Das Programmsystem FORTH

für 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

FORTH ist ein 1971 von Charles Moore entwickeltes Sprach- und Systemkonzept, das mit wenigen Betriebssystemschnittstellen arbeitsfähig ist und sich daher besonders für Kleinrechner, Heimcomputer, industrielle Steuerungen und ähnliche Anwendungsfälle 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 für den KC 85/1 (Z 9001), die an der Wilhelm Pieck Universität entwickelt wurde und dem Robotron Computerclub zur Verfügung stand.
Nach der Konvertierung auf einen Z 1013 wurde das System nahezu ausschließlich mit sich selbst neu erstellt. Dabei wurden zunächst die Schnittstellen angepaßt und vorhandene Adreßfehler in einigen Worten beseitigt. Als Vorbild für 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 übernommen und für das vorliegende System zugeschnitten.

1. Speicherkonzept

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

Damit wird vielen Anwendern die Möglichkeit gegeben, ohne Speichererweiterung arbeiten zu können. Einschränkungen des vorhandenen Wortschatzes treten dadurch nicht auf.
Bei Speichererweiterungen kann durch Ändern der Konstanten "LIMIT" auf die erste nicht mehr verfügbare Speicheradresse der Bereich für die Screen-Puffer beliebig geändert werden. Über einen anschließenden COLD oder WARM Aufruf wird die Anzahl der SCR-Puffer neu berechnet (#BUFF). Beim Abspeichern eines so geänderten Systems enthält das System die neuen Parameter!
Eine Verschiebung des Datenstacks und der Wörterbuchobergrenze für größere 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 ständig erweitert. Daher muß bei der Implementierung des Systems eine für den Anwender sichtbare Grenze geschaffen werden die kennzeichnet, wo der systembedingte Grundwortschatz 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 üblicherweise mit Hilfe eines Assemblers übersetzt. Dabei werden Assemblerworte (Primitive) direkt übersetzt und FORTH - Worte (Secondarys) mit DA (Definierte Adresse) aufgebaut.
Bei dieser Vorgehensweise ist es nicht zwingend notwendig in allen Fällen 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 Vorwärtsverweisen zu höheren Worten zu minimieren.
Aus diesem Grund wurden alle Worte oberhalb "(" neu geschrieben.
Das höchste mit Vorwärtsverweis adressierte Wort ist "MESSAGE". Für den Programmierer gilt die Wörterbuchgrenze "TASK", der erfahrene Systemprogrammierer hat die Möglichkeit alle Worte oberhalb "MESSAGE" zu vergessen, ohne einen Absturz befürchten zu müssen.

2.2 Kassettenarbeit mit FCB

FORTH ist ein Sprachsystem das normalerweise Diskettenarbeit beinhaltet. Die Diskette kann von FORTH als virtueller Speicher über ein Fenster (Screen-Puffer) verwaltet werden. Meist wird dabei die gesamte Diskette für 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 übersichtliches Sortieren von Screens für verschiedene Anwendungsfälle und deren Verwaltung vom Betriebssystem aus möglich.

In Anlehnung daran wurde das Kassetten-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 vollständig dem HEADER-Save/Load für den Z 1013. Als Anfangsadesse wird standardmäßig 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 einzelnen Worten eigene Worte und F79-Worte verwendet, um die Implementierung effektiver zu gestalten.
In anderen Worten wurde die innere Struktur so geändert, daß sich eine effektvollere Bedienung ergibt.
Das betrifft insbesondere folgende Worte:

EMPTY-BUFFERS	Dieses Wort berechnet zusätzlich die Anzahl der möglichen Screen-Puffer zwischen FIRST und LIMIT, auch wenn LIMIT willkürlich verändert wurde und trägt diese Zahl in #BUFF ein.
FLUSH	Dieses Wort ist so verändert, daß auch Screens oberhalb 8000H erreicht werden.
+BUF	Dieses Wort weist in Abhängigkeit von LIMIT auch Puffer oberhalb 8000H zu, LIMIT muß nicht mehr direkt auf Pufferobergrenze berechnet werden. Das wird von EMPTY-BUFFERS und +BUF übernommen.
VLIST	Damit der Bildschirm übersichtlich bleibt wurde VLIST in 3 Funktionen verändert: - 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 überschreitet, auch wenn eine andere Zahlenbasis eingestellt wurde. Werden Hexazahlen (Buchstaben) eingegeben, so werden sie in Dezimalzahlen umgerechnet. Alles darüber 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 enthält (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 Einschränkungen oder Fehler auf. Es wurde so erstellt, daß ein Vergessen über mehrere Vokabulare möglich ist. Die Vokabularzeiger werden dabei so korrigiert, daß es nicht zu Abstürzen 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 verändert, daß die Übergabe der Parameter beim Aufruf und beim Verlassen den Konventionen des FIG-Standards entsprechen. Der Wiedereintritt in das FORTH-System über Adresse 300H erfolgt 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	übergibt Tiefe der Stacknutzung
EXIT	vorzeitiges Verlassen eines Wortes vor ;S
ASCII	legt Wert des folgenden Zeichens auf Stack
FREZZE	Einfrieren der Systemdaten auf derzeitigen Stand

Weiterhin wurden zusätzlich folgende Worte implementiert:

USING	Änderung der Zuweisung des Filenamens für Screen.
QX	Quick-Index, es werden die 1. Zeilen (Index-zeilen) aller im Speicher befindlichen Screens aufgelistet. Bearbeitete Screens, (UPDATE) werden mit einem * gekennzeichnet.
BSPACES	Löscht die letzten Ausgaben durch Backspace in der Länge entsprechend OUT.
SAVE	Ohne Parameter legt das FORTH-System von 100H bis HERE auf Kassette ab. Zur Erleichterung beim Laden werden zuvor Anfangs- und Endadresse auf dem Bildschirm angezeigt und auf Bestätigung mit ENTER gewartet, andere Tasten brechen die Funktion ab. Die höchste abzuspeichernde Adresse wird auf XXFFH aufgerundet. Mit Namenangabe wird zusätzlich ein Kopfblock erzeugt, der dem Aufbau für HEADER-Save/Load entspricht.

3.2 Editor

Der Editor ist als ein zusätzliches Vokabular vereinbart. Er gehört also nicht zum Grundwortschatz des Systems. Er ist mit minimalem Aufwand für eine zeilenorientierte Eingabe von Screens ausgelegt. Für den Nutzer sind nur folgende Worte von Bedeutung:

n P	Eingabe des folgenden Textes in Zeile n.
n D	Löschen der Zeile n, alle anderen Zeilen rücken nach oben.

n I	Einfügen des folgenden Textes in Zeile n, alle anderen Zeilen rücken nach unten.
L	Listen des aktuellen Screens.
n CLR	Löschen des Screens n mit Leerzeichen und listen, es erfolgt kein Laden des Screens.
alt/neu RENUMBER	Umnumerieren 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 und erleichtert das Testen eines Wertes auf verschiedene Bedingungen.
Die vorliegende Version testet nur auf Gleichheit des TOS mit den Testwerten, was für 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 zugehörige Darstellung als ASCII-Zeichen, soweit möglich. Steuerzeichen und Grafikzeichen werden als Punkt dargestellt.
Der Aufruf erfolgt: anfang länge DUMP.
Jede Tastaturbestätigung läßt den Dump um 8 Adressen weiterlaufen, wobei die ENTER-Taste den Abbruch erzwingt.

3.5 Disco

Die Funktion Disco stellt einen DIS-Compiler für minimale Ansprüche dar. Disco ermöglicht das Betrachten des inneren Aufbaus von FORTH-Wörtern.
Der Aufruf erfolgt: *DISC 'Wort'*.

Es erscheint auf dem Bildschirm die Namensfeld-Adresse 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 Drücken der Taste 'N' für 'NEXT' wird zeichenweise der Speicherinhalt mit möglicher ASCII-Interpretation angezeigt. Das gleiche gilt für die Eingabe von 'B' für 'BLOCK'.
Das Betätigen der Leertaste bewirkt ein wortweises Interpretieren des Parameterfeldinhaltes mit Anzeige 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 Drücken 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 Aufwandsgründen 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 Erreichen des Semis (,S) am Wortende ab. Bei Worten ohne Abschluß (z.B. Endlosschleifen im System wie 'INTERPRET') ist es ratsam, die Länge des Wortes vorher im Dump anzusehen oder die Interpretation des Parameterfeldes zu vermeiden. Das gilt auch für die Analyse von Worten wie Konstanten, Variablen, Uservariablen und Worten mit Headerless-Code.

In solchen Fällen versucht 'Disco' den Inhalt des Parameterfeldes vergeblich zu interpretieren, findet jedoch kein zugehöriges Namensfeld. 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 gemäß 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 für den Returnstack steht. Von diesem Bereich werden beim Kaltstart die Daten von 312H bis 322H in den Userdatenbereich kopiert.
Dieser beginnt dann wie üblich mit den Eintragungen für:

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

usw.

Die Speicherplätze 32AH und 32CH innerhalb des Systemdatenbereiches werden nach dem Vorbild von "FORTHLE" für 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 verändert 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 ermöglicht, mit relativ geringem Aufwand eine Anpassung an andere Rechnertypen unter Beibehaltung aller Worte und Systemroutinen vorzunehmen. In diesem Speicherbereich sind die direkten Einsprünge in das jeweilige Betriebssystem des Rechners implementiert und mit Headerlessbezügen oder Sprüngen aus Primitivworten an das FORTH-System angebunden.
Die Adressbezüge zu den ausführenden 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 Änderungen und Ergänzungen ist zu beachten, daß der Bereich bis 300H noch vom FORTH-System durch den Userdatenbereich 2C0H bis 2FFH aufwärts und den Returnstack ab 2C0H abwärts genutzt wird.

Die derzeitig implementierten Routinen belegen einen Raum bis 250H, ein Stackkonflikt würde erst bei einer Verschachtelung von mehr als 50 Funktionen ineinander auftreten. Die Stacktiefe reicht für allgemeine Anwendungsfälle aus. Eine Änderung der Stackadresse bzw. des Userdatenbereiches kann gegebenenfalls durch Änderung 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 München 1983

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

Verschiedene Beiträge
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 für 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

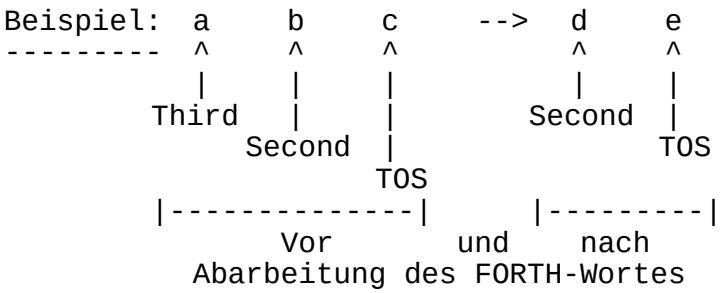
6. Glossar

Die häufigsten FORTH-Befehle sind:

6.1 Operanden

n , n1	16-Bit-Zweierkomplementzahlen
d , d1	32-Bit-Zweierkomplementzahlen
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



- 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 (zerstörend).
.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+1n)	Wandelt length-Byte-String in die Type Form.
?	(addr-->)	Druckt den Inhalt der Adresse.
?	TERMINAL (-->f)	Übergibt den Tatasturstatus ("bestätigt"<>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 gültigen Eingabe-Buffer.

6.4 Zahlensysteme

DECIMAL	(-->)	Deklariert Dezimalsystem
HEX	(-->)	Deklariert hexadezimales Zahlen-system
Base	(-->addr)	System-Variable, welche die Zahlenbasis enthält.

6.5 Eingabe-Ausgabe-Formatierung

NUMBER	(addr-->d)	Wandelt einen String in addr um in 32-Bit-Zahl.
<#	( )	Eröffnet Zahlenwandlung für Ausgabe (String).
#	(d --> d)	Wandelt nächste Stelle der Zahl und fügt dem Ausgabe-String eine Ziffer hinzu (32-Bit-Zahlen!).
#S	(d --> 00)	Wandelt alle signifikanten Stellen um in String.
SIGN	(nd --> d)	Fügt das Vorzeichen von n in den Ziffernstring ein.
#>	(d --> addr n)	Beendet Umwandlung in Ziffern-String (String hat passende Form für TYPE)
HOLD	(c -->)	Einfügung 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)	Systemkonstante (Blockgröße in Bytes)
BLK	(-->addr)	Systemvariable (aktuelle Block-Nummer)
SCR	(-->addr)	Systemvariable (enthält 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 (Zwischenspeicherung, Gebrauch mit Vorsicht !)
R>	(--> n)	Holt den Wert vom Return-Stack zum TOS zurück.
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 -->)	Fülle u Bytes im Speicher ab addr mit b.
ERASE	(addr u -->)	Fülle u Bytes im Speicher ab addr mit Null.
BLANKS	(addr u -->)	Fülle u Bytes im Speicher ab addr mit Blanks (20H).
TOGGLE	(addr b -->)	EXOR das Byte in Adresse addr mit Maske b.
SP@ (--> addr)		Übergibt aktuelle Position 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 (übergibt 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-genauem 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 Übergabe des 16-Bit-Restes und des 32-Bit-Quotienten.
MIN	(n1 n2 --> Minimum)	Übergibt die kleinere von zwei Zahlen.
MAX	(n1 n2 --> Maximum)	Übergibt die größere 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 größer n2
=	(n1 n2 -->f)	Flag=1 wenn n1 gleich n2
0<	(n -->f)	Flag=1 wenn TOS negativ 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-Verknüpfung
OR	(n1 n2 -->ODER)	Bitweise logische ODER-Verknüpfung
XOR	(n1 n2 -->EXOR)	Bitweise Exclusiv-ODER-Verknüpfung

6.12 Strukturierende Worte

DO...LOOP	(n1 n2 -->)	Schleife, Index läuft 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 zusätzlicher Parameter an +LOOP übergeben)
I	(--> Index)	Loop-Index --> TOS
LEAVE	( )	Erzwingt Abbruch der Schleife bei nächster Gelegenheit.
IF.. (wahr).. ENDIF	(f -->)	Führt Befehle aus wenn Flag=1 ist.
Ist..(falsch)..		dann ENDIF.
IF.. (wahr).. ELSE	(f -->)	dto., jedoch wird bei Flag=0 der FALSE-Teil ausgeführt
BEGIN...UNTIL	(--> f -->)	Schleife mit Abbruch, falls Flag für UNTIL=1
BEGIN.. WHILE.. REPEAT	(--> f -->)	Wie BEGINN...UNTIL, jedoch Abbruchtest am Anfang 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 übergibt die Adresse bei Aufruf).
CONSTANT yyy	(n -->)	Erzeugt eine Konstante yyy, mit dem Wert n (bei Aufruf von yyy wird Wert übergeben).
CREATE zzz	( )	Eröffnet 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)	Übergibt die Adresse eines Pointers zum Context-Vokabular (das zuerst abgesucht wird)
CURRENT	(-->addr)	Übergibt 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

(	( )	Eröffnet 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)	Übergibt die Adresse des nächsten freien Platzes im Dictionary

PAD	(-->addr)	Übergibt die Startadresse eines Zwischenspeichers, meist 68 Byte oberhalb von HERE
IN	(-->addr)	System-Variable, hält Input-Buffer-Offset für WORD
ALLOT	(n -->)	Hinterläßt eine ungenutzte Lücke (n Bytes) im Dictionary
`	(n -->)	Compiliert eine Zahl in das Dictionary (HERE)