Das Programmsystem FORTH

fuer den Kleinrechner-Bausatz Z 1013.

Bearbeiter: B. Schubert Leipzig 11/87

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

0. Einleitung
1. Speicherkonzept

2. Abweichungen vom FIG-Standard
2.1 Reihung, Sprachumfang
2.2 Kassettenarbeit mit FCB
2.3 Abweichungen im wortinternen Aufbau

3. Erweiterungen des Wortschatzes
3.1 Worte aus dem F 79 - Standard und Hilfsworte
3.2 Editor
3.3 Case-Struktur
3.4 Dump
3.5 Disco

4. Systemdatenbereiche
4.1 Cold-Start-Area
4.2 Anpassungs-Area

5. Literaturhinweise

6. Glossar
6.1 Operanden
6.2 Stackbewegungen
6.3 Terminal Eingabe/Ausgabe
6.4 Zahlensysteme
6.5 Eingabe-Ausgabe-Formatierung
6.6 Massenspeicher (Diskette/Kassette)
6.7 Stack-Manipulationen
6.8 Speicherbezogene Befehle
6.9 Arithmetik
6.10 Vergleichsoperatoren
6.11 Logische Befehle
6.12 Strukturierende Worte
6.13 Definitionsworte
6.14 Vokabulare
6.15 Systemworte und Diverses

0. Einleitung

FORTH ist ein 1971 von Chales Moore entwickeltes Sprach- und Systemkonzept, das mit wenigen Betriebssystemschnittstellen arbeitsfaehig ist und sich daher besonders fuer Kleinrechner, Heimcomputer, industrielle Steuerungen und aehnliche Anwendungsfaelle eignet.
In der DDR ist FORTH im Vergleich zu anderen Hochsprachen noch relativ wenig verbreitet.
Die vorliegende FORTH - Version entstand aus einer Arbeitsversion eines FORTH - Systems fuer den KC 85/1 (Z 9001), die an der Wilhelm Pick Universitaet entwickelt wurde und dem Robotron Computerclub zur Verfuegung stand.
Nach der Konvertierung auf einen Z 1013 wurde das System nahezu ausschliesslich mit sich selbst neu erstellt. Dabei wurden zunaechst die Schnittstellen angepasst und vorhandene Adressfehler in einigen Worten beseitigt. Als Vorbild fuer die Erweiterungen zu einem komfortablen Entwicklungssystem diente vorwiegend eine Version von "Microsystems Los Angeles", in der DDR bekannt als "FORTHLE.COM".
Von diesem System wurden einige Funktionen dem Sinn nach uebernommen und fuer das vorliegende System zugeschnitten.

1. Speicherkonzept

Das vorliegende System benutzt einen RAM - Breich ab 100H.
Der Kaltstart beginnt auf der Adresse 334H. Entsprechend der Konvention fuer CP/M ist fuer eine uebersichtliche Programmgestaltung der Kaltstart auf die Adresse 100H mit einem Zwischensprung vorverlegt.
Der Bereich von 100H bis 200H ist frei und kann fuer Anpassungen (z.B. fuer Save/Load Routinen anderer Formate) genutzt werden.
Im KC 85/1 ist dieser Bereich vom Betriessystem belegt.
Ab 200H beginnen die Routinen fuer die Anpassung an das Betriebssystem (Anpassungsarea), der Bereich der User-Variablen und der Returnstack.
Die Primitivworte fuer die Schnittstellenanpassung an das Betriebssystem wurden voellig neu gestaltet bzw.beseitigt.
Das System wurde so umgearbeitet, dass alle hardwarespezifischen Routinen fuer Ein- und Ausgaben auf diesen speziellen Anpassungsbereich zugreifen.
Auf Adresse 300H steht ein Sprung zum Warmstart des Systems (339H) und dahinter das zugehoerige Aufrufwort entsprechend dem Format fuer den KC 85/1 (WFORTH).
Gegenueber anderen Sprachkonzepten (z.B. CAOS-FORTH fuer KC 85/2) ist dieses FORTH so aufgebaut, dass es Hilfsmittel besitzt, sich selbst zu vervielfaeltigen (SAVE).
Dadurch kann die aktuelle Laenge des FORTH beim Abspeichern jeweils verschieden sein. Die Endadresse des abzuspeichernden Systems wird immer auf XXFFH aufgerundet.
Der vom FORTH - System waerend der Arbeit genutzte Speicherbereich ist in seiner Grundform auf 4000H d.h. 16 kByte eingestellt.

Damit wird vielen Anwendern die Moeglichkeit gegeben, ohne Speichererweiterung arbeiten zu koennen. Einschraenkungen des vorhandenen Wortschatzes treten dadurch nicht auf.
Bei Speichererweiterungen kann durch Aendern der Konstanten "LIMIT" auf die erste nicht mehr verfuegbare Speicheradresse der Bereich fuer die Screen-Puffer beliebig geaendert werden. Ueber einen anschliessenden COLD oder WARM Aufruf wird die Anzahl der SCR - Puffer neu berechnet (#BUFF). Beim Abspeichern eines so geaenderten Systems enthaelt das System die neuen Parameter!
Eine Verschiebung des Datenstacks und der Woerterbuch-obergrenze fuer groessere Erweiterungen obliegt den jeweiligen Anforderungen und den Systemkenntnissen des Anwenders.

2. Abweichungen vom FIG - Standard

2.1 Reihung, Sprachumfang

FORTH stellt eine sogenannte "offene" Sprache dar, d.h. ihr Sprachumfang ist theoretisch unbegrenzt und wird vom Anwender staendig erweitert. Daher muss bei der Implementierung des Systems eine fuer den nwender sichtbare Grenze geschaffen werden die kennzeeichnet, wo der systembedingte Grundwort-schatz endet und die Anwenderworte beginnen. Diese Grenze stellt im allgemeinen das Wort "TASK" dar, welches aus einer Leerdefinition besteht.
Unterhalb dieses Wertes wird das FORTH ueblicherweise mit Hilfe eines Assemblers uebersetzt. Dabei werden Assemblerworte (Primitive) direkt uebersetzt und FORTH - Worte (Secondarys) mit DA (Definierte Adresse) aufgebaut.
Bei dieser Vorgehensweise ist es nicht zwingend notwendig in allen Faellen die FORTH - typische Arbeitsweise ein Wort nur aus vorangegangenen zu erstellen einzuhalten.
Im vorliegenden System wurde darauf geachtet, diese FORTH - Reihung weitgehend einzuhalten und die Anzahl an Vorwaets-verweisen zu hoehern Worten zu minimieren.
Aus diesem Grund wurden alle Worte oberhalb "(" neu geschrieben.
Das hoechste mit Vorwaertsverweis adressierte Wort ist "MESSAGE". Fuer den Programmierer gilt die Woerterbuch-grenze "TASK", der erfahrene Systemprogrammierer hat die Moeglichkeit alle Wort oberhalb "MESSAGE" zu vergessen ohne einen Absturz befuerchten zu muessen.

2.2 Kassettenarbeit mit FCB

FORTH ist ein Sprachsystem das normalerweise Diskettenarbeit beinhaltet. Die Diskette kann von FORTH als virtueller Speicher ueber ein Fenster (Screen - Puffer) verwaltet werden. Meisst wird dabei die gesamte Diskette fuer Screens verwendet. Bei der Arbeit mit CP/M ist das mitunter ein Nachteil, der bei "FORTHLE.COM" durch Vergabe von Screen-File-Namen behoben wird. So ist ein ubersichtliches Sortieren von Screens fuer verschiedene Anwendungsfaelle und deren Verwaltng vom Betriebs-system aus moeglich.

In Anlehnung daran wurde das Kasetten - Screen - Konzept entwickelt.
Ein Screen beinhaltet 0,5 kByte die mit einem Kopfblock zusammen auf Kassette abgelegt werden. Der im Screenkopf verwendete Name wird beim Kalt- und beim Warmstart in der Systemausschrift angegeben und kann mit dem Wort "USING" dem System neu zugewiesen werden.
Der Aufbau des Kopfblockes entspricht vollstaendig dem HEADER-Save/Load fuer den Z 1013. Als Anfangsadesse wird Standardmaessig die Bildschirmadresse "EC00H" eingetragen, so dass ein Screen auch mit Hilfe des HEADER-Save/Load - Programms ohne Laden des FORTH schnell gesichtet bzw. kopiert werden kann.

2.3 Abweichungen im wortinternen Aufbau

Bei einigen implementierten Worten weicht der wortinterne Aufbau etwas von der Empfehlung des FIG ab. So wurden u.a. in einzellnen Worten eigene Worte und F 79 - Worte verwedet, um die Imlementierung effektiver zu gestalten.
In anderen Worten wurde die innere Struktur so geaendert, dass sich eine effektvollere Bedienung ergibt.
Das betrifft insbesondere folgende Worte:

EMPTY-BUFFERS	Dieses Wort berechnet zusaetzlich die Anzahl der moeglichen Screen-Puffer zwischen FIRST und LIMIT auch wenn LIMIT willkuerlich veraendert wurde nd traegt diese Zahl in #BUFF ein.
FLUSH	Dieses Wort ist so veraendert, dass auch Screens oberhalb 8000H erreicht werden.
+BUF	Dieses Wort weist in Abhaengigkeit von LIMIT auch Puffer oberhalb 8000H zu, LIMIT muss nicht mehr direkt auf Pufferobergrenze berechnet werden. Das wird von EMPTY-BUFFERS und +BUF uebernommen.
VLIST	Damit der Bildschirm uebersichtlich bleibt wurde VLIST in 3 Funktionen veraendert: -Auflisten in 2 Kolonnen mit Ausgabe der NFA. -Anhalten und Fortsetzen des Listens mit der Leertaste, andere Tasten brechen ab. -Analog zum FORTH 83 wird nur das jeweils aktuelle Vokabular aufgelistet.

LIST

Dieses Wort erkennt die eingegebene SCR - Nr. stets als Dezimalwert an, solange sie 2 Stellen nicht ueberschreitet, auch wenn eine andere Zahlenbasis eingestellt wurde. Werden Hexazahlen (Buchstaben) eingegeben, so werden sie in Dezimalzahlen umgerechnet. Alles darueber Hinausgehende wird mit Fehler quittiert.

.S

Die Darstellung erfolgt so, dass der TOS oben ist.

BYE

Ist unterteilt in eine innere Routine (BYE) die den Sprung zum Betriebssystem enthaelt (F000H) und eine Verkleidung.

QUIT/COLD/ABORT

Wurden weitgehend den Routinen des "FORTHLE.COM" nachgebildet.
So kann auf Adresse 2AH +ORIGIN die CFA eines Anwenderwortes eingetragen werden, welches beim Eintritt in das System durch ABORT aktiviert wird.
Eine zweite Startroutine kann auf der Adresse 2CH +ORIGIN eingetragen werden, die stets durch das Wort QUIT aktiviert wird (z.B. Einstellung HEX)

FORGET

Dieses Wort weist oft erhebliche Einschraenkungen oder Fehler auf.
Es wurde so erstellt, dass ein Vergessen ueber mehrere Vokabulare moeglich ist. Die Vokabularzeiger werden dabei so korrigiert, dass es nicht zu Abstuerzen kommt.

BLOCK,BUFFER,R/W
BLK-READ,BLK-WRITE
SET-IO

Wurden den Erfordernissen an die Kassettenarbeit mit dem Wort SAVE angepasst. Alle Worte wurden nur soweit veraendert, dass die uebergabe der Parameter beim Aufruf und beim Verlassen den Konventionen des FIG - Standards entsprechen.
Der Wiedereintritt in das FORTH - System ueber Adresse 300H erfogt so, dass die Screen - Puffer erhalten bleiben.

3. Erweiterung des Wortschatzes

3.1 Worte aus dem F 79 - Standard und Hilfsworte

2- , 2* , 2/	Arithmetik mit 2
ROLL	n - tief Stack rotieren (2 ROLL = SWAP , 3 ROLL = ROT)
PICK	n - tes Element vom Stack auf TOS (2 PICK - OVER)
DEPTH	uebergibt Tiefe der Stacknutzung
EXIT	vorzeitiges verlassen eines Wortes vor;S
ASCII	legt Wert des folgenden Zeichens auf Stack
FREEZE	einfrieren der Systemdaten auf derzeitigen Stand

Weiterhin wurden zusaetzlich folgende Worte implementiert:

USING	Aenderung der Zuweisung des File- namens fuer Screen.
QX	Quick-Index, es werden die 1. Zeilen (Indexzeilen) aller im Speicher be- findlichen Screens aufgelistet. Be- arbeitete Screens, (UPDATE) werden mit einem * gekennzeichnet.
BSPACES	Loescht die letzten Ausgaben durch Backspace in der Laenge entsprechend OUT.
SAVE	Ohne Parameter legt das FORTH - System von 100H bis HERE auf Kassette ab. Zur Erleichterung beim Laden wird zuvor Anfangs- und Endadresse auf dem Bildschirm angezeigt und auf Be- staetigung mit ENTER gewartet, andere Tasten brechen Funktion ab. Die hoechste abzuspeichernde Adresse wird auf XXFFH aufgerundet. Mit Namenangabe wird zusaetzlich ein Kopfblock erzeugt, der dem Aufbau fuer HEADER-Save/Load entspricht

3.2 Editor

Der Editor ist als ein zusaetzliches Vokabular vereinbart. Er gehoert also nicht zum Grundwortschatz des Systems. Er ist mit minimalen Aufwand fuer eine zeilenorientierte Eingabe von Screens ausgelegt. Fuer den Nutzer sind nur folgende Worte von Bedeutung:

n P	Eingabe des folgenden Textes in Zeile n.
n D	Loeschen der Zeile n, alle anderen Zeilen ruecken nach oben.
n I	Einfuegen des folgenden Textes in Zeile n, alle anderen Zeilen ruecken nach unten.
L	Listen des aktuellen Screens.
n CLR	Loeschen des Screens n mit Leerzeichen und listen, es erfolgt kein laden des Screens.
alt/neu RENUMBER	Umnummerieren des betreffenden Screen - Puffers im Speicher, gegebenenfalls wird der Screen geladen.

3.4 Case - Sruktur

Die Case Struktur stellt eine Erweiterung der strukturierenden Worte dar underleichtert das Testen eines Wertes auf verschiedene Bedingungen.
Die vorliegende Version testet nur auf Gleichheit de TOS mit den Testwerten, was fuer viele Zwecke schon eine grosse Erleichterung ist.
Die Anwendung sieht wie folgt aus:

```
          .
          .
          . Wert x auf TOS
CASE
n ON  ..... Anweisungen ..... OFF
o ON  ..... Anweisungen ..... OFF
p ON  ..... Anweisungen ..... OFF
q ON  ..... Anweisungen ..... OFF
          .
          .
ENDCASE
          .
```

Der Wert x bleibt auf dem TOS erhalten wenn keine der Bedingungen n...q zutrifft.

3.4 Dump

Der DUMP ist eine Funktion zur Anzeige des Speichers. Es erfolgt eine Anzeige in zwei Zeilen, erst die Hexadezimaldarstellung des Speicherinhaltes, dann die zugehoerige Darstellung als ASCII - Zeichen, soweit moeglich. Steuerzeichen und Grafikzeichen werden als Punkt dargestellt.
Der Aufruf erfolgt: anfang laenge DUMP
Jede Tastaturbetaetigung laesst den Dump um 8 Adressen weiterlaufen, wobei die ENTER - Taste den Abbruch erzwingt.

3.5 Disco

Die Funktion Disco stellt einen DIS - Compiler fuer minimale Ansprueche dar. Disco ermoeglicht das Betrachten des inneren Aufbaus von FORTH - Woertern.
Der Aufruf erfolgt: DISC `Wort'
Es erscheint auf dem Bildschirm die Namenfeldadresse des Wortes, das Wort selbst und die Codefeld - Adresse.
Wird das Wort nicht gefunden, erscheint eine Fehlermeldung.
Unter den Kopfangaben des Wortes wird die Parameterfeld-Adresse mit anschliessenden Doppelpunkt ausgegeben. Hier muss der Bediener entscheiden wie weiter analysiert werden soll. Durch druecken der Taste `N' fuer `NEXT' wird zeichenweise der Speicherinhalt mit moeglicher ASCII - Interpretation angezeigt. Das gleiche gilt fuer die Eingabe von `B' fuer `BLOCK'.
Das betaetigen der Leertaste bewirkt ein wortweises Interpretieren des Parameterfelinhaltes mit Anzeige des des jeweils hineincompilierten Wortes bzw. Textes. Ist die PFA identisch mit dem Inhalt der CFA, so handelt es sich um ein Primitivwort und `Disco' bricht ab.
Durch druecken der ENTER - Taste kann Disco jederzeit verlassen werden. Ist es von Bedeutung welche CFA in einem Wort enthalten sind bzw. welchen Wert eine Konstante oder eine Variable besitzt, so kann der Inhalt des Parameterfeldes durch Druck einer beliebigen anderen Taste wortweise (adressweise) angezeigt werden.
Da Disco aus Aufwandsgruenden die Worte nicht nach ihrer Art unterscheidet, liegt es in der Hand des Bedieners, zu entscheiden wie ein Wort analysiert werden soll. Anhand der angezeigten CFA kann der Bediener erkennen um welche Art Wort es sich handelt.
Bei normalen Secondary - Worten bricht `Disco' bei errichen des Semis ( ,S ) am Wortende ab. Bei Worten ohne Abschluss (z.B. Endlosschleifen im System wie `INTERPRET') ist es ratsam, die laenge des Wortes vorher im Dump anzusehen oder die Interpretation des Parameterfeldes zu vermeiden. Das gilt auch fuer die Analyse von Worten wie Konstanten, Variablen, Uservariablen und Worten mit Headerless - Code.

In solchen Faellen versucht `Disco'
den Inhalt des Parameterfeldes vergeblich zu interpretieren, findet jedoch kein zugehoeriges Namenfeld. Die Folge ist oftmals ein Programmabsturz. Ist der Wortaufbau (durch adressweise Anzeige oder Dump) bekannt, so kann bis zur kritischen Stelle analysiert und dann mit ENTER abgebrochen werden.

Zusammenfassung der Bedienung:

N	next Charakter
B	back Charakter
Space	Interpretation
Enter	Abbruch
andere	adressenweise Inhaltsangabe

4. Systemdatenbereiche

4.1 Cold - Start - Area

Die Cold - Start - Area beginnt bei dieser FORTH - Version gemaess FIG - Standard am Programmbeginn auf der Adresse 0CH +ORIGIN. Das ist in diesem Fall 30CH. Sie beginnt mit dem Eintrag von LATEST und reicht bis 328H wo der aktuelle Zeiger fuer den Returnstack steht. Von diesem Bereich werden beim Kaltstart die Daten von 312H bis 322H in den Userdatenbereich kopiert.
Dieser beginnt dann wie ueblich mit den Eintragungen fuer:

- Stack
- Returnstack
- Textinputbuffer (TIB)
- WIDTH
- WARNING

usw.

Die Speicherplaetze 32AH und 32CH innerhalb des Systemdatenbereiches werden nach dem Vorbild von "FORTHLE" fuer den Eintrag von Kaltstartroutinen genutzt. Ebenso wird die Adresse 32EH abgefragt welcher Rechnertyp vorliegt und danach die Systemausschrift und die Saveroutine modifiziert. Der Inhalt dieser Zelle sollte nicht veraendert werden, das ist einer Anpassung an den KC 85/X vorbehalten.

4.2 Anpassungs - Area

Die Anpassungs - Area ist ein neu ins System aufgenommener Programmbereich der es dem Systemprogrammierer ermoeglicht, mit relativ geringen Aufwand eine Anpassung an andere Rechnertypen unter Beibehaltung aller Worte und Systemroutinen vorzunehmen. In diesem Speicherbereich sind die direkten Einspruenge in das jeweilige Betriebssystem des Rechners implementiert und mit Headerlessbezuegen oder Spruengen aus Primitivworten an das FORTH - System angebunden.
Die Adressbezuege zu den ausfuehrenden Routinen sind geschlossen ab der Adresse 200H abgelegt.

Sie bedeuten im Einzelnen:

Adresse	Bezug	Routine (Assembler)
200H	DA	EMIT
202H	JR	KEY (mit Worten)
204H	JR	?TERMINAL (Status)
206H	JR	CR (Carridge-Return)
208H	DA	OPEN for READ
20AH	DA	READ (Kassette)
20CH	DA	OPEN for WRITE
20EH	DA	WRITE (Kassette)

Ab Adresse 210H beginnen dann die Anpassungroutinen an das Betriebssystem. Bei Aenderungen und Ergaenzungen ist zu beachten, dass der Bereich bis 300H noch vom FORTH - System durch den Userdatenbereich 2C0H bis 2FFH aufwaerts und den Returnstack ab 2C0H abwaerts genutzt wird.
Die derzeitig implementierten Routinen belegen einen Raum bis 250H, ein Stackkonflikt wuerde erst bei einer Verschachtelung von mehr als 50 Funktionen ineinander auftreten. Die Stacktiefe reicht fuer allgemeine Anwedungsfaelle aus. Eine Aenderung der Sackadresse bzw. des Userdatenbereiches kann gegebenenfalls durch Aenderung der Systemdaten (30CH bis 322H) von erfahrenen Systemprogrammierern vorgenommen werden.

5. Literaturhinweise

Ekkehard Floegel
Forth on the Atari
Hofacher-Verlag Holzkirchen 1983

Ronald Zech
Die Programmiersprache Forth
Franzis-Verlag Muenchen 1983

Gyoergy Varga, Michael Krapp
Forth - eine interessante Programmiersprache
Wissenschaftliche Zeitschrift der TH-Ilmenau 30(1984)H. 3

**Verschiedene Beitraege
Zeitschrift Mikroprozessortechnik
VEB Verlag Technik Berlin 1 (1987) H. 6/H. 8
2 (1988) H. 2/H. 11**

Zur Einarbeitung in die Problematik der Programmiersprache sowie zur Erarbeitung des Systems fuer Z 1013 wurde weiterhin fogendes Lehrmaterial verwendet:

**M. Balig
Forth kurz und knapp,
Unterrichtsmaterial/Literaturrecherche
TH Leipzig 1987**

**Dokumentation zum System Pop-Forth
WPU Rostock 1987**

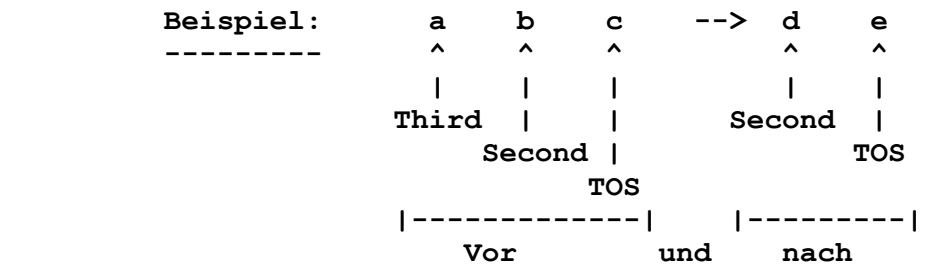
6. Glossar

Die haeufigsten FORTH - Befehle sind:

6.1 Operanden

n , n1	**16-Bit-Zweierkomplement-zahlen**
d , d1	**32-Bit-Zweierkomplement-zahlen**
u , u1	**vorzeichenlose 16-Bit-Zahlen**
ud	**vorzeichenlose 32-Bit-Zahlen**
addr	**Adresse**
b	**8-Bit-Byte**
c	**7-Bit-ASCII**
f	**Boolsches Flag (16-Bit #0 = wahr)**

6.2 Stackbewegungen



Abarbeitung des FORTH-Wortes

- Ein- und Ausgaben immer von links nach rechts.
- Der Top - Of - Stack (TOS) ist stets rechts aussen dargestellt.
- Second = Zahl unter dem TOS
- Third = Zahl unter dem Second

6.3 Terminal Eingabe / Ausgabe

.	(n -->)	Druckt die Zahl auf dem TOS aus (zerstoerend).
.R	(n Feldweite -->)	Druckt die Zahl (rechts adjustiert in Feld)
D.	(d -->)	Druckt doppelt genaue Zahl.
D.R	(d Feldweite -->)	Druckt 32-Bit-Zahl(rechts adjustiert in Feld).
CR	( )	Ausgabe eines Carridge-Return / Line-Feed.
SPACE	( )	Ausgabe eines Space-Character.
SPACES	(n -->)	Ausgabe von n Space-Characters.
."	( )	Druckt einen nachfolgenden Text aus, welcher mit " beendet wird.
Type	(addr u -->)	Druckt u Zeichen ab Adresse.
CONT	(addr-->addr+1 n)	Wandelt length-Byte-String in die Type Form.

?	(addr-->)	Druckt den Inhalt der Adresse.
?TERMINAL	(-->f)	Uebergibt den Tatatur-status ("bestaetigt"<>0)
KEY	(-->c)	Wartet auf Tastatureingabe und legt den Char. auf den Stack (ASCII).
EXPECT	(addr n -->)	Erwartet n Character (oder bis CR) und bringt sie nach addr.
EMIT	(c -->)	Gibt Character c aus.
WORD	(c -->)	Liesst ein Wort (bis zum Delimeter c) im gueltigen Eingabe-Buffer.

6.4 Zahlensysteme

DECIMAL	(-->)	Deklariert Dezimalsystem
HEX	(-->)	Deklariert hexadezimales Zahlensystem
Base	(-->addr)	System-Variable, welche die Zahlenbasis enthaelt.

6.5 Eigabe-Ausgabe-Formatierung

NUMBER	(addr-->d)	Wandelt einen String in addr um in 32-Bit-Zahl.
<#	( )	Eroeffnet Zahlenwandlung fuer Ausgabe (String).
#	(d --> d)	Wandelt naechste Stelle der Zahl und fuegt dem Ausgabe-String eine Ziffer hinzu (32-Bit-Zahlen !).
#S	(d --> 00)	Wandelt alle signifikanten Stellen um in String.
SIGN	(nd --> d)	Fuegt das Vorzeichen von n in den Ziffernstring ein.

#>	(d --> addr n)	Beendet Umwandlung in Ziffern-String (String hat passende Form fuer TYPE)
HOLD	(c -->)	Einfuegung eines ASCII-Characters in den String.

6.6 Massenspeicher (Diskette/Kassette)

LIST	(screen -->)	Ausdrucken eines Screen von Disk.
LOAD	(screen -->)	Laden eines Screen (Compilation oder Interpretation)
BLOCK	(block-->addr)	Liesst Disc-Block nach Adresse addr.
B/BUF	(-->n)	Systemkostante (Blockgroesse in Bytes)
BLK	(-->addr)	Systemvariable (aktuelle Block-Nummer)
SCR	(-->addr)	Systemvariable (enthaelt aktuelle Screen-Nummer)
UPDATE	( )	Markiert zuletzt benutzten Buffer als `updatet'.
FLUSH	( )	Schreibt alle `updated' Buffer auf die Disk.
EMPTY-BUFFERS	( )	Markiert alle Buffer als `leer'.

6.7 Stack-Manipulationen

DUP	(n --> n n)	Kopiert (dupliziert) den TOS.
-DUP	(n --> n ?)	Dupliziert nur dann, wenn ungleich Null.
DROP	(n -->)	Beseitigt den (aktuellen) TOS.
SWAP	(n1 n2 --> n2 n1)	Vertauscht die beiden oberen Zahlen des Stack.

OVER	(n1 n2 -->n1 n2 n1)	Kopiert den Second zum (neuen !) TOS.
ROT	(n1 n2 n3 -->n2 n3 n1)	Rotiert den Third zum TOS.
>R	(n -->)	Bringt den TOS zum Return-Stack (Zwischen-speicherung, Gebrauch mit Vorsicht !)
R>	(--> n)	Holt den Wert vom Return-Stack zum TOS zurueck.
R	(--> n)	Kopiert den Return-Stack-TOP zum TOS.

6.8 Speicherbezogene Befehle

@	(addr --> n)	Ersetzt Zellen-Adresse durch ihren Inhalt.
(@	(addr --> b)	Wie @, jedoch wird auf ein Byte zugegriffen.
!	(n addr -->)	Speichere Second in die Adresse auf dem TOS.
C!	(b addr -->)	Wie ! jedoch wird ein Byte abgespeichert.
+!	(n addr -->)	Addiere Second zum Inhalt der Adresse auf dem TOS.
CMOVE	(from to u -->)	Verschiebe u Bytes im Adressraum.
FILL	(addr u b -->)	Fuelle u Bytes im Speicher ab addr mit b.
ERASE	(addr u -->)	Fuelle u Bytes im Speicher ab addr mit Null.
BLANKS	(addr u -->)	Fuelle u Bytes im Speicher ab addr mit Blanks (20H).
TOGGLE	(addr b -->)	EXOR das Byte in Adresse addr mit Maske b.
SP@	(--> addr)	Uebergibt aktuelle Posi-tion des Stack-Pointers.

6.9 Arithmetik

+	(n1 n2 -->Summe)	Addition von 16-Bit-Zahlen (16-Bit-Summe).
D+	(d1 d2 -->Summe)	Addition von 32-Bit-Zahlen (32-Bit-Summe).
-	(n1 n2 -->Diff.)	Differenz n1 - n2
*	(n1 n2 -->Prod.)	16-Bit-Produkt zweier 16-Bit-Zahlen.
/	(n1 n2 -->Quot.)	16-Bit-Division mit 16-Bit-Ergebnis.
MOD	(n1 n2 -->Rest)	Modulo-Division (uebergibt Teiler-Rest)
/MOD	(n1 n2 -->Rest Quot.)	Division mit Rest und Quotient als Resultat.
*/MOD	(n1 n2 n3 -->Rest Quot.)	Multiplikation und anschliessende Division mit 32-Bit-genauen Zwischenergebnis (n1*n2/n3)
*/	(n1 n2 n3 --> Quot.)	Wie */MOD, jedoch lediglich Quotient.
M/MOD	(ud1 u2 --> u3 ud4)	Division einer vorzeichenlosen 32-Bit-Zahl mit Uebergabe des 16-Bit-Restes und des 32-Bit-Quotienten.
MIN	(n1 n2 --> Minimum)	Uebergibt die kleinere von zwei Zahlen.
MAX	(n1 n2 --> Maximum)	Uebergibt die groessere von zwei Zahlen.
ABS	(n --> u)	Bildet Absolutwert einer 16-Bit-Zahl.
DABS	(d --> ud)	Bildet Absolutwert einer 32-Bit-Zahl.
MINUS	(n --> -n)	Wechselt das Vorzeichen einer 16-Bit-Zahl.
DMINUS	(d --> -d)	Wechselt das Vorzeichen einer 32-Bit-Zahl.

1+	(n --> n+1)	Incrementiert den TOS mit 1.
2+	(n --> n+2)	Incrementiert den TOS mit 2.

6.10 Vergleichsoperatoren

<	(n1 n2 -->f)	Flag=1 wenn n1 kleiner n2
>	(n1 n2 -->f)	Flag=1 wenn n1 groesser n2
=	(n1 n2 -->f)	Flag=1 wenn n1 gleich n2
0<	(n -->f)	Flag=1 wenn TOS ngativ ist
0=	(n -->f)	Flag=1 wenn TOS gleich Null ist (negiert auch Wahrheitswert von Flags)

6.11 Logische Befehle

AND	(n1 n2 -->UND)	Bitweise logische UND-Verknuepfung
OR	(n1 n2 -->ODER)	Bitweise logische ODER-Verknuepfung
XOR	(n1 n2 -->EXOR)	Bitweise Exclusiv-ODER-Verknuepfung

6.12 Strukturierende Worte

DO...LOOP	(n1 n2 -->)	Schleife, Index laeuft von n2 bis n1-1 mit Increment = 1
DO...+LOOP	(n1 n2 -->)	Wie DO...LOOP, jedoch ist das Index-Increment hier (statt 1)nun beliebig (wird als zusaetzlicher Parameter an +LOOP ueber-geben)
I	(--> Index)	Loop-Index --> TOS
LEAVE	( )	Erzwingt Abbruch der Schleife bei naechster Gelgenheit.

IF.. (wahr).. ENDIF	(f -->)	Fuehrt Befehle aus wenn Flag=1 ist. Ist..(falsch).. dann ENDIF.
IF.. (wahr).. ELSE	(f -->)	dto., jedoch wird bei Flag=0 der FALSE-Teil ausgefuehrt
BEGIN...UNTIL	(--> f -->)	Schleife mit Abbruch, falls Flag fuer UNTIL=1
BEGIN.. WHILE.. REPEAT	(--> f -->)	Wie BEGINN...UNTIL, jedoch Abbruchtest am Anfang des des Schleifen-Kerns, REPEAT schliesst die Schleife bedingungslos nach BEGINN.
BEGIN...AGAIN		Endlos-Schleife

6.13 Definitionsworte

:xyz	( )	Beginn einer Colon-Definition mit Namen xyz.
;	( )	Abschluss einer Colon-Definition (SEMI-COLON).
VARIABLE XXX	(n -->)	Erzeugt eine Variable xxx, die mit n initialisiert ist (xxx uebergibt die Adresse bei Aufruf).
CONSTANT yyy	(n -->)	Erzeugt eine Konstante yyy, mit dem Wert n (bei Aufruf von yyy wird Wert uebergben).
CREATE zzz	( )	Eroeffnet Definition eines Primitive mit dem Namen zzz (Assembler- bzw. Maschinencode).
;CODE	( )	Abschluss einer Colon-Definition, wenn es sich um die Definition eines

		Definitionswortes handelte, wobei die Routine-Executive in Assembler definiert werden soll (Code hinter ;CODE).
<BUILDS..DOES> does:	(-->addr)	Wird zur Definition neuer Definitionsworte benutzt, wobei jedoch im Gegensatz zu ;CODE die Routine-Exective in high-level definiert wird.

6.14 Vokabulare

CONTEXT	(-->addr)	Uebergibt die Adresse eines Pointers zum Context-Vokabular (das zuerst abgesucht wird)
CURRENT	(-->addr)	Uebergibt die Adresse eines Pointers zum Current-Vokabular (das zur Zeit erweitert wird)
FORTH	( )	Name des Haupt-Vokabulars (setzt CONTEXT)
EDITOR, ASSEMBLER etc.	( )	Weitere Vokabular-Namen (setzen CONTEXT)
DEFINITIONS	( )	Macht Current-Vokabular zum Context-Vokabular
VOCABULARY xyz	( )	Deklariert ein neues Vokabular mit dem Namen xyz
VLIST	( )	Druckt die Namen aller Worte im Context-Vokabular

6.15 Systemworte und Diverses

(	( )	Eroeffnet Kommentar, der mit `)' abgeschlossen wird, nach `(' muss ein Space kommen
FORGET abc	( )	Vergisst alle neuen Definitionen ab (inclusive) abc
ABORT	( )	Erzwingt Fehler-Abbruch einer Operation
`xxx	(-->addr)	Findet die Adresse (PFA) des Wortes xxx im Dictionary (in Definitionen : compiliert die Adresse)
HERE	(-->addr)	Uebergibt die Adresse des naechsten freien Platzes im Dictionary
PAD	(-->addr)	Uebergibt die Startadresse eines Zwischenspeichers, meisst 68 Byte oberhalb von HERE
IN	(-->addr)	System-Variable, haelt Input-Buffer-Offset fuer WORD
ALLOT	(n -->)	Hinterlaesst eine ungenutzte Luecke (n Bytes) im Dictionary
`	(n -->)	Compiliert eine Zahl in das Dictionary (HERE)