
FORTH

INHALTSVERZEICHNIS

teil 1

editor

1.0 Allgemeines

FORTH organisiert seinen Programmspeicherplatz in "Screens". Jeder Screen umfaßt 1024 Zeichen.
ZX-FORTH kann zu ein und derselben Zeit nur einen Screen im Speicher haben.
Jeder Screen hat eine Nummer, welche im Bereich von 0 bis 32768 liegen muß.

Ein Screen besteht aus 16 Zeilen mit je 64 Zeichen.
Der FORTH-Screen ist nur eine Zusammenstellung von Speicherplätzen und hat nichts mit dem Bildschirmformat des ZX zu tun.

1.1 Anwählen eines Screens und Texteingabe

Da jedem Screen eine Zahl zugewiesen ist, kann man ihn mit dieser Zahl identifizieren und anwählen.
Um einen Screen anzuwählen gebraucht man
n CLEAR (löscht Screen n und wählt ihn zur Eingabe an).
Um irgendetwas einzugeben kann man Screen n nach CLEAR mit dem P (setz-) Kommando testen:

```
Ø P DIES ZEIGT
1 P WIE TEXT IN DIE
2 P ZEILEN Ø,1,2 EINGEGEBEN WIRD:
```

2.0 Zeileneditierung

Während dieser Beschreibung des Editors wird alles auf PAD bezogen. PAD ist ein Textpuffer, in welchem eine Textzeile gespeichert werden kann.

2.1 Zeileneditierungsbefehle

n D löscht Zeile n, hält sie jedoch in PAD. Zeile n+1 wird in Zeile 15 kopiert, um eine Zeile aufzurutschen.

n E füllt Zeile n mit Leerzeichen.

n I setzt den Text aus PAD in Zeile n, schiebt die alte Zeile n und alle folgenden Zeilen eine Zeile tiefer. Zeile 15 geht dabei verloren.

n R überschreibt Zeile n mit dem Text aus PAD.

n S verschiebt Zeile n und folgende um eine Zeile nach unten und füllt Zeile n mit Leerzeichen. Zeile 15 geht verloren.

3.0 Bildschirmeditierungsbefehle

n LIST	listet Screen n und wählt ihn zum Editieren an; wenn Screen n nicht im Speicher steht, wird versucht, ihn von Cassette zu laden.
n CLEAR	löscht Screen n und wählt ihn zum Editieren an.
L	listet den aktuellen Screen. Die Cursorzeile wird auch gelistet, um die aktuelle Position des Cursors zu zeigen.
FLUSH	wird am Ende einer Programmeingabe gebraucht um den aktuellen Screen auf Cassette zu speichern.

4.0 Cursorsteuerung und Zeichenkettenbearbeitung

Der Screen, an dem gerade gearbeitet wird, ist in einem Pufferspeicher festgehalten. Der Cursor ist eine Variable, die auf eine Stelle in diesem Puffer zeigt. Folgende Befehle sind dazu da, den Cursor zu bewegen, entweder direkt oder durch Suchen nach bestimmten Text im Puffer, auch kann Text an der Cursorposition eingefügt oder gelöscht werden.

4.1 Cursorsteuerung

TOP plaziert den Cursor an den Anfang des Screens.

n M bewegt den Cursor n Zeichen weit auf der Cursor-Zeile. Die Position des Cursors in der Zeile wird durch _ dargestellt.

4.2 Zeichenkettenbearbeitung

F text sucht ab der Cursorposition abwärts bis die Kette "text" gefunden ist. Der Cursor wird zum Ende der Kette versetzt und die Cursorzeile wird gedruckt. Wenn die Kette nicht gefunden ist, wird Fehlermeldung 0 ausgegeben und der Cursor wird an den Anfang des Screens gesetzt.

B wird nach F gebraucht, um den Cursor um die Länge des zuletzt eingegebenen Textes zurückzusetzen.

N findet das nächste Vorkommen der Kette, die mit F gefunden wurde.

X text findet und löscht das nächste Vorkommen der Kette "text".

C text setzt den text in die Cursorzeile an der Cursorposition ein.

TILL text löscht in der Cursorzeile am dem Cursor bis zum Ende von "text".

Achtung: Eingabe C ohne Text setzt eine Null in den Text an der Cursorposition. Dies führt zum sofortigen Abbruch bei einer späteren Compilation. Um diesen Fehler zu beseitigen, gibt man TOP X ein.

Befehlszusammenstellung Editor

#LAG --- Cursoradresse Länge-von-Cursor-bis-EOL

#LEAD --- Zeilenadresse Länge-bis-Cursor

-MOVE addr zeilennr. ---
bringt eine Textzeile von addr zur Zeile des Screens.

-TEXT addrl zähler addr2 --- wahrheitswert
Wahr, wenn beide Ketten ab Adressen gleich sind.

lLINE --- f
durchsucht die Cursorzeile nach einem Vergleich mit PAD-Text.

B ---
bringt den Cursor so weit wie Text in PAD zurück.

C ---
verschiebt ab Cursor und bringt den folgenden Text in die Cursorzeile.

CLEAR n ---
löscht Screen n.

D n ---
löscht Zeile n, hält sie jedoch in PAD.

DELETE n ---
löscht n Zeichen bis zum Cursor.

E n ---
füllt Zeile n mit Leerzeichen.

F ---
bringt den Cursor so viele Zeichen zurück, wie Text in PAD lang ist.

FIND ---
sucht nach einem Vergleich mit PAD-Text, ab Cursor bis Screenende. Wenn keine Gleichheit gefunden wurde, wird eine Fehlermeldung ausgegeben und der Cursor zum oberen Teil des Cursors zurückgesetzt.

FLUSH ---
schreibt alle Daten des Screens auf Cassette.

H
n ---
hält Zeile n in PAD.

I
n ---
verschiebt ab Zeile n nach unten und setzt Text von PAD auf Zeile n.

L
---
listet den aktuellen Screen.

LINE
n --- addr
hinterläßt Adresse von Zeile n. Diese Adresse liegt in der Speichergegend des Disketten-puffers.

M
n ---
verschiebt Cursor um n Zeichen und druckt Cursorzeile.

MATCH
Cursor-addr. bytes-bis-EOL text-addr text-zähler
--- tf Cursor-vorrückung-bis-ende-d.-vergleichstext
--- ff Bytes-bis-zeilenende
Vergleicht die Kette von text-addr mit der Kette ab Cursor.

N
---
sucht nächstes Vorkommen von PAD-text.

P
n ---
setzt folgenden Text in Zeile n.

R
n ---
bringt den Text von PAD in Zeile n.

R#
--- addr
Eine Benutzervariable, die die Länge zwischen Cursor und dem Anfang des Screens enthält.

S
n ---
Zeile n und folgende werden nach unten ver-schoben, Zeile n wird mit Leerzeichen gefüllt.

T
n ---
Eingabe von Zeile n und Abspeichern in PAD.

TEXT
c ---
bringt folgenden Text nach PAD. c ist die Textlänge.

TILL
---
löscht in der Cursorzeile Text vom Cursor bis zum Ende des folgenden Textes.

TOP
---
bringt den Cursor zum oberen Teil des Screens.

X
---
löscht nächstes Vorkommen von folgendem Text.

teil 2

forth

1.0 Grund-Operationen

Der leichteste Weg FORTH zu lernen ist damit zu arbeiten. Da FORTH eine interaktive Sprache ist, kann man sich hinsetzen und damit herumexperimentieren. In diesem Handbuch sind viele Beispiele, die die Möglichkeiten von FORTH zeigen.

1.1 Der Anfang

FORTH stellt sich selbst vor und zeigt an, wieviel freier Speicherplatz zur Verfügung steht. Das Zeichen ▙ ist der Cursor und er erscheint, wenn das System für Eingaben über die Tastatur bereit ist. Man kann nun ein Kommando, welches mit NEW LINE beendet wird, eingeben. So lange, bis NEW LINE gedrückt wurde, kann mit RUBOUT korrigiert werden.
Das einfachste FORTH-Kommando ist eine Leerzeile. Wenn man nun NL drückt, antwortet FORTH mit OK, da es gesehen hat, daß nichts zu tun ist, und wartet auf ein neues Kommando.

1.2 Wörter

Die Befehlseinheit wird in FORTH Wort genannt. Ein Wort besteht aus einer Reihe von Zeichen und wird mit einem Freiraum beendet. Die einzigen Bedingungen bei Wörtern sind, daß kein SPACE, kein NL, kein RUBOUT und kein Grafikzeichen darin vorkommen darf. Das Wort darf beliebig lang sein, doch nur die ersten 31 Zeichen sind signifikant.
Nach Beenden einer Textzeile mit NL unterteilt der FORTH TEXT INTERPRETER die Zeile in Wörtern, die nach der Reihenfolge der Eingabe ausgeführt werden. Jedes FORTH-Wort hat einen Namen (mit dem es aufgerufen wird) und eine Definition (was es macht).

Um ein Wort ausführen zu können, sucht der Interpreter in seinem Wörterbuch nach der Definition des Wortes. Wenn das Wort gefunden ist, wird die Definition interpretiert. Falls nicht, versucht der Interpreter das Wort in einen 16-Bit-Integer zu verwandeln. Wenn auch dies nicht geht, entweder, weil das Wort nicht nur Zahlzeichen enthält oder nicht im richtigen Zahlensystem eingegeben wurde, wird eine Fehlermeldung Ø ausgegeben. Dann meldet sich das System mit dem Cursor zurück.

Dieses Wörterbuch kann erweitert werden, indem man neue Worte definiert, die existierende Worte aufrufen (s. 1.7).

1.3 Zahlen

Zahlen können in jedem Zahlensystem von 2 - 36 ausgedrückt werden. Nach dem Laden rechnet FORTH im Dezimalsystem. Jederzeit kann mit HEX ins Hexadezimalsystem gesprungen werden und mit DECIMAL wieder zurück, man kann natürlich auch eigene Basen definieren. Grundsätzlich sollte man in einer Definition nur eine Basis verwenden, sonst kann es leicht zu Verwirrungen kommen.

Zahlen können als positive oder negative Integer dargestellt werden, positive Integer werden im Bereich von 0 - 65536 und vorzeichenbehaftete Integer im Bereich von -32768 - 32767 akzeptiert. Man kann auch doppelt genaue Zahlen im Bereich von -2147483648 - 2147483647 verwenden. Diese 32-Bit-Zahlen müssen mit einem Punkt '.' markiert werden.

Da alle Zahlen in binärer Form abgespeichert werden, kann man ganz leicht Zahlen von einem System in ein anderes umwandeln. Z.B. dezimal in hexadezimal:
DECIMAL 258 HEX . ´NL`
und als Antwort wird 102 OK erscheinen.
(Beachte: FORTH rechnet jetzt hexadezimal!)

1.4 Der Stapel

Alle Computerprogramme bearbeiten Daten durch Gebrauch von Parametern. In FORTH werden die meisten Parameter in einem PUSH DOWN-Stapel festgehalten, welcher einfach STACK genannt wird. Dieser Stapel ist nach dem LIFO (Last in First out) - Verfahren aufgebaut, d.h. das zuletzt gespeicherte Element wird als erstes Element wieder abgerufen.

Um eine Zahl auf den Stack zu geben, kann man sie als Teil der Eingabe eingeben. Das FORTH-Wort '.' holt die oberste Zahl vom Stapel und druckt sie in der aktuellen Zahlenbasis auf den Bildschirm.
Z.B.: Zahlen auf den Stapel setzen 2 4 6 8 ´NL`
Der Stack sieht nun so aus:

8
6
4
2

Wenn man nun . ´NL` eingibt, druckt FORTH 8 OK aus.
Der Stapel sieht jetzt so aus:

6
4
2

Nach der Eingabe . . . ´NL` erscheint
6 4 2 OK
Der Stapel ist nun leer.
Nach der Eingabe . ´NL` antwortet der Computer jetzt mit
. ? MSG # 1 (Stack leer)

FORTH hat auch noch einen anderen Stapel, welcher Return-Stack genannt wird. Dieser Stapel wird vom Interpreter zum Abspeichern von Return-Adressen bei der Interpretation benutzt. Die Fehlermeldung 1 wird für beide Stapel benutzt.

1.5 Arithmetik

FORTH hat einen vordefinierten Satz von Rechenoperatoren (s. Tabelle 1). Da FORTH einen Stack benutzt und mit umgekehrter polnischer Notation arbeitet, müssen die Parameter auf dem Stapel vorhanden sein, bevor der Rechenoperator ausgeführt werden soll. Um z.B. zwei Zahlen zu addieren, gibt man ein

5 27 + . ´NL` 32 OK

Die gleiche Zeile in ihre Bestandteile zerlegt:

5 Setzt das oberste Element des Stapels (TOS) auf 5

27 Setzt das oberste Element des Stapels auf 27

\+ Holt die beiden obersten Elemente vom Stapel, addiert sie und gibt die Summe auf den Stapel zurück. Beachte: Der Stapel hat nur noch 1 Element!

. Holt das oberste Element vom Stapel und druckt es: 32 OK

So verläßt man den Stapel genau so, wie man ihn vorgefunden hat.

FORTH kennt auch logische Operatoren:
Die Wahrheitswerte sind
Ø - Falsch
≠ Ø - Richtig
Die Relationsworte sind <, >, = etc.
Sie arbeiten genauso mit dem Stapel wie die Rechenoperatoren. Der Test A < B zum Beispiel wird mit A B < ausgedrückt. Die Zahlen A und B werden vom Stapel geholt, verglichen und der Wahrheitswert zurück auf den Stapel gesetzt.

1.6 Stapelveränderungen

Andere häufig benutzte Operatoren sind Operatoren, mit denen man den Stapel verändern kann. Diese Worte (s. Tabelle III) werden hauptsächlich zur Aufrechterhaltung

des Stacks, wenn er Parameter enthält, benutzt. Wenn man diese Operatoren benutzt, müssen zwei Regeln beachtet werden:

1. Unbedingt auf die Reihenfolge achten.
2. Nie mehr Elemente vom Stapel holen, als vorhanden sind.

Wenn man mit den arithmetischen und den Stack-Operatoren vertraut ist, kann man schon einige Worte definieren.
Z.B. zum Quadrieren einer Zahl:

```
: SQUARE DUP * . ; ´NL` OK
```

1.7 Definitionen

Die Stärke von FORTH liegt in der Möglichkeit, eigene Worte zu definieren.
Nehmen wir an, man müßte öfter die dritte Potenz einer Zahl ausrechnen. Es ist ganz einfach, für diese Arbeit ein Wort zu definieren:

```
: CUBE DUP DUP * * . ; ´NL` OK
```

Hier ist aufgeführt, was jedes Element bewirkt:

:	Zeigt an, daß eine Definition beginnt.
CUBE	Der Name des Wortes, das neu ins Wörterbuch eingetragen werden soll.
DUP DUP * * .	Die FORTH-Worte, mit denen das neue Wort arbeiten soll.
;	Ende einer Definition.

Nach der Eingabe kann man jederzeit die dritte Potenz ausrechnen. Z.B.:

```
4 CUBE ´NL` 64  OK
3 CUBE ´NL` 27  OK
```

Was passiert nun, wenn man das Wort CUBE benutzen will, bevor es definiert wurde?
FORTH erlaubt es nicht, es druckt dann CUBE? MSG # Ø (undefiniertes Wort - Fehler).

Glücklicherweise hat FORTH für den Programmier-Anfänger ein reiches Vokabular an vordefinierten Worten. Z.B.

? druckt den Wert der Adresse, die auf dem Stack steht (in BASIC PEEK)

? hat eine ganz einfache Definition:

```
: ? @ . ;
```

Eine andere einfache Kombination, die vordefiniert ist, ist 1+, welche 1 zum obersten Element des Stapels addiert. 1+ ist definiert mit

```
: 1+ 1 + ;
```

1.8 Modi

Der FORTH Text Interpreter arbeitet in zwei Modi: Sofort-Ausführung und Compilation. Bei der Sofortausführung wird jedes Wort aus der Eingabe-Zeile aus dem Wörterbuch herausgesucht und sofort ausgeführt.

Bei der Compilation werden die meisten Worte nicht ausgeführt, sondern nur ins Wörterbuch eingetragen. Das Wort : setzt den Interpreter in den Compilier-Modus, ; setzt ihn zurück zum Sofort-Ausführen.
Die compilierte Form der Definition besteht aus Adressen der Routinen, die ausgeführt werden sollen, wenn die Definition ausgeführt werden soll. Diese Interpretationsform ist sehr schnell. Um zwischen beiden Modi zu unterscheiden, versuche folgendes:

```
905 . ´NL` 905  OK
```

führt es sofort aus.

```
: SHOW 905 . ; ´NL` OK
```

compiliert und nichts passiert.

```
SHOW ´NL` 905  OK
```

führt das Compilierte aus.

Um FORTH schnell zu lernen, muß man mit Grund-FORTH-Worten und mit Worten, die bei Experimenten gefunden werden, arbeiten.

2.0 Datenvereinbarungen

FORTH erlaubt, auch Konstanten, Variablen und Felder zu definieren.

2.1 Konstanten

Um Namen als Konstanten zu vereinbaren, gebraucht man CONSTANT. Konstanten werden oft gebraucht, um Zahlen leichter abzurufen oder Werte, die oft gebraucht werden.
Z.B.:
5280 CONSTANT FT/MILE ´NL` OK
definiert ein neues Wort FT/MILE mit dem Wert 5280. Nachdem man FT/MILE den Wert 5280 zugewiesen hat, kann man es benutzen, wenn man 5280 auf den Stack bringen will. Z.B.:
3 FT/MILE * .
errechnet und druckt den Wert der Anzahl von Feet in drei Meilen.
Beachte: Wenn ein Wert definiert worden ist, ist es egal, in welchem Zahlensystem er abgerufen wird.

2.2 Variablen

Das FORTH-Wort VARIABLE benennt eine Stelle, deren Wert beliebig geändert werden kann.

Wenn man z.B. den Score eines Spiels festhalten und ändern können möchte, dann definiert man
0 VARIABLE SCORE ´NL` OK
↑
Anfangswert
Wenn man eine Variable mit ihrem Namen aufruft, wird ihre Adresse auf den Stack gebracht. Das Wort @ holt die Adresse vom Stack und setzt den Wert der Adresse auf den Stack. Um den Wert von SCORE auf den Stapel zu

bekommen, benutzt man
SCORE @ ´NL` OK
Um den Inhalt einer Variablen ausgedruckt zu bekommen, gibt man
SCORE ? ´NL`
ein.
FORTH antwortet mit
0 OK
Das Wort ! wird zum Abspeichern eines 16-Bit-Wertes an einem Ort verwendet. '!' gebraucht den Wert, der das 2. Element auf dem Stack ist und speichert ihn in die Adresse, welche als erstes Element auf dem Stapel steht. Um den Score jetzt z.B. auf 100 zu setzen:
100 SCORE ´NL` OK
Das Wort +! addiert einen Wert zu einer Variablen (einem Ort). Um den Score jetzt um 100 zu erhöhen:
100 SCORE +! ´NL` OK

2.3 Zahlensysteme

FORTH hat eine Benutzer-Variable, welche das laufende Zahlensystem speichert. Man kann diese Variable mit DECIMAL in 10, mit HEX in 16 ändern. Es sind aber auch alle Werte zwischen 2 und 36 (inklusive) erlaubt. Wenn man z.B. im Binärsystem arbeiten möchte, kann man dies mit 2 BASE ! ´NL` OK erreichen. Alle folgenden Werte werden nun binär ausgegeben. Man darf nur nicht vergessen, daß die Eingabe auch binär erfolgen muß.

2.4 Felder

In manchen Programmen sind Datenfelder wichtig. Wenn man z.B. 10 Variablen TØ, T1, T2 usw. hat, ist es besser, 10 Datenelemente namens T zu benutzen. Durch entsprechende Adreßberechnung kann man nun die Adresse eines bestimmten Elements ausrechnen. Dies ist flexibler zu programmieren und man spart viel Platz im Wörterbuch.

Um Platz für Felder zu reservieren, benutzt man das Wort ALLOT. Im Falle T schreibt man

Ø VARIABLE T 18 ALLOT ´NL` OK

wobei die Worte folgende Funktion haben:

Ø VARIABLE T	definiert eine Variable (2 Bytes lang) namens T
18	gibt den Wert 18 auf den Stapel
ALLOT	teilt T noch 18 Bytes zu

Um das n-te Element auszulesen, gibt man n auf den Stack und dann

2 * T + ? ´NL` (Wert) OK

Achtung: Die Elemente sind von Ø - 9 numeriert, nicht von 1 - 10.

Um einen Wert in ein n-tes Element hineinzuschreiben, benutzt man

(Wert) n 2 * T + ! ´NL` OK

2.5 Andere Speicheroperationen

In FORTH gibt es vier Worte, mit denen man Speicherbereiche ändern kann:

a) CMOVE - verschiebt eine bestimmte Menge von Zeichen (1. Element) von Adresse (3. Elem.) zu Adresse (2. Elem.). Inhalt der kleinsten Adresse wird zuerst verschoben.
Beispiel: 16396 8000 64 CMOVE ´NL` OK
verschiebt 64 Bytes von 16396 zu 8000.

b) FILL - füllt eine bestimmte Menge Bytes (2. Elem.) ab Adresse (3. Elem.) mit einem Zeichen (1. E
Beispiel: 8000 64 0 FILL ´NL` OK
setzt 64 Bytes ab 8000 auf 0.

c) ERASE - füllt einen Speicherblock mit Nullen.
Beispiel: 8000 64 ERASE ´NL` OK
löscht 64 Bytes ab 8000.

d) BLANKS - füllt einen Speicherbereich mit Leerzeichen (≡ 32 FILL)
Beispiel: 8000 64 BLANKS ´NL` OK
setzt 64 32en von 8000 ab aufwärts.

3.0 Ein- und Ausgabe

Um etwas mit dem Computer anfangen zu können, ist es notwendig, Daten einzugeben und Ergebnisse zu erhalten.

FORTH hat verschiedene Worte, die dies ermöglichen.

3.1 Zeichensatz

ZX-FORTH benutzt standard ASCII für seine Zeichencodes (Tabelle V). Dieser Code ist Standard auf den meisten anderen Computern und unterscheidet sich von Sinclairs Zeichencodes. Der Grund zur Wahl des ASCII ist, daß FORTH damit wesentlich rechnerunabhängig ist.
ZX-FORTH benutzt jedoch die Codes 16 - 25 und 128 - 191 für Grafikzeichen, welche kein ASCII-Standard sind.

3.2 Eingabe von Informationen

FORTH hat keine richtigen Eingabe-Anweisungen für Zahlen, da Zahlen ja als Parameter auf dem Stack stehen. Sie werden gewöhnlich vor Ausführung des Befehls auf den Stapel gebracht. Wenn man z.B. von der Funktionsgleichung $f(x) = 4x^3-3x+2$ eine Wertetabelle erstellen möchte, ist es günstig, erst ein Wort zur Errechnung von x^3 und eins zum Ausrechnen des Übrigen zu definieren:

```
: CUBE DUP DUP * * ; ´NL` OK
: FX DUP CUBE 4 * SWAP 3 * - 2 + . ; ´NL` OK
```

Man braucht nur noch jeweils den x-Wert und FX eingeben. Es reicht hier also, den Parameter vor dem Kommando

einzugeben:
8 CUBE ´NL` 2026 OK

FORTH hat auch ein Kommando, welches ein Zeichen vom Tastenfeld akzeptiert, nämlich KEY.

Dieses Wort ist ähnlich wie INKEY$ in BASIC, KEY wartet jedoch auf ein Zeichen, was INKEY$ ja nicht tut.

Angenommen, während einer Ausführung braucht ein Programm Eingaben vom Tastenfeld, z.B.
BRAUCHST DU INFORMATIONEN? J/N
KEY bringt nun den ASCII-Wert der gedrückten Taste auf den Stapel.

In FORTH ist kein Wort vorhanden, welches als numerische Eingabe mehr als eine Stelle erlaubt, im Anhang ist jedoch ein Wort INPUT aufgeführt, welches dies erledigt.

3.3 Ausgabe und Drucken

FORTH bietet verschiedene Wege zur Ausgabe von Informationen. Das meistbenutzte Wort ist das Wort zur zeilenweisen Ausgabe von Text. Dieses Wort heißt ." , gefolgt von dem Auszudruckenden, welches mit " beendet wird.
Beispiel:
." DIES IST EINE TEXTZEILE " druckt
DIES IST EINE TEXTZEILE auf den Bildschirm.

Jeder weitere Ausdruck wird auf der gleichen Zeile fortgesetzt. Das Wort CR (Carriage Return) entspricht NEWLINE, das heißt, alle folgenden Ausdrücke starten in der nächsten Zeile.

ZX-FORTH scrollt automatisch, wenn die Druckposition die unterste Zeile überschritten hat.

Ein weiterer Weg, ein Zeichen zu drucken ist mit dem

Wort EMIT gegeben. EMIT druckt das Zeichen, wessen ASCII-Code auf dem Stapel oben liegt. EMIT kann man gut in Verbindung mit KEY gebrauchen.

Es ist mit EMIT möglich, Zeichen zu drucken, welche nicht direkt über das Tastenfeld eingegeben werden können. Beispiel:
128 EMIT ´NL` OK
druckt das Zeichen mit dem Code 128, in diesem Fall ein "graphic space".

3.4 Zahlenausdrücke

Der einfachste Weg eine Zahl zu drucken ist . (Punkt), wie vorher schon zu ersehen war. . druckt die oberste Zahl auf dem Stack in der kleinstmöglichen Weite, also ohne führende Nullen und mit einem Leerzeichen nach der Zahl. Formatierte Ausgabe ist mit folgenden Worten möglich:

.R druckt das zweite Element als Zahl rechtsbündig in einem Feld mit der Weite des Wertes des ersten Elements, z.B.:
112 4 .R ´NL` 1120K
druckt 112 in einem Feld mit der Weite 4.

FORTH gibt auch die Möglichkeit, Zahlen zu zerlegen und Teilstücke von Zahlen zu drucken.
<# startet den Ausdruck und erwartet eine doppelt genaue Zahl auf dem Stapel.

Wenn man gewöhnlich mit normalen Zahlen arbeitet, so ist es kein Problem, diese in doppelt-genaue umzuwandeln. Das Wort S→D erledigt dies.
In dem Teilausdruck können folgende Worte verwendet werden:

setzt die nächste Dezimalstelle in den Ausgabe-Puffer, gesehen vom kleinsten Stellenwert aus, z.B. bei 112 bewirkt #, daß 2 in den Puffer gesetzt wird, das nächste # setzt die 1 hinein usw.

#S setzt die restlichen Stellen in den Ausgabepuffer.

HOLD, gebraucht als 46 HOLD, setzt das Zeichen, wessen Code 46 ist, als nächstes in den Ausgabe-Puffer.

#> beendet den Teilausdruck und hinterläßt Adresse und Länge des Ausgabe-Puffers auf dem Stapel. Ab jetzt kann diese Kette über TYPE ausgegeben werden.
Beispiel:

```
: TEIL <# # # 46 HOLD # #S #> ; ´NL` OK
```

druckt jede doppelt genaue Zahl als (... x.xx).
Beispiel:

```
11473. TEIL TYPE ´NL` 114.73 OK
    Ø. TEIL TYPE ´NL` Ø Ø.ØØ OK
```

Der Punkt hinter der Zahl daklariert sie als doppelt genaue Zahl.

Tabelle I

Arithmetische Operatoren

Wort	Beschreibung	Beispiel davor	Stack danach
+	addiert	9 6 2	9 8
-	subtrahiert	9 6 2	9 4
*	multipliziert	9 6 2	9 12
/	dividiert	9 6 2	9 3
1+	addiert 1	9 6 2	9 6 3
2+	addiert 2	9 6 2	9 6 4
ABS	Betrag	9 -6 -2	9 -6 2
MAX	erhält größeren von zwei Werten	9 6 2	9 6
MIN	erhält kleineren von zwei Werten	9 6 2	9 2
MINUS	2er Komplement	9 6 2	9 6 -2
MOD	ergibt Modulus (Divisionsrest)	9 6 2	9 0
*/	multipliziert 2. mit 3. Element, dividiert durch erstes	9 6 2	27
*/MOD	wie oben, ergibt auch Rest	9 6 2	27 0
+-	gibt dem zweiten Element das Vorzeichen des ersten	9 6 -2	9 -6
/MOD	ergibt Rest und Quotient	9 6 2	9 0 3
AND	bitweise logisches UND	9 6 3	9 2
OR	bitweise logisches ODER	9 6 3	9 7
XOR	bitweise exclusive ODER	9 6 3	9 5

Tabelle II

Vergleichsoperatoren

Wort	Beschreibung	davor	danach
<	wenn 1. Element kleiner 2. Element, logisch 1, sonst logisch Ø	9 6 2	9 0
>	wenn 1. Element größer 2. Element, logisch 1, sonst logisch Ø	9 6 2	9 1
Ø=	testet, ob 1. Element =Ø	9 6 2	9 6 0
Ø<	testet, ob 1. Element negativ	9 6 2	9 6 0
=	testet, ob 1. Element gleich 2. Element	9 6 2	9 0

Befehle zur Verarbeitung von Werten mit doppelter Genauigkeit existieren auch!
(siehe Befehlszusammenstellung.)

Tabelle III

Stapelveränderungs-Operationen

Wort	Beschreibung	davor	danach
.	druckt oberstes Element	1 2 3	1 2
DROP	entfernt oberstes Element	3 2 1	3 2
DUP	dupliziert oberstes Element	3 2 1	3 2 1 1
-DUP	dupliziert oberstes Element, wenn es ungleich Null ist	3 2 1	3 2 1 1
		3 2 0	3 2 0
OVER	kopiert 2. Element	3 2 1	3 2 1 2
ROT	verdreht oberste 3 Elemente	4 3 2 1	4 2 1 3
SWAP	vertauscht oberste beiden Elemente	3 2 1	3 1 2
R	kopiert oberstes Element des Return-Stacks auf Rechen-Stack	r20 30	r20 30
		1 2 3	1 2 3 30

Tabelle V

Zeichencodes

Ø-7	Kontrollzeichen	
8	RUBOUT	-shift Ø
9-12	Kontrollzeichen	
13	RETURN	
14	Cursor an	
15	Cursor aus	
16		
17		
18		
19		
2Ø		
21		
22		
23		
24		
25		
26-32	nicht benutzt	
33	!	-shift W
34	"	" P
35	#	" E
36	£	" U
37	$	
38	-	
39	'	" A
4Ø	(	" I
41	)	" O
42	*	" B
43	+	" K
44	,	" .
45	-	" J
46	.	
47	/	" V
48	Ø	
49	1	

5Ø	2		
51	3		
52	4		
53	5		
54	6		
55	7		
56	8		
57	9		
58	:	-shift	Z
59	;	"	X
6Ø	<	"	N
61	=	"	L
62	>	"	M
63	?	"	C
64	@	"	Q
65	A		
66	B		
67	C		
68	D		
69	E		
7Ø	F		
71	G		
72	H		
73	I		
74	J		
75	K		
76	L		
77	M		
78	N		
79	O		
8Ø	P		
81	Q		
82	R		
83	S		
84	T		
85	U		
86	V		
87	W		
88	X		

89	Y	
9Ø	Z	
91	[	-shift S
92		
93	]	" G
94	↑	
95	–	
96-127	wie 64-95 (Kleinbuchstaben)	
128		
129		
13Ø		
131		
132		
133		
134		
135		
136		
137		
138		
139	@	
14Ø	!	
141	[	
142	]	
143		
144	'	
145	_	
146	↑	-shift 7
147	↓	" 6
148	→	" 8
149	←	" 5
15Ø		" S
151		" D
152		" R
153		" T
154		" 1
155		" 2
156		" 3
157		" 4
158		" 9

159 

16Ø 

161 

162 

163 

164 

165 

166 

167 

168 

169 

17Ø 

171 

172 

173 

174 

175 

176 

177 

178 

179 

18Ø 

181 

182 

183 

184 

185 

186 

187 

188 

189 

19Ø 

191 

```
-shift Y
   "    RETURN
   "    H
```

Anhanng A - ZX-FORTH Fehler Codes

Error Code	Bedeutung
Ø	Kommando ist nicht im Wörterbuch zu finden
1	Stapel ist leer
2	Wörterbuch ist voll bzw. überfüllt
3	Falscher Adress-Modus
4	Warnung: Name schon vorhanden
7	Stack ist voll bzw. überfüllt
17	Wort muß in einer Definition benutzt werden
18	Nur zur Sofortausführung verwenden
19	Unerreichte Bedingung
2Ø	Definition noch nicht beendet
21	Im geschützten Wörterbuch
22	Nur beim Laden gebrauchen
23	Über laufendem Arbeitsschirm
24	Deklariertes Wörterbuch bzw. deklarierte Vokabel

Achtung: Falls nach einer Definition eine andere Fehlermeldung als 4 gedruckt wird, darf dieses Wort keinesfalls benutzt werden! Sofort FORGET name eingeben und neu definieren.

3.5 Andere Druckoperationen

ZX-FORTH besitzt noch 4 weitere Kommandos zur Beeinflussung des Bildschirms.

SPACE bringt ein Leerzeichen auf den Bildschirm (= 32 EMIT).

SPACES druckt eine vorgegebene Anzahl von Leerzeichen auf den Bildschirm. Beispiel:

```
5 SPACES ´NL` ____ OK
```

druckt fünf Leerzeichen.

HOME bringt die Druckposition in die linke obere Bildschirmecke. Alle weiteren Drucke erfolgen von dort aus.

CLS löscht den Bildschirm und bringt den Cursor in die linke obere Ecke des Bildschirms.

4.0 Bedingte Verzweigungen und Schleifen

Es gibt in FORTH verschiedene Möglichkeiten, bei bestimmten Bedingungen bestimmte Sachen ausführen zu lassen. FORTH hat auch Schleifenstrukturen, um eine Sequenz von Befehlen wiederholen zu können.

Achtung: Verzweigungen und Schleifen dürfen nur innerhalb von Definitionen gebraucht werden!

4.1 Bedingte Verzweigungen

Die drei Worte IF ELSE ENDIF (oder THEN) werden zum Compilieren von bedingten Verzweigungen benötigt.
Das Wort IF untersucht das oberste Element des Stapels um festzustellen, welcher Zweig gewählt wird. Eine bedingte Verzweigung hat folgende Struktur:

```
: DEFINITION bedingung IF (wahr) dies ELSE (falsch)
das THEN weiter ;
```

wobei

```
: DEFINITION eine Definition beginnt.
```

bedingung	einen Wahrheitswert auf den Stapel läßt (Null = falsch/nicht Null = wahr).
IF	holt die Nummer vom Stapel und testet sie.
dies	führt dies aus, wenn Element nicht Null (wahr).
ELSE	
das	führt das aus, wenn Element Null (falsch).
THEN	
weiter	macht nach beiden Bedingungen weiter.

IF markiert die Stelle, von wo das Element vom Stapel geholt wurde. Wenn der Wert nicht Null ist, wird alles bis ELSE ausgeführt, bei ELSE angekommen wird nach THEN gesprungen. Wenn der Wert Null ist, wird alles ab ELSE ausgeführt. Die ELSE-das Einheit ist eine Erweiterung, sie kann, wenn sie nicht benötigt wird, wegfallen.

Der Wahrheitswert ist meistens das Ergebnis eines Vergleichs, welcher die FORTH-Operatoren < > = benutzt (siehe Kapitel 1).

Zwei Wahrheitswerte Können kombiniert werden mit den Worten AND OR XOR.

AND ergibt wahr, wenn die beiden obersten Elemente des Stapels wahr sind.
OR ergibt wahr, wenn ein oder beide Elemente wahr sind.
XOR ergibt wahr, wenn ein Element wahr ist und das andere falsch.

Beispiel:
1 1 AND ergibt 1
1 0 AND ergibt 0
(siehe Wahrheitswert-Tabelle VI).

Angenommen, man möchte ein Wort definieren, welches bei einer Division einen Rest, falls vorhanden, ausdruckt:

```
: DIV MOD DUP Ø= IF . "KEIN REST" DROP ELSE . "REST" .
THEN CR ; ´NL`
```

gibt man nun z.B. 9 2 DIV ´NL` ein, antwortet FORTH
REST 1 OK.

4.2 Unbedingte Schleifen

FORTH beinhaltet auch einige Schleifenstrukturen, welche Befehle so oft wiederholen bis eine gegebene Bedingung erfüllt ist oder so oft, wie am Anfang der Schleife festgelegt wurde.
Der erste Typ ist

```
: DEFINITION BEGIN ausführung bedingung UNTIL weiter ;
```

wobei

: DEFINITION	die Definition beginnt.
BEGIN	den Anfang einer unbedinten Schleife markiert.
ausführung	das definiert, was ausgeführt werden soll.
bedingung	einen Wahrheitswert auf dem Stapel zurückläßt.
UNTIL	sich den Wert vom Stapel holt und nach BEGIN springt, wenn der Wert Null ist.
weiter	macht weiter, wenn der Wert ungleich Null ist.

END ist ein Synonym für UNTIL und kann beliebig für UNTIL benutzt werden.
Beispiel: Man möchte den Speicher nach einer gegebenen 16-Bit-Zahl durchsuchen. Im Beispiel ist ein Wort definiert, welches den Speicher ab Adresse Ø nach dem Wert Ø durchsucht:

```
: SUCHE Ø BEGIN DUP @ SWAP 1+ SWAP Ø = UNTIL 1 - . ;
```

Die Befehle zwischen BEGIN und UNTIL werden so oft wiederholt, bis der gefundene Wert Ø ist. Die Null vor dem Schleifenanfang ist die Startadresse. Der Code nach UNTIL druckt die Adresse, in der die Null gefunden wurde.

Die zweite Form der unbedingten Schleife ist

```
: DEFINITION BEGIN bedingung WHILE ausführung REPEAT
weiter ;
```

wobei

: DEFINITION	die Definition beginnt.
BEGIN	markiert den Start der unbedingten Schleife.
bedingung	läßt einen Wahrheitswert auf dem Stapel zurück.

Wenn man vorzeitig, also bevor der Endwert erreicht ist, die Schleife verlassen möchte, benutzt man LEAVE. LEAVE beendet die Schleife am nächsten LOOP oder +LOOP.

4.5 Geschachtelte Schleifen

DO ... LOOP und IF ... THEN können einander enthalten, es können auch mehrere DO ... LOOP und IF ... THEN - Strukturen in sich selbst geschachtelt sein, doch sie dürfen sich keinesfalls überlappen.

Beispiel:

Richtig: ... IF 1Ø Ø DO ... LOOP THEN

Falsch: ... IF 1Ø Ø DO ... THEN LOOP

5.0 Bandspeicherung

Normalerweise arbeitet FORTH interaktiv und immer, wenn eine Definition eingegeben wurde, gibt es keinen Weg, sie zu ändern, als, sie neu einzugeben. FORTH kann den Code aber auch in einem numerierten Bild, SCREEN genannt, abspeichern. Ein Screen besteht aus 16 Zeilen mit je 64 Zeichen. Programme werden mit dem FORTH-EDITOR im Screen abgespeichert, siehe auch EDITOR-HANDBUCH.

5.1 Abspeichern von Programmen

Wenn ein Programm auf den laufenden Screen eingegeben worden ist, ist es möglich, es auf Cassette abzuspeichern. Zuerst muß man den Computer für's Abspeichern vorbereiten (siehe entspr. Kapitel im Computer-Handbuch). Das Kommando FLUSH sagt FORTH, daß der laufende Screen auf Cassette abgespeichert werden soll. FORTH meldet sich dann mit READY CASSETTE safebereit. Man muß nun den

Recorder auf Aufnahme einschalten und NEWLINE drücken.

Ein Screen braucht 30 Sekunden zum Abspeichern. Wenn irgendeine Taste außer NL gedrückt wird, wird das Kommando sofort abgebrochen und der Cursor erscheint wieder auf dem Bildschirm.

5.2 Laden von Programmen

ZX-FORTH besitzt zwei Worte, um ein Programm vom Band zu laden. LIST wird gebraucht, um einen Screen aufzulisten, nämlich in der Form n LIST, wobei n die Screen-Nummer ist.

Wenn Screen n schon im Speicher ist, wird er sofort auf den Bildschirm gelistet. Wenn nicht, versucht FORTH, ihn von Cassette zu laden. Man muß das Band so einstellen, daß der zu ladende Screen fast anfängt, drückt NL und startet den Recorder.

Wenn der Screen korrekt geladen wurde, wird er auf dem Bildschirm gelistet. Wenn nicht, entweder durch Störungen oder weil es die falsche Screen-Nummer war, erscheint wieder READY CASSETTE. Man kann nun wieder NL oder eine andere Taste gebrauchen, genau wie bei FLUSH.

Die Variable FIRST enthält die Screen-Nummer von dem Screen, der sich gerade im Speicher befindet.

Wenn man den Ladeprozeß stoppen möchte, BREAK drücken.

LOAD wird gebraucht, um die Definitionen in einem Screen zu compilieren. Dies geschieht wie bei LIST.

5.3 Screen-Format

Jeder Screen hat eine Nummer und besteht aus 16 Zeilen

zu 64 Zeichen. Damit FORTH nun bestmöglich damit arbeiten kann, ist es notwendig, den Screen durch spezielle Worte zu begrenzen.

-> am Ende der letzten Zeile des Screens befiehlt FORTH den nächsten Screen zu LOADen, wenn dieser Screen durch LOAD vom Band geladen wurde.

Dieser Befehl wird benutzt, wenn ein Programm mehr als einen Screen an Länge braucht. ZX-FORTH compiliert den Screen und druckt dann READY CASSETTE, um den nächsten Screen zu laden.

;S am Ende der letzten Zeile beendet das LOAD-Kommando.

Dies ist beim letzten Screen des Programms unbedingt notwendig. Falls dieses Wort vergessen wird, wird das System höchstwahrscheinlich zusammenbrechen.

6.0 Andere gebräuchliche Befehle

Da FORTH einen sehr reichhaltigen Befehlssatz hat, ist es unmöglich, alle ausführlich zu erklären. Der einzige Weg, die ganzen Befehle genau zu verstehen ist, die Befehlszusammenstellung zu lesen und die Befehle auszuprobieren.

FORGET wird gebraucht, um Definitionen zu löschen und zwar in der Form von FORGET wort ´NL. Das Wort wort und alle danach eingegebenen Definitionen werden vergessen.

VLIST listet alle Wörter des Wörterbuchs. Das zuletzt eingegebene Wort wird zuerst gelistet. Das Auflisten kann jederzeit mit der BREAK-Taste gestoppt werden. (SHIFT + BREAK)

BYE verläßt ZX-FORTH und kehrt zum BASIC-Interpreter zurück.

IMMEDIATE Normalerweise, wenn ein Wort in einer Definition enthalten ist, wird es als Teil der Definition compiliert. Wenn man nun wünscht, daß ein Wort ausgeführt wird, wenn es in einer Definition enthalten ist, muß das Wort als IMMEDIATE deklariert werden, was nach der Definition eingegeben werden muß.

Beispiel:

: BSP ." JETZT WIRD COMPILIERT" ; ´NL` OK

druckt JETZT WIRD COMPILIERT immer, wenn das Wort BSP in einer Definition vorkommt und nicht compiliert ist.

[ ] und LITERAL Manchmal ist es notwendig, eine Konstante in einer Definition auszurechnen, ohne daß sie bei jeder Ausführung ausgerechnet werden muß. Das Wort] setzt den Computer wieder in den Interpretationsmodus und alles nachfolgende wird sofort ausgeführt. Das Wort [bringt den Computer zurück zum Compilieren, in eine Definition.
Das Wort LITERAL holt den obersten Wert vom Stapel und setzt es anstelle von LITERAL in die Definition.

Beispiel:

Die folgenden beiden Definitionen sind äquivalent.

: D1 3 [1 3 + 2 *] LITERAL + . ;

: D2 3 8 + . ;

Man gebraucht diese Wörter, wenn das Ergebnis einer Berechnung nicht bekannt ist und rettet den Benutzer vor dem Ausrechnen.

VOCABULARY FORTH ermöglicht es, eigene Wörterbücher zu definieren, so daß alle Worte von einem Programm zu einem Wörterbuch zusammengefaßt werden können.

Wörterbücher sollten den Zusatz IMMEDIATE tragen.
Beispiel: Es wird ein Wörterbuch namens W definiert:
VOCABULARY W IMMEDIATE

Um Definitionen in das Wörterbuch einzutragen, gibt man den Wörterbuchnamen ein, gefolgt von DEFINITIONS:
Beispiel:
W DEFINITIONS
Alle nachfolgenden Definitionen werden nun ins Wörterbuch W eingetragen bis das Wörterbuch gewechselt wird.

Falls man von einem Wörterbuch aus ein Wort aus einem anderen Wörterbuch benutzen möchte, benötigt man den Namen des Wortes und den des Wörterbuchs.

Das Grundwörterbuch ist FORTH.

FAST und SLOW Mit diesen beiden Befehlen kann von Fast auf SLOW und umgekehrt umgeschaltet werden.

MEM druckt aus, wieviel freier Speicherplatz noch zur Verfügung steht. Die Anzahl der Bytes wird im laufenden Zahlensystem gedruckt.

Anhang B: nützliche Routinen

Die hier aufgelistete INPUT-ROUTINE erwartet eine Zahl bei der Ausführung, die über die Tastatur eingegeben wird und nach NL auf dem Stapel gelassen wird.

```
: INPUT PAD 1+ 64 EXPECT .0 PAD (zahl) DROP DROP;
```

INPUT. erwartet eine doppelt genaue Zahl.

```
: INPUT. PAD 1+ 64 EXPECT .0 PAD (zahl) DROP ;
```

Hier sind Cursor-Routinen aufgelistet:
D - runter, U - rauf, V - vor, Z - zurück

```
: D 16442 C@ 3 > IF 16442 C@ 1 - 16442 C! 33 16398
+! THEN ;
: U 16442 C@ 24 < IF 16442 C@ 1+ 16442 C! -33 16398
+! THEN ;
: V 16441 C@ 1 > IF 16441 C@ 1 - 16441 C! 1 16398
+! THEN ;
: Z 16441 C@ 33 < IF 16441 C@ 1+ 16441 C! -1 16398
+! THEN ;
```

Die folgenden Routinen definieren PLOT und UNPLOT und zeigen den Gebrauch von Maschinencode in FORTH.

```
HEX
: PR 4030 C! ; CODE E1, D1 C, C5 C, 4B C, 45 C, CD C,
B2 C, C1 C, C3 C,
NEXT, SMUDGE
: PLOT 9B PR ;
: UNPLOT AØ PR ;
```

Man sollte vorsichtig sein, daß die Zeilen- und Spaltenwerte nicht außerhalb des Bildschirmformats liegen, weil FORTH sonst zurück zu BASIC springt.

Befehlszusammenstellung

Diese Zusammenstellung enthält alle Definitionen, die in ZX-FORTH existieren. Sie sind in der ASCII-Reihenfolge geordnet.

Die erste Zeile in jedem Eintrag zeigt eine symbolische Beschreibung von der Veränderung des Parameterstapels. Die Symbole zeigen die Reihenfolge der Elemente an, die drei Bindestriche --- markieren Ausführungspunkt. In dieser Notation ist das oberste Stapelelement rechts.

Die Symbole Bedeuten:

addr	Speicheradresse
b	8-bit-Byte
c	8-bit-ASCII-Zeichen
d	32-bit-Doppel-Integer
f	Wahrheitswert, 0 = falsch, sonst wahr
ff	falscher Wahrheitswert
n	16-bit-Integer mit Vorzeichen
u	16-bit-Integer ohne Vorzeichen
tf	wahrer Wahrheitswert

Die Großbuchstaben rechts zeigen folgende Eigenschaften:

C	kann nur in einer : ... ; Definition benutzt werden. Eine Zahl zeigt die Anzahl der benutzten Speicherplätze an, sonst wird nur eine Adresse benötigt.
E	nur zur Sofortausführung
LO	Ausgabe Null von FORTH 78
L1	Ausgabe Eins von FORTH 78
P	setzt Vorrang-Bit. Wird ausgeführt, wenn compiliert wird.
U	eine Benutzervariable

Wenn nicht besonders vermerkt, wird jede Rechenoperation auf 16-Bit-Integer mit Vorzeichen bezogen. Bei 32-Bit-Zahlen ist der ausschlaggebende Teil (mit dem Vorzeichen) das obere Element auf dem Stapel.

!

n addr --- LO
Speichert die 16 Bit von n an der Adresse.

!CSP

Speichert die Stapelposition in CSP. Wird benutzt zur Absicherung des Compilers.

#

d1 --- d2 LO
Erzeugt aus der Zahl d1 das nächste ASCII-Zeichen für die Ausgabekette. Das Ergebnis d2 ist der Quotient nach der Division durch BASE und ist wichtig für weitere Bearbeitung. Wird zwischen <# und #> benutzt.

#>

d --- addr anzahl LO
Beendet die numerische Ausgabe durch Wegnahme von d und Abspeichern der Textadresse und Zeichenanzahl, geeignet für die Ausgabe mit TYPE.

#S

d1 --- d2 LO
Erzeugt ASCII-Text im Textausgabepuffer durch Gebrauch von # bis eine doppelt genaue Zahl mit dem Wert Ø herauskommt. Wird zwischen <# und #> benutzt.

,

--- addr P, LO
Benutzt in der Form ,nnnn. Hinterläßt die Parameterfeldadresse des Wörterbuchs-Wort nnnn.Bei Ausführung in einer Doppelpunkt-Definition wird die Adresse als ein Literal compiliert.

(

Benutzt in der Form (cccc) P, LO
Der Kommentar, welcher mit) begrenzt wird, wird bei der Compilation ignoriert. Die geschlossene Klammer muß in der gleichen Zeile wie die geöffnete Klammer stehen. Nach der offenen Klammer muß ein Leerzeichen erfolgen.

(.")

C
Die Laufzeitprozedur, compiliert durch ."
, welches den nachfolgenden Text zur Ausgabe bringt. Siehe ."

(; CODE)

Die Laufzeitprozedur, compiliert durch
; CODE, was das Codefeld des zuletzt definierten Wortes mit nachfolgendem Maschinencode überschreibt.
Siehe ; CODE

(+LOOP)

n --- C2
Die Laufzeitprozedur, compiliert durch
+LOOP, welches den Schleifenindex um n erhöht und auf Schleifenvollständigkeit prüft.
Siehe +LOOP

(ABORT)

Führt nach dem Fehler WARNING -1 normalerweise ABORT aus, aber es kann (mit Vorsicht) zu einer vom Benutzer beliebig wählbaren Prozedur definiert werden.

(DO)

Die Laufzeitprozedur, compiliert durch DO, welches die Schleifenparameter zum Returnstapel bringt.
Siehe DO

(FIND)

addr1 addr2 --- pfa b tf (richtig)
--- ff (falsch)
Durchsucht das Wörterbuch, startend ab der Namenfeldadresse addr2, vergleicht den Text ab addr1. Läßt die Parameterfeldadresse pfa, der Bytelänge des Namenfeldes b und des wahren Wahrheitswert auf dem Stapel zurück oder nur einen falschen Wahrheitswert.

(LINE)

n1 n2 --- addr zähler
Wandelt die Zeilennummer n1 und den Screen n2 zu einer Kassettenpufferadresse, die die Daten enthält, um. Ein zähler mit dem Wert 64 weist auf die volle Zeilenlänge hin.

(LOOP)

C2
Die Laufzeitprozedur, compiliert durch LOOP, welches den Schleifenindex erhöht und auf Schleifenvollständigkeit prüft.
Siehe LOOP

(NUMBER)

d1 addr1 --- d2 addr2
Wandelt den ASCII-Text, beginnend mit addr1+1, im Hinblick auf die Basis, um. Der neue Wert wird mit der doppelt genauen Zahl d1 zusammengezählt und als d2 auf dem Stapel zurückgelassen. Benutzt von NUMBER.

*

n1 n2 --- prod LO
Hinterläßt das vorzeichenbehaftete Produkt von zwei vorzeichenbehafteten Zahlen.

*/

n1 n2 n3 --- n4 LO
Hinterläßt das Ergebnis n4=n1*n2/n3, wobei alle Zahlen vorzeichenbehaftet sind. Es wird mit einem 31-bit Zwischenprodukt gerechnet, welches größere Genauigkeit erlaubt als würde man folgendermaßen vorgehen: n1 n2 * n3 /

*/MOD

n1 n2 n3 --- n4 n5 LO
Hinterläßt den Quotienten n5 und den Rest n4 der Operation n1*n2/n3. Es wird wie bei */ mit einem 31-bit Produkt gearbeitet.

+

n1 n2 --- sum LO
Hinterläßt die Summe von n1+n2.

+! n addr --- LO
Addiert zu dem Wert der Adresse den Wert n.

+- n1 n2 --- n3
Hinterläßt n3, was aus dem Wert n1 und dem Vorzeichen von n2 besteht.

+LOOP n1 --- (Lauf)
Wird in einer Doppelpunkt-Definition in der folgenden Form gebraucht:
DO ... n1 +LOOP
In der Laufzeit kontrolliert +LOOP den Rücksprung zum anführenden DO mit der Schrittweite n1, begrenzt durch den Schleifenindex und dem Schleifenende. Das vorzeichenbehaftete n1 wird zum Schleifenindex addiert und dann mit dem Schleifenende verglichen. Der Rücksprung nach DO erfolgt, bis der neue Index gleich oder größer als das Schleifenend ist (n1>0), oder bis der neue Index gleich oder kleiner als das Schleifenende ist (n1<0
Bei LOOP wird der Parameter, falls das Ende der Schleife nicht erreicht ist, wieder abgelegt und die Ausführung wird vorne fortgesetzt.

+ORIGIN n --- addr
Hinterläßt die entsprechende Speicheradresse von n bis zum Anfang des Parametergebietes. n ist die kleinste Adresseneinheit, entweder Byte oder Wort.

, n --- LO
Speichert n in die nächste mögliche Wörterbuchspeicherzelle, den Wörterbuchzeiger vorrückend.

- n1 n2 --- diff LO
Hinterläßt die Differenz von n1-n2.

-- Fährt mit der Interpretation fort mit dem nächsten Screen.

-DUP n1 --- n1 (wenn Null) LO
n1 --- n1 n1 (ungleich Null)
Verdoppelt n1 nur, wenn es ungleich Null ist. Dies ist gewöhnlich nötig, um einen Wert vor IF zu kopieren, um den Gebrauch des ELSE-Teils, der ein Element vom Stapel nimmt, unnötig zu machen.

-FIND --- pfa b tf (gefunden)
--- ff (nicht gefunden)
Akzeptiert das nächste Textwort in die Eingabekette bei HERE und sucht das CONTEXT- und dann das CURRENT-Wörterbuch nach einem gleichen Eintrag durch. Wenn es gefunden wurde,

wird die Eingabe-Parameterfeldadresse des Wörterbuchs, die Länge und ein wahrer Wert hinterlassen. Im anderen Fall wird nur ein falscher Wahrheitswert abgelegt.

-TRAILING addr n1 --- addr n2
Berichtigt den Zeichenzähler n1 einer Textkette, welche an addr beginnt, um bei der Ausgabe nachfolgender Leerstellen zu unterdrücken, das heißt die Zeichen auf addr+n1 bis addr+n2 sind Leerstellen.

. n --- LO
Druckt die Zahl eines 16-bit-Zweierkomplement-Wertes in der laufenden Zahlenbasis aus. Ein Leerzeichen folgt.

." Gebraucht in der Form ."cccc" P, LO
Compiliert eine Zeichenkette in der Eingabezeile (die Zeichenkette wird mit " beendet), so daß sie bei der Ausführung auf dem Bildschirm ausgegeben wird. Wenn der Befehl außerhalb einer Definition eingegeben wird, druckt ." den nachfolgenden Text sofort bis zum beendenden ".

.R n1 n2 ---
Druckt die Zahl n1 rechtsbündig in einem Feld mit der Weite n2. Es wird kein nachfolgendes Leerzeichen gedruckt.

/ n1 n2 --- quot LO
Hinterläßt den vorzeichenbehafteten Quotient von n1/n2.

/MOD n1 n2 --- rest quot LO
Hinterläßt den Rest und den vorzeichenbehafteten Quotient von n1/n2. Der Rest hat das Vorzeichen des Dividenden.

Ø 1 2 3 --- n
Diese Zahlen werden so oft benutzt, daß es günstig ist, sie im Wörterbuch als Konstanten zu definieren.

Ø < n --- f LO
Hinterläßt einen wahren Wert, wenn die Zahl kleiner Null (also negativ) ist, sonst einen falschen.

Ø= n --- f LO
Hinterläßt einen wahren Wert, wenn die Zahl gleich Null ist, sonst einen falschen.

ØBRANCH f --- C2
Die Laufzeitprozedur für bedingte Zweige. Wenn f falsch (Null) ist, wird der folgende in der Zeile stehende Parameter zum Zeiger

addiert um vor oder zurück zu verzweigen. Compiliert bei IF, UNTIL und WHILE.

1+ n1 --- n2 L1
Erhöht n1 um 1.

2+ n1 --- n2
Erhöht n1 um 2.

: P, E, LO
Wird in folgender Form gebraucht und Doppelpunkt-Definition genannt:
: cccc ... ;
Nimmt einen Wörterbucheintrag vor, daß cccc äquivalent ist mit der Sequenz der folgenden FORTH-Worte ... bis zum nächsten ; oder ; CODE. Der Compilierprozess wird vom Interpreter so lange beibehalten wie STATE ungleich Null ist. Andere Details sind, daß das CONTEXT-Wörterbuch zum CURRENT-Wörterbuch gesetzt wird und daß Wörter, bei denen das Vorrangsbit (p) gesetzt ist, schneller ausgeführt als compiliert werden.

; P, C, LO
Beendet die Doppelpunkt-Definition und stoppt weitere Compilation.

; CODE Wird im Zusammenhang mit FORTH-Assembler benutzt.

;S Beendet Interpretation eines Screens. ;S ist auch das Laufzeitwort am Ende einer Doppelpunkt-Definition und bringt die Ausführung zurück zur Aufrufprozedur.

Hinterläßt einen wahren Wert, wenn n1 kleiner n2 ist, sonst einen falschen.

<# Beginnt die formatierte numerische Teilausgabe. Es können folgende Wörter folgen:
#S SIGN HOLD #>
Es wird mit einer doppelt genauen Zahl gearbeitet.

<BUILDS C, LO
Wird in einer Doppelpunkt-Definition gebraucht:
: cccc <BUILDS ... DOES> ... ;
Immer wenn cccc ausgeführt wird, definiert <BUILDS ein neues Wort mit schneller Ausführungszeit. Ausführung von cccc in der Form cccc nnnn benutzt <BUILDS, um einen Wörterbucheintrag von nnnn vorzunehmen mit einem Aufruf zum DOES> -Teil für nnnn. Wenn nnnn später ausgeführt wird, ist die Adresse seines Parametergebietes auf dem Stapel und führt die Worte nach DOES> in cccc aus.

=
n1 n2 --- f LO
Hinterläßt einen wahren Wert wenn n1=n2, sonst einen falschen.

>
n1 n2 --- f LO
Hinterläßt einen wahren Wert, wenn n1 größer als n2 ist, sonst einen falschen.

> R
n --- C, LO
Bringt eine Zahl vom Rechenstapel auf den Returnstapel. Beim Gebrauch sollte es mit R > zusammen in einer Definition verwendet werden.

?
addr --- LO
Druckt den in addr enthaltenen Wert unformatiert im laufenden Zahlensystem aus.

?COMP
Ausgabe einer Fehlermeldung, wenn nicht compiliert wird.

?CSP
Ausgabe einer Fehlermeldung, wenn sich die Stapelposition vom Wert in CSP unterscheidet.

?ERROR
f n ---
Ausgabe einer Fehlermeldung # n, wenn der Wahrheitswert wahr ist.

?EXEC
Ausgabe einer Fehlermeldung, wenn nicht ausgeführt wird.

?LOADING
Ausgabe einer Fehlermeldung, wenn nicht geladen wird.

?PAIRS
n1 n2 ---
Ausgabe einer Fehlermeldung, wenn n1 ungleich n2 ist. Die Meldung zeigt an, daß compilierte Bedingungen nicht verglichen werden.

?STACK
Ausgabe einer Fehlermeldung, wenn der Stapel über der Stapelgrenze hinaus ist.

@
addr --- n LO
Hinterläßt den 16-bit Inhalt der Adresse.

ABORT
Löscht die Stapel und geht in den Ausführungsmodus. Die Tastatureingabe ist frei, wenn eine Meldung zur Bereitschaft gedruckt wird.

ABS
n --- u LO
Hinterläßt den Betrag von n.

AGAIN
Wird in einer Doppelpunkt-Definition in folgender Form gebraucht: BEGIN ... AGAIN
Bei der Ausführung bewirkt AGAIN einen Rücksprung zum anführenden BEGIN. Am Stapel wird

nichts verändert. Diese Schleife kann programmmäßig nicht verlassen werden.

ALLOT — n --- LO
Addiert n zum Wörterbuchzeiger DP. Kann benutzt werden, um Platz im Wörterbuch zu reservieren oder Speicherplatz zu löschen.

AND — n1 n2 --- n3 LO
Hinterläßt das bitweise logische UND von n1 und n2.

B/SCR — --- n
Diese Konstante hinterläßt die Zahl der Blöcke pro Screen. Normalerweise besteht ein Screen aus 1024 Bytes, welche sich aus 16 Zeilen mit je 64 Zeichen zusammensetzen.

BACK — addr ---
Errechnet die Rückverzweigung durch Differenz von HERE zu addr und compiliert sie in die nächste freie Wörterbuchadresse.

BASE — --- addr U, LO
Eine Benutzervariable, welche die laufende Zahlenbasis, die für die numerische Ein- und Ausgabe benötigt wird, enthält.

BEGIN — Kommt in einer Doppelpunkt-Definition in folgenden Formen vor:
BEGIN ... UNTIL
BEGIN ... AGAIN
BEGIN ... WHILE ... REPEAT
Bei der Laufzeit markiert BEGIN den Anfang einer Sequenze, welche wiederholt werden soll Es dient als ein Rücksprungspunkt von UNTIL, AGAIN oder REPEAT. Wenn UNTIL ausgeführt wird, wird nach BEGIN gesprungen, wenn das oberste Stapelelement falsch ist. Bei AGAIN und REPEAT wird immer nach BEGIN gesprungen.

BL — --- c
Eine Konstante, welche den ASCII-Wert von "blank" enthält.

BLANKS — addr zähler ---
Füllt einen Speicherblock ab addr mit Leerzeichen.

BLK — --- addr U, LO
Eine Benutzervariable, welche die Nummer des Blocks, der interpretiert wird, enthält. Wenn Null, wird gerade in den Tastatureingabepuffer eingegeben.

BLOCK n --- addr LO
Hinterläßt die Speicheradresse des Blockpuffers, der Block n enthält. Wenn der Block nicht schon im Speicher ist, wird es vom Band geladen.

BRANCH C2, LO
Die Laufzeitprozedur für unbedingte Verzweigungen. Ein Wert wird zum Interpretationszeiger addiert, ob vor- oder rückverzweigt werden soll. BRANCH wird bei ELSE, AGAIN und REPEAT compiliert.

BYE Springt zurück nach BASIC.

C! b addr ---
Speichert 8 bit an addr.

C, b ---
Speichert 8 bit im nächstfolgenden Wörterbuch-Byte und erhöht den Wörterbuchzeiger.

C @ addr --- b
Hinterläßt den 8-bit Inhalt von addr.

CFA pfa --- cfa
Hinterläßt die Codefeldadresse von der Parameterfeldadresse einer Definition.

CLS Löscht den Bildschirm.

CMOVE von nach zähler ---
Bringt die in zähler festgehaltene Menge beginnend ab Adresse von nach Adresse nach.
Der Inhalt der Adresse von wird zuerst bewegt.

COLD Die Kaltstartprozedur, um den Wörterbuchzeiger auf das Minimum zurückzustellen und neu zu starten wie ABORT.

COMPILE C2
Wenn das Wort, das COMPILE enthält, ausgeführt wird, wird die Ausführungsadresse des Wortes nach COMPILE ins Wörterbuch compiliert. Dies erlaubt, spezielle Compiliersituationen und normale Compilation einfach zusammenzubringen.

CONSTANT n --- LO
Ein Definierungswort, welches in der Form
n CONSTANT cccc gebraucht wird. Das Wort cccc wird gebildet mit einem Parameterfeld, welches n enthält. Wenn cccc später ausgeführt wird, bringt es den Wert n auf den Stapel.

CONTEXT --- addr U, LO
Eine Benutzervariable, die einen Zeiger enthält, in welchem Wörterbuch zuerst gesucht werden soll.

COUNT addr1 --- addr2 byte LO
Hinterläßt die Byteadresse addr2 und den Bytezähler byte eines Textes, welcher an addr1 beginnt. Es ist anzunehmen, daß das erste Byte an addr1 den Textbytezähler enthä und daß der richtige Text an dem zweiten Byt beginnt. Normalerweise folgt TYPE nach COUNT

CR CR
Bringt ein NEWLINE zur Ausgabe.

CREATE Ein Definitionswort, daß in der Form CREATE cccc benutzt wird. Wird wie bei CODE und CONSTANT dazu benutzt, einen Kopfeintrag im Wörterbuch vorzunehmen. Das Codefeld enthält die Adresse des Wortparameterfeldes. Da neue Wort wird im CURRENT-Wörterbuch eingetragen.

CSP --- addr U
Eine Benutzervariable, welche zeitweise die Position des Stapelzeigers enthält, um Compilationsfehler zu finden.

D+ d1 d2 --- dsum
Hinterläßt die doppelt genaue Zahl sum als Summe der beiden doppelt genauen Zahlen.

D+- d1 n --- d2
d2 setzt sich zusammen aus dem Wert von d1 mit dem Vorzeichen von n.

D. d --- L1
Druckt eine vorzeichenbehaftete doppelt genaue Zahl von einem 32-bit-zweierkomplement-Wert. Die hochwertigen 16 bit sind das obere Element auf dem Stapel. Ein Leerzeichen folgt

D.R d n ---
Druckt eine vorzeichenbehaftete doppelt genaue Zahl rechtsbündig in einem Feld mit der Größe n.

DABS d --- ud
Hinterläßt den Betrag ud von d.

DECIMAL LO
Setzt die Zahlenbasis von BASE auf 10 zur dezimalen Ein- und Ausgabe.

DEFINITIONS Wird in der Form L1
cccc DEFINITIONS gebraucht. Setzt das CURRENT-Wörterbuch zum CONTEXT-Wörterbuch. Im Beispiel, die Ausführung von cccc macht das Wörterbuch zum CONTEXT-Wörterbuch und die Ausführung von DEFINITIONS bestimmt das spezielle Wörterbuch cccc.

DIGIT c n1 --- n2 tf (richtig)
c n1 --- ff (falsch)
Ändert das ASCII-Zeichen c durch Gebrauch von der Zahlenbasis n1 in seine binäre Form n2 und hinterläßt noch einen wahren Wahrheitswert. Wenn das Ändern unmöglich ist, wird nur ein falscher Wert hinterlassen.

DLITERAL d --- d (beim Ausführen) P
d --- (beim Compilieren)
Beim Compilieren wird die doppelt genaue Zahl vom Stapel anstelle von DLITERAL. Spätere Ausführung der Definition, die den Wert enthält, bringt den Wert auf den Stapel. Bei der Ausführung bleibt die Zahl auf dem Stapel.

DMINUS d1 --- d2
Ändert d1 zum Zweierkomplement d2.

DO n1 n2 --- (ausführung) F, C2, LO
Kommt in einer Doppelpunkt-Definition in folgender Form vor:
DO ... LOOP
DO ... +LOOP
Bei der Laufzeit startet DO eine Folge von wiederholten Ausführungen, welche durch das Schleifenende n1 und dem Schleifenanfang n2 kontrolliert werden. DO holt diese vom Stapel. Beim Erreichen von LOOP wird der Zähler um eins erhöht. Wenn der neue Zähler kleiner oder gleich dem Schleifenende ist, wird zurück nach DO gesprungen, sonst werden die Schleifenparameter gelöscht und es wird nach LOOP mit der Ausführung weitergemacht. Beide Parameter n1 und n2 werden erst bei der Laufzeit bestimmt und können daher das Ergebnis anderer Berechnungen sein. In einer Schleife bringt das Wort I den aktuellen Schleifenzähler auf den Stapel. Siehe I, LOOP, +LOOP, LEAVE.

DOES> Ein Wort, welches die Laufzeithandlung in einem sehr schnell definierten Wort definiert. DOES> ändert das Codefeld und das Parameterfeld des neuen Wortes um die Folge der compilierten Wortadressen nach DOES> auszuführen. Wird zusammen mit <BUILDS gebraucht. Wenn der DOES> -Teil ausgeführt wird, beginnt er mit der Adresse des ersten Parameters des neuen Wortes auf dem Stapel. Dies erlaubt die Inter-

pretation durch Gebrauch dieser Stelle oder ihrer Inhalte. Gewöhnlich wird es gebraucht für den FORTH assembler, viel-dimensionale Felder und Compilerdefinitionen.

DP --- addr U, LO
Eine Benutzervariable, der Wörterbuchzeiger, enthält die Adresse des nächsten freien Speicherplatzes über dem Wörterbuch. Dieser Wert kann mit HERE gelesen werden und mit ALLOT geändert werden.

DPL --- addr U, LO
Eine Benutzervariable, welche die Anzahl der Stellen einer Zahleneingabe enthält. Es wird auch dazu benutzt, die Spalte des Dezimalpunktes bei der Ausgabe mit Formatierung zu halten. Der Wert bei der Eingabe einer Stelle ist -1.

DROP n --- LO
Löscht das oberste Element vom Stapel.

DUP n --- n n LO
Dupliziert das oberste Element des Stapels.

ELSE Erscheint in einer Doppelpunkt-Definition in der Form
IF ... ELSE ... ENDIF
Bei der Laufzeit wird ELSE nach dem wahren IF ausgeführt. Es fordert dann das Überspringen des folgenden falschen Teils und die Ausführung nach ENDIF. ELSE verändert nichts am Stapel.

EMIT c --- LO
Bringt das ASCII-Zeichen des Wertes c zur Ausgabe.

EMPTY-BUFFERS Markiert alle Blockpuffer als leer, hat keine Einwirkung auf den Inhalt. Dies ist auch eine Initialisierungsprozedur, die vor dem ersten Gebrauch des Bandes gebraucht wird.

ENCLOSE addr c ---
addr1 n1 n2 n3
Der ursprünglich durchsuchte Text gebraucht von WORD. Von der Text Adresse addr1 und einem ASCII-Begrenzungszeichen c wird das Unterschiedsbyte zum ersten Nichtbegrenzungszeichen n1 bestimmt, die Differenz zum ersten Begrenzer nach dem Text n2 und die Differenz zum ersten nicht beinhalteten Zeichen. Diese Prozedur findet nicht nach einem ASCII "Null" statt, sie behandelt es als einen unbestimmten Begrenzer.

END P, C2, LO
Dieses Wort ist gleichbedeutend mit UNTIL.

ENDIF Erscheint in einer Doppelpunkt-Definition in der Form
IF ... ENDIF
IF ... ELSE ... ENDIF
Bei der Laufzeit dient ENDIF nur als Bestimmungsort einer Vorwärtsverzweigung von IF oder ELSE. ENDIF markiert das Ende einer Bedingungsstruktur. THEN ist ein anderer Name für ENDIF. Beide Namen sind im ZX-FORTH vorhanden. Siehe auch IF und ELSE.

ERASE addr n ---
Löscht eine Region des Speichers zu Null, und zwar von addr n Bytes weit.

ERROR line --- in blk
Führt Fehlermeldung aus und startet das System neu. Zuerst wird WARNING untersucht. Wenn 1, wird der Text der Zeile n, bezogen auf Screen 4 des Laufwerks Ø gedruckt. Diese Zeilennummer kann positiv oder negativ sein, über Screen 4 hinaus sein. Wenn WARNING Ø ist, wird n als Meldungsnummer gedruckt (keine Diskette angeschlossen). Wenn WARNING -1 ist, wird die Definition (ABORT) ausgeführt, welche ABORT ausführt. Der Benutzer kann vorsichtig diese Ausführung abändern, indem (ABORT) geändert wird. ZX-FORTH sichert die Inhalte von IN und BLK, um beim Bestimmen des Ortes, wo der Fehler auftrat, zu helfen. Die letzte Handlung ist die Ausführung von QUIT.

EXECUTE addr ---
Führt die Definition aus, deren Adresse sich auf dem Stapel befindet. Die Codefeldadresse wird auch Compilationsadresse genannt.

EXPECT addr zähler ---
Bringt Zeichen von der Tastatur zur Adresse, bis ein NEWLINE eingegeben wurde oder die in zähler stehende Anzahl der Zeichen erreicht ist. Eine oder mehrere Nullen werden hinter den Text gesetzt.

FAST Bringt den ZX81 in den Fast-Modus.

FENCE --- addr U
Eine Benutzervariable, welche die Adresse enthält, wo FORGET erfolgte. Um unter diesem Punkt zu vergessen, muß der Benutzer den Inhalt von FENCE ändern.

FILL addr menge b ---
Füllt Speicherplätze ab der Adresse menge weit mit dem Byte b.

FIRST --- n
Eine Konstante, welche die Adresse des Blockpuffers hinterläßt.

FORGET E, LO
Wird in der Form FORGET cccc ausgeführt. Löscht die Definition mit dem Namen cccc aus dem Wörterbuch mit allen nachfolgenden Einträgen. In ZX-FORTH erfolgt eine Fehlermeldung, wenn die CURRENT- und CONTEXT-Wörterbücher zur Zeit nicht die gleichen sind.

FORTH P, L1
Der Name des Elementarwörterbuchs.
Die Ausführung macht FORTH zum CONTEXT-Wörterbuch. Bis weitere Benutzerwörterbücher definiert sind, werden neue Definitionen Teil des FORTH-Wörterbuchs. FORTH ist sofortausführend, so daß es möglich ist, während des Erschaffens einer Doppelpunkt-Definition dieses Wörterbuch bei der Compilierung anzuwählen.

HERE --- addr LO
Hinterläßt die Adresse der nächstfolgenden freien Wörterbuchadresse.

HEX LO
Setzt BASE auf 16 (hexadezimal).

HLD --- addr LO
Eine Benutzervariable, welche die Adresse des letzten Zeichens eines Textes bei einer numerischen Ausgabe enthält.

HOLD c --- LO
Wird zwischen < # und # > gebraucht, um ein ASCII-Zeichen in eine zerlegte numerische Ausgabekette einzusetzen, z.B. 46 HOLD setzt einen Dezimalpunkt.

HOME Bringt den Cursor in die linke obere Ecke des Bildschirms.

I --- n C, LO
Wird in einer DO-LOOP-Schleife gebraucht, um den Schleifenindex auf den Stapel zu kopieren. Anderer Gebrauch ist ausführungsabhängig. Siehe R.

ID. addr ---
Druckt den Definitionsnamen von seiner Normalfeldadresse.

IF f --- (laufzeit) P, C2, LO
Kommt in Doppelpunkt-Definition in folgender Form vor:

IF (wahr) ... ENDIF
IF (wahr) ... ELSE (falsch) ... ENDIF
Bei der Laufzeit wählt IF zwischen Ausführungen, wobei die Wahl von einem Wahrheitswert abhängig ist. Wenn f falsch ist, wird der nächste Befehl nach ELSE ausgeführt und dort weitergemacht. Nach beiden Möglichkeiten wird nach ENDIF die Ausführung fortgesetzt. ELSE und der falsche Teil sind nicht erforderlich; bei Nichtvorhandensein springt die Ausführung bei falschem Wahrheitswert nach ENDIF.

IMMEDIATE — Markiert die zuletzt eingegebene Definition, so daß, wenn sie in einem zu compilierenden Teil steht und compiliert wird, sie schneller ausgeführt und nicht compiliert wird, d.h. daß das Vorrangbit im Kopf der Definition gesetzt ist. Dies ermöglicht den Gebrauch von ungewöhnlichen Compilationssituationen besser als wenn man es mit in den Compiler eindefiniert hätte. Der Benutzer kann auch IMMEDIATE-Definitionen compilieren lassen, indem er sie mit [COMPILE] bezeichnet.

IN — --- addr LO
Eine Benutzervariable, welche den Byterest im Eingabetextpuffer (Tastatur oder Diskette), von dem der nächste Text geholt wird, enthält. Der Gebrauch von WORD bringt den Wert von IN.

INTERPRET — Der äußere Textinterpreter, welcher entweder den Text der Eingabekette (Tastatur oder Diskette) ausführt oder compiliert, abhängig von STATE. Wenn der Wortname nach einer Suche im CONTEXT und dann im CURRENT-Wörterbuch nicht gefunden wurde, wird versucht, das Wort in eine Nummer umzuwandeln. Geht auch dies nicht, wird eine Fehlermeldung ausgeworfen in der Form Wortname?.
Texteingabe erfolgt in Übereinstimmung mit WORD. Wenn ein Dezimalpunkt als Teil einer Zahl gefunden wird, wird ein doppelt genauer Wert auf den Stapel gesetzt. Der Dezimalpunkt hat keine andere Funktion als diese. Siehe NUMBER.

KEY — --- c LO
Hinterläßt den ASCII-Code des nächsten Tastendrucks.

LATEST — --- addr
Hinterläßt die Namenfeldadresse des obersten Wortes im CURRENT-Wörterbuch.

LEAVE — Fordert Beendung einer DO-LOOP-Schleife bei der nächsten Gelegenheit durch Setzen des Schleifenendes auf den Wert des Index. Der Index selbst bleibt unverändert und die Ausführung erfolgt normal weiter bis LOOP oder +LOOP ausgeführt wird.

LFA

pfa --- lfa
Ändert die Parameterfeldadresse eines Wörterbucheintrages zu einer Verbindungsfeldadresse

LIMIT

--- n
Eine Konstante, welche die Adresse, welche direkt nach der höchsten Adresse des Bandpuffers kommt. Gewöhnlich ist dies die höchst Systemadresse.

LIST

n --- LO
Druckt den ASCII-Text des Screen n auf dem Bildschirm. SCR enthält die Screennummer während und nach diesem Prozess.

LIT

--- n C2, LO
In einer Doppelpunkt-Definition wird LIT automatisch vor jedem Vorkommen eines 16-Bit-Literals im Eingabetext compiliert. Spätere Ausführung von LIT veranlaßt, den Inhalt der nächsten Wörterbuchadresse auf den Stapel zu bringen.

LITERAL

n --- (compilieren) P, C2, LO
Bei der Compilation wird der oberste Stapelwert als 16-bit-Literal compiliert. Diese Definition ist sofortausführend, so daß sie ausgeführt wird, wenn sie in einer Doppelpunkt-Definition vorkommt.
Der gewöhnliche Gebrauch ist:
: xxx (Berechnung) LITERAL ;
Die Compilation ist für die Zeit der Berechnung aufgehoben. Bei der Weitercompilation wird das Ergebnis der Berechnung durch LITERAL compiliert.

LOAD

n --> LO
Startet die Interpretation des Screens n. Das Laden wird mit ;S beendet.
Siehe ;S und --

LOOP

Kommt in einer Doppelpunkt-Definition in der Form DO ... LOOP vor.
Bei der Laufzeit wird aufgrund des Schleifenendes und des Index zurückgesprungen zum anführenden DO oder mit der Ausführung nach LOO fortgesetzt. Der Schleifenindex wird um eins erhöht und mit dem Ende verglichen. Der Sprun zurück zum DO erfolgt so lange, bis der Index größer oder gleich dem Ende ist. Ist dies der Fall, werden die Parameter gelöscht und die normale Ausführung geht weiter.

M*

n1 n2 --- d
Ein gemischter-Größen mathematischer Operator welcher aus zwei vorzeichenbehafteten Zahlen das Produkt als doppelt genaue Zahl auf dem Stapel ablegt.

M/ d n1 --- n2 n3
Ein gemischter-Größen mathematischer Operator, welcher den Rest n2 und den Quotienten n3 eines doppelt genauen Dividenden und dem Divisor n1 hinterläßt. Der Rest hat das Vorzeichen des Dividenden.

M/MOD ud1 u2 --- u3 ud4
Ein positiver gemischter-Größen mathematischer Operator, welcher von dem doppelt genauen Dividenden ud1 und dem Divisor u2 den Rest u3 und den doppelt genauen Quotienten ud4 hinterläßt.

MAX n1 n2 --- max LO
Hinterläßt die größere der beiden Zahlen.

MESSAGE n .....
Druckt ?MSG # n

MIN n1 n2 --- min LO
Hinterläßt die kleinere der beiden Zahlen.

MINUS n1 --- n2 LO
Hinterläßt das Zweierkomplement einer Zahl.

MOD n1 n2 --- rest LO
Hinterläßt den Rest von n1/n2 mit dem Vorzeichen von n1.

NEXT Wird mit FORTH-Assembler gebraucht.

NFA pfa --- nfa
Ändert die Parameterfeldadresse einer Definition in die Namenfeldadresse.

NUMBER addr --- d
Ändert eine Zeichenkette ab addr mit einem vorgegebenen Zähler in eine vorzeichenbehaftete doppelt genaue Zahl. Wenn ein Dezimalpunkt im Text vorkommt, steht seine Position in DPL und nichts anderes passiert. Wenn die Umwandlung unmöglich ist, wird eine Fehlermeldung ausgegeben.

OR n1 n2 --- oder LO
Hinterläßt das bitweise logische ODER zweier 16-bit Werte.

OUT --- addr U
Eine Benutzervariable, welche den erhöhten Wert von EMIT enthält. Der Benutzer kann OUT ändern und untersuchen, um die Bildschirmformatierung zu steuern.

OVER n1 n2 --- n1 n2 n1 LO
Kopiert den zweiten Stapelwert zur Spitze.

PAD --- addr LO
Hinterläßt die Adresse des Texausgabepuffers welche eine feste Differenz zu HERE ist.

PFA nfa --- pfa
Ändert die Namenfeldadresse einer Definition in die Parameterfeldadresse.

QUERY Ermöglicht die Eingabe von 80 Zeichen (oder bis ein NEWLINE erfolgt) von der Tastatur. Der Text wird an der Adresse, welche in TIB gespeichert ist, festgehalten und IN wird auf Null gesetzt.

QUIT Löscht den Returnstapel, stoppt die Compilation und gibt die Kontrolle zurück an die Tastatur. Keine Meldung wird ausgegeben.

R --- n
Kopiert das oberste Element des Returnstapels zum Rechenstapel.

R# --- addr U
Eine Benutzervariable, welche die Position des Editor-Cursors enthalten kann oder andere Örtlichkeiten.

R/W addr blk f ---
Der fig-FORTH Standard Schreib/Lese Befehl. addr zeigt auf den Quellblockpuffer, blk ist die aufeinanderfolgende Nummer des bezogenen Blocks und f ist ein Flag für 0=Schreiben und 1=Lesen. R/W bestimmt den Ort der Massenspeicherung, führt Lesen/Schreiben aus und die Fehlerkontrollierung.

R > --- n LO
Bringt den obersten Wert des Returnstapels auf den Rechenstapel. Siehe > R und R

RØ --- addr U
Eine Benutzervariable, welche den Anfangsort des Returnstapels enthält. Siehe RP!

REPEAT P, C2
Wird in einer Doppelpunkt-Definition in folgender Form gebraucht:
BEGIN ... WHILE ... REPEAT
Bei der Laufzeit fordert REPEAT den unbedingten Sprung zurück, direkt nach dem anführenden BEGIN.

RSMUDGE Setzt das SMUDGE-bit des letzten Eintrags des Wörterbuchs zurück.

ROT n1 n2 n3 --- n2 n3 n1 LO
Verdreht die obersten drei Werte des Stapels, bringt den 3. Wert nach oben.

RP!
Eine vom Computer abhängige Prozedur, um dem Returnstapelzeiger der Benutzervariable RØ zu initialisieren.

SLOW
Bringt den ZX81 in den Slow-Modus.

S→D
n --- d
Wird zum Umformen einer normalen Zahl in eine doppelt genaue gebraucht.

SØ
--- addr U
Eine Benutzervariable, welche den Anfangswert des Stapelzeigers enthält. Siehe SP!

SCR
--- addr U
Eine Benutzervariable, welche die zuletzt durch LIST aufgerufene Screennummer enthält.

SIGN
n d --- d LO
Speichert ein ASCII "-" -Zeichen direkt vor der numerischen Ausgabekette des Textausgabepuffers, wenn n negativ ist. n wird weggenommen, d wird beibehalten. Muß zwischen <# und #> gebraucht werden.

SMUDGE
Wird während einer Wortdefinition benutzt, um das "SMUDGE-bit" in dem Definitionsnamenfeld zu setzen. Dies verhütet unvollständige Definitionseinträge bis die Definition ohne Fehler compiliert wurde.

SP!
Eine vom Computer abhängige Prozedur, um den in SØ enthaltenen Stapelzeiger zu initialisieren.

SP
--- addr
Eine vom Computer abhängige Prozedur, um die Adresse der Stapelposition auf die Spitze des Stapels, wo sie bevor SP ausgeführt wurde, war, zurückzusetzen.

SPACE
LO
Bringt ein ASCII-Leerzeichen zur Druckausgabe.

SPACES
n --- LO
Bringt n ASCII-Leerzeichen zur Druckausgabe.

STATE
--- addr LO, U
Eine Benutzervariable, welche den Compilationsstatus enthält. Ein Wert ungleich Null zeigt Compilation an.

SWAP
n1 n2 --- n2 n1 LO
Vertauscht die obersten beiden Werte des Stapels.

TASK

Ein nichts bewirkendes Wort, welches zum Markieren der Grenze zwischen Bedeutungen genommen werden kann. Wenn TASK "forget"tet wird und zurückcompiliert wird, kann eine andere Bedeutung an seinem Platz eingesetzt werden.

THEN P, CO, LO

Gleichbedeutend mit ENDIF.

TIB --- addr U

Eine Benutzervariable, welche die Adresse des Tastatureingabepuffers enthält.

TOGGLE addr b ---

Vervollständigt die Inhalte von addr durch das Bitmuster b.

TRAVERSE addr1 n --- addr2

Bringt über das Namenfeld einer ZX-FORTH-Variable die Länge des Namenfeldes. addr1 ist die Adresse, entweder des Längenbytes oder die des letzten Buchstabens. Wenn n=1 ist die Bewegung in Richtung höherem Speicher wenn n=-1 in Richtung tieferem Speicher. Die addr2 ist die Adresse des anderen Endes des Namens.

TYPE addr count --- LO

Bringt count Zeichen von addr zur Bildschirmausgabe.

U* u1 u2 --- ud

Hinterläßt eine positive doppelt genaue Zahl von zwei Zahlen ohne Vorzeichen.

U/ ud u1 --- u2 u3

Hinterläßt den Rest u2 und den Quotienten u3 von dem doppelt genauen Dividenden und dem Divisor n1.

UNTIL f --- (laufzeit)

Erscheint in einer Doppelpunkt-Definition in der Form:
BEGIN ... UNTIL
Bei der Laufzeit steuert UNTIL die Rückverzweigung zum anführenden BEGIN. Wenn f falsch ist, wird nach BEGINN zurückgesprungen; wenn f wahr ist, wird nach BEGIN weiter ausgeführt.

USER n --- LO

Ein Definierungswort, welches in der Form
n USER cccc
gebraucht wird. Dies definiert eine Benutzervariable namens cccc. Das Parameterfeld von cccc beinhaltet n als bestimmte Differenz zum Benutzerzeigerregister UP dieser Benutzervariable. Wenn cccc später ausgeführt wird,

bringt es die Summe der Differenz und der Benutzergegendadresse auf den Stapel als Speicheradresse dieser Variable.

VARIABLE E, L1

Ein Definierungswort, welches in der Form
n VARIABLE cccc
gebraucht wird. Wenn VARIABLE ausgeführt wird, erstellt es die Definition cccc mit dem Parameterfeld, in dem n steht. Wenn cccc später ausgeführt wird, wird die Adresse des Parameterfeldes auf dem Stapel abgelegt, so daß sich ein Auslesen oder Abspeichern auf diese Adresse bezieht.

VOC-LINK --- addr U

Eine Benutzervariable, welche die Adresse eines Feldes in der Definition eines zuletzt definierten Wörterbuchs enthält. Alle Wörterbuchnamen sind mit diesen Feldern verbunden, um das Vergessen durch vielfache Wörterbücher zu erlauben.

VOCABULARY E, L

Ein Definitionswort, gebraucht in der Form
VOCABULARY cccc,
um ein Wörterbuch namens cccc zu schaffen. Späterer Gebrauch von cccc macht es zum CONTEXT-Wörterbuch, welches durch INTERPRET zuerst durchsucht wird. Die Folge "cccc DEFINITIONS" macht cccc zum CURRENT-Wörterbuch, so daß neue Definitionen in dieses Wörterbuch eingetragen werden. In ZX-FORTH wird cccc so zusammengekettet, daß alle Befehle in dem Wörterbuch, in dem cccc selbst enthalten ist, enthalten sind. Alle Wörterbücher werden zuletzt an FORTH gehängt. Normalerweise werden Wörterbuchnamen als IMMEDIATE deklariert.

VLIST Listet alle Namen der Definitionen im CONTEXT-Wörterbuch.

WARNING Muß Ø sein, wenn kein Diskettenlaufwerk angeschlossen ist.

WHILE f --- (laufzeit) P, C2

Erscheint in einer Doppelpunkt-Definition in der Form:
BEGIN ... WHILE ... REPEAT
Bei der Laufzeit wählt WHILE zwischen bedingten Ausführungen, welche durch f bedingt sind. Wenn f wahr ist, führt WHILE den Teil bis REPEAT, von wo wieder nach BEGIN gesprungen wird, durch. Wenn f falsch ist, wird direkt nach REPEAT gesprungen und die Struktur verlassen.

WIDTH — Maximale Länge eines Wortnamens, 31 in ZX-FORTH.

WORD — c ---
Liest die nächsten Zeichen der Eingabekette, welche ausgeführt wird, bis der Begrenzer c gefunden ist. Dann wird diese Zeichenkette, beginnend ab dem Wörterbuchpuffer HERE, abgespeichert. WORD hinterläßt den Zeichenzähler in den ersten beiden oder mehr Leerzeichen. Vorherige Vorkommen von c werden ignoriert. Wenn BLK null ist, wird der Text vom Tastatureingabepuffer geholt, sonst vom Block, auf den BLK zeigt. Siehe BLK, IN.

XOR — n1 n2 --- exclus.oder L1
Hinterläßt das bitweise logische exclusiv-Oder der beiden Zahlen n1 und n2.

[— P, L1
Wird in einer Doppelpunkt-Definition in der Form:
: xxx wörter weiter ;
gebraucht. Verläßt Compilation. Die Wörter nach [werden ausgeführt, nicht compiliert. Dies erlaubt das Ausrechnen von Ergebnissen bevor mit] weitercompiliert wird. Siehe LITERAL,] .

[COMPILE] — P, C
Gebraucht in einer Doppelpunkt-Definition in der Form
: xxx [COMPILE] FORTH ;
[COMPILE] fordert die Compilation einer als IMMEDIATE deklarierten Definition, sonst würde sie während der Compilation ja nur ausgeführt.

] — L1
Setzt die Compilation fort, zum Beenden einer Ausführungsphase. Siehe [.