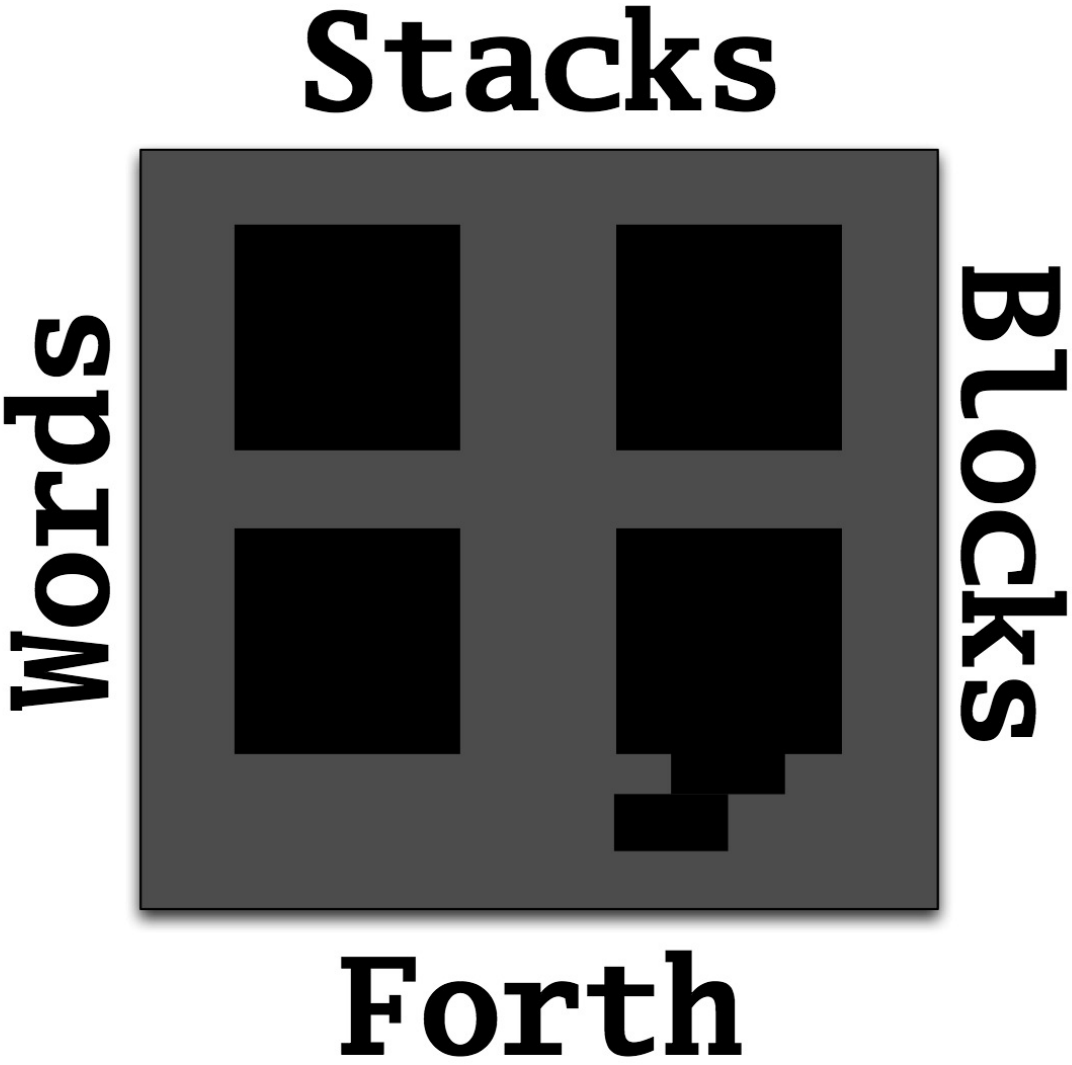
Arcus Forth 0.1 - 1987 Arcussoftware (Michael Balig)

Anwendungsbeschreibung: © 2021 Harald Lack

Erstellt mit Hilfe von Microsoft Word 6 und FreePDF2

Verwendete Hardware: Original ZX Spectrum 48K

Veröffentlicht im Thomas Lienhard Spectrum Forum - Januar 2021 für den Spectrum und Sam Profi Club



Liebe Mituser und Programmierfreunde!!!

Unser Spectrum ist ja bekannterweise ein recht vielseitiger Computer, den man für die unterschiedlichsten Zwecke einsetzen kann. Man braucht ja nur die einzelnen Projekte in den Foren anzusehen, um sich davon zu überzeugen. Aber die grundlegende Arbeit am Computer stellt normalerweise ja die Programmierung dar (das Benutzen von Spielen und Anwendungsprogrammen wollen wir an dieser Stelle mal vernachlässigen - auch wenn es die Hauptbeschäftigung vieler Computerbesitzer sein wird). Was jetzt die Programmiersprachen

betrifft, so spielte und spielt sich wohl das Meiste in BASIC oder Assembler ab. Allerdings gibt es noch eine Reihe nicht uninteressanter (Hoch)Sprachen für die es auch verschiedenste Spectrum-Versionen bzw. Implementierungen gibt. Eine wenig beachtete und oft wegen ihrer Eigentümlichkeiten vernachlässigte davon ist Forth. Ich möchte mich deshalb heute ein wenig mit dieser Sprache beschäftigen und sie kurz anhand von Arcus Forth vorstellen. Dieser Beitrag soll auch kein Programmierkurs für Forth werden, das würde den Rahmen sprengen. Als Grundlage meiner Betrachtungen dient uns die Implementierung von Arcus-Forth V 0.1 aus dem Jahre 1987. Dieses Programm wurde seinerzeit von Michael Balig unter dem Label Arcus Software vertrieben, der auch der treibende Pol hinter der damaligen Forth-Bewegung war. Leider ist er im Juni 2020 gestorben und die Forthgemeinde um einen anerkannten Spezialisten ärmer. Anhand dieser Implementierung möchte ich ein bischen auf die Bedienung dieser Programmierumgebung eingehen. Damit sei Interessierten dann die Möglichkeit an die Hand gegeben, auf dieser Basis ihre Kenntnisse in Forth zu entwickeln oder gar zu vervollständigen.

Zunächst möchte ich ein paar allgemeine, sozusagen geschichtliche Informationen dazu geben. Forth an sich wurde um ca. 1970 entwickelt. Die Entwicklung dieser Sprache wird Charles H. Moore zugeschrieben und sie zählt zu den imperativ stackorientierten Programmiersprachen (hört sich ziemlich hochtrabend an, ist aber die rein „technische" Beschreibung). Bei unseren Betrachtungen werden wir später (in einem separaten Beitrag) sehen, dass Forth eine Entwicklungsumgebung in Form eines Editors bereitstellt. Ganz ähnlich war es ja auch mit QBasic unter DOS (das nur nebenbei und erläuternd). Bei seiner Entwicklung war - so ist zu lesen - vorgesehen, die Sprache eigentlich Fourth zu nennen, da es auf der seinerzeit 4. Generation von Computern laufen sollte. Nun hatte Moore aber das Problem, dass das System auf dem er arbeitete nur Dateinamen mit maximal fünf Buchstaben zuließ. So wurde aus Fourth eben dann Forth. Man muss sich halt an die eigenen Restriktionen halten. Forth ist sehr genügsam was den Speicherbedarf betrifft und ein vollständiges Forth System läßt sich mitunter in wenigen Kilobyte unterbringen. Kein Wunder bei den damals vorherrschenden Systemen, die ja kaum Speicherplatz bzw. Systemspeicher zur Verfügung hatten. Das macht es natürlich zu einer idealen Programmierumgebung für den Spectrum, der ja nicht mit üppigem Speicher gesegnet ist.

Wer sich ein bischen mit Forth auseinandersetzt wird feststellen, dass der sogenannte Stack ein wesentliches Element in Forth ist um den sich sozusagen alles dreht. Immer wieder werden in der Literatur Begriffe wie Data Stack und Return Stack auftauchen. Daneben muss man sich noch mit Registern, dem Dictionary (das man in mehrere Vocabularys aufteilen kann), Sprungtabellen und Token herumschlagen. Das aber nur, am Rande. Ich möchte an dieser Stelle ja noch keinen Leser verlieren. Was, und da muss ich aus eigener leidvollen Erfahrung berichten, am Anfang durchaus einiger Gewöhnung bedarf ist, dass Forth mit der sogenannten umgekehrten polnischen Notation (UPN) arbeitet. Nun wird sich mancher fragen, was ist denn das schon wieder? Kurz erläutert handelt es sich hierbei um folgendes. Die UPN, auch Postfixnotation genannt, ist eine bestimmte Eingabeweise von Anweisung für Operationen. Vielleicht kennt der eine oder andere noch dieses Thema von den Hewlett-Packard Taschenrechnern wie dem HP 41CX die Anfang der 80er Jahre sehr beliebt und verbreitet waren. Ich möchte es an einem trivialen Beispiel erläutern und denke, dass es damit leicht verständlich und somit transparenter wird.

Es sollen die Zahlen 5 und 9 addiert werden. Das menschliche Gehirn steht hier einer sequentiellen Abarbeitung nahe, denn es rechnet 5 + 9. Bei der UPN ist dies anders. Hier lautet die Anweisung 5 9 +. An dieser Stelle kommt jetzt die weiter oben bereits erwähnte Stapelverarbeitung wieder ins Spiel. Zuerst werden die beiden Operanden (5, 9) nacheinander auf den Stapel gelegt, dann holt sie der Operator (+) vom Stapel, berechnet das Ergebnis und legt dieses wieder auf dem Stapel ab. Das Ergebnis liegt also immer oben auf dem Stack. Last in - First out. Alles kein Hexenwerk, man muss sich nur mal damit vertraut machen. In der

Mathematik haben wir uns ja früher oft mit den Klammern herumgeärgert. Zumindest war das zu meiner inzwischne lang zurückliegenden Schulzeit so. Forth kommt uns hier mit der UPN sehr entgegen, denn es benötigt keine Klammern. Schade, dass wir das erst jetzt erfahren. Dazu noch ein kleines erläuterndes Beispiel. Es soll folgendes berechnet werden. Zwei Zahlen (3, 5) sind zu addieren und mit einer dritten Zahl (2) anschließend zu multiplizieren. Im sequentiellen Verfahren, so wie wir das aus der Realität kennen, braucht es hier dann schon Klammern, um eindeutig zu sein. Wir rechnen/denken also: (3+5)*2 - und nur durch das Setzen der Klammern kommen wir zu dem einzig logischen Ergebnis, nämlich 16. Würden wir keine Klammern setzen und die axiomatisch definierte Regel Punkt vor Strich anwenden würden wir 13 erhalten. Bei Forth genügt hier die Eingabe 3 5 + 2 * um alles klar zu definieren. Klammern nicht notwendig!!! Man sagt auch oft, dass die UPN die kompakte, klammerfreie Schreibweise der Aussagenlogik darstellt. Abschließend sei noch kurz erwähnt, dass die UPN auf den polnischen Mathematiker Jan Lukasiewicz zurückgeht, der sie in den 20er Jahren des letzten Jahrhunderts entwickelt hat.
Zurück zu Forth. Weiter oben habe ich den Begriff des Dictionary eingeführt. Darunter versteht man den in Forth enthaltenen Wortschatz an Befehlen, die das System kennt. Analog der internen Befehle in Command.Com eines DOS Systems. Die Programmierung in Forth unterscheidet sich also schon mal grundlegend von der in BASIC oder anderen Sprachen, die nicht auf UPN bauen. Aber das macht die Arbeit mit Forth natürlich auch interessant. Kurze Nebenbetrachtung: Forth wurde in den 70ern von Xerox zu Interpress weiterentwickelt aus der sich wiederum Postscript entwickelt hat. Man könnte also fast sagen, jeder der mit pdf-Dateien arbeitet greift damit auf dessen Wurzeln in Forth zurück.

So, wir wollen unseren Exkurs in die Entwicklungsgeschichte von Forth hiermit auch wieder beenden und uns wieder unserem eigentlichen Thema zuwenden.

Arcus-FORTH ist an den sogenannten f.i.g.-Standard (Forth Interest Group) angelehnt. Es bietet einige über den üblichen Standard hinausgehende Worte, auf die ich später noch speziell eingehen möchte. Wer sich nach diesem kurzen Überblick näher mit FORTH beschäftigen will (und ich hoffe, ich konnte ein paar Interessenten anwerben), der sei auf die am Ende angeführte Literatur verwiesen. In wie weit die dort aufgeführten Bücher heute noch erhältlich sind, müsst ihr bitte mit eurem Buchhändler abklären oder aber im Internet danach suchen. Quellen sind dort genügend zu finden. Es gibt auch bestimmt einige Referenzen zu Forth die in den Tiefen des World Wide Web zu finden sind. Recherchieren lohnt auf jedem Fall. Etwas näher möchte ich dann auch noch die Bedienung des mitgelieferten Editors beschreiben. Dies möchte ich aber in einem separaten Beitrag abhandeln. Also immer schön der Reihe nach.

Empfohlen von Michael Balig für das Erlernen dieser Sprache ist das Buch #4. Es ist recht gut geschrieben doch sei auf eine Besonderheit hingewiesen. In diesem Buch beschäftigt sich der Autor mit dem FORTH-83-Standard. Dies ist eine gegenüber dem f.i.g.-FORTH neuerer Standard, doch sind die Unterschiede marginal. Trotzdem soll dies hier informativ erwähnt werden.

Arcus-FORTH ist jedoch ein speziell auf den Spectrum+ zugeschnittenes Prograrnmiersystern. Damit muss es sich natürlich dann auch gleich mal den Spectrum-typischen Restriktionen anpassen. Besonders muß hier auf die Anpassung der Quelltexteinteilung in den Screens hingewiesen werden und die Darstellung dieser auf dem angeschlossenen Bildschirm. Ab der Systemadresse LIMIT, die am Ende des Diskbuffers steht, plaziert das Programm eine 16 KByte große RAM-Disk in der 32 Screens a 512 Byte untergebracht sind. Es sind dies die Screens 0-31. Sie haben eine Größe von 16 Bildschirmzeilen a 32 Zeichen. Dies stellt zwar eine Verletzung des f.i.g.-Standards dar der 64 Zeichen pro Zeile vorsieht, ist aber auf dem Spectrum so Standard. Weil wir gerade bei den Screens sind, sollten wir gleich mal eine Besonderheit beachten. Screen 0 sollte aus Sicherheitsgründen nur Kommentartexte

enthalten da von ihm systembedingt kein Quelltext geladen werden kann. Auch das Abspeichern in diesem Screen ist mit Unwägbarkeiten verbunden, deshalb sollte man zunächst in einen anderen Screen schreiben und erst dann nach Screen 0 kopieren. Vorsicht hat ja bekanntlich noch nie geschadet. Ich hoffe mal, das wird dann später noch ein wenig verständlicher.

Als Ein-/Ausgabegeräte benutzt Arcus-FORTH neben der Tastatur/Bildschirm und dem spectrumüblichen Kassettenrecorder (evtl. Anpassungen an neuere Disk(Speicher)systeme sind durchaus möglich) noch einen Centronics-Drucker. Dieser wird über Adresse 0DFH angesprochen. Es besteht jedoch die Möglichkeit über die Worte P$ und P! weitere selbst geschriebene Gerätetreiber zu programmieren. Ebenso läßt sich so an dieser Adresse auch eine Kempston-kompatible Joystick Abfrage aufbauen. Das ist dann aber schon eher was für Experten und soll hier nur ergänzend erwähnt werden.

Die von mir benutzte Version hat den Stand Juni 1987 und wurde als Public Domain Version vertrieben. Sicherlich hat sich der eine oder andere von Euch schon über die merkwürdige Versions Nummer (V 0.1) gewundert. Hierzu sei folgendes angemerkt. Arcus-FORTH geht hier nicht den sonst bei f.i.g.-Anwendungen üblichen Weg, sondern baut auf ein eigenes System auf. Alle Arcus-(f.i.g.)-FORTH Versionen 0.x kennzeichnen Systeme, die auf dem BASIC-System logisch aufsitzen und einzelne Routinen aus diesem System (mit)benutzen. Arcus Forth bedient sich also der vorhandenen ROM Routinen des darunter liegenden Systems. Damit macht Arcus-Forth nichts anderes als z. B. Windows 3.X bei PCs.

Im Rahmen von Weiterentwicklungen waren noch folgende Schritte geplant:

Version 1.x
FORTH-Systern das nur noch physisch auf dem BASIC-System aufsitzt, jedoch logisch selbständig ist und deshalb keine ROM-Routinen mehr benutzt.

Version 2.x
Diese Versionen sollen FORTH-Systeme sein, die ROM-fähig sind und damit für den Zielrechner selbst Betriebssystern sein können.

Version 3.x
Hier soll es sich dann schon um autonome RAM-System handeln, die nur noch von einem Speichermedium (hier am besten wohl Diskette) mittels einem sogenannten Urlader in den Speicher geladen werden.

In wie weit dies umgesetzt wurde habe ich ich nicht nachverfolgt. Wer näheres zu berichten weiß, kann dies gerne tun.

Kommen wir jetzt zu den Worten aus dem Vokabular von FORTH, die vom Standard abweichen oder in diesem nicht enthalten sind (keine Regel ohne Ausnahme):

.BASE ()
Gibt die aktuell gültige Zahlenbasis für uns besser lesbar als Dezimalzahl an.

.DUMP (addr1 addr2 ->)
Gibt den Speicherbereich zwischen addr1 und addr2 (einschließlich) hexadezimal und als Zeichen aus. Das ist dann erst mal schon ein bischen verwirrend.

.S ()
Gibt die Belegung des Parameterstacks in der Form (Adresse) (Inhalt) aus. Dabei entspricht die Adresse mit dem Wert 1 dem top of stack und so weiter.

.voc ()
Gibt die Namen der im Dictionary enthaltenen Vokabeln aus.

?PRINTER (-> F)
Prüft, welches Ausgabegerät angesprochen bzw. definiert ist (Drucker (f=1) oder der Bildschirm (f=0)).

?READY (b ->)
Wartet, bis der Druckerstatus gleich dem Byte b ist. Wenn man ungeduldigerweise in der Zwischenzeit die Taste BREAK drückt, so wird nach QUIT gesprungen und der Parameterstack zurückgesetzt.

ASCII (-> c)
Legt den Zahlenwert c eines Zeichens, das im INPUT-Buffer erwartet wird, sofort auf den top of stack (TOS). Beispiel: Das ASCII-Zeichen „A" würde eine 65 auf dem TOS legen.

BEEP (n1 n2 ->)
Erzeugt einen Ton von der Höhe n1 und der Länge n2. Eigentlich ganz logisch. Mehr soll dazu gar nicht gesagt werden.

BYE ()
Springt (wie der Name vermuten läßt) in das BASIC-System zurück.

C/L (-> n)
Bringt die Länge einer Ausgabezeile auf den top of stack. Der f.i.g.-Standard definiert hier jedoch C/L als Länge einer Eingabezeile.

CASE (n -> n)
Stellt ein CASE-Construct (Verbundanweisung) dar. Siehe nachfolgendes Beispiel:

```
CASE n1 OF ... (Fall 1) ... ENDOF
       n2 OF ... (Fall 2) ... ENDOF
       .....................
       nk OF...(Fall k)...ENDOF
      ......(Fehlerfall)......
ENDCASE
```

Während der Fehlerbehandlung liegt das Eingangsargument für CASE noch auf dem top of stack. Damit wäre eine Auswertung möglich, die uns ggfs. weiterhilft. Das Wort ENDCASE nimmt aber in jedem Fall einen Wert vom Stack, wenn keiner der vorher definierten Fälle zutraf. Die Fehlerbehandlung kann damit auch wegfallen. Tritt jedoch einer der definierten Fälle ein, so wird vom jeweiligen ENDOF hinter ENDCASE verzweigt und das Construct verlassen.

CLEAR (n ->)
Löscht den Screen Nr. n.

CLS ()
Löscht den Bildschirm.

CONSOLE ()
Definiert den Bildschirm als Ausgabegerät (Standard).

COPY (n1 n2 ->)
Kopiert Screen n1 nach n2.

CR ()
Ist ein Hilfswort für die Zeichenausgabe und sollte nicht verwendet werden.

EDIT (n ->)
Ruft den Full-Screen-Editor (siehe dazu meinen nächsten Beitrag) zur Bearbeitung des Screen n auf.

ENDCASE (n ->)
Siehe CASE

ENDOF ()
Siehe CASE

Editor ()
Ist das Vokabular für die Nicht-Top-Worte des Editors.

FREE (-> n)
Bringt die Anzahl der freien Bytes zwischen Dictionary-Ende und Parameter-Stack auf den TOS.

FREEZE ()

Erzeugt einen Kaltstartparameter mit dem aktuellen Dictionary-Zustand und friert so die zuletzt gegebenen Definitionen, also den aktuellen Systemzustand ein bzw. schützt so vor der Kaltstartprozedur. So ist es möglich eine neue Version (mit dem aktuellen Zustand) abzuspeichern. Damit man das aber erfolgreich durchführen kann muss man vorher mit dem Befehl SIZE U. den aktuellen Umfang des Dictionary bestimmen. Anschließend verlässt man das Forth System über den Befehl BYE. Der eben ermittelte Wert des Dictionary wird nun der BASIC Variablen size zugewiesen. Danach kann man mit GO TO 10 die neue Version abspeichern.

HEADER ()
Überträgt einen maximal 10 Zeichen langen Datei-Namen aus dem aktuellen INPUT-Buffer und speichert ihn ab Adresse HERE. Siehe dazu auch bei den Befehlen SAVETAPE, LOADTAPE, SAVESCREENS, LOADSCREENS.

IST (addr ->)
Gibt den Vocabular-Namen zur übergebenen Adresse aus. Wird in folgender Form genutzt.

CONTEXT IST
CURRENT IST
VOC-LINK IST

J (-> n)
Bringt den Index eines äußeren LOOPs zum top of stack.

K (-> n)
Bringt den Index eines zweitäußeren LOOPs zum top of stack.

LEAP (->)

Steigt sofort (ohne wenn und aber) aus einer LOOP-Schleife aus. Aber VORSICHT!! Kann nicht in geschachtelten LOOPs verwendet werden.

LINK (-> addr)
Ist eine Variable und kann die Werte 0 oder 1 annehmen. Sie enthält 0 wenn der Bildschirm und 1 wenn der Drucker als Ausgabegerät eingestellt ist.

LINKER - RAND (n ->)
Gibt eine Linke-Rand-Einstellung (Left Margin) an einen Epson-kompatiblen Drucker aus. Bei Ausgabe auf den Bildschirm (anstelle des Druckers) kommt es in der Regel zu einem Fehler oder sogar zu einem System-Crash. Also aufgepasst!!

LIST (n ->)
Gibt den Screen n aus. Das erfolgt bei der Bildschirmausgabe anders als der Standard ohne Zeilen-Nr. um bei 32 Zeichen pro Zeile sowohl in den Screens als auch auf dem Bildschirm das Druckbild nicht zu verzerren. Bei einer Ausgabe auf dem Drucker werden die Zeilennummern natürlich gedruckt.

LOADSCREENS ()
Liest ein Screenfile in der Art ein, in der es ausgegeben wurde (Umfang und Screen-Nr.). Der Name des Files muss im aktuellen INPUT Buffer stehen (z. B. "LOADSCREENS XYZ").

LOADTAPE (addr ->)
Speichert ein Bytefile, das auch ein Screenfile sein kann ab der Adresse addr. Der Name des Files muss im INPUT Buffer stehen.

LTAPE (addr ->)
Liest ein Bytefile von der Kassette ein. Die Parameter (Header) müssen ab der Adresse addr im Spectrum-Standard-Format abgespeichert sein.

LX-86 (c ->)
Gibt das Byte c über Centronics Drucker aus.

NOOP ()
Hat keine Wirkung (no operation).

OF (n1 n2 ->)
Siehe CASE

P$ (b1 -> b2)
Liest am Eingabeport mit Adresse b1 das Byte b2 ein.

P! (bl b2 -->)
Gibt über den Port mit Adresse b2 das Byte b1 aus.

PRINTER ()
Bestimmt den Drucker als Ausgabegerät.

SAVESCREENS (n1 n2 ->)
Speichert n2 Screens ab Screen n1 auf Kassette ab (Achtung: Die Aufzeichnung startet sofort ohne vorherige Rückfrage wie aus dem BASIC-Interpreter bekannt). Der Name des Files muss im INPUT-Buffer stehen.

SAVETAPE (addr n ->)

Speichert n Bytes ab Adresse addr auf Kassette (Sofortstart - siehe oben). Der Name des Files muss im INPUT-Buffer stehen.

SHOW (n1 n2 ->)
Druckt die Screens n1 bis n2 auf dem Drucker aus.

SIZE (-> n)
Legt die Anzahl der vom Arcus-FORTH Programmsystem belegten Bytes auf den top of stack ab.

STAPE (addr ->)
Speichert die Bytes auf Kassette.

THRU (nl n2 -->)
Lädt die Screens von n1 bis n2 inklusive (das Wort -> (arrow) darf nicht in ihnen enthalten sein, ausser im Screen n2, ab dem dann in Folge weitergeladen werden soll).

ZNR (n ->)
Gibt n dreistellig aus gefolgt von einem Leerzeichen.

\()
Ein Backslash kennzeichnet den Rest der Zelle als Kommentar.

Soweit zu den Abweichungen vom f.i.g.-Standard. Abschließend jetzt noch zu den am Anfang angesprochenen Büchern die Michael zur weiteren Lektüre für all diejenigen empfielt, die sich näher mit Forth beschäftigen möchten.

#1 Ronald Zech - Die Programmiersprache FORTH (Franzis Verlag)
#2 Charles E. Eaker - The Case Statement (FORTH Dimensions)
#3 Leo Brodie - In FORTH denken (Hanser Berlag)
#4 Leo Brodie - Programmieren in FORTH (Hanser Velag)
#5 Andreas Goppold/Roger Bouteiller - FORTH ein Programmiersystem ohne Grenzen (Edition Aragon)
Weitere Informationen in den einschlägigen Seiten des World Wide Web!!

Soviel zu meiner kurzen Übersicht über die Eigenheiten von Arcus Forth. Im folgenden Beitrag sprechen wir dann noch über den Full Screen Editor. Da ich, wie gesagt hier keinen Forth Programmierkurs machen wollte sondern nur Arcus Forth näher beschreiben, wäre es schön, wenn sich ein Forth-Kundiger finden würde, der mal einen Programmierkurs zu dieser recht interessante Sprache anbieten würde. Bis dahin viel Spaß mit Arcus-Forth.

English summary:

Today´s article is about programming the Spectrum especially using different programming languages. Most applications and routines for our computer were written using Basic and machine code/assembler. But there are also some very interresting other languages. As you can see, I will this time have a closer look on Forth and here the f.i.g-Forth standard of the Arcus Forth language. In this article I give you a very perfunctory overview of Arcus Forth and some hints for books about Forth that may help you to go any further. Maybe the World Wide Web is also a good source. In the second part of the article I give you a perspective of the vocabulary that is different as the well known Forth standard. In a supplementary article I will in the near future show you how to use and operate the Arcus Forth editor.