

Glossar

Glossarium des Z80 FORTH 83 der IG Forth

Der Aufbau einer Wortbeschreibung im Glossarium:

Wortname – M, „ausprache“

ausfuehrliche Beschreibung des Wortes Wortname

Dabei bedeutet Wortname das jeweilige FORTH-Wort und „ausprache“ gibt die englische Sprechweise desselben an, wenn diese nicht aus dem Wort selbst hervorgeht.

– entspricht der Stackbilanz des Wortes. Links von – stehen die Stackparameter (Angaben ueber den Stack) vor und rechts nach Abarbeitung des Wortes. Dabei ist zu beachten, dass auf der linken und rechten Seite von – der am weitesten rechts stehende Wert die Datenstackspitze darstellt.

Die jeweiligen Wortnamen sind durch Fettdruck hervorgehoben. Die in den Erlaeuterungen der Worte ebenfalls durch Fettdruck her- vorgehobenen Woerter verweisen auf andere Definitionen, die gleichfalls im Glossarium aufgefuehrt und beschrieben sind.

Die letzte Positon der Kopfzeile gibt weitere Informationen ueber die Worte:

I	Immediate-Wort
C	nur im Compilemode benutzen
E	nur im Executemode
M	diese Worte koennen Multitask-Applikationen beeinflussen.
U	User-Variable
Voc	Vokabular
Vec	Vektor, der auf andere Worte zeigen kann.

Fals das Wort nicht im Grundvokabular FORTH enthalten ist, erfolgt die Angabe des entsprechenden Woerterbuches.

Erlaeuterung der Stackparameter:

Stackabkuerzung	Zahlentyp	Zahlenbereich	min. Feldweite (Bit)
flag	Boolsche Groesse	0=falsch, sonst wahr	16
wahr	Boolsche Groesse	-1 (als Ergebnis)	16
falsch	Boolsche Groesse	0	16
b	Bit	0 ... 1	1
char	Zeichen	0 ... 127	7
8b	8 beliebige Bits (Byte)	nicht angebbbar	8
16b	16 beliebige Bits	nicht angebbbar	16
n	Zahl (gewichtete Bits)	-32768 ... 32767	16
+n	positive Zahl	0 ... 32767	16
u	vorzeichenlose Zahl	0 ... 65535	16
w	nichtspezifizierte Zahl (n oder u)	-32768 ... 65535	16

Stackabkuerzung	Zahlentyp	Zahlenbereich	min. Feldweite (Bit)
adr	Adresse (gleich u)	0 ... 65535	16
32b	32 beliebige Bits	nicht angebbar	32
d	doppelt genaue Zahl	-2147483648 ... 2147483647	32
+d	positive doppelt genaue Zahl	0 ... 2147483647	32
ud	vorzeichenlose doppelt genaue Zahl	0 ... 4294967295	32
wd	nichtspezifizierte doppelt genaue Zahl (d oder u)	-2147483648 ... 4294967295	32
sys	0, 1 oder mehrere systemabhaengige Stackeintragungen	nicht angebbar	nicht angebbar

! 16b adr -- "store"
16b wird in adr abgespeichert

!CSP -- "store-c-s-p"
Mit diesem Befehl wird der Inhalt des Datenstackzeigers in CSP gerettet. Er wird zur Erhoehung der Uebersetzungssicherheit verwendet.

" --
Compiliert in :-Definitionen einen String ins Worterbuch mit der Laufzeitroutine ("). Waehrend Runtime werden Adresse und Laenge des Strings auf den Stack hinterlegt.

+d1 -- +d2 "sharp"
Ausgabekonvertierungsschritt (+d1 / Zahlenbasis), Ergebnis +d2; aus Divisionsrest generiertes Zeichen wird vor dem ersten Zeichen der Ausgabekette angefuegt.

#> 32b -- adr +n "sharp-greater"
Beendigung der Ausgabekonvertierung, Entfernen des 32b-Wertes. adr ist Adresse, +n Laenge der Zeichenkette (Format fuer TYPE).

#LINE -- adr U "sharp-line"
Ist eine System-Variable. Der Inhalt der Adresse (adr) entspricht der Anzahl der CR, die seit dem letzten Ruecksetzen ausgesandt wurden.

#OUT -- adr U
Ist eine Systemvariable, die einen Wert enthaelt, der durch EMIT inkrementiert wird. Der Benutzer kann #OUT auslesen oder veraendern, um die Display-Formatierung zu steuern.

#S +d -- 0 0 "sharp-s"

Folge von Konvertierungsschritten an +d, Generierung der Ausgabezeichenkette so lange, bis bei #-Teilschritt 0 entsteht. Fuer +d = 0 Uebernahme einer einzelnen 0 in Ausgabezeichenkette. Vgl. <# #>

#TIB -- adr "number-t-i-b"

Adresse der Variablen, die die aktuelle Byteanzahl im Texteingabepuffer enthaelt.

#THREADS -- n "sharp-threads"

Ist eine System-Konstante (Anzahl der Hash-Faeden). Vgl. HASH

#USER -- addr

Systemvariable, die den Offset zum ersten unbelegten Feld im USER-Area enthaelt. Sie wird bei Definition von USER-Variablen und -Vektoren automatisch nachgestellt.

#VOCS -- n "sharp-vocs"

Ist eine System-Konstante, maximale Anzahl der Woerterbuecher, die durch CONTEXT in der Reihenfolge festgelegt und durchsucht werden (siehe auch ORDER).

' -- adr M, "tick"

Mit ' Wortname wird die Compilationsadresse (CFA) von Wortname ermittelt. Fehler: Wortname kann in der aktuellen Suchreihenfolge nicht gefunden werden.

'RAM n -- adr

Vok.: I/O

rechnet die Blocknummer n in die Adresse des Blockes in der RAM-Disk um. Prueft dabei, ob der Block ueberhaupt verfuegbar ist; sonst ABORT.

'TIB -- adr "tick-t-i-b"

Ist eine Variable, welche die Anfangsadresse des TIB enthaelt. Durch Aenderung des Inhalts von 'TIB kann der Anfang des TIB beliebig veraendert werden.

(-- I, M, "paren"

Kommentarzeichen. Mit (ccc) wird der Kommentar ccc eingegeben, Abschluss durch). Keine Verarbeitung von ccc; Das Leerzeichen nach (zaehlt nicht zum Text.

(.) n -- a c

Ist die Kern-Routine von ., Konvertierung von n als String in PAD.

(.) --

Vok.: HIDDEN

Die durch ." compilierte Laufzeit-Routine, uebertraegt den

nachfolgenden Text bis " auf die Konsole.

(") -- a c
Vok.: HIDDEN
Die durch " compilierte Laufzeit-Routine, hinterlaeszt
Adresse und Count des folgenden Textes auf dem Datenstack.

(+LOOP) n --
Vok.: HIDDEN
Ist die Laufzeit-Routine von +LOOP (siehe dort).

(;CODE) --
Vok.: HIDDEN
Ist die durch ;CODE (siehe dort) compilierte Laufzeit-Rou-
tine.

(;USES) --
Vok.: HIDDEN
Ist die durch ;USES (siehe dort) complilierte Laufzeit-
Routine.

(?DO) n1 n2 --
Vok.: HIDDEN
Ist die Laufzeit-Routine von ?DO.
Sind beide Zahlen gleich, so wird die Schleife uebersprungen.
Im anderen Fall beinhaltet sie die gleiche Funktion wie (DO)

(?LEAVE) flag --
Vok.: HIDDEN
Ist die Laufzeit-Routine von ?LEAVE.

(ABORT") flag --
Vok.: HIDDEN
Ist die Laufzeit-Routine von ABORT" (siehe dort).

(CONSOLE) c -- I/O
Vok. I/O
gibt das Zeichen c nur auf dem Bildschirm aus.

(D.) d -- a c
Ist die Kern-Routine von D..Konvertierung von d in einen
String in PAD.

(DO) w1 w2 --
Vok.: HIDDEN
Ist die Laufzeit-Routine von DO (siehe dort).

(EMIT) c -- I/O
Vok.: I/O
gibt Zeichen auf Bildschirm und bei PRINTING ON auch auf dem
Drucker aus. Standartvoreinstellung fuer EMIT.

(FIND) adr1 -- adr2 flag
Vok.: HIDDEN
Ist die Laufzeit-Routine von FIND (siehe dort).

(FORGET) adr --
Vok.: HIDDEN
Vergisst Teil des Woerterbuches ab spezifizierter Adresse.

(IS) adr -- (comp.Mod.)
Ist die Laufzeit-Routine von IS (siehe dort).
Schreibt die cfa des folgenden DEFER-ierten Wortes (Vector)
in die Adresse (adr).

(LEAVE) --
Vok.: HIDDEN
Ist die Laufzeit-Routine von LEAVE (siehe dort).

(LIT) n --
Vok.: HIDDEN
Wird innerhalb einer Doppelpunktdefinition automatisch vor
jede 16-Bit-Zahl n, die im Eingabetext bereitgestellt wird,
compiliert. Bei der Ausfuehrung von (LIT) wird der Inhalt der
auf (LIT) folgenden Woerterbuchadresse auf den Datenstack
gelegt.

(LOOP) --
Vok.: HIDDEN
Ist die Laufzeit-Routine von LOOP (siehe dort).

(KEY) -- c I/O
Vok.: I/O
holt ein Zeichen von der Tastatur. Standartvoreinstellung
fuer KEY.

(KEY?) -- f I/O
Vok.: I/O
Prueft den Status der Tastatur. f=TRUE, wenn eine Taste
gedrueckt war. Standartvoreinstellung fuer KEY?.

(NUMBER?) adr -- d f
Vok.: HIDDEN
wandelt einen Zifferstring in adr+1 in eine doppeltgenaue
Integer-Zahl, die ein fuehrendes Minuszeichen haben darf. Der
String muss mit BL abgeschlossen sein. Als Decimalpunkt,
dessen Position in DPL uebergeben wird, gelten: .-,/
Das Flag teilt mit, ob eine konvertierbare Zahl vorlag.

(PRINT) c -- I/O
Vok.: I/O
gibt das Zeichen nur auf den Drucker aus.

(U.) u -- a c
Ist die Kern-Routine von U.. Konvertiert u in einen String in

PAD.

* w1 w2 -- w3 "star"
w3 = niederwertige 16 Bit des arithmetischen Produkts $w1 * w2$

*/ n1 n2 n3 -- n4 "times-divide"
Berechnung $(n1 * n2)$ mit 32-Bit-Ergebnis, anschl. Division durch $n3$. $n4$ ist der Quotient entsprechend den Rundungsregeln. Fehler: Division durch 0 oder $n4$ ausserhalb des Bereichs $-32768...+32767$.

*/MOD n1 n2 n3 -- n4 n5 "times-divide-mod"
Funktion wie bei */, jedoch zusaetzlich Uebergabe des Divisionsrests $n5$ (gleiches Vorzeichen wie $n4$ bzw. gleich 0).

*D n1 n2 -- d "star-d"
Multipliziert zwei 16-bit-Zahlen ($n1$ und $n2$) und hinterlaesst auf dem Datenstack eine 32-bit-Zahl (d).

+ w1 w2 -- w3
w3 ist die arithmetische Summe $w1 + w2$.

+! w1 adr -- "plus-store"
w1 wird zum Wert w in $\langle\text{adr}\rangle$ addiert Vgl. +), die Summe ersetzt w in $\langle\text{adr}\rangle$.

+LOOP n -- C, I, "plus-loop"
 sys -- (comp.Mod.)
n wird zum Schleifenindex addiert. Schleifenabbruch (mit Beseitigung der Schleifenparameter auf dem Returnstack), wenn Index die Grenze zwischen Limit - 1 und Limit ueberschreitet, sonst Fortsetzung der Schleife hinter D0. sys wird mit D0 ausbalanciert. Vgl. D0

, 16b -- "comma"
Reserviere Platz fuer 16 Bit, dann speichere 16b in Adresse HERE 2-.

," --
Vok.: HIDDEN
Fuegt den folgenden Text bis " ins Woerterbuch ein.

- w1 w2 -- w3 "minus"
w3 ist das Subtraktionsergebnis $w1 - w2$.

--> -- I, M, "next-block"
Fortsetzung der Interpretation auf naechst-folgendem Block. Verwendung in Doppelpunktdefinitionen, die eine Blockgrenze ueberschreiten.

-ROT 16b1 16b2 16b3 -- 16b3 16b1 16b2 "minus-rote"

Die 3 obersten Stackeinträge werden rotiert, der Oberste gelangt an die 3. Stackposition.

`-TRAILING adr +n1 -- adr +n2 "dash-trailing"`

Der Zeichenzähler der ab `adr` stehenden Zeichenkette wird so korrigiert, dass nachfolgende Leerzeichen unterdrückt werden. `+n2 = 0`, falls `+n1 = 0` oder wenn die Zeichenkette nur Leerzeichen enthält.

`. n -- M, "dot"`

Formatfreie und vorzeichenbehaftete Ausgabe von `n`.

`." -- C, I, "dot-quote"`

Zeichenkette `ccc` in `."ccc"` wird kompiliert und bei späterer Ausführung ausgegeben. Abschluss durch `"`. Das Leerzeichen nach `."` gehört nicht zum Text.

`.(-- I, M, "dot-paren"`

Zeichenkette `ccc` in `.(ccc)` wird bis `)` ausgegeben. Das Leerzeichen nach `.(` gehört nicht zum Text.

`.2 n --`

Druckt `n` zweistellig ohne Vorzeichen.

`.ID adr -- "dot-id"`

Gibt den Definitionsnamen der Namensfeldadresse `adr` aus.

`.R n +n -- M, "dot-r"`

Konvertierung von `n` bezüglich der aktuellen Zahlenbasis, rechtsbündige Ausgabe in einem gedachten Feld der Weite `+n` mit vorangestelltem `'-'` für negatives `n`.

`.S -- "dot-s"`

Anzeige des Inhaltes des Datenstacks (zerstörungsfrei).

`.VERSION --`

Druckt Versionsbezeichnung des Systems aus.

`/ n1 n2 -- n3 "divide"`

`n3` ist der gerundete Wert des Quotienten (`n1 / n2`). Fehler: Division durch 0 oder Quotient ausserhalb des Bereichs `-32768 ... 32767`.

Vgl "Division mit Rundung" in Anlage Definitionen

`/LINE -- n "per-line"`

Konstante hält die Anzahl der je Dumpzeile ausgegebenen Bytes. Damit ist eine leichte Anpassung an das Bildschirmformat möglich. s. DUMP

`/MOD n1 n2 -- n3 n4 "divide-mod"`

Funktion wie bei `/`, jedoch zusätzlich Übergabe des Divi-

sionsrestes n4 (gleiches Vorzeichen wie n3 bzw. gleich 0).

/STRING adr1 n1 n3 -- adr2 n2 "up-string"

Liefert Adresse adr2 und die Laenge n2 des noch zu durchsuchenden Eingabestromes (adr1 Quelladresse, n1 maximale Laenge, n3 schon durchsuchte Zeichen).

0 -- n

Ist eine System-Konstante mit dem Wert n=0.

0< n -- flag "zero-less"

Flag ist wahr, falls n negativ ist, d.h. $n < 0$.

0= w -- flag "zero-equals"

Flag ist wahr, falls $w = 0$.

0> n -- flag "zero-greater"

Flag ist wahr, falls n nicht negativ ist, d.h. $n > 0$.

1 -- n

Ist eine System-Konstante mit dem Wert n=1.

1+ w1 -- w2 "one-plus"

w2 ist das Ergebnis der Addition $w1+1$ entsprechend der Operation +.

1- w1 -- w2 "one-minus"

w2 ist das Ergebnis der Subtraktion $w1 - 1$ entsprechend der Operation -.

2 -- n

Ist eine System-Konstante mit dem Wert n=2.

2! 32b adr -- "two-store"

32b wird ab adr abgespeichert. Vgl. "Zahlen"

2+ w1 -- w2 "two-plus"

w2 ist das Ergebnis der Addition $w1 + 2$ entsprechend der Operation +.

2- w1 -- w2 "two-minus"

w2 ist das Ergebnis der Subtraktion $w1 - 2$ entsprechend der Operation -.

2* w1 -- w2 "two-times"

w2 ist das Ergebnis der arithmetischen Linksverschiebung von w1 um 1 Bit. In die frei werdende Bitposition wird 0 geschrieben.

2/ n1 -- n2 "two-divide"
 n2 ist das Ergebnis der arithmetischen Rechtsverschiebung von n1 um 1 Bit. Das Vorzeichen wird mit verschoben und nicht veraendert.

2@ adr -- 32b "two-at"
 32b ist das Bitmuster ab adr.

2CONSTANT 32b -- M, "two-constant"
 Definitionswort fuer 32b Konstanten. Bei Ausfuehrung von Wortname wird 32b auf dem Stack abgelegt.

2DROP 32b -- "two-drop"
 32b wird vom Stack entfernt.

2DUP 32b -- 32b 32b "two-dup"
 Duplizieren von 32b

2OVER 32b1 32b2 -- 32b1 32b2 32b3 "two-over"
 32b3 ist eine Kopie von 32b1.

2ROT 32b1 32b2 32b3 -- 32b2 32b3 32b1 "two-rote"
 Die 3 obersten 32-Bit-Eintraege auf dem Stack werden rotiert, der drittoberste Eintrag gelangt an die Spitze.

2SWAP 32b1 32b2 -- 32b2 32b1 "two-swap"
 Die 2 obersten 32-Bit-Eintraege auf dem Stack werden ver-tauscht.

2VARIABLE -- M, "two-variable"
 Definitionswort fuer 32-Bit-Variablen: 2VARIABLE Wortname . Zuweisung von 4 Byte im Parameterfeld zur Abspeicherung des Variableninhalts. Bei Ausfuehrung von Wortname wird die Parameterfeldadresse auf dem Stack abgelegt.

3 -- n
 Ist ein System-Konstante mit dem Wert n=3.

: -- M, "colon"
 Definitionswort, mit dem durch :Wortname...; ein Eintrag (Doppelpunktdefinition) im Compilationswoerterbuch erfolgt; anschl. Einschalten des compilierenden Modus (nachfolgender Text wird compiliert). Das Wort Wortname kann erst nach Abschluss der Definition mit ; oder ;CODE im Woerterbuch gefunden werden.

; -- C, I, "semi-colon"
 -- (comp.Mod.)
 Beendigung der Compilation einer Doppelpunktdefinition Wortname , Einschalten des ausfuehrenden Modus, Compilation von EXIT .

Wortname kann nun im Woerterbuch gefunden werden. Vgl EXIT :

```
;CODE -- C, I, "semi-colon-code"
```

```
-- (comp.Mod.)
```

Wird in folgender Form verwendet:

```
Wortnamex...<create>...;CODE ... END-CODE
```

Beendet die Compilation, schliesst das Definitionswort Wortnamex ab und fuehrt ASSEMBLER aus. Nachfolgende Zeichenfolge wird in Maschinencode uebersetzt. Bei Ausfuehrung von Wortnamex Wortname wird Wortname definiert. Dieses Wort enthaelt als Ausfuehrungsadresse die Adresse der Kodesequenz, die auf ;CODE in Wortnamex folgt, so dass spaeter dieser Maschinencode ausgefuehrt wird.

```
;USES -- I,C,Def,
```

Wird folgendermassen verwendet:

```
: name ... <create> ... ;USES label ,
```

Beendet die Compilation. Bei Ausfuehrung des mit <name> angelegten Wortes wird die Codesequenz auf der Adresse <label> zur Ausfuehrung gebracht. s. ;CODE

```
< n1 n2 -- flag "less"
```

flag ist wahr fuer $n1 < n2$. $-32768 \ 32767 <$ und $-32768 \ 0 <$ muss jeweils wahr ergeben.

```
<# -- "less-sharp"
```

Initialisierung der Ausgabekonvertierung einer 32-Bit-Zahl in ASCII-Darstellung mit # #> #S <# HOLD SIGN.

```
<MARK -- adr C, "backward-mark"
```

Verwendung am Zielpunkt eines Rueckwaertssprungs. adr wird typisch nur von <RESOLVE fuer die Compilation der Sprungadresse verwendet.

```
<RESOLVE adr -- C, "backward-resolve"
```

Verwendung am Ausgangspunkt eines Rueckwaertssprungs hinter BRANCH oder ?BRANCH. Compilation einer Sprungadresse unter Verwendung der Zieladresse adr.

```
= w1 w2 -- flag "equals"
```

Flag ist wahr fuer $w1 = w2$.

```
> w1 w2 -- flag "greater-than"
```

Flag ist wahr fuer $n1 > n2$. $-32768 \ 32767 >$ und $-32768 \ 0 >$ muss jeweils falsch ergeben.

```
>BODY cfa -- pfa "to-body"
```

die Parameterfeldadresse des Wortes wird aus dessen Codefeldadresse ermittelt.

```
>DOES -- n
```

Ist eine System-Konstante. Entspricht der Adresse der Laufzeit-Routine von DOES>.

>IN -- adr U, "to-in"

Adresse einer Variablen, die den aktuellen Zeichenoffset im Eingabestrom enthaelt. Vgl WORD

>IS adr1 -- adr2 "to-is"

Teilt einem Kode-Feld (adr1) ein Daten-Feld (adr2) zu. Wenn sich das Wort im Nutzerbereich der Worte (words) befindet, muss die Datenadresse (adr2) relativ zum CURRENT-Nutzer-Zeiger berechnet werden. Andernfalls ist adr1 sofort das Parameterfeld.

>LINK cfa -- lfa "to-link"

Ausgehend von der cfa eines Wortes wird die lfa berechnet.

>MARK -- adr C, "forward-mark"

Verwendung am Ausgangspunkt eines Vorwaertssprungs, und zwar meist hinter BRANCH oder ?BRANCH. Compiliert Platz im Woerterbuch fuer eine Sprungadresse, die spaeter von >RESOLVE aufgeloeset wird.

>NAME cfa -- nfa "to-name"

Ausgehend von der cfa eines Wortes wird die nfa berechnet.

>NEXT -- adr

Ist eine Konstante. adr entspricht der Adresse des inneren Interpreters.

>R 16b -- C, "to-r"

Transfer von 16b zum Returnstack. Vgl. "Returnstack"

>RESOLVE adr -- C, "forward-resolve"

Verwendung am Zielpunkt eines Vorwaertssprungs. Berechnet die Adresse eines Sprungs zur aktuellen Woerterbuchposition unter Verwendung von adr, traegt diese Adresse in den von >MARK freigelassenen Platz ein. .

>TYPE adr n -- M

s. TYPE, jedoch wird vor dem Drucken der String nach PAD transportiert. Ermoglicht Multitaskarbeit mit Blockinhalten.

? adr -- "question"

Druckt den Inhalt der Adresse adr aus (in freiem Format und in Uebereinstimmung mit der aktuellen Zahlenbasis).

?BRANCH flag -- C, "question branch"

Compilation eines bedingten Sprungs, der ausgefuehrt wird, wenn flag falsch ist, ansonsten Fortsetzung der Abarbeitung auf der Compilationsadresse, die der Sprung-

adresse folgt. Vgl BRANCH

?CONDITION n1 n2 --

Erzeugt eine Fehlermeldung falls n1 ungleich n2 ist. Die Meldung zeigt an, dass die Compilationsbedingungen nicht eingehalten wurden.

?CSP --

Erzeugt eine Fehlermeldung wenn die Stackposition (Inhalt von SP) sich von dem in CSP geretteten Wert unterscheidet.

?DNEGATE d1 d2 -- d3
-- d1

Negiert die vorzeichenbehaftete Doppelzahl d1, falls d2 negativ ist (d2 kann z.B. auch das Duplikat von d1 sein) und hinterlaesst sie als d2 auf dem Datenstack, andernfalls wird der Wert d1 beibehalten.

?DO w1 w2 --

Testet, ob w1=w2. Ist dies der Fall, wird hinter LOOP bzw. +LOOP gesprungen. Ansonsten Abarbeitung wie DO.

?DUP 16b -- 16b 16b "question-dupe"
oder 0 -- 16b, falls 16b =| 0 ist.

?ENOUGH +n --

Gibt eine Fehlermeldung aus, wenn nicht genug Werte (+n ist die Anzahl) auf dem Datenstack vorhanden sind. Wird z.B. am Anfang eines Programms zur Stackkontrolle benutzt.

?ERROR adr n f --

Fuehrt bei wahrem Flag zur Ausgabe des Strings auf adr als Fehlermeldung und Abbruch nach QUIT.

?LEAVE flag --

bedingter vorzeitiger Abbruch einer DO...LOOP(bzw....+LOOP)-Schleife (siehe auch LEAVE).
Anwendung: DO...?LEAVE...LOOP (bzw. ...+LOOP)

?LINE +n --

Kontrolliert bei Ausgabe einer Zeile mit +n Zeichen, ob das Zeilenende erreicht wird. Wenn dies der Fall ist, wird vorher CR (carriage return) ausgefuehrt.

?MISSING flag --

Erzeugt eine Fehlermeldung, wenn das Flag flag wahr ist, d.h. z.B. das Wort Wortname unbekannt ist.

?NEGATE n1 n2 -- n3
-- n1

Ist die Zahl n2 negativ, wird die Zahl n1 negiert und als n3

auf den Datenstack gelegt. (n2 kann z.B. auch das Duplikat von n1 sein), andernfalls bleibt die Zahl n1 in ihrer Form erhalten.

?STACK --

Erzeugt eine Fehlermeldung, wenn der Stackpointer ausserhalb der vorgegebenen Grenzen liegt.

@ adr -- 16b "at"

16b ist das Bitmuster auf der Adresse adr.

ABORT

Datenstack saubern, keine Ausgabe einer Nachricht und Ausfuehrung von QUIT.

ABORT" flag -- C, I, "abort-quote"
-- (comp.Mod.)

Mit ABORT" ccc" wird ccc ausgegeben und ABORT ausgefuehrt, falls das Flag wahr ist, ansonsten wird die normale Abarbeitung fortgesetzt. Das Leerzeichen nach ABORT" gehoert nicht zur Zeichenkette ccc.

ABS n -- u "absolute"

u ist der absolute Betrag von n. ABS -32768 ergibt -32768.

AGAIN -- C, I

sys -- (comp.Mod.)

Ausfuehrung eines unbedingten Ruecksprungs zum Anfang einer BEGIN...AGAIN - Schleife. sys wird mit BEGIN ausbalanciert. Vgl. BEGIN

ALLOT w --

Belegung von w Byte Speicherplatz im Woerterbuch.

ALLOT n -- U

Vok.: USER

Reserviert n Byte im USER-Area. Zusammen mit USER CREATE benutzt koennen Felder im USER-Area angelegt werden.

ALSO -- Voc

Vok.: ROOT

Fuegt ein weiteres Vocabular in die z. Z. gueltige Suchreihenfolge ein.

ONLY voc#1 ALSO voc#2

macht das Vocabular #1 in der Suchordnung resident. Die Durchsuchung des Woerterbuches beginnt mit Vocabular #2 und wird mit #1 und ROOT fortgesetzt. Es koennen maximal #VOCs (hier 8) Vocabulare durch mehrfache Anwendung von ALSO nacheinander durchsucht werden.

AND 16b1 16b2 -- 16b3

Bitweise logische UND-Verknuepfung von 16b1 und 16b2 zu 16b3.

ASCII -- char I, M "as-key"
-- (comp.Mod.)

Mit ASCII ccc wird der ASCII des ersten Zeichens von ccc auf dem Stack eingetragen (ausfuehrender Modus) oder ins Woerterbuch als Literal compiliert (compilierender Modus).

ASSEMBLER -- Voc,

Das erste Woerterbuch in der Suchreihenfolge wird durch ASSEMBLER ersetzt. Das ASSEMBLER-Vokabular enthaelt im Grundzustand nur das Wort END-CODE. Weiterhin ist es zur Aufnahme des FORTH-Assembler-Wortschatzes vorgesehen, der nachgeladen werden kann.

AVOC -- adr "a-voc"

Ist eine Variable. Wird systemintern zur Umschaltung auf ASSEMBLER verwendet.

B/BUF -- +n "bytes-per-buffer"

Ist eine Konstante und entspricht der Anzahl der Bytes je Blockpuffer.

BASE -- adr U,

Adresse einer Variablen, die die Zahlenbasis fuer Eingabe- und Ausgabekontvertierungen enthaelt. Zahlenbasis: 2...72

BEGIN -- C, I,
-- sys (comp.Mod.)

Eroeffnung einer Strukturierungssequenz: BEGIN ...flag UNTIL. Wird abgearbeitet, bis flag wahr ist; BEGIN...flag WHILE...REPEAT wird abgearbeitet, bis flag falsch ist. sys wird mit UNTIL bzw. WHILE ausbalanciert.

BELL -- n M

Konstante, deren Ausgabe ein akustisches Zeichen hervorruft.

BETWEEN n3 n2 n1 -- flag

Der Vergleich der 16-Bit-Zahl n3 mit dem Minimum n2 und dem Maximum n1 wird ausgewertet. Liegt n3 ausserhalb dem Bereich von n1 und n2, ist Flag wahr andernfalls falsch.

BL -- 32 "b-l"

Uebergabe des ASCII-Code fuer ein Leerzeichen.

BLANK adr u --

u Speicherbytes ab adr werden mit Leerzeichen gefuellt, falls u > 0.

BLK -- adr U, "b-l-k"

Adresse der Variablen, die die Nummer des Massenspeicher-

blocks enthaelt, der als Eingabestrom interpretiert wird (Nummer = 0 impliziert Eingabestrom vom Texteingabepuffer).

BLK# -- adr

Vok.: HIDDEN

Variable, die die Nummer des Blockes im Blockpuffer enthaelt. Der Blockpuffer beinhaltet nur einen Block, da dies bei Arbeit mit einer RAM-Disk ausreichend ist.

BLOCK u -- adr M,

adr zeigt auf erstes Byte in Block u im Blockpuffer. Steht darin zunaechst ein anderer Block, so wird Block u eingelesen. Der alte Blockinhalt wird zuvor zurueckgeschrieben, falls dieser als aktualisiert gekennzeichnet ist. Nur Dateninhalte, die zuletzt mit BLOCK oder BUFFER angesprochen wurden, sind gueltig.

BODY> pfa -- cfa "from-body"

Ausgehend von der pfa eines Wortes wird die cfa berechnet.

BOOT -- Vec

erstes Wort beim Kaltstart. Fuehrt alle Nutzerspezifischen Initialisierungen aus. Da es ein Vector ist, kann seine Funktion jederzeit geaendert werden, z.B. um Turn-Key-Applikationen zu erstellen.

BRANCH -- C,

Vok.: HIDDEN

In COMPILE BRANCH Compilation eines unbedingten Sprungs mit nachfolgender Sprungadresse, die meist von <RESOLVE oder >MARK aufgeloeset wird.

BS -- n

Konstante, die den Wert 8 fuer Backspace liefert.

BUFFER u -- adr M,

Zuweisung eines Blockpuffers zu Block u . Diese Funktion ist vollstaendig mit BLOCK spezifiziert, es sei denn, der Block befindet sich noch nicht im Puffer. Dann wird der Block n nicht vom Massenspeicher gelesen. Der Inhalt eines mit BUFFER ausgewaehlten Blockpuffers ist noch unbestimmt.

BYE --

Bewirkt das Verlassen des FORTH-Systems. Rueckkehr in den Monitor. Vorher werden alle Blockpuffer gerettet.

C! 16b adr -- "c-store"

Die niederwertigen 8 Bits von 16b werden im Byte auf adr abgespeichert.

C, 16b -- "c-comma"
Zuweisung von 1 Byte Speicherplatz im Woerterbuch, danach
Abspeicherung der niederwertigen 8 Bit von 16b auf Adresse
HERE - 1.

C/L -- n "character-per-line"
Entspricht der Anzahl der Zeichen pro Screen-Zeile.

C@ adr -- 8b "c-at"
8b ist der Inhalt des Bytes auf adr .

CMOVE adr1 adr2 u -- "c-move"
Transfer von u Bytes ab adr1 in Bereich ab adr2. Das erste
Byte wird zuerst transferiert (Vorwaertstransfer). Transfer
erfolgt nur, wenn u ungleich 0 .

CMOVE> adr1 adr2 u -- "c-move-up"
Transfer von u Bytes ab adr1 nach adr2. Transfer beginnt am
Ende bei adr1 + u - 1 (Rueckwaertstransfer).

CODE -- M,
Definitionswort fuer Woerterbucheintraege, deren Funktionsum-
fang durch Assemblertext beschrieben wird (Codedefinitionen):
CODE Wortname ... END-CODE. Vor Abschluss der Definition mit
END-CODE kann Wortname nicht aufgefunden werden.

COLD --
Das FORTH-System wird schrittweise initialisiert und
laeuft ueber BOOT nach ABORT, wo die restliche Initialisier-
ung erfolgt, in den Textinterpreter.

COMPARE adr1 adr2 +n -- flag
Fuehrt einen Zeichenkettenvergleich durch (Zeichenkette1 ab
adr1 und Zeichenkette2 ab adr2 mit der Laenge +n).
Sind die beiden Zeichenketten:
Zeichenkette1 = Zeichenkette2 ==> flag:=0
Zeichenkette1 < Zeichenkette2 ==> flag:=-1
Zeichenkette1 > Zeichenkette2 ==> flag:=1
Alle Vergleiche sind relativ zur ASCII-Ordnung.

COMPILE -- C,
Typisch verwendet in:Wortname...COMPILE Wortnamex...;.
Bei Ausfuehrung von Wortname wird die Compilationsadresse von
Wortnamex nicht ausgefuehrt, sondern compiliert.

CONSTANT 16b -- M,
Definitionswort fuer Konstanten: 16b CONSTANT Wortname . Bei
spaeterer Ausfuehrung von Wortname wird 16b auf dem Stack
uebergeben.

CONTEXT -- adr U,
Adresse des Feldes, das die Suchreihenfolge im Woerterbuch festlegt. s.ORDER ALSO

CONTROL -- n
Ermittlung des Kodes n der "Control-Tasten"-Belegung.
s. auch ASCII

CONVERT +d1 adr1 -- +d2 adr2
Eingabekonvertierung einer FORTH-Zeichenkette (Laengenbyte auf adr, Zeichenkette ab adr + 1) mit Startwert +d1 in 32-Bit-Zahl +d2. Die Adresse des ersten nichtkonvertierbaren Zeichens ist adr2.

COPY n1 n2 --
kopiert Block n1 in RAM-Disk nach n2.

COUNT adr1 -- adr2 +n
Ermittlung von Anfangsadresse adr2 und Laenge +n (0...255) einer FORTH-Zeichenkette ab adr1.

CR -- M, Vec "c-r"
Loest einen Zeilenvorschub auf dem Ausgabegeraet aus, loescht #OUT und erhoehrt #LINE um eins.

CRASH --
Bewirkt die Ausgabe einer Systemmitteilung: (no execution vector), wenn Vektor noch nicht initialisiert wurde.

CREATE -- M,
Definitionswort. CREATE Wortname richtet Woerterbucheintrag ein. Die darauf folgende freie Adresse im Speicher ist die Parameterfeldadresse. Bei Ausfuehrung von Wortname Uebergabe dieser Adresse auf dem Datenstack, d.h. keine Reservierung von Speicherplatz im Parameterfeld.

CREATE -- Vok.: USER
Definiert ein Wort, welches die Adresse des naechsten freien Eintrages im USER-Area uebergibt. s. USER

CRESET w adr --
Logische Verknuepfung des Inhaltes von adr. mit der Maske w nach der Gleichung: <adr>:= /w AND <adr>

CRLF -- I/O
Vok.: I/O
gibt Hex 0D und 0A aus. Incrementiert den Zeilenzaehler #LINE und loescht den Spaltenzaehler #OUT. Standartvoreinstellung fuer CR.

CSET w adr --

Logische Verknuepfung des Inhaltes von adr mit der Maske w nach der Gleichung: $\langle \text{adr} \rangle := w \text{ OR } \langle \text{adr} \rangle$

CSP -- adr

Ist eine System-Variable. Sie dient zur Sicherung des beim Aufruf gerade aktuellen Stackpointers. Der Inhalt von adr wird am Ende des Compilationsvorganges fuer einen Fehlertest benoetigt.

CTOGGLE w adr --

Logische Verknuepfung des Inhaltes von adr mit der Maske w nach der Gleichung:
 $\langle \text{adr} \rangle := w \text{ XOR } \langle \text{adr} \rangle$

CURRENT -- adr U,

Adresse der Variablen, die das Compilationswoerterbuch spezifiziert.

D+ wd1 wd2 -- wd3 "d-plus"

wd3 ist die arithmetische Summe $wd1 + wd2$.

D- wd1 wd2 -- wd3 "d-minus"

wd3 ist das Ergebnis der Subtraktion $wd1 - wd2$.

D. d -- M, "d-dot"

Formatfreie Ausgabe des Betrags von d mit vorangestelltem '-' fuer negatives d.

D.R d +n -- M, "d-dot-r"

Konvertierung von d bezueglich der aktuellen Zahlenbasis, rechtsbuendige Ausgabe in einem gedachten Feld der Weite +n mit vorangestelltem '-' fuer negatives d.

D0= wd -- flag "d-zero-equals"

flag ist wahr, falls $wd = 0$.

D2/ d1 -- d2 "d-two-divide"

d2 ist das Ergebnis der arithmetischen Rechtsverschiebung von d1 um 1 Bit. Das Vorzeichen wird mit verschoben und nicht veraendert.

D< wd1 wd2 -- flag "d-less"

flag ist wahr, falls $d1 < d2$. Vgl. <

D= wd1 wd2 -- flag "d-equal"

Das Flag flag ist wahr, falls $wd1 = wd2$.

D> d1 d2 -- flag "d-greater"

Das Flag ist true, falls $d1 > d2$ (vgl. >).

DABS d -- ud "d-absolute"

ud ist der Betrag von d . Fuer d = -2 147 483 648 ist d = ud.
Vgl. "Zweierkomplementarithmetik"

DECIMAL --

Zahlenbasis fuer Eingabe-/Ausgabekontrollierungen auf 10 setzen

DEFER --

Definiert einen Woerterbucheintrag fuer spaetere Verwendung als Vektor.

DEFER --

Vok.: USER

Definiert einen Ausfuehrungsvektor, der fuer jedes Task unterschiedlich sein kann. s. USER

DEFINED -- adr flag

Sucht nach dem naechsten Wort im Eingabestrom:

flag:=-1 - es existiert und adr:=cfa und normales Wort

flag:= 1 - es existiert und adr:=cfa und IMMEDIATE-Wort

flag:= 0 - es existiert nicht und adr:=HERE.

DEFINITIONS --

Zum Compilationswoerterbuch wird das erste Woerterbuch der Suchreihenfolge.

DEFINITIONS --

Vok.: ROOT

Redefinition s. Vokabular FORTH

DEPTH -- +n

+n ist die Anzahl der Datenstackeintraege vor Uebergabe von +n.

DIGIT n1 n2 -- n3 flag

Es wird ein Flag zurueckgegeben, welches anzeigt, ob das Zeichen eine gueltige Dezimalstelle in gegebener Zahlenbasis n2 ist oder nicht. Wenn es ein gueltiges Zeichen ist, wird ein konvertierter Wert n3 und das Flag wahr zurueckgegeben. Andernfalls das Zeichen n3=n1 und das Flag falsch.

DLITERAL d --

Compilation einer 32-Bit-Zahl in die naechste freie Woerterbuchposition.

DMAX d1 d2 -- d "d-max"

d3 ist die groessere der beiden Zahlen d1 und d2.

DMIN d1 d2 -- d3 "d-min"

d3 ist die kleinere der beiden Zahlen d1 und d2.

DNEGATE d1 -- d2 "d-negate"

d2 ist das Zweierkomplement von d1.

DO w1 w2 -- C, I,
-- sys (comp.Mod.)

Anwendung in DO ... LOOP oder DO ... +LOOP. Eröffnung einer Schleife, die im Laufbereich des Indexes zwischen w2 und w1 (Limit) abgearbeitet wird, und zwar wenigstens einmal. sys wird mit LOOP bzw. +LOOP ausbalanciert.

DOES> -- adr C, I,
-- (comp.Mod.)

Festlegung der Laufzeitaktivität eines Definitionswortes mit : Wortname ... <create> ... DOES> ... ; und anschl. Wortname Wortname. <create> ist CREATE oder ein beliebiges anderes Wort, welches CREATE ausführt. DOES> schliesst den Teil fuer die Spezifikation der Compilationsaktivität des Definitionswortes ab und eröffnet den Teil fuer die Ausführungaktivität. Zur Laufzeit wird die Parameterfeldadresse von Wortname auf dem Datenstack eingetragen und anschliessend die Sequenz zwischen DOES> und ; abgearbeitet.

DOUBLE? -- f

f ist TRUE, wenn die von NUMBER konvertierte Zahl doppelte Genauigkeit hatte; sonst FALSE

DP -- adr U

Ist eine Systemvariable (Woerterbuchzeiger), welche in adr die Adresse des naechsten freien Speicherzelle am Ende des Woerterbuches enthaelt. Der Wert von DP kann mit HERE gelesen und mit ALL0T geaendert werden.

DPL -- adr U

Ist eine Variable, die die Anzahl der Stellen nach dem Dezimalpunkt bei Eingabekontvertierungen enthaelt.

DROP 16b --

16b wird vom Stack entnommen.

DU< ud1 ud2 -- flag "d-u-less"

Das Flag flag ist wahr, falls ud1 < ud2. Beide Zahlen sind vorzeichenlos.

DU adr -- adr +40H

Ausgabe von 40H Bytes ab Adresse adr als Hexdump. Die Folgeadresse wird auf dem Stck hinterlassen und kann fuer einen weiteren Aufruf von DU verwendet werden.

DUMP adr u -- M

Ausgabe von u Bytes, die im Speicher ab adr stehen. Vor jeder Zeile wird die Anfangsadresse ausgegeben.

DUP 16b -- 16b 16b "dup"

Dupliziert 16b.

ELSE -- C, I,
 sys1 -- sys2 (comp.Mod.)
 ELSE wird innerhalb IF...ELSE...THEN hinter dem IF- Zweig
 abgearbeitet, Ausfuehrung setzt sich bis hinter THEN fort.
 sys1 wird mit IF, sys2 mit THEN ausbalanciert.
 Vgl. IF, THEN

EMIT 16b -- M, Vec
 Ausgabe der niederwertigen 8 Bit auf dem aktuellen
 Ausgabegeraet. #OUT wird incrementiert.

EMPTY --
 Letztes Wort im Kernvokabular. Bei Aufruf werden alle spaeter
 definierten Worte geloescht.

EMPTY-BUFFERS -- M, "empty-buffers"
 Freigabe aller Blockpuffer. Aktualisierte Bloecke werden
 nicht zum Massenspeicher zurueckgeschrieben. Vgl. BLOCK

END-CODE --
 Vok.: ASSEMBLER
 Schliesst eine Primitivdefinition ab und macht das vor CODE
 gueltige Vokabular wieder zum aktuellen CONTEXT-Vokabular.
 s. ASSEMBLER

ENTRY -- adr U
 eine USER-Variable mit spezieller Bedeutung fuer Multitask.
 Hier wird ein Stueck MC eingetragen, welches bestimmt, ob das
 Task aktiv ist oder nicht.

EOD -- adr
 Vok.: I/O
 Uebergibt die Adresse im Boot-Bereich, auf der die erste von
 der RAM-Disk nicht mehr benutztbare Adresse steht.

ERASE adr u --
 u Speicherbytes ab adr werden auf 0 gesetzt, falls u > 0

EXECUTE adr --
 Ausfuehrung der mit adr bezeichneten Wortdefinition.

EXIT -- C,
 Rueckkehr zu einer aufrufenden Doppelpunktdefinition hinter
 die Aufrufstelle aus der aufgerufenen Doppelpunktdefinition
 heraus; (entspricht ;S in fig) in Schleifen verboten!

EXPECT adr +n -- M, Vec
 Fuer +n > 0 Eingabe und byteweises Abspeichern von Zeichen ab
 adr in Richtung hoeherer Adressen, bis der Steuerkode der

ENTER-Taste [RET] erkannt wird oder +n Zeichen eingegeben sind. Ausgabe (Echo) aller Zeichen (ausser <ret>, statt dessen Ausgabe eines Leerzeichens). Als Korrekturcode sind 08H und 7FH installiert. Siehe (EXPECT). Vgl. SPAN

FALSE -- n
Ist eine Konstante, welche den Wert n=0 liefert (false entspricht falsch).

FENCE -- adr
Ist eine Systemvariable, die eine Adresse enthaelt. Unterhalb dieser Adresse wird FORGET nicht ausgefuehrt. Um unter die angegebene Adresse zu gelangen, muss der Benutzer den Inhalt von FENCE aendern.

FH n -- "from-here"
Dient der relativen Adressierung des zu ladenden Screens von dem Screen in dem FH steht aus gerechnet.
1 FH LOAD laedt den naechsten Screen, 2 FH 4 FH THRU die Reihenfolge vom uebernaechsten bis zum 4. folgenden.

FILL adr u 8b --
u Speicherbytes ab adr werden mit 8b gefuellt, falls u > 0 .

FIND adr1 -- adr2 flag
Suche des Wortes, das in der FORTH-Zeichenkette ab adr1 enthalten ist, in der aktuellen Suchreihenfolge. Kann das Wort gefunden werden, so ist adr2 dessen Compilationsadresse und n = 1 (IMMEDIATE-Wort) oder n = -1. Wurde das Wort nicht gefunden, so ist adr2 = adr1 und n = 0 .

FIRST -- adr
Bringt die Adresse des ersten Blockpuffers auf den Datenstack.

FLUSH -- M,
Ausfuehrung von SAVE-BUFFERS und EMPTY-BUFFERS

FORGET -- M,
Mit FORGET Wortname werden Wortname und alle nachfolgenden Eintraege (unabhaengig von ihrer Woerterbuchzugehoerigkeit) entfernt, falls Wortname im Compilationswoerterbuch gefunden werden kann. Fehler: Name wird nicht gefunden bzw. Compilationswoerterbuch soll entfernt werden. s. (FORGET)

FORTH --
Name des Basis- (Primaer-) Woerterbuchs. Bei Ausfuehrung von FORTH wird dieses zum ersten Woerterbuch in der Suchreihenfolge. Bei Systeminitialisierung ist FORTH das erste Woerterbuch und gleichzeitig Compilationswoerterbuch. Alle Definitionen werden darin eingetragen, bis ein anderes Compilationswoerterbuch eingerichtet wird.
Vgl. VOCABULARY

FORTH --
Vok.: ROOT
Redefinition von FORTH um vom ROOT-Vocabular auf FORTH umschalten zu koennen.

FORTH-83 --
Sicherung der Verfuegbarkeit eines FORTH-83-Systems, anderenfalls Fehler!

GET --
Laedt eine Screendatei vom Magnetband in die RAM-Disk. Beachte je nach Rechnertyp installationsabhaengige Besonderheiten.

HASH adr1 adr2 -- +n
Es ist eine Zeichenkettenadresse (adr1) und ein Zeiger(adr2) zu einem Woerterbuch gegeben. Als Resultat erscheint der aktuelle Faden (+n). Es wird das erste Zeichen der Zeichenkette genutzt, um den Faden zu bestimmen.

HEADER --
Teilfunktion von CREATE. Baut einen Namensheader auf.

HELLO --
Aktuell von BOOT benutzt, um die Systemmeldung auszugeben und EMPTY-BUFFERS und DECIMAL auszufuehren.

HERE --
Adresse der naechsten verfuegbaren Adresse im Woerterbuch.

HEX --
Setzen der Zahlenbasis fuer Ein- / Ausgabekonvertierungen auf 16.

HIDDEN -- Voc
Das Vocabular enthaelt alle normalerweise "verborgenen" Laufzeitroutinen und Teilaktionen, die fuer den Nutzer meist uninteressant sind und die nur benoetigt werden, wenn das Compile-Verhalten erweitert werden soll.

HIDE --
Verbirgt den letzten Worteintrag im Woerterbuch, so dasz er von (FIND) erst nach Aufruf von REVEAL gefunden werden kann.

HLD -- adr U
Ist eine Systemvariable, die die Zieladresse enthaelt, auf der das naechste erzeugte ASCII-Zeichen abgespeichert wird.

HOLD char --
Einfuegen von char in die Ausgabezeichenkette bei Ausgabekonvertierungen mit <# ... #>.

I -- w C,
 w ist die Kopie des Schleifenindexes in DO...I...LOOP (bzw. ... +LOOP).

I/O -- Voc
 Woerterbuch welches alle Worte, die zur Anpassung an die systemspezifischen Ein- und Ausgabefunktionen erforderlich sind, enthaelt. Dazu gehoert auch die Behandlung des Blocklesens und -schreibens (in der Grundversion fuer RAM-Disk installiert). Diese Worte sind meist Zielpunkt fuer Vektoraufrufe aus dem Stammvokabular. Nutzerspezifische Aenderungen der Ein- und Ausgabe bzw. Massenspeicherbehandlung sollten ebenfalls dementsprechend in diesem Vokabular untergebracht werden.

IF flag -- C, I,
 -- sys (comp.Mod.)
 Wird in Strukturen der Form: IF...ELSE...THEN oder IF...THEN verwendet. Wenn flag wahr ist, Ausfuehrung der Woerter zwischen IF und THEN bzw. IF und ELSE mit Fortsetzung hinter THEN . sys wird mit ELSE bzw. THEN ausbalanciert.

IMMEDIATE --
 Kennzeichnung des zuletzt gebildeten Woerterbucheintrags als ein Wort, welches auch im compilierenden Modus ausgefuehrt und nicht compiliert wird.

INDEX u1 u2 -- M
 Ausgabe der jeweils ersten Zeile jedes Screens von Screen Nr. u1 bis Screen u2 (Titelzeile).

INTERPRET -- M,
 Beginn der Interpretation bei dem Zeichen, welches mit >IN relativ zum Blockanfang des Massenspeicherblocks mit der Nummer, die in BLK steht, indiziert wird. Wenn die Blocknummer 0 ist, erfolgt Interpretation vom Texteingabepuffer. s. auch (INTERPRET) und (COMPILE).

IS --
 Abhaengig von STATE wird entweder das folgende DEFER-ierte Wort (Vector) sofort gesetzt oder zum spaeteren Setzen compiliert. z.B. ' .S IS STATUS stellt den Vector so um, dasz nach jeder Eingabezeile der Datenstack gedruckt wird; sehr geeignet fuer Tests.

J -- w C,
 w ist eine Kopie des Indexes der naechst aeusseren Schleife. Nur in genesteten Schleifen DO...LOOP (bzw. ...+LOOP) anwendbar, wenn dazwischen keine Werte auf dem Returnstack eingetragen werden, z.B. DO...DO...J...LOOP...LOOP .

KEY -- 16b M, Vec,
Die niederwertigen 7 Bit sind die Kodierung des naechsten eingegebenen ASCII-Zeichens. Alle ASCII-Zeichen sind gueltig, auch CTRL-Zeichen (keine Echofunktion, keine Aufbereitung).

KEY? -- f Vec
liefert TRUE, wenn eine Taste betaetigt wurde. Entspricht ?TERMINAL in FIG.

L/SCR -- u "lines-per-screen"
Ist eine Konstante. Der Wert stellt die Zeilenanzahl eines Screens dar.

L>NAME lfa -- nfa "l-to-name"
Ausgehend von der lfa eines Wortes wird die nfa berechnet.

LABEL --
Wird zur Bildung von Marken im Maschinencode genutzt. Vgl. ASSEMBLER

LAST -- adr U
Ist eine Variable, die die Anfangsadresse des letzten (u.U. unvollstaendigen oder ungueltigen) Woerterbucheintrags enthaelt.

LEAVE -- C, I,
Sofortiges Verlassen einer Schleife DO...LEAVE...LOOP (bzw. ...+LOOP) mit Beseitigung der Schleifenparameter auf dem Returnstack. Auch in genesteten Strukturierungssequenzen in Schleifen anwendbar, auch mehrere LEAVE in einer Schleife.

LINK -- adr
USER-Variable die die Adresse des naechsten Tasks in der Round-Robin-Schleife enthaelt.

LINK> lfa -- cfa "from-link"
Ausgehend von der lfa eines Wortes wird die cfa berechnet.

LITERAL -- 16b C, I,
16b -- (comp.Mod.)
Typische Anwendung im compilierenden Modus: [16b] LITERAL. Compilation von 16b als Literal, bei der spaeteren Ausfuehrung erfolgt eine Eintragung von 16b auf dem Datenstack.

LIST u -- M,
SCR wird auf u gesetzt, Screen u wird ausgegeben. Vgl. BLOCK

LOAD u -- M,
Sicherung der aktuellen Parameter des Eingabestroms >IN und BLK ; IN wird auf 0 und BLK auf Screennummer u gesetzt, damit Interpretation von Screen u . Beendigung der Interpretation

und Rueckkehr zu den alten Werten von >IN und BLK explizit mit \S oder bei Blockende.

Fehler: u = 0 Vgl. >IN BLK BLOCK

LOOP -- C, I,
sys -- (comp.Mod.)

Inkrementieren des Schleifenindex um 1. Schleifenabbruch, wenn Index die Grenze zwischen Limit - 1 und Limit ueberschreitet, sonst Fortsetzung der Schleife hinter D0.

sys wird mit D0 ausbalanciert.

M/MOD d n1 -- n2 n3

die doppelgenaue Zahl d wird durch n1 dividiert. n3 ist Quotient, n2 Rest. ACHTUNG: nicht identisch mit M/MOD in FIG.

MARK --

MARK <name> erzeugt einen Woerterbucheintrag <name>, bei dessen Aufruf alle nach <name> definierten Worte geloescht werden. s. EMPTY

MAX n1 n2 -- n3 "max"

n3 ist die groessere der beiden Zahlen n1 oder n2 entsprechend der Operation >.

MAX# -- n

Vok.: I/O

Anzahl der maximal in der RAM-Disk verfuegbaren Bloecke.

MIN n1 n2 -- n3 "min"

n3 ist die kleinere der beiden Zahlen n1 oder n2 entsprechend der Operation <.

MOD n1 n2 -- n3

n3 ist der Rest der Division n1 / n2 mit gleichem Vorzeichen wie n2 oder gleich 0 . Fehler: Divisor = 0 oder n1 / n2 ausserhalb des Bereichs -32768 ... +32767.

MOVE adr1 adr2 u --

Falls u > 0, werden u Bytes ab adr1 in den Bereich ab adr2 transferiert.

MU/MOD ud1 u1 -- u2 ud2

Ist eine mathematische Operation mit gemischten Zahlentypen ohne Vorzeichen, die den Rest u2 und den Doppelzahlquotienten ud2 bei der Division einer Doppelzahl ud1 durch einen Divisor u1 liefert. ud1/u1=:ud2 Rest u2

N>LINK nfa -- lfa "n-to-link"

Ausgehend von der nfa eines Wortes wird die lfa berechnet.

NAME> nfa -- cfa "from-name"

Ausgehend von der nfa eines Wortes wird die cfa berechnet.

NEGATE n1 -- n2
n2 ist das Zweierkomplement von n1.

NIP w1 w2 -- w2
Loescht den zweiten Stackeintrag.

NOOP --
Ist eine leere Anweisung.

NOT 16b1 -- 16b2
16b2 ist das Einerkomplement von 16b1.

NUMBER? adr -- d f Vec
Konvertiere die Zeichenkette, deren erstes Zeichen auf adr steht, in eine 32-Bit-Zahl entsprechend dem aktuellen Inhalt von BASE . Standartmaessig auf (INUMBER?) vektorisiert. Flag gibt Auskunft ueber Erfolg der Konvertierung.

OFF adr --
Setzt Inhalt von adr auf FALSE.

OFFSET -- n U,
Ist eine Variable, die den Blockoffset enthaelt, welcher bei Ausfuehrung von BLOCK oder BUFFER zur Blocknummer auf dem Stack addiert wird.

ON adr --
Setzt Inhalt von adr auf TRUE.

ONLY --
Vok.: ROOT
Das erste Woerterbuch in der Suchreihenfolge wird durch ROOT ersetzt. ROOT wird gleichzeitig zum einzigen durchsuchten Vocabular. Vgl. VOCABULARY und Teilwoerterbuch ROOT

OR 16b1 16b2 -- 16b3
16b3 ist die bitweise ODER-Verknuepfung von 16b1 mit 16b2 .

ORDER -- Vok.: ROOT
zeigt die z. Z. gueltige Vokabular-Suchreihenfolge.

OS n -- Vok.: I/O
Aufruf der Systemroutinen im Installationsbereich. n ist der Adressoffset zum Aufruf von CI (Zeicheneingabe).

OVER 16b1 16b2 -- 16b1 16b2 16b3
16b3 ist die Kopie von 16b1 .

P! n adr --
Direkte Portausgabe, wobei n der auszugebende Wert und adr die Adresse des Ports ist.

P@ **adr -- n**
Direkte Porteingabe, wobei n der einzugebene Wert und adr die Adresse des Ports ist.

PAD **-- adr**
Anfangsadresse eines mindestens 84 Byte langen Textzwischenpuffers.

PARSE **+n1 -- adr +n2**
Separiert im Eingabestrom eine Zeichenkette, die mit +n1 endet, liefert die Anfangsadresse adr und die Laenge +n2 der Zeichenkette. Aktualisiert den >IN-Zeiger.

PARSE-WORD **+n1 -- adr +n2**
Separiert im Eingabestrom eine Zeichenkette, die mit +n1 beginnt, liefert die Anfangsadresse adr und die Laenge +n2 der Zeichenkette. Aktualisiert den >IN-Zeiger.

PAUSE **--**
Ist eine leere Anweisung. Sie ist fuer Multitask reserviert, wo sie eine Umschaltung auf das naechste Task bewirkt.

PICK **+n -- 16b**
16b ist die Kopie des n-ten Stackeintrags (+n nicht mitgerechnet). 0 PICK entspricht DUP , 1 PICK entspricht OVER .

PLACE **adr1 n adr2 --**
Lagert Zeichenkette der Laenge n von adr1 (Quelladresse) nach adr2 (Zieladresse) um.

PREVIOUS **--**
Vok.: ROOT
entfernt als Gegenstueck zu ALSO das zuletzt deklarierte Woerterbuch aus der Suchreihenfolge.

PRINTING **-- adr U**
USER-Variable, die anzeigt, ob Drucker als Echo zugeschaltet ist. PRINTING ON bzw. OFF schaltet den Drucker zu bzw. ab.

PRIOR **-- adr**
Ist eine Variable, die zur Woerterbuchsuche verwendet wird. Zeigt zum letzten Woerterbuch, das durchsucht wurde.

PUT **--**
Speichert den Inhalt der RAM-Disk auf Magnetband ab. Beachte rechnerspezifische Unterschiede.

QUERY **-- M,**
Empfangen von 80 Zeichen in den TIB und zwar so lange, bis <ret> erkannt wird oder der Textpuffer gefuellt ist. >IN und BLK werden auf 0 gesetzt, #TIB enthaelt den Wert von SPAN .
Vgl. EXPECT

QUIT --
Fuehrt folgende Operationen aus: Returnstack saeuubern, ausfuehrenden Modus einschalten, Eingabe ueber das aktuelle Eingabegeraet, Beginn der Wortinterpretation. Keine Systemauschrift.

R# -- adr U "r-sharp"
Ist eine Systemvariable, die den Ort eines Editcursors oder anderer mit Files zusammenhaengender Funktionen enthalten kann.

R> -- 16b C,
Entnahme von 16b vom Returnstack und Transfer zum Parameterstack.

R@ -- 16b C,
Kopiert den obersten Wert des Returnstacks als 16b auf den Datenstack.

READ adr n --
Vok.: I/O
liest aus der RAM-Disk Block n und schafft ihn auf Adresse adr. Standartvoreinstellung fuer READ-BLOCK.

READ-BLOCK adr n -- Vec
liest Block n von (RAM) Disk nach Adresse adr.

RECURSE -- C, I,
Compiliert die Compilationsadresse der Definition, die gerade compiliert wird. Ausfuehrung erfolgt spaeter rekursiv.

REPEAT -- C, I,
sys -- (comp.Mod.)
Verwendung in BEGIN...WHILE...REPEAT. Zur Ausfuehrungszeit Sprung von REPEAT zur Stelle hinter BEGIN. sys wird mit WHILE ausbalanciert. Vgl. BEGIN

REVEAL --
Replaziert die letzte Definition in das Hauptwoerterbuch.

ROLL +n --
Der +n - te Stackeintrag (+n nicht mitgerechnet) wird an die oberste Stackposition rotiert, die folgenden Eintraege gelangen auf die jeweils frei werdenden Plaetze. 2 ROLL entspricht ROT , 0 ROLL ist eine Nulloperation.

RMARGIN -- adr
Ist eine Variable. Der Inhalt der Adresse (adr) entspricht der Anzahl der Stellen, die bei einer Ausgabe auf Bildschirm oder Drucker rechts maximal ausgegeben werden. Wird bisher nur von WORDS genutzt.

ROOT -- Vok
Das Vokabular ROOT enthaelt die zur Steuerung der Suchreihenfolge notwendigen Worte und basiert auf den Experimentalvorschlaegen von ...
(s. auch Der Standart FORTH 83)

ROT 16b1 16b2 16b3 -- 16b2 16b3 16b1
Die 3 obersten Stackeintraege werden rotiert, der drittoberste gelangt an die Spitze.

RP0 -- adr U "r-zero"
Ist eine Systemvariable, die die Ausgangsposition des Returnstack-Zeigers enthaelt.

RP! adr -- "r-p-store"
Setzt Returnstackanfang auf die Adresse adr.

RP@ -- adr "r-p-at"
Hinterlaesst den aktuellen Returnstack-Zeiger-Wert auf dem Datenstack.

S>D n -- d
Wandelt eine vorzeichenbehaftete Einfachintegerzahl n in eine Doppelintegerzahl d.

SAVE-BUFFERS -- M, "save-buffers"
Die Inhalte aller als aktualisiert gekennzeichneten Blockpuffer werden in die entsprechenden Massenspeicherbloecke zurueckgeschrieben, alle Puffer werden nicht laenger als aktualisiert gekennzeichnet, bleiben aber u.U. zugewiesen.

SCAN adr1 +n1 n2 -- adr2 +n3
Aus einer Zeichenkette mit der Adresse adr1 und der Laenge +n1 wird nach einem Zeichen n2 gesucht. Auf dem Datenstack wird nach der Ausfuehrung des Wortes die Adresse adr2 des gefundenen Zeichens n2 und die Laenge der verbleibenden Zeichenkette (+n3).

SCR -- adr "s-c-r"
Adresse der Variablen, die die Nummer des zuletzt ausgegeben Screens enthaelt. Vgl. LIST

SEAL -- Vok.: ROOT
SEAL <name> macht das Woerterbuch <name> zum alleinig durchsuchten. Kann z.B. fuer versiegelte Applikationen verwendet werden.

SIGN n --
Bei $n < 0$ Einfuegen eines ASCII-Zeichens '-' (Minuszeichen) an die Zeichenkette der Ausgabekonvertierung. Anwendung innerhalb <# ... #>.

SKIP adr1 +n1 n2 -- adr2 +n3

Aus einer Zeichenkette mit der Adresse adr1 und der Laenge +n1 wird nach einem Zeichen +n2 gesucht. Auf dem Stack liegen nach Ausfuehrung des Wortes die Adresse adr2 des ersten Zeichens nach dem gefundenen Zeichen +n2 und die Laenge der verbleibenden Zeichenkette (+n3).

SOD -- adr Vok.: I/O

Uebergibt die Adresse im Boot-Bereich, auf der die erste von der RAM-Disk benutzte Adresse steht.

SOURCE -- adr n

Liefert Quelladresse adr und Laenge n des Eingabestromes.

SP0 -- adr U "s-zero"

Variable, die den Initialisierungswert des Datenstackpointers enthaelt.

SP! adr -- "s-p-store"

Setzt Datenstackanfang auf adr.

SP@ -- adr "s-p-at"

adr ist die Adresse des obersten Stackeintrags, bevor SP@ ausgefuehrt wurde.

SPACE -- M,

Ausgabe eines ASCII-Leerzeichens.

SPACES +n -- M,

Ausgabe von +n ASCII-Leerzeichen.

SPAN -- adr

Adresse der Variablen, die die Laenge der zuletzt mit EXPECT empfangenen Zeichenkette enthaelt. Vgl. EXPECT

STATE -- adr

Adresse der Variablen, die die Systemzustandsinformation enthaelt (compilierender Modus: Inhalt ungleich 0)

STATUS -- Vec

Ausfuehrungsvektor, der jeweils nach Abarbeitung einer Eingabezeile ausgefuehrt wird und den Systemstatus meldet. (Normalerweise CR, sinnvoll ist fuer Testfaelle die Umstellung mit ' .S IS STATUS)

SWAP 16b1 16b2 -- 16b2 16b1

Die 2 obersten Stackeintraege werden getauscht.

THEN -- C, I,

sys -- (comp.Mod.)

In IF...ELSE...THEN oder IF...THEN verwendet markiert THEN

die Stelle, wo die Ausfuehrung nach der Abarbeitung des IF - bzw. des ELSE - Zweigs fortgesetzt wird. sys wird mit IF bzw. ELSE ausbalanciert.

THRU u1 u2 -- M,

Lade die aufeinanderfolgenden Bloecke von u1 bis u2.

TIB -- adr "t-i-b"

Adresse des Texteingabepuffers, darin Speicherung der Zeichen des Eingabestroms vom aktuellen Geraet.

TOS -- adr

User-Variable, die bei Taskumschaltung den TOS des jeweiligen Task zwischenspeichert.

TRIM adr voc-adr --

Vok.: HIDDEN

Stutzt die 4 Hash-Faeden eines Vocabulars so, dasz sie kleiner als adr sind.

TRUE -- n

Ist eine Konstante und liefert den Wert n=-1 (0FFFFH).

TUCK w1 w2 -- w2 w1 w2

Verdoppelt den letzten Stackeintrag und bringt diesen Wert auf die 3. Stackposition.

TYPE adr +n -- M,

+n Zeichen werden fortlaufend ab adr aus dem Speicher gelesen und ausgegeben, falls +n > 0.

U. u -- M, "u-dot"

Formatfreie Ausgabe von u als vorzeichenlose Zahl.

U.R u +n -- M, "u-dot-r"

Konvertierung von u bezueglich der aktuellen Zahlenbasis, rechtsbuendige Ausgabe in einem gedachten Feld der Weite +n. Fehler: Feldweite reicht nicht zur Darstellung aller Zeichen aus. Vgl. "Zahlenkonvertierung"

U2/ u1 -- u2

u2 ist das Ergebnis der arithmetischen Rechtsverschiebung von u1 um 1 Bit. In die frei werdende Bitposition wird eine Null geschoben.

U< u1 u2 -- flag "u-less"

Das flag ist wahr, falls u1 < u2.

UM* u1 u2 -- ud "u-star"

ud ist das vorzeichenlose Produkt u1 * u2.

UM/MOD ud u1 -- u2 u3 "u-m-divide-mod"
 u2 ist der Rest und u3 der Quotient nach der Division u1/u2. Fehler: Division durch 0 oder Quotient ausserhalb 0...65 535 . Vgl. "Division mit Rundung"

UNTIL flag -- C, I,
 sys -- (comp.Mod.)
 Markiert das Ende in einer Schleife BEGIN...flag UNTIL. Schleifenabbruch, falls flag wahr ist, sonst Fortsetzung der Abarbeitung hinter BEGIN. sys wird mit BEGIN ausbalanciert. Vgl. BEGIN

UP -- adr "user-pointer"
 Ist eine System-Variable. Enthaelt die Adresse des aktuellen USER-Areas.

UPD -- adr Vok.: HIDDEN
 Variable, die anzeigt, ob der Block im Buffer UPDATE ist.

UPDATE --
 Kennzeichnung des aktuell gueltigen Blockpuffers als aktualisiert. Vor Aenderung des Blockinhalts (Beschreiben mit einem anderen Blockinhalt oder Ausfuehrung von FLUSH oder SAVE-BUFFERS) erfolgt automatisches Rueckschreiben zur RAM-Disk.

USER -- Voc, M
 Vokabular zur Definition von USER-Variablen und -Vektoren. Die Definition dieser Worte weicht vom bisher bekannten etwas ab. Die Position im USER-Area wird vom System selbst verwaltet. Die Definition einer Variablen lautet dann:
 USER VARIABLE name
 Die eines USER-Vektors:
 USER DEFER name

VARIABLE -- M,
 Definitionswort fuer 16-Bit-Variablen: VARIABLE Wortname. Bei Ausfuehrung von Wortname wird die Parameterfeldadresse auf dem Stack abgelegt.

VARIABLE --
 Voc.: USER
 Definiert eine USER-Variable und ist somit aequivalent zum Befehl USER in FIG.

VOCABULARY -- M,
 Definitionswort fuer Woerterbuecher: VOCABULARY Wortname.
 Eintragen von Wortname ins Woerterbuch zur Spezifizierung einer neuen geordneten Liste von Wortdefinitionen. Die Ausfuehrung von Wortname ersetzt das erste Woerterbuch in der Suchreihenfolge durch Wortname. Wenn Wortname Compilationswoerterbuch ist, so werden neue Definitionen an die Liste von

Wortname angefügt.

Vgl. DEFINITIONS

VOC-LINK -- adr

Ist eine Systemvariable, welche die Adresse eines Feldes in der Definition des zuletzt erzeugten Woerterbuches enthaelt. Alle Woerterbuchnamen sind ueber diese Felder verkettet, um eine Steuerung von FORGET ueber mehrere moegliche Woerterbuecher zu ermoeöglichen.

VOCS -- Vok.: ROOT

zeigt alle definierten Vokabulare.

WARM --

Die Warmstartprozedur liefert eine Systemmeldung und fuehrt ABORT aus.

WARNING -- addr

Systemvariable. Wenn ihr Inhalt $\neq 0$ ist, wird bei Redefinition eines Wortes eine entsprechende Warnung ausgegeben.

WHERE off n -- Vec

Lokalisiert Screen n und Position des Wortes, dass einen Fehler verursachte.

WIDTH -- adr U

Ist eine Systemvariable, welche die Hoechstzahl von Zeichen enthaelt, die bei der Uebersetzung eines Definitionsnamens abgespeichert werden. Der Wert muss zwischen 1 und 31 liegen und wird mit 31 vorbelegt.

WHILE flag -- C, I,
sys1 -- sys2 (comp.Mod.)

In BEGIN...flag WHILE...REPEAT wird die Abarbeitung fuer flag = wahr zwischen WHILE und REPEAT fortgesetzt, anschliessend Ruecksprung zur Stelle hinter BEGIN. Bei flag = falsch Fortsetzung der Abarbeitung hinter REPEAT.

sys1 wird mit BEGIN, sys2 mit REPEAT ausbalanciert.

WORD char -- adr M,

Erzeugen einer Teilzeichenkette (Token) als FORTH-Zeichenkette aus dem aktuellen Eingabestrom. Abschluss durch char oder bei Erschoepfung des Eingabestroms. Fuehrende Trennzeichen char werden ignoriert. Die Laenge des Token wird auf adr, die Zeichenkette ab adr+1 als Bytefolge mit nachfolgendem Leerzeichen (zaehlt nicht zur Laenge) abgespeichert. Laenge der Zeichenkette: maximal 255 Byte, sonst unbestimmt. Bei erschoeptem Eingabestrom zum Zeitpunkt des Aufrufs von WORD Generierung einer Zeichenkette der Laenge 0. Wenn char nicht gefunden wird, ist der Inhalt von >IN die Laenge des Eingabestroms, sonst wird >IN so korrigiert, dass der Offset hinter

die Teilzeichenkette zeigt. #TIB wird nicht veraendert. Die generierte FORTH-Zeichenkette wird nach HERE transportiert und ihre Adresse uebergeben.

WORDS -- M
Ausgabe aller Wortnamen des ersten Woerterbuchs der aktuellen Suchreihenfolge.

WORDS --
Vok.: ROOT
Redefinition. s. Vokabular FORTH.

WRITE adr n -- Vok.: I/O
schreibt Block n von Adresse adr in die RAM-Disk. Standart-Voreinstellung fuer WRITE-BLOCK.

WRITE-BLOCK adr n -- Vec
schreibt Block n, welcher auf Adresse adr steht zur (RAM) Disk.

XOR 16b1 16b2 -- 16b3 "x-or"
16b3 ist das Ergebnis der bitweisen Exklusiv-ODER-Verknuepfung von 16b1 mit 16b2.

[-- I, "left-bracket"
Einschalten des ausfuehrenden Modus, nachfolgend Interpretation des Textes aus dem Eingabestrom. Typische Anwendung: Vgl. LITERAL und]

['] -- adr C, I, M, "bracket-tick"
Mit ['] Wortname wird die Compilationsadresse von Wortname in eine Doppelpunktdefinition compiliert. Bei deren spaeterer Ausfuehrung wird adr auf dem Stack abgelegt. Fehler: Wortname kann nicht in der aktuellen Suchreihenfolge gefunden werden. Vgl. LITERAL

[COMPILE] -- C, I, M, "bracket-compile"
In [COMPILE] Wortname wird das IMMEDIATE-Wort Wortname compiliert und nicht ausgefuehrt.

\ -- "skip-line"
Auf \ folgender Kommentar in der Screenzeile wird bei der interpretativen Abarbeitung des Screens ignoriert.

\S -- "skip-screen"
Beendet die Abarbeitung des Screens.

] -- "right-bracket"
Einschalten des compilierenden Modus mit nachfolgender Compilation des Textes aus dem Eingabestrom. In diesem Wort steckt die Hauptcompilerschleife (anders als in fig). s. RUN INTERPRET



From:
<https://hc-ddr.hucki.net/wiki/> - **Homecomputer DDR**

Permanent link:
<https://hc-ddr.hucki.net/wiki/doku.php/forth/fgforth/glossar?rev=1367391242>

Last update: **2013/05/01 06:54**

