zur Speicherung von Daten. Damit man diese Befehle richtig und effektiv verwenden kann, muß man folgendes beachten:

Aufbau des File-Control-Blocks ab Adresse adr :

```
adr   ...adr+7    Name des Files in ASCII-Zeichen
adr+8 ...adr+10   Typ des Files in ASCII-Zeichen    - bei Screendatei: "(F)"
adr+17...adr+18   Anfangsadresse                    - bei Screendatei: 0001
adr+19...adr+20   Endadresse                        - bei Screendatei: Screenanzahl
adr+21...adr+22   Startadresse                      - bei Screendatei: 0000
adr+23            Schreibschutz                     - bei Screendatei: 00
```

(adr+17...adr+23 sollten in eigenen Routinen mit 00 belegt werden.)

Die Adresse des Standard-File-Control-Blockes wird durch die Systemkonstante **FCB** übergeben. Sie beträgt 92 (05CH).

Verwendete Systemzellen des Betriebssystems:

Adresse Funktion

27 (01BH) Enthält die Speicheradresse des zu lesenden oder zu schreibenden Records. Wird nach jedem RDS bzw. WRS automatisch um 128 erhöht.

107 (06BH) Enthält beim Lesen die Nummer des gelesenen Records und beim Schreiben die Nummer des zu schreibenden Records. Wird nach WRS automatisch um 1 erhöht.

108 (06CH) Enthält beim Lesen die Nummer des zu lesenden Records. Wird nach RDS automatisch um 1 erhöht.

Das letzte Record hat die Recordnummer 255 (0FFH).

WFREC (adr --) adr ist die Adresse des nach obigem Muster vorbelegten File-Control-Block, der als erstes Record mit der Recordnummer 1 ausgegeben wird.

WRS Gibt 128 Bytes ab der Adresse, die in der Speicherzelle 27 steht, auf Kassette aus. Handelt es sich um hintereinander stehende Daten, so kann WRS mehrmals aufgerufen werden, ohne die Systemzellen jedesmal neu belegen zu müssen.

RFREC (adr --) Erwartet ein Record mit der Nummer **1** und speichert es ab der Adresse 128 (080H). Anschließend werden die ersten 11 Zeichen als Name und Typ angezeigt und mit Name und Typ des ab Adresse adr stehenden und nach obigem Muster vorbelegten File-Control-Block verglichen. Bei Gleichheit werden die Fileparameter aus den Adressen 139 (091H) bis 147 (099H) in adr+17 bis adr+24 übertragen.
Solange nicht das richtige File gelesen wurde, wartet der Befehl in einer Schleife, die sich nur bei anliegendem Geräuschpegel durch die 'STOP'-Taste abbrechen läßt.

RDS Liest 128 Bytes in den Hauptspeicher ab der in der Speicherzelle 27 stehende Adresse. Der Befehl ist erst beendet, wenn das Record

fehlerfrei gelesen wurde.

Die Kassettenbefehle lassen sich durch die 'STOP'-Taste abbrechen.
Der gerade laufende Lese- oder Schreibbefehl wird in jedem Fall noch beendet.

Editorkommandos

Kommando	Taste	Beschreibung / Bemerkungen
<CONTR> A		PUSH LINE, Zeile wird auf den Zeilenstapel gelegt
<CONTR> B	CL LN	Zeile löschen
<CONTR> D		Zeile streichen (**D**elete)
<CONTR> E		**E**ndemarkierung für INSERT setzen
<CONTR> F		Zeile ein**F**ügen
<CONTR> G		aktuellen Screen physisch streichen (Vorsicht !) Der Screen wird sofort in der RAM-Disk gelöscht.
	<-	Cursor nach links
	->	Cursor nach rechts
	ðexpnd-76	Cursor eine Zeile nach oben
	ðexpnd-76	Cursor eine Zeile nach unten
<CONTR> L		Screen **L**öschen (d.h., mit Leerzeichen überschreiben)
<CONTR> M	ENTER	Cursor auf den Anfang der nächsten Zeile
<CONTR> N	STOP	**N**ächster Screen (anlegen, falls nicht vorhanden)
<CONTR> O		Zeile vom Zeilenstapel holen und auf aktuelle Cursorposition schreiben
<CONTR> Q	RUN	**Q**uit, Arbeit mit dem aktuellen Screen beenden und den nächsten vorhandenen Screen aufrufen
<CONTR> R		alten Screeninhalt zu**R**ückladen
<CONTR> S	PAUSE	Zeichen an Cursorposition **S**treichen
<CONTR> T	COLOR	Suche einer Zeichenfolge (ab Cursorposition)
<CONTR> V	LIST	**V**orhergehender Screen (anlegen, falls nicht vorhanden)
<CONTR> W		**W**eiter suchen (siehe <CONTR> T)
	->|	vorwärts zum Anfang des nächsten Wortes
	|<-	rückwärts zum Anfang des vorhergehenden Wortes
	DEL	Zeichen vor Cursorposition streichen
	INS	INSERT-Mode ein/aus schalten
	ESC	Arbeit im Editor beenden