VIERTE DIMENSION

Volume V/Nr.4 - Dezember 1989

Themen

- ANS FORTH
- Graphtale Pflanzen
- ASYST
- Das "GREENHORN-Modul"
- Stringstack mit Heap
- DRUMA-FORTH

FORTH MAGAZIN

7,50 DM

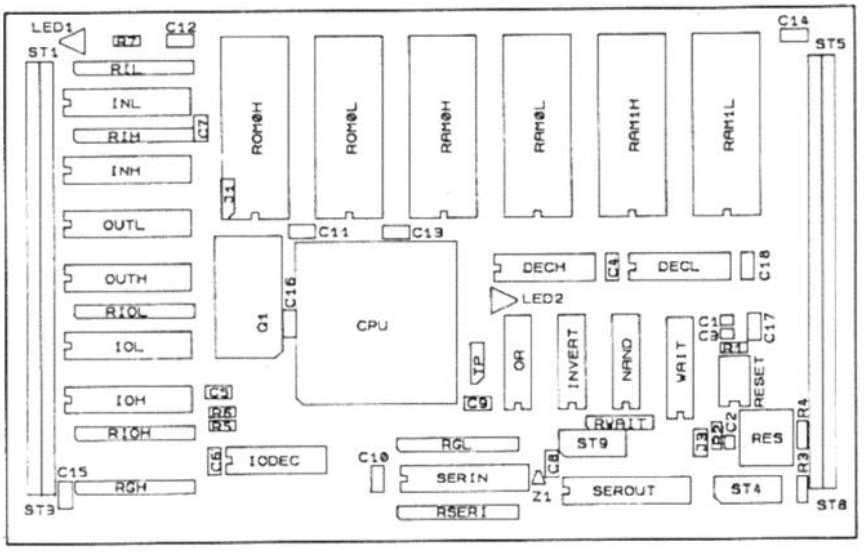
Real Time Express Real Time Express Real Time Express

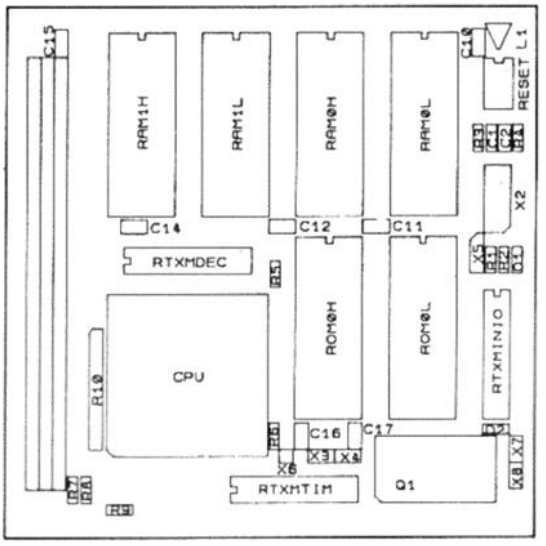
10.000.000 FORTH-Operationen pro Sekunde!

Der REALTIME-EXPRESS läuft weiter !

Die "Emulatorkarte": sämtliche Signale des RTX2000 auf einer 96-poligen VG-Leiste. Serielle Softwareschnittstelle, Systemtakt 6,7 MHz, 64 kByte RAM bestückt, erweiterbar on Board auf 128 kByte, FG-FORTH, kompaktes System 100mm x 100mm durch PALs

Die mc-RISC-Karte: vorgestellt in mc 6/89, 16 Bit Input-Port, 16 Bit-Output-Port, 16 Bit I/O-Bus, serielle Softwareschnittstelle, 8 MHz Systemtakt, 64 kByte RAM bestückt, on Board auf 128 kByte erweiterbar, FG-FORTH, Format 100mm x 160mm.





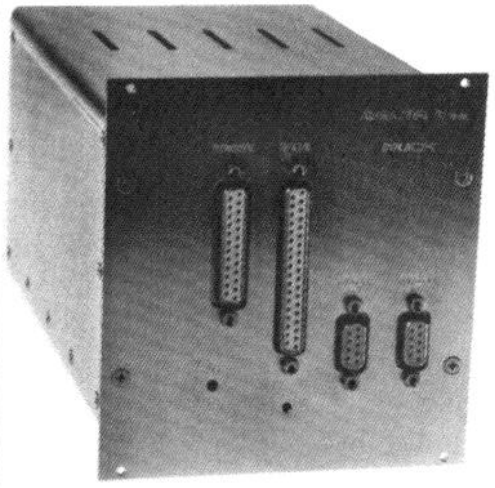
SONDERPREIS für Mitglieder der FORTH-Gesellschaft:
DM 1.777,--
(incl. Versand, incl. MWSt)
Hochschulrabatt auf Anfrage!

FORTH

REAL TIME EXPRESS und RTX ist Trademark der HARRIS CORPORATION, Palm Bay, Florida

RTX2000

Karten zum Einsteigen erhalten Sie ab sofort bei:
BRÜHL ELEKTRONIK ENTWICKLUNGS-GESELLSCHAFT mbH, Hegelstraße 10, 8500 Nürnberg 10, Tel. 0911/359088
RTX2000, 64 KB PROM, 64 KB RAM, FG-FORTH-Compiler: DM 2.000,–



DELTA t

Mit MUCK wird der Zufall immer greifbarer

Viele zeichnen Daten schnell auf.
Wir verarbeiten bis zu einer Million Samples/Sekunde mit unserem Multiprozessorsystem auf der Basis von Forth-RISC-Prozessoren.

DELTA t
Ulrich Hoffmann Marina Kern Klaus Schleisiek-Kern
Entwicklungsgesellschaft für computergesteuerte Echtzeitsysteme mbH
Telefon 040/229 64 41 · Uhlenhorster Weg 3 · D - 2000 Hamburg 76

EDITORIAL

Es ist kaum zu glauben - doch es ist schon wieder ein ganzes Jahr vergangen. Und nicht nur das, eine Dekade neigt sich ihrem Ende zu. Mit großen Schritten nähern wir uns der Jahrtausendwende. Alles in allem kann man für das Jahr 1989 eine durchaus positive Bilanz ziehen. Denken wir nur einmal an die Geschehnisse in der DDR. Für die FORTH-Gesellschaft ist die Öffnung der DDR sicherlich auch von Vorteil, können doch jetzt Verbindungen viel leichter geknüpft werden. Wie wir wissen herrscht ja in der DDR ein reges Interesse an FORTH.

In diesem Heft finden Sie eine Reihe von aktuellen Berichten, in denen Autoren ihre Erfahrungen mit FORTH-Systemen beschreiben. Auch fehlt diesesmal der obligatorische Graphik-Artikel von Christoph Krinninger, der die Titelgraphik beschreibt, nicht.

Wir wünschen allen Lesern der 'Vierten Dimension' ein frohes Weihnachtsfest und einen guten Rutsch ins neue Jahrzehnt.

Denise Luda

Rainer Aumiller

IMPRESSUM

Titel:
FORTH MAGAZIN 'Vierte Dimension'
Zeitschrift der Mitglieder der FORTH-Gesellschaft e.V. © 1989

Herausgeber:
FORTH-Gesellschaft e.V.

Redaktion:
D. LUDA Software, Gustav-Heinemann-Ring 42, 8000 München 83,
Tel. 089/670 83 55, FAX 089/679 22 71

Kontaktadresse:
Entweder direkt die Redaktion anrufen bzw. anschreiben, das FORTH-Büro in München, Postfach 1110, 8044 Unterschleißheim, Tel.: 089/3173784 kontaktieren oder die FORTH-Mailbox München (s.u.) 'Konferenz Vierte Dimension' benutzen.

Quelltextservice:
Der Quelltext von Beiträgen, die mit diesem Symbol gekennzeichnet sind, ist auf der Leserservice-Diskette zur jeweiligen Ausgabe oder in der FORTH-Mailbox in München Tel. 089/7259625 8N1 zu finden.

Quelltext Service

Autoren dieser Ausgabe:
M. Schultheis, Jörg Staben, Claus Vogt, Martin Holzapfel, Denise Luda, Christoph Krinninger, Konrad Scheller, Ulrike Schnitter, Frank Raschke.

Erscheinungsweise:
Vierteljährlich

Redaktionsschluß:
Die zweite Woche im mittleren Quartalsmonat

Auflage:
ca. 1000 Stück

Druck:
Buch- und Offsetdruckerei Bickel Söhne, Frankfurter Ring 243, 8000 München 40

Bezugspreis:
Einzelheft DM 7,50, Abonnement 4 Hefte DM 40,- inklusive Versand.

Vierte Dimension Inhalt

Leserbriefe und Zuschriften

Lesermeinung zum Thema "Warum Postfix?"

Die Postfix-Notation wird gern als Argument sowohl für als auch gegen FORTH angeführt. Meiner Meinung nach wird diese Eigenheit von FORTH in beiden Fällen überbewertet. Auf unterster Ebene kommt man kaum in die Verlegenheit, mathematische Ausdrücke zu berechnen, und in einer Applikation ist man ohnehin frei in der Wahl der Dateneingabe. Der Programmierer, der sich ein Bild von den inneren Abläufen machen muß, hat keine Schwierigkeit mit Postfix, ihm erscheint diese Reihenfolge natürlich.

Hierzu ein Gleichnis aus der Hardware-Ecke (und es hinkt wie fast alle Gleichnisse): Schaltpläne werden in aller Regel so gezeichnet, daß der Signalfluß von links nach rechts erfolgt. Handelt es sich um einen Zähler mit seinen Überträgen, so ergibt sich eine Anordnung, bei der die "Einer" ganz links stehen. Das ist aber weniger störend, als wenn man den Plan umgekehrt zu zeichnen hätte. Wer solche Pläne lesen muß, sieht das Gerät eben "von innen", und damit steht die Einerstelle goldrichtig.

Mir geht es jedenfalls so, daß ich die Reihenfolge der Argumente in FORTH gar nicht als "Notation" empfinde, sondern vielmehr als eine Folge von Anweisungen. Deshalb würde ich auch das Wort "+" nicht (wie das FORTH-83-Standard-Dokument) als "plus" bezeichnen, sondern als "add". Das mag spitzfindig klingen, aber es charakterisiert gut die unterschiedliche Sichtweise. (Der Ausdruck "5 7 +" kommt in der Praxis ohnehin kaum vor, da man dafür besser "12" schreibt.)

Wenn man vorhat, sich gegen "Angriffe" zu verteidigen, ist es gut, die eigenen Waffen richtig einzuschätzen. Zuweilen beruft sich die Verteidigung der Postfix-Notation auf "Klammerfreiheit". Aber gerade dieses Argument scheint mir wenig stichhaltig zu sein.

Wenn im Infix-Ausdruck "2 * (3 + 4)" die Klammer verschwinden soll, so kann ich mir so helfen, daß ich die Regel "Punkt vor Strich" fallen lasse und an ihre Stelle die Regel "links vor rechts" setze:

```
3 + 4 * 2  -->  14
```

Das wäre dann eine Zusammenziehung von "3 + 4 = 7" und "7 * 2 = 14". Wem das absonderlich erscheint, der möge sich klarmachen, daß genau dies die Vorgehensweise eines Infix-Taschenrechners ist (von den Klammern mal abgesehen): ich drücke "3 + 4 * 2 =" und erhalte das Ergebnis 14. Und was gewinne ich nun durch Postfix?

```
3 4 + 2 *  -->  14
```

Gar nichts. Und deshalb bin ich dafür, dieses fadenscheinige Argument fallen zu lassen. Fairerweise sollte man der Infix-Notation nicht ankreiden, sie verdrehe irgendwas, denn wenn ich "2 * 3" sehe, dann lese ich das nicht von links nach rechts, sondern auf einen Schlag.

Also: dem Rechner setze ich (in richtiger Reihenfolge) Anweisungen vor, keine mathematische »Notation« (die begreift er ja doch nicht). Und im Kommentar tue ich gut daran, die Infix-Notation zu verwenden.

Johannes Teich, Murnau

PS: Hier noch ein simpler Infix-Kalkulator:

```
: task ;     variable mode     0 mode !
: calc   mode @  1 =  if  +  exit then
         mode @  2 =  if  -  exit then
         mode @  3 =  if  *  exit then
         mode @  4 =  if  /       then
;

: +   mode @  if calc then  1 mode ! ;
: -   mode @  if calc then  2 mode ! ;
: *   mode @  if calc then  3 mode ! ;
: /   mode @  if calc then  4 mode ! ;
: =   mode @  if calc then  0 mode ! ;

\  17 * 2 + 16 / 2 - 17  -->  8
```

An die Redaktion

Sehr geehrte Redaktion der Vierten Dimension, zuerst möchte ich Sie ganz dick loben: So, wie die Vierte Dimension jetzt aussieht, stelle ich mir ein FORTH-Magazin vor. Das zeigt wohl auch, daß die FORTH-Gesellschaft als Ganzes auf dem richtigen Weg ist.

Schade nur, daß die Resonanz der Leser gering ist. Eigentlich erstaunlich, denn ich hatte in meinen Beiträgen immer klar herausgestellte Schwachpunkte, mit denen ich die Leser zu Verbesserungsvorschlägen anregen wollte. War diese Art zu aufdringlich?

Vielleicht sollte man, auch im Hinblick auf das drohende FORTH-Buch der FORTH-Gesellschaft, ganze Themen als Arbeitsaufgaben ausschreiben. Das müßte doch Aktivitäten auslösen, wo doch auch der Winter vor der Türe steht.

Allgemein wollte ich auch mal anfragen, wie man dem volksFORTH eine kontext-sensitive Hilfe verpassen kann? Vielleicht das PD-Programm PopHelp nehmen und die Files der Dokumentation dafür entsprechend straffen und aufbereiten??

Generell sollte man auch prüfen, inwieweit Beiträge zur 'Vierten Dimension' in die volksFORTH-Dokumentation einfließen könnten: Der Beitrag »Assembler im Vergleich« war ein Musterbeispiel für das, was die volksFORTH-Dokumentation benötigt.

Jörg Staben, 4010 Hilden

Leserbrief zum Artikel "Der fleißige Biber"

Sehr geehrter Herr Prinz, in Ihrem Artikel über den fleißigen Biber schreiben Sie, daß der »... bisher fleißigste Biber ... nach 134.467 Schritten 501 Einsen« produziere. Dies ist aber nicht mehr richtig. In dem Spektrum-Sonderheft "Computer-Kurzweil II" von 1988 ist unter anderem ein Artikel über die Biber enthalten. Dort wird auch ein Biber vorgestellt, der

nach 2.133.492 (!) Schritten 1915 Einsen produziert (s. Abb.). Dieser wurde im Dezember 1984 von G. Uhing (New York) entdeckt. Dafür baute Uhing einen »Turingmaschinen-Simulator, ... der in jeder Sekunde sieben Millionen Turingmaschinenübergänge ausführt«. Die Kosten für diesen Simulator waren unter 100 Dollar(!).

Auf meinem Rechner (ein XT mit 8086 CPU und 8 MHz Takt) braucht der Biber für die 2.113.492 Schritte ca. 4 min. Mein Simulator wurde in Turbo-Pascal programmiert und ist um einiges schneller, als der FORTH-Biber von F. Prinz. Allerdings wird bei mir erst nach dem Halt die Anzahl der Einsen gezählt und nicht während des Abarbeitens.

Es wäre ein interessantes Projekt, zu versuchen, den produktivsten Biber für n Zustände herauszufinden. Die mögliche Anzahl von Bibern ist übrigens $[16n^2+32n+16]^n$, wobei n die Anzahl der Zustände angibt. Falls sich noch andere Interessenten finden, wäre es eine spannende Sache, einen schnellen Turingsimulator herzustellen.

Zustand	Eingabe	nächster Zustand	Ausgabe	Richtung
A	0	B	1	rechts
	1	C	1	links
B	0	A	0	links
	1	D	0	links
C	0	A	1	links
	1	H	1	links
D	0	B	1	links
	1	E	1	rechts
E	0	D	0	rechts
	1	B	0	rechts

Andreas Filmann
Postfach 14 43
August-Bebel-Str. 11
6454 Bruchköbel
Tel. (06181) 72861

Leserkritik

Ich bin von FORTH begeistert und darf mich inzwischen zu den Anhängern zählen.

Ich habe auch schon einige Programme geschrieben - Anwendungen in der Meßdatenerfassung und Auswertung.

Einerseits schätze ich die Vorteile von FORTH, die modulare Programmierung, die getrennte Prüfung jedes Bausteines, die im Vergleich zu anderen Sprachen meist höhere Ausführungsgeschwindigkeit.

Doch dann gibt es die Momente des Zweifels und der Enttäuschung. Vergebens sucht man nach einem Befehl, der einen Zahlenwert entgegennimmt, geschrieben in beliebiger Schreibweise mit oder ohne Komma, mit oder ohne 'E'. Man sucht vergebens.

Oder man braucht eine formatierte Ausgabe, z.B. für die Bildschirmausgabe eines veränderlichen Meßwertes. Mühsam bastelt man sich einen Befehl zusammen, um dann festzustellen, daß der Geschwindigkeitsvorteil von FORTH, gerade durch diesen Befehl, fast verschwunden ist!

Gerade FORTH-Neulinge kämpfen mit FORTH-Versionen, die meist nicht ausgereift sind. Da fehlen Befehle für Gleitkomma-Operationen. Nach schnellen Winkelfunktionen sucht man ebenfalls vergebens!

Ich denke, daß es hier ein weites Betätigungsfeld für die FORTH-Gesellschaft und ihr Magazin gibt.

Zeigen Sie doch einmal, wie man am besten Eingaben ohne Bremswirkung entgegennimmt oder Ausgaben formatiert.

Zeigen Sie, wie man Masken und Eingabefenster aufbaut (mit Hilfsprogrammen).

Ich würde mich freuen, wenn ich in FORTH Befehle hätte, die mir eine WINDOW-Technik erlauben, wie beispielsweise das Becker-Basic auf dem C64.

Rubrik Leserfragen:

Fragen (und natürlich die Antworten dazu) bringen drei wichtige Aspekte mit sich:

- der Leser erhält Hilfestellung, wenn er mal Schwierigkeiten hat
- aus einfachen Beispielen lernt man einen FORTH-typischen Programmierstil (Thinking FORTH!)
- die Redaktion erfährt, wo den Leser der Schuh drückt

Berichte aus der Praxis:

Bitten Sie die 'alten Hasen' unter den Lesern aus ihrer Praxis zu berichten. Ein gutes Beispiel war schon immer der beste Lehrmeister.

Josef Rieger
Rubensstr. 14
7560 Gaggenau 13

Lokale Gruppe

Ich hätte gerne Kontakte zu FORTH-Interessierten, auch der noch-nicht-Mitglieder, weil ich gerne eine lokale Gruppe aufmachen möchte. Dazu zählen besonders alle Benutzer des volksFORTH83.

Martin Holzapfel
Bassestr.17
3300 Braunschweig

Die neuesten Neuigkeiten

Anläßlich der euroFORML haben Christoph Krinninger und Ulrich Hoffmann die Fachgruppe Object-Orientiertes FORTH ins Leben gerufen. Die Arbeit und Kommunikation findet über die Mailbox der FORTH-Gesellschaft statt, Konferenz OOF.

Seit kurzem werden die beiden internationalen Konferenzen comp. lang. forth aus dem UUCP-Netz und FORTH-Interest-Group International FIGI von der Mailbox MFB gepollt. Es lohnt sich also, mal hereinzuschauen. Tel: 089/725 96 25, 24h, 300-2400 Baud, 8N1 .

Bericht aus der Darmstädter Gruppe

In der Umgebung von Darmstadt und Frankfurt greifen einige kräftig in die FORTH-Tasten, andere können es nur gelegentlich zum Treffen schaffen. Nachdem wir bei der Volkshochschule keinen Computer zur Verfügung haben und die Zeit so schnell um ist, trafen wir uns am 15.7. bei mir mit Computerbenutzung. Vorhaben wurden diskutiert, Lösungsmöglichkeiten betrachtet, kritisiert usw. Mit seinen enormen Möglichkeiten ist mir das volksFORTH im Vergleich zum fig-FORTH vielfach unübersichtlich und an manchen Stellen überfettet, dem Leserbrief von Claus Vogt, Berlin schließe ich mich über große Strecken an. Der Editor zur V 3.81.41 ist mit seinen jüngsten Fähigkeiten Zeilen aufzutrennen und zusammenzulöten enorm leistungsfähig. Aber er hat Tücken: eines Tages sehe ich mit Erstaunen, daß mein KERNEL.SCR 270 Screens hat, lauter leeres Zeug. Der Editor zur V 3.81.1, sagte einer am Telefon, macht das nicht - tatsächlich. Am 15.7. gab Frank Stüss mit einer Änderung einen Anstoß, damit weitere und ich denke außer MORE alle Möglichkeiten zu ungewollten Screens beseitigt sind. Allerdings nicht so gut wie bei der alten Version mit *piep* und Meldung. Wie beseitigt man überflüssige Screens? Ich fand nichts, vielleicht habe ich was übersehen, jedenfalls entstanden dazu 1SCRWEG und NSCRWEG. Im Taunus wird mit Hochdruck an Beiträgen für die nächste VD gearbeitet, darum habe ich auch die Gemeinschaftsarbeit übernommen, die SWORDS von Ronald Zech nach volksFORTH zu übertragen - mit seiner Genehmigung übrigens - und diesen Quelltext hier einzusenden (steht über die FORTH-Mailbox zur Verfügung und enthält einige Hilfsmittel wie z.B. ein .S, das gleichsinnig, wie die Stackkommentare ausgibt. Alles zusammengefaßt im Quelltext AS-TOOLS.SCR. Hat vielleicht jemand ein P6PLUS.PRN anzubieten?)

Das ist ein unvollständiger Bericht, die anderen Arbeiten sprechen für sich, da will ich nicht vorgreifen.

Andreas Soeder

euroFORML 1989

von Christoph Krinninger

Das »european FORTH Research and Modification Laboratory« euroFORML fand in diesem Jahr in Neunkirchen am Brand statt. Diese internationale Veranstaltung fand zum drittenmal in Deutschland statt und war sehr zur Freude der FORTH-Gesellschaft die Größte und Vielfältigste ihrer Art. Die Teilnehmer kamen aus folgenden Ländern: Belgien, Bulgarien, Bundesrepublik Deutschland, DDR, Frankreich, Großbritannien, Schweden, Schweiz, USA. Besonders positiv ist die Beteiligung aus Bulgarien, England und USA aufgefallen.

Aus der reichen Anzahl von Beiträgen möchte ich nur ein paar wenige herausgreifen, die mir positiv aufgefallen sind: Das traditionelle Screen-Konzept von FORTH wird schrittweise durch Werkzeuge des modernen Computer-Aided-Software-Engineering (CASE) ersetzt. Wolf Weijgaard aus der Schweiz stellte sein System "Holon" vor, indem nur noch eine Einteilung in Worte, Gruppen und Module existiert. Die Benutzeroberfläche dieses Systems hat gewisse Ähnlichkeiten mit SMALLTALK, ferner ist es in der Lage, einzelne Worte auch nachträglich neu zu kompilieren oder im Sourcecode zu verschieben, ohne neu kompilieren zu müssen. Dies wird dadurch erreicht, indem die Header getrennt vom Code kompiliert werden. Findet eine Änderung statt, so muß meist nur der Header neu eingefädelt werden. Die Struktur der Header ist weit aufwendiger als in Standard-FORTH Systemen, so ist die Linked-List in beiden Richtungen gefädelt, es existiert noch eine spezielle alphabetische Link sowie Pointer zum Sourcecode und zum FORTH-Code.

Frank Stüss, Frank Raschke und Detlef Heid, alle aus Frankfurt, stellten das Konzept eines Datenbank-orientierten FORTH-Systems vor. Auch hier wird das Problem, nämlich die Verwaltung von riesigen Software-Mengen, sehr unkonventionell und elegant gelöst. Näheres wird man sicher noch hier in der VD erfahren. In Sachen FORTH-Maschinen stellte Rick van Noorman von Harris USA den RTX-4000 vor. Dieser 32-Bit Prozessor ist an die WISC-Maschine von Phil Koopman angelehnt, erlaubt also die Schaffung neuer Maschinenbefehlen. Der RTX-4000 ist also intern microcodiert, der RTX-2000 im Gegensatz aber extern microcodiert.

Man darf gespannt sein, wann die ersten Exemplare auf dem Markt sind. Die viel spannendere Neuigkeit hatte George Shaw, bekannt u.a. durch die ACM SIGFORTH und das ANSI-FORTH Komittee, auf Lager. Der Vater von FORTH, Charles Moore hat in seiner Küche wieder einen Knüller vorbereitet: Eine 32-Bit FORTH-Maschine, die von OKI, Japan gefertigt werden wird. Die Sensation dabei: 20 MIPS für 20 Dollar (so lautete auch ein Workshop: What would you do with it?). Der Preis wird zu Beginn bei etwa $40 liegen, der Prozessor ist so ausgelegt, das man auf Scheckkartengröße den Prozessor sowie 1 MByte RAM unterbringt. OKI wird selbst ca. 2 Millionen Prozessoren abnehmen, vermutlich für Geräte wie Laserdrucker und ähnliches. Eigentlich ist der Chip ein DMA-Kontroller, da aber noch etwas Platz in einer Ecke war, hat man noch eine FORTH-Maschine auf den Chip gepackt, typisch FORTH. Wer jetzt noch unschlüssig ist, ob er sich eine FORTH-Maschine zulegen soll, soll ruhig auf diesen Chip warten, er wird sicher eine Revolution einleiten.

George Shaw berichtete ferner über die Arbeit im ANSI-FORTH Komitee. Die Arbeitsgruppe trifft sich viermal im Jahr, leider in immer wechselnder Besetzung. So werden je nach Mehrheitsverhältnisse Entschlüsse mal gefällt und anschließend gleich wieder widerrufen. Es liegt noch mindestens für ein Jahr Arbeit vor, wenn die Gruppe ihre Arbeit ab-

geschlossen hat, so dauert es noch 1-2 Jahre, bis der Standard veröffentlicht wird. Es dauert also noch mindestens zweieinhalb Jahre, bis das ANSI-FORTH fertig ist.

Sehr gut in Erinnerung habe ich noch den Vortrag von Jens Storjohann, der die philosophischen Hintergründe in Sachen FORTH beleuchtet hat. Nach seiner Darlegung ist es nur sehr natürlich, wenn man ein fanatischer FORTH-Anhänger ist. Neben den Berichten über diverse FORTH-Anwendungen gab es auch einige fundamentale Betrachtungen, so etwa über das Für und Wider bei diversen DO-LOOP's. Chris Hainsworth aus England, bekannt für seine Kontrollstrukturen, hat in unvergleichlich klarer Weise dargestellt, daß auch ganz selbstverständliche Strukturen, wie das DO-LOOP voll von Fallstricken ist. Bei den obligatorischen Sitzungen bei "Wine and candlelight" hat er auch ausgedehnt über die Unstimmigkeiten im ANSI-FORTH hingewiesen.

Leider kann ich nur einen ganz kleinen Ausschnitt aus dem reichhaltigen Programm zeigen, Interessenten verweise ich auf die Proceedings der euroFORML.

Die nächste euroFORML findet turnusgemäß in England statt, voraussichtlich im Herbst '90 in Southampton. Zusammenfassend kann ich nur wieder feststellen, daß noch keine Veranstaltung der FORTH-Gesellschaft so vielfältig und international war, ein herzlicher Dank noch an die Organisatorin, Marina Kern, die dafür gesorgt hat, daß alles reibungslos über die Bühne ging.

Kurzübersicht der Beiträge

- Some Experiences Implementing Floating-Point in FORTH, Frank Stüss
- Object Oriented Menu Creating System, K. V. Vassiliev
- DOing it right, Chris Hainsworth
- Module FORTH, Ulrich Hoffmann
- HAVE, Dot-IF, Dot-ELSE, Dot-THEN, Klaus Schleisiek-Kern
- Two levels of Parallelism or a way to think for Control System Design, L. Blagoev et. al.
- A Comparison of the Cooperative and preemptive Concurrent Scheduling Algorithms, Harvey Glass
- Software Development System DSW, L. Blagoev et. al.
- A Modular Approach to Robotic Control Systems, Paul E. Bennett
- Interactive Remote Compilation for Development and Machine Integration, Alan M. Robertson
- XSHELL: A Cross-development User Interface
- RTX 4000, Rick van Noorman
- DBGX - Harris RTX 2000 C-Compiler Debugger. Tom Hand
- The MARC4 family, Gerhard Goettle
- Programming and out view of man, Jens Storjohann
- Not Screens nor Files but Words, Wolf Weijgaard
- The future of FORTH, Bengt Grahn
- An RTX Recompiler for on-line Maintenance, Jonathan Lee
- From Block Files to the Twentieth Century, Andrew Waters
- A Data-Based FORTH, Frank Raschke
- A vision of Transparence, Stephen Pelc
- Memory Cards and FORTH: An update, Paul Frenger
- ZUG POWER STATION: An application of trainable Neural Nets, John Carpenter
- Ocean Bottom Seismic, Karsten Roederer

Testbericht: Ein Weg zum professionellen M680X0 FORTH

von M. Schultheis

Stichworte

- 32-Bit
- M680x0
- UNIX
- Multi I/O

Auf der Suche nach einem preiswerten FORTH, das die Vorteile eines großen Adreßraums und eine Fileverwaltung nach Unix- oder DOS-Art bietet, stieß ich auf das FORTHmacs. Die Firma Bradley FORTHware mit ihrem Besitzer Mitch Bradley entwickelte dieses System für Atari ST, Macintosh, Sun Workstation und 680X0 Unix Systeme.

Die Version 1.1 für Atari ST ist in der FORTH-Mailbox zur privaten Nutzung verfügbar, und als Disk über die FORTH-Gesellschaft ohne Lizenzgebühr zu bekommen. Es handelt sich jedoch nicht um ein Public-Domain-Programm. Ein User's Guide und den Source Code erhält man für je $50. Wer Luftpost erwartet, denke an einen Versandkostenzuschlag bis zu 10%.

Nun zu den Details, die einen alten DV'ler und FORTH-Fan so begeistern:

Ein Relativlader bringt das FORTH ins RAM und rechnet alle 32-Adressen aus. FORTHmacs kann sich daher in der Arbeitsgeschwindigkeit mit jedem gefädelten FORTH messen.

Die Fileverwaltung versteht alle wichtigen Unix- und MS-DOS-Anweisungen. Als Editor wird eine einfache Version des EMACS eingesetzt. Bei Verwendung mehrerer EMACS-Puffer kommt auch in FORTH Freude auf. Dies senkt die Hemmschwelle für Nicht-nur-FORTH-Anwender die mit »FORTH for fun« einen Versuch wagen wollen. Mitch Bradley ist kein Screen-Anhänger. Sein FORTH bietet aber Hilfsworte zum Laden und Umsetzen von Screens.

Worte wie APPEND-TO-FILE für die Ausgabeumleitung, EOPEN für Zugriffe auf EMACS-Puffer und +LOAD zum Interpretieren ab der letzten Cursorposition im RAM sind nur einige Beispiele für die Erfüllung fast aller Wünsche. Gleichzeitiges Öffnen mehrere Files ist kein Problem und sieht dann noch einfacher als in C aus.

Die Verwaltung der Befehlszeile genügt höchsten Ansprüchen. Über die Tasten UP and DOWN sind die letzten acht Eingaben wiederholbar. Diese Zeile kann mit den entsprechenden EMACS-Anweisungen editiert werden. Das Ganze ist die reine Freude für tippfaule Anwender.

Das Wort 50LINES kann dazu verwendet werden auf der oberen Hälfte des Bildschirms den FORTH-Code des Editors und darunter die Testkommandos und deren Ergebnisse gleichzeitig anzuzeigen. Eine eindrucksvolle Demonstration für modulares Vorgehen, im besten FORTH-Stil, der sich kein objektiver Betrachter entziehen kann.

Assembler, Multitasking, schnelle Graphik, GEMDOS-Calls und ein durchdachtes String-Konzept sind vorhanden.

Bradley FORTHware bietet außerdem verschiedene Programme und Dienstleistungen an; darunter z.B. Floating Point, Music Language For MIDI, Spreadsheet und ein GEM Window Interface.

Das System verwendet leider (noch) keinen IBM-Zeichensatz und daher fehlen die deutschen Sonderzeichen.

Hier finden sicher viele FORTH Fans ein Aufbau-System, wenn Sie mit ihrem volksFORTH ausreichend Erfahrungen gesammelt haben und an dessen Grenzen stoßen. Profis können ihrem MAC oder ihrer SUN jetzt, zu kleinen Preisen, mal zeigen wie einfach und schnell ein Problem in FORTH gelöst wird. Ein Nachsatz zu den Aktivitäten der 32-Bitler in der FORTH-Gesellschaft. Bei uns in München konnte ich bisher keine M680x0-Aktivitäten ausmachen die ähnlich begeistern wie die von Mitch Bradley.

Testbericht: DRUMA-FORTH - ein interessantes FORTH-System ?

von Jörg Staben, 4010 Hilden, Tel.:02103-55609

Seit März 1988 wirbt eine Anzeige in der Zeitschrift BYTE für das FORTH-System der Firma **DRUMA FORTH Inc.** mit der Aussage:

"Ein auf 320KB konfiguriertes DRUMA-FORTH ist so schnell wie ein konventionelles 64KB FORTH-System. Diese 320KB stellen für DRUMA-FORTH keine Grenze dar, Programme können bis 4 GigaByte Code und Daten enthalten. Dennoch gibt es keine spürbaren Unterschiede in der Benutzung zu einem 'normalen' System."

Speicherverwaltung

Dies ist dann auch die Stärke von **DRUMA-FORTH v1.1**, wobei die Aussage über eine Programmgröße von 4GigaByte mangels Anwendung und Hauptspeicher nicht nachgeprüft werden konnte. Die Überschreitung der 64KB-Grenze wird durch eine Unterteilung in 9 logische Segmente erreicht. So liegen der Terminal Input Buffer, andere Puffer, die Code und die Colon Definitionen, die Variablen und beide Stacks sowie die Daten in jeweils eigenen Segmenten.

Da DRUMA FORTH strikt dem FORTH-83 Standard entspricht, ist diese logische Segmentierung für den Anwender völlig transparent; die Speicherzugriffe über @ und ! arbeiten innerhalb der 320KB-Grenze wie gewohnt. Lediglich darüberhinaus und für einige spezielle Segmente, die mit der memory allocation Funktion des MSDOS verkleinert oder vergrößert werden, stehen verschiedene Speicherzugriffsmöglichkeiten über 32Bit-Adressen oder segmentierte Adressen zur Verfügung.

Die Segmentierung des Compilers bringt noch weitere Vorteile mit sich: Da auch die Namensfelder (header) der Worte in einem eigenen Segment abgelegt werden, kann dieses Segment später gelöscht werden und DRUMA-FORTH so headerless Code erzeugen. Dies spart Speicher, da die Namen für Worte - bis auf den Namen der Anwendung selbst - nach dem Kompilieren überflüssig sind. Sogar dieser eine Name kann gelöscht werden, wenn man mit **DEFAULT.WORD** die Startroutine festlegt.

Durch die Verlagerung von Variablen und Stacks in jeweils eigene Segmente findet eine Trennung statt zwischen unveränderlichem Speicher und solchem, der veränderlich ist. Dies vereinfacht das Erstellen von ROM-fähigem Code.

Allerdings ist der DRUMA-System auch entsprechend groß: Der KERNEL hat als EXE-File etwa 45 KB; mein Arbeitssystem belegt 90KB, wobei diese erst zu 70% genutzt sind. Dennoch kann die fertige Anwendung so klein als möglich gehalten werden, indem man headerlosen Code generiert und die Möglichkeiten dynamischer Speicherzuweisung für die einzelnen Segmente nutzt:

```
2 KBYTES TOKENS +ALLOCM

40 KBYTES HEADERS -ALLOCM
```

Geschwindigkeit

Auch der zweite Punkt der Werbe-Aussage trifft zu: DRUMA Inc. versichert, ihr FORTH System hätte bis 320K Code die volle Geschwindigkeit eines 16-Bit FORTH-Systems. Dies läßt sich leicht mit den unvermeidlichen Benchmarks überprüfen, wobei die Programmausführung von DRUMA-FORTH gegenüber dem volksFORTH oder dem F-PC etwa 10% schneller ist. Dem ist noch hinzuzufügen, daß DRUMA-FORTH eine stabile Programmierumgebung ist, in der noch keine merkwürdigen Abstürze festzustellen waren.

Sprachstandard

DRUMA ist nach FORTH-83 "strictly standard". Diesen Vorteil kann man nutzen, um fehlende oder gewohnte Funktionen einfach aus dem volks4TH (>EXPECT,PLACE) oder dem F83 (besseres DUMP) einzubinden. Abweichende Namen für Worte lassen sich über die ALIAS-Funktion gleichnamig machen:

```
ALIAS <altname> <neuname>
```

Erfreulich ist auch, wenn man auch neuere Ideen des FORTH verwirklicht findet und auf vertraute Syntax stößt. Die mitgelieferten Beispiele werden teilweise sogar Zeile für Zeile durch Kommentare erläutert:

```
Defer DEMO.BEHEAD
\ vectored keyword to execute
\ beheaded keyword
' MAIN.BHD Is DEMO.BEHEAD
\ vector DEMO.BEHEAD to execute
```

Diejenigen, die ihren Brodie lieben, finden auch:

```
DOER XXX
: YYY MAKE XXX ... ; oder
: YYY MAKE XXX ... ;AND ... ;
```

Stichworte

- DRUMA-FORTH-83
- Standard-FORTH
- segmentierter Compiler

Auch finden sich z.B. mit BOUNDS oder COMMA QUOTE (,") Erweiterungen, die sich oftmals mit dem volks4TH vollständig oder zumindest inhaltlich decken.

Ebenso ist das ONLY/ALSO-Vokabularkonzept vollständig enthalten, so daß durch **SEAL** nur noch nach bestimmte Vokabulare durchsucht werden und eine Anwendung mit einem genau auf den Benutzer zugeschnittenen Interpreter versehen werden kann.

Unter den Erweiterungen und Hilfsprogrammen begegnen dem Anwender gute alte Bekannte, mögen sie auch in anderen Systemen andere Namen tragen: MANY und TIMES heißen im volksFORTH zwar OFTEN und TIMES, aber viel wichtiger ist, daß auch hier die Worte im Quelltext vorliegen. Ebenso findet sich mit CPACK ein alter Bekannter wieder (das PLACE des volksFORTH), mit dem ein verschobener Speicherbereich zu einem counted String wird.

Für's Stringhandling stehen auch eine Reihe von besonderen Worten zur Verfügung. Man kann Strings splitten, verbinden und lexikalisch vergleichen, wobei die Amerikaner auf diese abstrusen deutschen Grafikzeichen mit den kleinen Pünktchen obendrauf natürlich keinerlei Rücksicht nehmen. Dennoch ist COMPARE sicherlich ein brauchbares Wort, für die Berücksichtigung der Umlaute braucht man nur den Quelltext zu ändern. Das Wort MATCH prüft, ob ein (Teil)String in einem gegebenen String enthalten ist und dient auch als Beispiel für eine umfangreichere Assemblerprogrammierung.

Quelltextformate

DRUMA FORTH arbeitet bevorzugt mit Textdateien, die auch sehr schnell kompiliert werden. Um der langen Diskussion um die 1K-Blöckchen ein Ende zu bereiten, steht parallel (!) ein entsprechendes Interface gleichzeitig Verfügung, wobei der Blockeditor zwar quälend ist, aber vollständig als Beispielprogramm im Quelltext vorliegt. Um auch bei stream files zu akzeptablen turn-around-Zeiten zu kommen, hat sich die Kombination aus einem residenten Editor und DRUMA FORTH bewährt. Als ASCII-Editor muß prinzipiell ein Fremdeditor eingesetzt werden, aber über EDLIN verfügt eh' jeder.

Alle Erweiterungen und Utilities - auch der Assembler - liegen im Quelltext vor. Damit sind die Hilfsprogramme zugleich Programmierbeispiele.

Mit **VIEW** und **DOC** sind vertraute Tools zur on line Hilfe und Dokumentation vorhanden, wobei hier die high-level Definitionen dieser HELP-Utilities sogar mit den entsprechenden Assembler-Definitionen verglichen werden können. Listing 1 ist ein Beispiel für Code in DRUMA FORTH.

Ebenso gibt es die Möglichkeit eines dynamischen [SAVE], mit dem während des Kompilierens oder in einer Zeitabhängigkeit immer wieder Systeme herausgeschrieben werden; so soll man eine MAKE-Utility realisieren können.

Interrupthandling

Der Fehler- und Ausnahme-Behandlung wird in DRUMA-FORTH viel Aufmerksamkeit geschenkt. Der Umgang mit Interrupts wird von »high level service«-Routinen unterstützt, mit denen man Interrupts abfragen, sichern oder umsetzen kann. DRUMA FORTH-83 stellt auch die vordefinierten Interrupt-Routinen **SET.0DIV**, **SET.OV**, **SET.BREAK** zur Verfügung, um die Gastrechner-Interrupts **divide-by-zero**, **overflow**, und **break** Interrupts auf eigenen DRUMA FORTH Service Routinen umzulenken. Zusätzlich können die ursprünglichen Routinen mit einem entsprechenden Satz an Worten, z.B. **RESET.0DIV**, wiederhergestellt werden.

Mit den Worten zur Interruptbehandlung kann ein Passwort-gesicherter Softwareschutz für die Tastatur realisiert werden, indem der CTRL.C-Interrupt umgesetzt wird:

```
Code PRINT.BIN ( u --)
\ Prints w as a binary number. Demonstrates the usage of assembly
\ BEGIN-UNTIL loop. The algorithm is as follows:
\ Shift BX, the top of the stack, left by one bit. The left most
\ bit is shifted out into the carry. If the bit was a zero, the
\ carry will be a zero, otherwise the carry will be one. Add 30h
\ to the carry to convert it to its equivalent ASCII character
\ code and print it. Do this sixteen times.
02  # AH MOV     \ dos display output function
010 # CX MOV     \ initialize cx with # of bits
BEGIN
0   # DL MOV     \ initialize dl to 0
BX SHL           \ shift bx left by one
030 # DL ADC     \ add carry the carry bit to 30h in dl
021  INT         \ call DOS character output function 02h
CX DEC           \ decrement CX
0= UNTIL         \ loop until cx=0
BX POP           \ adjust stack
next c;
```

Listing 1

Fazit

Mittlerweile ist DRUMA FORTH über ein Jahr auf dem Mark; es werden Zusatzpakete zum Modulmanagement, zur Windowverwaltung und ein Debugger/Profiler angeboten, wobei allerdings bis jetzt nur das Window-Paket erhältlich ist.

Hier relativiert sich die Preiswürdigkeit des DRUMA-FORTH. So günstig das doch sehr reichhaltige Grundpaket mit $79 auch ist, so sehr geht der Preis für das gesamte System in die Höhe, wenn man sich für Zusatzpakete entscheidet:

Ein Window-Tool schlägt mit $69, das Handbuch dazu (!) mit $25 zu Buche. Leider muß auch das wichtigste Werkzeug - der Disassembler/Tracer/Profiler - für teures Geld ($149 !) zusätzlich erworben werden, so daß man für das komplette System mit dem Modulmanagement nach aktuellem Dollar-Kurs fast 1000 DM bezahlt hätte.

Die Dokumentation auf Disk läßt ebenfalls noch Wünsche offen, da mehr Worte vorhanden als beschrieben sind. Wenn man sich schon in guter shareware-Manier die Dokumentation selbst ausdrucken muß, sollten wenigstens alle Worte dokumentiert sein.

```
HEX                             \ -=-
023 INAME LOCK.CC
DECIMAL                         \ -=-
: LOCKCC.SERVICE
>R >R
DROP DROP DROP                  \ because DOS leaves 3 extras items
R> R>
( ...)
IRETURN ;

: MAKE.LOCK ( --)               \ usage  <name> <password>
( ...)
PAD SPAN @ PASSWORD S!          \ from, count, to, len --;
LOCK.WARN
LOCK.CC ISET                    \ redirect ctrl-c
SET.Z;                          \ disable ctrl-z

: LOCK.TERMINAL ( --)
LOCK.CC ISERVICE LOCKCC.SERVICE
LOCK.CC IPRESERVE
MAKE.LOCK
POLL.PASS
LOCK.CC IRESTORE                \ restore ctrl-c
RESTORE.Z ;                     \ restore ctrl-z
```

Listing 2

Da stellt sich die Frage, ob man sich nicht mit dem respektablen Speicherangebot des PD-Systems **F-PC** von Tom Zimmer zufrieden gibt und den 10%igen Geschwindigkeitsverlust in Kauf nimmt.

Hier bekommt man für, ich glaube auch $79, das neueste System, für $49 ein Technical Reference Manual, das immerhin von C.H.Ting geschrieben ist und somit insgesamt viel mehr FORTH für sehr viel weniger Geld.

Quellenangaben:

[1] DRUMA-FORTH example Disk

Hinweise für Autoren

Auch in Zukunft möchten wir Beiträge veröffentlichen, die Sie uns hoffentlich in großer Zahl liefern werden. Schicken Sie Ihre Manuskripte bitte an die Redaktion der 'Vierten Dimension' D.LUDA Software, Gustav-Heimann-Ring 42, 8000 München 83, Tel. 089/6708355 oder legen Sie sie in der FORTH-Mailbox München 'Konferenz Vierte Dimension' ab (8N1 Tel. 089/7259625).

Am liebsten hätten wir die Manuskripte auf einer Diskette 5 1/4" (360 Kbyte oder 1,2 Mbyte) im IBM-Format oder einer 3 1/2" Diskette (Atari-Format oder 720 Kbyte IBM-Format). Ist Ihnen das nicht möglich, können Sie auch normale Texte auf Papier einsenden. Bei Bildern sollte allerdings darauf geachtet werden, daß ein möglich guter Kontrast vorliegt. Die Arbeiten sollten in dieser Reihenfolge enthalten:

- Kurzer Titel,
- Autor,
- Zusammenfassung (ca. 50 Worte),
- Schlüsselworte (ca. 5), Text,
- Quellenangaben,
- Illustrationen,
- Tabellen,
- Quellcode.

Die Beiträge werden überarbeitet. Falls ein ausführliches Lektorieren erforderlich ist, erhalten Sie vor der Wiedergabe den Beitrag zur Korrektur und Zustimmung zurück. Layouts werden nicht mehr zur Prüfung durch die Autoren vorgelegt. Autoren erhalten auf Wunsch ein kostenloses Exemplar der 'Vierten Dimension' mit ihrem Artikel.

Erfahrungsbericht: ASYST oder ein FORTH-Programm muß nicht immer klein sein

von Claus Vogt, Berlin

Ein sicher nicht alltägliches Experiment startete die lokale Gruppe Berlin (immer noch im Aufbau) bei ihrem Treffen am 28.09.89. Statt preiswerte kleine FORTH-Programme selbst zu schreiben und sie anschließend in der Publikums-Domäne zu plazieren, lud die Gruppe zwei Vertreter der Firma Keithley Instruments ein, um sich über das Meß-, Steuer- und Grafik-Paket ASYST informieren zu lassen. Gerne folgten der Einladung Mathias Hall (Keithley, Köln) und Michael Döring (Keithley, Berlin) und ließen uns einen Blick in das Innere ihres verdongelten 6.000.-DM-Päckchen werfen. Der Abend war sicher für beide Seiten lehrreich. Es gab mehr Anknüpfungspunkte als erwartet.

Asyst stammt aus Rochester. Beschäftigte der dortigen Universität suchten eine geschickte Möglichkeit, mathematische Standardmethoden sinnvoll ablegen und bedienen zu können. Daß sie dazu ausgerechnet die Sprache FORTH verwendeten, erstaunt wenig. Die Stadt hat immerhin auch in FORTH-Kreisen einen Ruf. Erst später erhielt Asyst die bekannten Grafikfähigkeiten, wurde von HP und später von Keithley Instruments übernommen. Schon diese kurze geschichtliche Einleitung zeigt, daß Herr Hall keine Standard-Vorführung des Systems machte, sondern auf die speziellen Wünsche der FORTH-Freunde zuschnitt. Asyst tat selbiges und meldete sich mit einem standesgemäßen 'ok'.

Asyst ist ein Paket zum Erfassen, mathematischen Aufbereiten und grafischen Darstellen von Meßdaten. Weiterhin dient es zur Steuerung. Es läuft auf kompatiblen PCs und reizt den Speicher trotz Overlays bis zur 640kB-Grenze aus. Die Bedienungsfreundlichkeit wird außer durch den wundervoll heimeligen 'ok'-Prompt durch Pull-down-Menüs und eine wirklich praktische Hilfe-Funktion gewährleistet. Das zuletzt eingegebene Wort wird (nach Drücken der Hilfe-Taste) als Suchmuster verwendet und alle der über 2800 Worte ausgegeben, die Teile davon enthalten. Auch die Glossare sind abrufbar.

Die Meßwerterfassung erfolgt über IEC-Bus und RS 232, Treiber stehen für 70 verschiedene Karten zur Verfügung, eigene Karten können angepaßt werden. Diverse mathematische Filterfunktionen stehen zur Verfügung, eigene Routinen können entweder in der Asyst-Sprache (FORTH or not FORTH?) oder in Microsoft Fortran und C geschrieben werden. Zur grafischen Ausgabe stehen 2-dimensionale XY-Plots, 3-dimensionale Isolinien und Netzgrafiken, sowie Torten und Balken bereit. Alle Parameter, gegen deren Änderung sich andere Grafikpakete gerne sperren, sind variabel - ob kleine oder große Häkchen an den Achsen, jede zweite oder nur jede dritte beschriftet, alles geht (ging jedenfalls während der Vorführung). Das Ganze natürlich mit vielen Fenstern, Kurven, Farben und in durchaus akzeptabler Geschwindigkeit, zumindest habe ich schon erheblich langsamere Fortran- und Pascal-Grafik gesehen.

Warum Asyst? Weil Asyst sich anpaßt

...sagt der Prospekt oft und gerne. Damit hat er Recht. Wer im obigen Absatz die aufgezählten Möglichkeiten zur Anpassung zählt, stellt fest: Alles kann angepaßt werden. Wer in FORTH programmiert, hat keine Probleme den Anwendenden eine komplette Programmiersprache zur Verfügung zu stellen. (Eher kostet es ihn Mühe, diese zu verstecken.) Dies ist ein zentrales Argument für das Programmieren in FORTH. (Oder dagegen.) In einem anderen Punkt muß ich den Prospekt allerdings kritisieren. Von FORTH wird hier nichts gesagt.. halt, es wird doch etwas gesagt. Beim Vergleich von Asyst mit anderen Programmiersprachen taucht FORTH auf. Allerdings erreicht es von 10 blauen Kästchen nur drei und liegt damit auf dem letzten Platz hinter der Masse (Basic, Fortran..) mit 4 und Asyst mit (wer hätte das gedacht?) 10 Kästchen. Aber wir wollen nicht nachtragend sein. Die anwesenden Firmenvertreter verraten ihren KundInnen zumindest, daß Asyst in FORTH geschrieben ist.

Stichwort

» ASYST

Ist Asyst FORTH oder eine eigene Programmiersprache?

Der kundige FORTH-Sophist merkt sofort: Die Frage ist falsch gestellt. Da ein FORTH-Programm aus Erweiterungen des FORTH-Kerns besteht, ist jedes FORTH-Programm eine andere Sprache und doch FORTH. Asyst ist aber nicht ganz so FORTH. Der FORTH-Kern ist headerless und damit versteckt. Es läßt sich zwar immer noch ' 2 3 * ' eingeben, aber die Bedeutung hat sich verschoben. Der '*'-Operator läßt sich nicht nur auf ganze Zahlen, sondern auch auf reelle und komplexe, ja sogar auf mehrdimensionale Felder anwenden. Diese Typüberprüfung ist sicher konzeptionell eins der interessantesten Merkmale von Asyst. Es scheint so zu sein, daß hier mit der Methode der multiplen Codefelder (ein Codefeld für jede erlaubte Typkombination) und dem Ablegen von Typinformation auf dem Stack aus dem generischen '*' der typ-spezifische Operator ausgewählt wird. Dies läßt sich vermuten, da (a) die Anzahl der Typen konstant ist, neue Typen

dürfen nicht eingeführt werden und (b) es Methoden gibt, bei bekannten Typen den spezifischen Operator kompilieren zu lassen. Andere FORTH-Worte arbeiten wie bekannt, die »Colondefinition (Asyst Terminologie)« wird auch mit einem ':' eingeleitet. Sogar ein COMPILE_OFF und COMPILE_ON ist vorhanden (wenn auch unter anderem Namen, nicht als '[' und ']'). Allerdings fehlt das Wort 'Literal' leider, so daß sie ihres Hauptzwecks beraubt sind. Dies war die einzige Kritik des Abends, die auf Probleme aufgrund zu wenig FORTH im Asyst zielte. Asyst hat einen zweiten Stack, auf dem Zeichenketten und Bitfelder residieren.

Wer programmiert Asyst ?

Asyst ist kein Anwendungsprogramm, es ist eine Programmiersprache. Typische KundInnen schreiben einige KBytes Code für ihre Anwendung. Allerdings wird hier oft im Fortran-Stil programmiert. Das Programm beginnt mit endlosen Variablen-Deklarationen. Das Mittel der »Unter-Colon-Definition« zwecks Entlastung der »Haupt-Colon-Definition« wird viel zu wenig verwendet. Ein wenig Leo Brodie lesen, würde da sicher weiterhelfen, aber wie soll der Mensch darauf kommen, wenn ihm kaum verraten wird, daß er gerade ein FORTH-Programm schreibt? Andere BenutzerInnengruppen nähern sich auch von der anderen Seite dem Problem. Sie sind FORTH-Profis und programmieren jetzt Asyst. Die Diskussion guten FORTH-Stils könnte vielleicht ein Ansatzpunkt für eine nützliche Zusammenarbeit zwischen FORTH-Gesellschaft und Asyst-Kundendienst sein. Eine Regel für die Arbeit mit typsensiblem FORTH hat uns Herr Hall auch schon beigebracht. Überall, wo möglich müssen DO..LOOP's durch die entsprechende Feldoperation ersetzt werden. Ist das das Geheimnis der Geschwindigkeit von Asyst?

Was fehlt?

Asyst ist zu umfangreich, um es an einem Abend vorzustellen. Deshalb nur Stichworte: Turnkey-Fähig, Menü-Tools, einfaches Multitasking, Felder max. 64KB, FFT, Dbase- und Lotus-Files, LIM-Memory, Editor, Statistik, Approximation

Schluß

Der Abend war interessant. Wir FORTHler haben ein ziemlich großes FORTH-Programm gesehen, das durchaus interessante Konzepte beinhaltet. Insbesondere für Herrn Hall war es eine Anregung sich näher mit den Wurzeln seines Systems zu beschäftigen. (Er bestellt jetzt ein volksFORTH, um FORTH kennenzulernen [..und bedauert sehr, daß es keine Grafik hat]). Anknüpfungspunkte zwischen FORTH-Gesellschaft und Keithley als Asyst-Kundendienst existieren z.B. beim Programmierstil. Eine eventuelle Zusammenarbeit mit der (in Gründung befindlichen) User-Group könnte ebenfalls geprüft werden. Denn Asyst verbreitet FORTH. Wenn das Land Schleswig-Holstein seine Universitäten in größerem Umfang mit Asyst ausstattet, lernen einige StudentInnen FORTH, die es sonst nicht täten.

Comparison Chart*

	ASYST	FORTRAN	BASIC	APL	PASCAL	C	FORTH
Compiled	■	■	■	some	■	■	■
Structured Flow of Control	■	■	some	■	■	■	■
Interactive	■	no	■	■	no	no	■
Completely Supports Complex Numbers	■	no	no	no	no	no	no
Integrated Data Manipulation/Analysis	■	no	no	no	no	no	no
High-Level Interactive Integrated Graphics	■	no	no	no	no	no	no
Integral Array Operations	■	no	no	■	no	no	no
Integrated Data Acquisition Hardware Interface	■	no	no	no	no	no	no
Integral Support of 8087/80287 Chips	■	■	■	some	■	■	some
Integral Support of Expanded Memory Boards	■	no	no	some	some	some	no

■ full capabilities

Das "GREENHORN-MODUL"

Was FORTH von LOGO lernen kann, wodurch LOGO überflüssig wird.

von Manorainjan - Martin Holzapfel

Daß viele Leute FORTH gegenüber Vorurteile haben, weil es so anders ist oder gar FORTH überhaupt nicht kennen, ist ein offenes Geheimnis. Die Erkenntnis, daß der Umgang mit Computern bald in den Rang von Kulturtechniken wie Lesen und Schreiben aufsteigen wird und man damit gar nicht früh genug anfangen kann, setzt sich auch langsam durch. Mit Hilfe dessen, was ich das Greenhorn-Modul nennen möchte, würde sowohl die Akzeptanz von FORTH, als auch der Wert von FORTH als Lernsprache für Kinder erhöht.

LOGO wurde vom MIT als Lernsprache für Kinder entwickelt und deshalb recht konsequent von der Maschine weg, hin zum Bildhaften Lernen gestaltet. Damit sind einerseits gute Erfolge erzielt worden, andererseits ist Logo wohl aber eine der langsamsten Sprachen geworden. Unter ATARI Logo (schlechte Implementation!) dauert ein 6 zeiliges Programm, daß den Bildschirm nach dem Zufallsprinzip mit S/W-Pixeln komplett ausfüllt ca 90 min! Mit FORTH kann ich das in ca. 1 min 30 sec d. h. in 1/60 der Zeit machen. Mit anderen Worten: Was in FORTH eine Minute dauert, das kann in LOGO leicht mal eine Stunde dauern.

Warum soll ein Greenhorn erst einen begeisterten Einstieg durch LOGO finden und sich dann nach einer »richtigen« Programmiersprache umschauen müssen? Warum soll sich der FORTH-Einsteiger, der sich ein volksFORTH hat kommen lassen, erst ins Studium der VDI-Aufrufe verbeißen müssen, bis er sein erstes Pixel auf den Schirm bekommt, so wie ich es tat?

Es wird Zeit, daß FORTH das Image als Geheimsprache für Technik-Freaks los und der Allgemeinheit zugänglich wird. Zu diesem Zwecke habe ich mich entschlossen, das Greenhorn-Modul zu bauen, daß dem Kernel sozusagen als freundliche Maske aufgesetzt wird, bei voller Gewährleistung der Transparenz des Systems.

Bild 1

Stichworte

» GREENHORN-Modul
» LOGO ersetzen
» Akzeptanz
» Lernsprache
» Turtle-Grafik

Nun bin ich aber selber noch ein Greenhorn, was zwei Folgen hat:

⇒Erstens brauche ich Rat und Tat der Eingeweihten

⇒Zweitens weiß ich, was ein Greenhorn wirklich braucht, denn: Wem von Euch alten Hasen würde es je einfallen an der Tauglichkeit der Begriffe FALSE/TRUE zu zweifeln? Ist nicht UNWAHR/WAHR eher ein Begriffspaar? Sind Werte WAHR? (Theologie der Gatter) Als noch nicht Abgehobener sage ich: "Das zu verwendende Begriffspaar lautet JA/NEIN, das kann man jedem Greenhorn ohne syntaktische Verrenkungen begreiflich machen"

- 1=1? JA!
- Bedingung Erfüllt (z.B. x>5 bei x=4)? NEIN!
- Bit 4 von Adresse xyz gesetzt? JA!
- Spannung am Ausgang? NEIN!

Ich bitte hiermit lautstark um Unterstützung, Anregung, Mitarbeit und Kritik. Die vorläufigen Vorgaben sind wie folgt:

- Das Modul ist ein File, das auf das normale FORTH einfach draufgeladen werden kann.
- Es wird mit der jeweiligen Public Domain Version als Kombination von Kernel, Fileinterface, Editor usw. ausgeliefert.
- Vollständig eingedeutscht.
- Umfangreiche Igelgrafik (die bei FORTH eigentlich Hasengrafik heißen müßte "Ick bin all hier!"), die aber Stiftgrafik heißen soll, denn hier soll nicht rumgeTurtelt werden. (Schild)Kröten und anderes Viehzeug haben in diesem Modul ebensowenig etwas zu suchen wie theologische Fragen (TRUE), denn:
- Die Wortwahl soll so sinnfällig sein wie nie zuvor (hier meine ich Wortsinn und nicht Sinnfrage, also verschont mich bitte mit spöttischen Bemerkungen!).
- Alle Zeichenfunktionen mit abschaltbarer Verzögerung, sonst macht es einfach WHAP! und die Zeichnung steht und ist damit aber nicht nachvollziehbar.

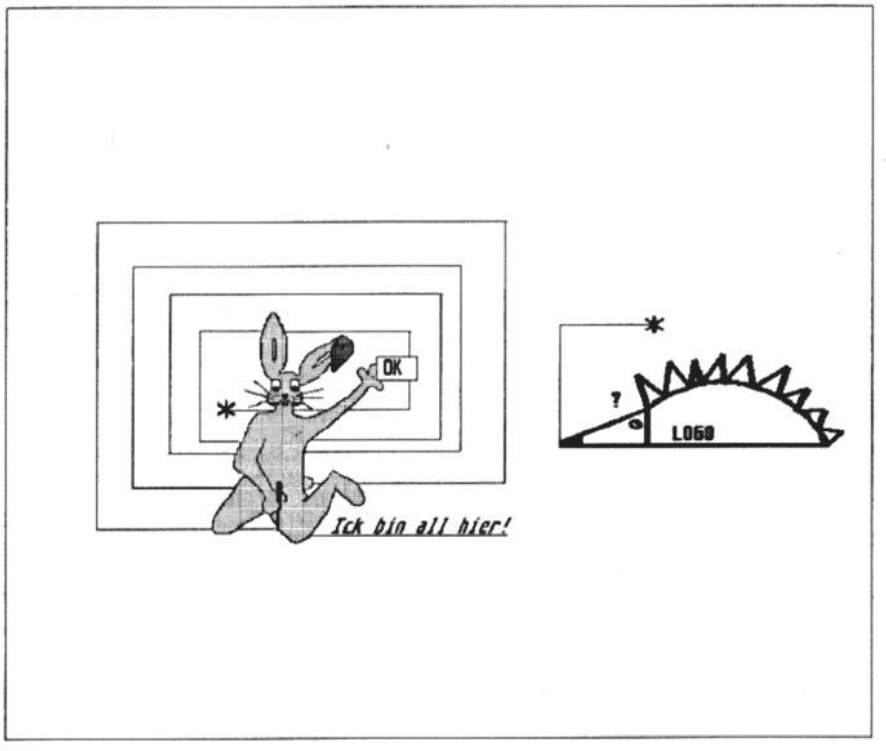


Bild 2

- Leicht zu handhabende Sound-Funktionen, so daß man ruckzuck ein kleines Klimperklavier bauen kann.
- Eventuell auch ein anderer Editor jenseits des 1024 Formats.
- Viele eingebaute Hilfstexte und (abschaltbare!) Fehlerabfang-Routinen, sowie ein Luxus - .S, ein Luxus-Dump und dergleichen, um alle Vorgänge und Zustände zu veranschaulichen.

Ich bitte die Leser dieses Magazins auch ganz ausdrücklich, mit ihren Frauen, Töchtern, Freundinnen, Müttern, Lehrerinnen und Großmüttern über den Sinn dieses Artikels zu diskutieren und sie zur Zusammenarbeit mit mir anzuregen. Es ist kein Zufall, daß ich in der Mitgliederliste der FORTH-Gesellschaft e.V. nur 2(!) weibliche Namen entdecken konnte. Man hat mir mal gesagt, daß Frauen eine natürliche Begabung für Sprachen haben, ich halte das für richtig. Das analytische Denken hingegen scheint ihnen verhältnismäßig schwer zu fallen (sagt die Statistik). Zum Training desselben ist die Beschäftigung mit dem Programmieren aber definitiv das effektivste Mittel.

Euer neuer

Manorainjan ALIAS
Martin Holzapfel,
Bassestraße 17,
3300 Braunschweig,
Tel: 0531-35 12 62

Die nächste 'Vierte Dimension' erscheint im März '90

ANS-FORTH

Required Words

Übersetzt von: Denise Luda

John R. Hayes - Laurel, Maryland

aus: FORTH Dimensions Volume XI, Nr. 4, Nov./Dez. 1989

Die Kreation eines neuen FORTH-Standards ist ein Drahtseilakt. Das Standardkomitee muß einen Mittelweg finden zwischen dem Verlangen FORTH auf den neuesten Stand der Computertechnologie zu bringen und der Notwendigkeit die Investitionen zu schützen, die in Verbindung mit FORTH-83 getätigt wurden. Dazu gehört auch der Zeitaufwand, der zum Erlernen von FORTH-83, sowie zum Erstellen von Applikationen, die mit Hilfe von FORTH-83 entwickelt wurden, notwendig war. Es ist schwer, eine Entscheidung zu treffen, da man es nicht jedem recht machen kann. In diesem Artikel kommen einige der Unterschiede zur Sprache, die es zwischen FORTH-83 und ANS FORTH gibt (Stand Juli 1989).

Die virtuelle FORTH-83 Maschine wurde so genau so ausgelegt, daß sie mit 16-Bit Daten unter Verwendung der Zweier-Komplement-Arithmetik, sowie mit einem Adreßspeicher, der als sukzessive Abfolge von 8-Bit Bytes organisiert ist, arbeitet. Dieses Modell entspricht den meisten gebräuchlichen Computern, die es 1983 gab. Die Computertechnologie entwickelte sich seither jedoch wesentlich weiter. Heutzutage sind 32-Bit Mikroprozessoren gang und gebe und es gibt verschiedene FORTH-Systeme, die direkt in die Hardware integriert sind. Unglücklicherweise haben gerade diese Computertypen Schwierigkeiten mit dem FORTH-83 Standard zu leben. Eines der Hauptziele des Standards besteht darin, zu ermöglichen, daß effiziente Implementationen des ANS FORTH auf möglichst vielen verschiedenen Prozessorvariationen installiert werden können.

FORTH entwickelte sich genauso weiter, wie die gesamte Computertechnologie. Es entstehen immer neue Implementationstechniken. So sind im Augenblick z.B. Implementationen mit Subroutine-threaded- oder Nativ-Code sehr gebräuchlich. Mit Hilfe von ANS FORTH soll es möglich sein, viele verschiedene Implementationsoptionen verwirklichen zu können. Außerdem wurden neue FORTH-Sprachkonstruktionen und Programmiertechniken entwickelt. Die ausgereiften und dadurch unentbehrlich gewordenen, werden standardisiert.

In Tabelle 1 sehen Sie alle Ergänzungen zum »Required Word Set«. Tabelle 2 zeigt was daraus gestrichen wurde. Der Rest dieses Artikels beschäftigt sich detailliert mit einigen dieser Änderungen. Andere Abweichungen von FORTH-83, wie z.B. Ergänzungen zur Fließkommaarithmetik und zum »File-Extension Word Set« werden in zukünftigen Artikeln behandelt werden.

Ergänzungen

Viele Ergänzungen zum »Required Word Set« sind minimal. So wurden z.B. die zweizelligen Operatoren um die Operatoren `2>R`, `2DROP`, `2DUP`, `2OVER`, `2R>` und `2SWAP` ergänzt. `2!`, `2@`, sowie `2*` stehen bei den meisten FORTH-Systemen bereits zur Verfügung und wurden deshalb in den Standard mitaufgenommen. `C,` vervollständigt den Satz der Zeichenoperatoren (`C@`, `C!` und `C,`).

Einige Ergänzungen übersteigen die Möglichkeiten von FORTH-83. Mit ANS FORTH ist es möglich, Stringliterals und Zeichenliterals zu verwenden. " (doppelte Anführungszeichen) leitet ein Stringliteral innerhalb einer Doppelpunktdefinition ein:

```
: HELLO
  " hello world" TYPE ;
```

`CHAR` bringt das erste Zeichen des nächsten Wortes im Eingabestrom auf dem Stack:

```
CHAR A CONSTANT 'A'
```

`[CHAR]` entspricht `CHAR` mit der Ausnahme, daß es das Zeichen als Literal kompiliert:

```
: FOO
  ... [CHAR] A EMIT ... ;
```

ANS FORTH verfügt über größere Möglichkeiten der Ablaufkontrolle. Mit `RECURSE` wird das Wort rekursiv aufgerufen, das `RECURSE` enthält (bei manchen Systemen heißt diese Anweisung `MYSELF`). Bei FORTH-83 war der Gebrauch von `EXIT` innerhalb einer `DO ... LOOP` Schleife nicht möglich. Der Grund bestand darin, daß es keine akzeptable Möglichkeit gab, um die Schleifenkontrollparameter vom Returnstack zu entfernen, bevor `EXIT` ausgeführt wurde. Diesem Mangel wurde in ANS FORTH durch die Hinzufügung von `UNLOOP` abgeholfen. `UNLOOP` ermöglicht es, daß ein Wort von einer Schleife aus mit `EXIT` verlassen werden kann.

```
DO ... LOOP:
DO
   ... IF
   ... UNLOOP
   EXIT THEN
LOOP
```

Dadurch werden viele heikle Probleme in Bezug auf den Programmablauf beseitigt.

Bei ANS FORTH ist es möglich, daß Programme auf den FORTH-Interpreter zugreifen. Viele FORTH-Systeme verfügen über ein Wort mit dem Namen `INTERPRET`. ANS FORTH beinhaltet ein ähnliches Wort mit dem Namen `EVALUATE`, das, sobald ihm ein String übergeben wird, diesen String als FORTH Quelltext interpretiert. Z.B.:

```
: 2+
  " 2 +" EVALUATE ;
    IMMEDIATE
```

2+, das aus dem ANS FORTH entfernt wurde, kann wie oben gezeigt definiert werden, damit eine Rückwärts-Kompatibilität hergestellt wird. Überall dort wo ein 2+ im darauf folgenden geladenen Quelltext auftaucht, wird die Phrase 2 + ausgewertet. Das entspricht dem Einsatz eines Texteditors, der nach allen im Quellcode vorkommenden 2+ sucht und sie durch 2 + ersetzt. Die Funktion ist wirklich sehr leistungsfähig.

Ein Hauptziel der ANS FORTH Verbesserungen ist es, daß sowohl 16-Bit als auch 32-Bit FORTH-Systeme dem Standard entsprechen können. Das wird dadurch erreicht, indem die Größe einer Daten-"Zelle" von der Implementation abhängig sein darf (z.B. 16 Bits bei FORTH-83). Sobald jedoch diese Verallgemeinerung gemacht wird, passiert etwas Bemerkenswertes. Zusätzlich zu den 16- und 32-Bit Prozessoren können eine Menge anderer Rechner, wie z.B. 18-, 20-, 24- oder 36-Bit Prozessoren durch dieses etwas allgemeinere Konzept einer Zelle verwandt werden. Deshalb wächst die Zahl der Computer, auf denen Standard-FORTH lauffähig ist, praktisch ins Unendliche.

ANS FORTH verfügt über Operatoren für portable Adressierung von Zellen im Speicher. `CELL+` wird benutzt, um von Zelle zu Zelle im Speicher zu wandern und `CELLS` dient dazu auszurechnen, wieviel Speicherplatz von einer bestimmten Anzahl von Zellen belegt wird.

`S>D` und `D>S` werden dazu benutzt, um einfach und doppelt genaue Zahlen ineinander umzurechnen. Bei Systemen, die mit der Zweierkomplement-Arithmetik arbeiten, sind diese Worte ganz einfach als `DUP 0<` bzw. `DROP` definiert. `S>D` und `D>S` jedoch machen den Quelltext leichter verständlich und ermöglichen es, daß der Code auf Systemen läuft, bei denen die zwei oben genannten Zweierkomplement-Tricks nicht funktionieren[1].

Wort	Art/Grund der Ergänzung
`2>R 2DROP 2DUP 2OVER 2R> 2SWAP`	Vervollständigen den Satz der Worte, die mit zwei Zellen arbeiten.
`2! 2@ 2*`	Essentiell
`C,`	Vervollständigt den Satz der Zeichenworte
`" CHAR [CHAR]`	String- und Zeichenliterals
`RECURSE UNLOOP`	Verbessert Ablaufkontrolle
`EVALUATE`	Der FORTH Interpreter
`CELL+ CELLS`	Portable Adressierung
`BYTE+ BYTES`	Portable Adressierung
`ALIGN REALIGN`	Portable Adressierungsausrichtung
`S>D D>S`	Portable Konversion
`POSTPONE`	Ersetzt COMPILE [COMPILE]
`MOVE`	Ersetzt CMOVE [CMOVE]

Tabelle 1: Ergänzungen zum Required Word Set

Streichungen

Streichungen aus dem »Required Word Set« müssen mit Vorsicht vorgenommen werden. Das Hinzufügen von neuen Worten geht problemlos, aber Streichungen können dazu führen, daß ansonsten funktionierende FORTH-83 Programme plötzlich auf ANS FORTH Systemen nicht mehr laufen. Deshalb sind die meisten der in Tabelle 2 aufgeführten Streichungen lediglich Reorganisationen der Wortsätze. So wurden z.B. die Block-Worte dem »Block Extension Word Set« und `VOCABULARY` wurde dem»Vocabulary Extension Word Set«hinzugefügt.

Einige veraltete FORTH-83 Worte wurden gestrichen, da sie entweder nicht effektiv genug oder zu schwierig zum Implementieren waren. In den meisten Fällen, wurden die gestrichenen Worte von Worten gleicher oder besserer Qualität ersetzt. So wurden z.B. `COMPILE` und `[COMPILE]` durch das einzelne Wort `POSTPONE` ersetzt. `COMPILE` stellte das größte Hindernis bei der Implementation von ANS FORTH in Verbindung mit Subroutine-threaded-Code dar. Diese Implementationstechnik wird aber bei der Implementation von FORTH auf FORTH-Chips bevorzugt. Da ein Standard-FORTH, das auf FORTH-Chips nicht ablauffähig ist, nicht sehr nützlich ist, wurde `POSTPONE` eingeführt. In allen Fällen - außer bei ein paar ganz speziellen - kann `POSTPONE` anstelle von `COMPILE` oder `[COMPILE]` verwendet werden. Damit die rückwärtige Kompatibilität gewährleistet ist, kann `COMPILE` folgendermaßen definiert werden:

```
: COMPILE
  POSTPONE POSTPONE ;
  IMMEDIATE
```

`[COMPILE]` kann dementsprechend wie folgt definiert werden:

```
: [COMPILE]
  POSTPONE POSTPONE ;
  IMMEDIATE
```

`CMOVE` und `CMOVE>` wurden durch den Operator `MOVE` ersetzt. FORTH-83 legt fest, daß sich mit Hilfe von `CMOVE` Muster im Speicher bilden, sobald sich Quellgebiet und Zielgebiet überlappen. So füllt z.B.:

```
CREATE X 10 ALLOT
0 X ! X X 1+
9 CMOVE
```

das Array X mit Nullen. Das bedeutet, daß `CMOVE` Byte für Byte verschieben muß. Bei vielen Computern, bei denen mehrere Bytes gleichzeitig übertragen werden können, ist das sehr uneffektiv. Dadurch wird das Leistungsvermögen von `CMOVE` als Blockoperator - seiner Hauptfunktion - verringert. Dementsprechend ist `MOVE` so konzipiert, daß es einen ganzen Speicherblock so schnell wie möglich transportieren kann. Die Er-

1 Ein zukünftiger Artikel wird die Möglichkeiten des ANS FORTH bzgl. der Portabilität noch ausführlich beschreiben.

zeugung von Mustern kann besser durch die Verwendung von C@, C! und DO...LOOP erreicht werden.

PICK und ROLL sind problematisch. Bei manchen Rechnerarchitekturen sind sie nicht sehr effektiv und sie haben allgemein einen sehr schlechten Ruf. Deshalb wurden sie aus dem »Required Word Set« entfernt und im »Extension Word Set« untergebracht. Unglücklicherweise wurden keine gleichwertigen Funktionen, wie z.B. lokale Variablen hinzugefügt. Es empfiehlt sich daher, daß Implementationen von ANS FORTH PICK und ROLL beinhalten, damit alte Programme lauffähig bleiben. Neue Programme sollten gleich so geschrieben werden, daß sie ohne PICK und ROLL auskommen. Zur Not können PICK und ROLL wie folgt definiert werden:

```
: PICK
  ?DUP IF SWAP >R
  1- RECURSE R>
  SWAP ELSE DUP THEN ;
: ROLL
  ?DUP IF SWAP >R
  1- RECURSE R>
  SWAP THEN ;
```

Zusammenfassung

ANS FORTH ist ein Abkömmling von FORTH-83. Deshalb kann das Wissen, das im Zusammenhang mit FORTH-83 erworben wurde, auf ANS FORTH übertragen werden. Meiner Meinung nach sind die größten Vorteile gegenüber FORTH-83 einmal darin zu sehen, daß es beim ANS FORTH mehr Implementationsmöglichkeiten gibt und andererseits, daß es viel mehr Computer gibt, auf denen das ANS FORTH lauffähig ist. Andererseits müssen FORTH-83 Programme nur leicht abgeändert werden, damit sie auf einer 16-Bit Implementation von ANS FORTH laufen.

John Hayes ist der Autor von verschiedenen FORTH Artikeln und eine Schlüsselfigur des VLSI FORTH Mikroprozessor Projektes.

Wort	Art/Grund der Streichung
BLOCK BLK BUFFER	Wurden in den Block Extension Word Set gebracht
FLUSH LOAD SAVE-BUFFERS UPDATE	Wurden in den Block Extension Word Set gebracht
VOCABULARY	Wurde in den Vocabulary Extension Word Set gebracht
2+ 2-	Überflüssig
FORTH-83	Überflüssig
COMPILE [COMPILE]	Nicht mit Native-Code kompatibel
CMOVE CMOVE>	Uneffizient
PICK ROLL	Uneffizient
PAD	Unsicher
FORGET	Wurde in den Reserved Wort Set gebracht

Tabelle 2: Streichungen aus dem Required Word Set

Graphtale Pflanzen

von Christoph Krinninger

Quelltext Service

Bei der Spezies der graphtalen Pflanzen handelt es sich um künstliche Geschöpfe, die realistisch und unheimlich zugleich wirken. Realistisch, weil das Muster ihrer Verästelungen dem wirklicher Pflanzen ähnelt, und unheimlich, weil es eben doch keine vertrauten Pflanzen sind. Ihr Name rührt daher, weil sie sowohl auf Graphen beruhen, als auch quasifrakaler Natur sind. Quasifraktal bedeutet, daß das Bildungsprinzip zwar bis auf die Auflösungsgrenze angewandt werden kann, dies aber aus praktischen Gründen nicht geschieht. Ein Ast verzweigt sich nicht endlos in immer kleinere Zweige, vielmehr findet die Unterteilung irgendwann ein willkürlich gesetztes Ende.

Die zu einer gegebenen Pflanze gehörenden Graphen werden durch »L-Systeme« erzeugt: eine Klasse von Grammatiken, die die dänische Biologin und Mathematikerin Astrid Lindenmeyer 1968 eingeführt hat. Ein L-System ist im wesentlichen ein Regelwerk zur Erzeugung neuer Zeichenketten aus alten. Die Regeln schreiben dabei vor, wie einzelne Zeichen

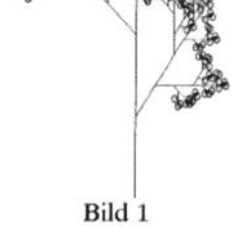


Bild 1

durch Folgen von Zeichen zu ersetzen sind. Benutzt man beispielsweise die Zahlen 0 und 1 sowie die Symbole [und], so läßt sich mit den folgenden Regeln ein breites Sortiment botanischer Formen erzeugen:

```
0  ->  1[0]1[0]0
1  ->  11
[  ->  [
]  ->  ]
```

Verwendet man als Startsymbol eine "0", so erhält man in der ersten Generation den Graphen:

```
1[0]1[0]0
```

In der zweiten Generation bildet sich dann daraus der folgende Graph:

```
11[1[0]1[0]0]11[1[0]1[0]0]1[0]1[0]0
```

Solche Zeichenketten kann man dann in baumartige Graphen verwandeln, indem man jede Zahl (0 oder 1) als Kante (Linie) und jede Klammer als Verzweigung auffaßt. Variiert man den Winkel und die Richtung der Verzweigung sowie die Länge der Verbindungsgeraden, so erhält man eine reiche Auswahl an Pflanzenarten. Genaueres über die Mechanismen erfährt man in der angegebenen Literatur.

Eingabevorschriften

Die Regeln für die Zeichenketten werden so definiert:

```
decl: rule1
      1 , 1 ,  ," 0"  ," 1[0]1[0]0"
      ( Start, Länge, Suchstring,
        Ergebnisstring )
      ende ,
```

In der ersten Zeile wird ein FORTH-Wort *rule1* im Dictionary abgelegt, anschließend die Parameter mit den Worten »,« (Komma) und »,"« (Komma-Quote) abgelegt. Die erste Zahl bestimmt, ab welcher Position in der Zeichenkette das Symbol beginnen soll, die zweite Zahl legt fest, wie lang das eigentliche Symbol ist. Anschließend werden der Such- und Ergebnisstring angegeben. Zur Verdeutlichung noch ein Beispiel: Wünscht man als eigentliches Symbol im Suchstring "axyzb" die Kombination "xyz" so lautet die Regel beispielsweise:

```
decl: rule2
      2 , 3 , ," axyzb"  ," 1234"
      ende ,
```

Wenn der zu bearbeitenden String "xxaxyzbxx" lautet, dann ergibt sich daraus "xxy1234bxx". Jede Regel muß mit *ende* , abgeschlossen werden. Das mit *decl:* erzeugte Wort kann dann mehrmals aufgerufen werden, beispielsweise:

```
: generationen  ( -- )
   rule1 rule1 rule1 ;
```

Mit dem Wort *arbeite* wird ein Anfangsstring im Speicher abgelegt und kann dann mit den mit *decl:* geschaffenen Worten bearbeitet werden:

```
: arbeiten      ( string -- )
   " abc" arbeite generationen ;
```

Für den so geschaffenen String müssen dann noch die Regeln für Abzweigungen, Astenden, Geraden und Blätter definiert werden. Der Aufbau ist sehr einheitlich:

```
Verzweig: verzweig1
      1 , 1 ,  ," ["  35 ,
      ( Start, Länge, String, Winkel )
      ende ,

Blatt: blatt1
   1 , 1 ,  ," ]"  1 ,
   ( Start, Länge, String, Blatt-Typ )
   ende ,

Astende: ende1
      1 , 1 ,  ," ]"  0 ,
      ( Start, Länge, String, Dummy )
      ende ,

Vorwärts: vor1
      1 , 1 ,  ," 1"  8 ,
      ( Start, Länge, String,
        Pixel-Anzahl )
      ende ,
```

Wie bei den Regeln für die Symbolbearbeitung sind die ersten beiden Zahlen die Position des Symbols und die Länge im Suchstring, dann folgt der Suchstring und zuletzt je nach Regel der Abzweigungswinkel, der Blatt-Typ oder die Pixelanzahl bei Geraden. Im Falle der Astenden ist dieser Parameter nur ein Dummy. Anschließend muß noch festgelegt werden, welche Attribute zu einer Pflanzenart gehören:

```
Art: art1  ( -- )
     vor1 verzweig1 blatt1 ende1 ;
```

Das Defining-Word *art:* legt ein FORTH-Wort an und legt die Worte für Geraden, Abzweigungen, Blätter und Astenden fest. Als letzten Arbeitsschritt muß nur noch die Grafik gezeichnet werden:

Stichworte

- Graphtale Pflanzen,
- volksFORTH,
- quasifraktal

```
: zeichnel      ( -- )
        arbeiten
        ( Regelstring erzeugen )
        art1
        ( Pfanzenart festlegen )
        320 370 -90
        ( X-pos  Y-pos Startwinkel )
        zeichne
        ( Pflanze zeichnen )
;
```

Leider kann in diesem Artikel nur oberflächlich auf das Problem der Graphtale eingegangen werden, Interessenten verweise ich auf die angegebene Literatur. Man kann das Programm noch beliebig ausbauen, etwa Zufallswerte für die Winkel festlegen oder abwechselnd nach links und nach rechts verzweigen. Anregungen findet man in der angegebenen Literatur. Das Programm wurde wie immer mit volksFORTH für den Atari ST geschrieben. Sollte ein Speichermangel auftreten, so kann man das Problem etwa durch Verringerung der Diskbuffer lösen (INCLUDE RELOCATE.SCR 2 BUFFERS <return>).

Literatur:

[1] Graphtale Pflanzen, Vinzenz Tragut, c't 4/89, S.210 ff

[2] Computer Kurzweil, A.K. Dewdney, Spektrum der Wissenschaft, 3/87, S.6

[3] Mathematical models for cellular interaction in development I and II, A. Lindenmayer, Journal of Theoret. Biology, 18/1968

SCR# 1

```
\                                                       25nov89 ck


Onlyforth gem also      \needs pline '      2 loadfrom vdi.scr
Onlyforth gem also      \needs overwrite    8 loadfrom vdi.scr


decimal

2 28 thru
```

SCR# 2

```
\ Zeichenbuffer                                         25nov89 ck

$2000 Constant bufflen
Create charbuffer       bufflen 2* allot    \ Zeichenbuffer

Variable frombuffer     Variable tobuffer   \ Div. Pointer
Variable fromlen        Variable tolen

: $>buffer      ( string -- )
   count dup dup 0= Abort" Nullstring! "
   fromlen !
   charbuffer swap move
   charbuffer dup frombuffer !   bufflen + tobuffer ! ;

' $>buffer Alias arbeite        \ for readability
```

SCR# 3

```
\ Regelbuffer                                           25nov89 ck

Variable >from.string   Variable >to.string

Variable s.start        Variable s.length

Create search.string    $80 allot   \ Such- und Regelbuffer
Create replace.string   $80 allot

: get.rule      ( adr -- adr' )     \ Regel in Parameter zerlegen
   dup @ s.start !
   2+ dup @ s.length !
   2+       dup count dup >r search.string place
   r> 1+ + dup count dup >r replace.string place
   r> 1+ + ;
```

SCR# 4

```
\ Regel verarbeiten                                     25nov89 ck

: found? ( -- flag )  \ Text mit Regel vergleichen
   search.string count  frombuffer @ >from.string @ +
   -capstext 0= ;

: search.char   ( rule -- rule flag )  \ Alle Regeln benutzen
   dup
   BEGIN
      dup @ not
   WHILE
      get.rule found? IF drop true exit THEN
   REPEAT drop
   false ;
```

SCR# 5

```
\ Zeichen unverändert kopieren                          25nov89 ck

: copy.char     ( -- )
   frombuffer @ >from.string @ + c@
   tobuffer @ >to.string @ + c!
   1 >from.string +!
   1 >to.string +!   1 tolen +! ;

: copy.leading  ( -- )
   s.start @ 1 ?DO copy.char LOOP ;     \ copy leading chars
```

SCR# 6

```
\ Zeichen kopieren/ersetzen                             25nov89 ck

: copy.trailing ( -- )
   search.string count nip          \ copy trailing chars
   s.length @ s.start @ + 1-
   ?DO copy.char LOOP ;

: replace.char  ( -- )
   copy.leading
   replace.string count dup >r
   tobuffer @ >to.string @ + swap move
   r> dup >to.string +! tolen +!
   s.length @ >from.string +!
   copy.trailing ;
```

SCR# 7

```
\ Regeln durchführen                                    25nov89 ck

: perform.rule  ( rule -- )
   >from.string off  >to.string off  tolen off
   BEGIN
      search.char
      IF    ( found )
         replace.char
      ELSE
         copy.char
      THEN ." ."
      fromlen @ >from.string @ =
   UNTIL drop
   frombuffer @ >r tobuffer @ frombuffer ! r> tobuffer !
   fromlen @ >r    tolen @ fromlen !       r> tolen ! ;
```

SCR# 8

```
\ Sinus-Berechnung                                      25nov89 ck

Create sintab
    0 ,   349 ,   698 ,  1045 ,  1392 ,  1736 ,  2079 ,  2419 ,
 2765 ,  3090 ,  3420 ,  3746 ,  4067 ,  4384 ,  4695 ,  5000 ,
 5299 ,  5592 ,  5878 ,  6157 ,  6428 ,  6691 ,  6947 ,  7193 ,
 7431 ,  7660 ,  7880 ,  8090 ,  8290 ,  8480 ,  8660 ,  8829 ,
 8988 ,  9135 ,  9272 ,  9397 ,  9511 ,  9613 ,  9703 ,  9781 ,
 9848 ,  9903 ,  9945 ,  9976 ,  9994 , 10000 ,

: ((sin ( n1 -- n2 )
   dup dup 2/ 2* =
   IF sintab + @ ELSE sintab + 1- dup @ swap 2+ @ + 2/ THEN ;
```

Graphtale Pflanzen

SCR# 9

```
\ Sinus- und Kosinusberechnung                          25nov89 ck

: (sin  ( n1 -- n2 )
   dup 180 > IF  180 - true  ELSE  false  THEN
   swap dup 90 > IF  negate 180 +  THEN
   ((sin  swap IF  negate  THEN ;

: sin   ( n1 -- n2 )
   dup 0<    IF  BEGIN 360 + dup    0> UNTIL  THEN
   dup 360 > IF  BEGIN 360 - dup 360 < UNTIL  THEN  (sin ;

: cos   ( n1 -- n2 )    90 + sin ;
```

SCR# 10

```
\ Turtle                                                25nov89 ck

Variable angle      Variable xcood    Variable ycood
Variable stepsize

Variable #lines          10 Constant #maxlines

Create line-array        #maxlines 10 * allot

: init.lines    ( -- )
   2 sf_interior 4 sf_style   true sf_perimeter
   overwrite solid   xcood @ ycood @ 2dup 2 pline
   xcood @ ycood @ line-array 2!  1 #lines !  -90 angle ! ;

: turn  ( deltaangle -- )       angle +! ;
```

SCR# 11

```
\ Turtle-Bewegung                                       25nov89 ck

: draw-line     ( -- )
        #lines @ 0
        DO line-array I 4 * + 2@ LOOP
        #lines @ [ gem ] pline
        xcood @ ycood @ line-array 2!  1 #lines +! ;

: move-turtle   ( -- )
        stepsize @ dup
        angle @ cos 10000 */  xcood @ + dup xcood ! swap
        angle @ sin 10000 */  ycood @ + dup ycood !
        line-array #lines @ 4 * + 2!
        1 #lines +!
        #lines @ #maxlines < not IF draw-line THEN ;
```

SCR# 12

```
\ Vektorisierte Regel-Parameter                         25nov89 ck

: Decl: ( -- )
        Create
        Does>   perform.rule ;

\ Diverse ALIAS'e für bessere Lesbarkeit

' Create Alias Vorwärts:
' Create Alias Verzweig:
' Create Alias Blatt:
' Create Alias Astende:

-1 Constant ende
```

SCR# 13

```
\ Defining word für Pflanzen-Art                        25nov89 ck

Variable Art

: Art:          ( -- )
        Create:   Does>  Art ! ;

| : (Art:   Create dup c, 2+ Does> c@ art @ + perform ;
0 (Art: vorwärts        (Art: verzweig
  (Art: blatt           (Art: astende    drop

Variable turtle.value

: inc.from.string       ( dummy -- )
        drop search.string count nip >from.string +! ;
```

SCR# 14

```
\ Zeichenregeln bearbeiten                              25nov89 ck

: get.draw.rule ( adr -- adr' )
        dup @ s.start !
        2+ dup @ s.length !
        2+ dup count dup >r search.string place
        r> + 1+ dup @ turtle.value ! 2+ ;

: zeichnen?     ( rule -- flag )
        BEGIN
                dup @ not
        WHILE
                get.draw.rule
                found? IF inc.from.string true exit THEN
        REPEAT drop false ;
```

SCR# 15

```
\ Turtle-Bewegungen                                     25nov89 ck

: move.forward  ( -- )  turtle.value @ stepsize !  move-turtle ;

: turn.turtle   ( -- angle xcood ycood )
   angle @  xcood @  ycood @
   turtle.value @
   turn ;

: thread.back?  ( -- flag )
  false
  0 #lines @ DO
        -rot line-array I 4 * + 2@ 2over d=
        IF rot drop #lines @ I - 1- true leave THEN  rot
  -1 +LOOP ;
```

SCR# 16

```
\ Zurückfädeln                                          25nov89 ck

: thread.back   ( n -- )
   dup >r 0 DO line-array #lines @ I - 2 -  4 * + 2@
   line-array #lines @ I + 4 * + 2! LOOP r> #lines +!
   ycood ! xcood ! angle ! ;

\\

Da das VDI-Metafile nur begrenzte Aufnahmekapazität bietet, mu
man versuchen, möglichst wenige Vektoren zu verwenden.
Durch das zurückfädeln wird versucht, Vektoren zusammenzufassen.
```

SCR# 17

```
\ Möglichst lange Vektorenkette bilden                  25nov89 ck

: branch.back   ( -- )
  thread.back?
  IF
     thread.back
     #lines @ #maxlines < not IF draw-line THEN
  ELSE
     draw-line     depth 0> IF  ycood !  xcood !  angle ! THEN
     xcood @ ycood @ line-array 2!  1 #lines !
  THEN ;
```

SCR# 18

```
\ Verschiedene "Blatt-Formen"                           25nov89 ck

Variable kugel.count

: .kugel ( -- )
        kugel.count @ abs 4 >
        IF
           xcood @ ycood @ 6 circle  branch.back
           kugel.count off
        ELSE
           1 kugel.count +!   branch.back
        THEN ;

: .blatt ( -- )
   xcood @ ycood @ 3 circle branch.back ;
```

Graphtale Pflanzen

SCR# 19

```
\ Blätter zeichnen, Zeichenkette zeichnen          25nov89 ck

: blatt.zeichnen        ( -- )
        turtle.value @ 1 case? IF .blatt exit THEN
                       3 case? IF .kugel exit THEN
        Abort" Falsches Blatt " ;

: draw.char     ( -- )
   vorwärts zeichnen? IF move.forward   exit THEN
   verzweig zeichnen? IF turn.turtle    exit THEN
   blatt    zeichnen? IF blatt.zeichnen exit THEN
   astende  zeichnen? IF branch.back    exit THEN
   1 >from.string +! ;
```

SCR# 20

```
\ Hauptwort zum Zeichnen                           25nov89 ck

: zeichne       ( x-coord y-coord angle -- )
   page
   >r  ycood !  xcood ! init.lines r> angle !
   >from.string off
   BEGIN
      draw.char
      >from.string @ fromlen @ =
   UNTIL draw-line  ;
```

SCR# 21

```
\                                                  25nov89 ck

decl: rule1     \ Zeichenregel für Tannenbaum
        1 , 1 ,   ," 0"  ," 1[0]{0)0"
        1 , 1 ,   ," 1"  ," 1"          ende ,

decl: rule2     \ Zeichenregel für Laubbaum
        1 , 1 ,   ," 0"  ," 1[0]1(0)0"
        1 , 1 ,   ," 1"  ," 11"         ende ,
```

SCR# 22

```
\ Generationen berechnen                           25nov89 ck

: generation1   ( -- )   rule1 rule1 rule1 rule1 ;
: generation2   ( -- )   rule1 rule1 rule1 rule1 rule1 ;

: generation3   ( -- )   rule2 rule2 rule2 ;
: generation4   ( -- )   rule2 rule2 rule2 rule2 rule2 ;

Verzweig: verzweig1     \ Zeichenregel für Verzweigungen
        1 , 1 ,  ," ["    35 ,
        1 , 1 ,  ," {"   -35 ,          ende ,

Verzweig: verzweig2
        1 , 1 ,  ," ["    45 ,
        1 , 1 ,  ," {"   -45 ,          ende ,
```

SCR# 23

```
\                                                  25nov89 ck

Blatt: blatt1   \ Blätter für Weinachtsbaum (Kugeln)
        1 , 1 ,  ," )"   3 ,
        1 , 1 ,  ," ]"   3 ,            ende ,

Blatt: blatt2   \ Blätter für Laubbaum
        1 , 1 ,  ," )"   1 ,
        1 , 1 ,  ," ]"   1 ,            ende ,

Astende: ende1  \ Früchte/Blüten für Schnittlauch
        1 , 1 ,  ," ]"   0 ,
        1 , 1 ,  ," )"   0 ,            ende ,
```

SCR# 24

```
\                                                  25nov89 ck

Vorwärts: vor1  \ Zeichenregeln für Vorwärtsbewegung
        1 , 1 ,  ," 1" 10 ,
        1 , 1 ,  ," 0" 10 ,             ende ,

Vorwärts: vor2
        1 , 1 ,  ," 1" 4 ,
        1 , 1 ,  ," 0" 4 ,              ende ,


Art: art1       \ Pflanzenart für Weihnachtsbaum
        vor1 verzweig1 blatt1 ende1 ;

Art: art2       \ Pflanzenart für Laubbaum
        vor2 verzweig2 blatt2 ende1 ;
```

SCR# 25

```
\                                                  25nov89 ck

\ Zeichenketten ermitteln

: arbeite1      ( -- )    " 0" arbeite generation1 ;
: arbeite2      ( -- )    " 0" arbeite generation2 ;
: arbeite3      ( -- )    " 0]" arbeite generation3 ;
: arbeite4      ( -- )    " 0]" arbeite generation4 ;

\ Zeichenketten zeichnen

: zeichne1      ( -- )    arbeite1  320 50 90 art1 zeichne ;
: zeichne2      ( -- )    arbeite2  320 50 90 art1 zeichne ;
: zeichne3      ( -- )    arbeite3  320 350 -90 art2 zeichne ;
: zeichne4      ( -- )    arbeite4  320 350 -90 art2 zeichne ;
```

SCR# 26

```
\ Zeichenregeln für Schnittlauch                   25nov89 ck

decl: rule3
        1 , 1 ,   ," A]"  ," AA]B]0"
        1 , 1 ,   ," B]"  ," BB]A]0"
        1 , 1 ,   ," 0"   ," A]B]10"
        1 , 1 ,   ," 1"   ," 11"        ende ,

decl: rule4
        1 , 2 ,   ," A]"  ," AA]B]0"
        1 , 2 ,   ," B]"  ," BB]A]0"
        1 , 1 ,   ," 0"   ," A]B]10"
        1 , 1 ,   ," 1"   ," 11"        ende ,
```

SCR# 27

```
\ Zeichenregeln für Schnittlauch                   25nov89 ck

decl: rule5
        1 , 1 ,   ," A"   ," -A"
        1 , 1 ,   ," B"   ," +A"        ende ,

: generation5 ( -- ) rule4 rule3 rule3 rule3 rule3 rule3 rule5 ;

Verzweig: verzweig3
        1 , 1 ,  ," -"   -15 ,
        1 , 1 ,  ," +"    15 ,          ende ,

Vorwärts: vor3
        1 , 1 ,  ," A" 15 ,
        1 , 1 ,  ," 1" 10 ,
        1 , 1 ,  ," 0" 20 ,             ende ,
```

SCR# 28

```
\ Zeichenregeln für Schnittlauch                   25nov89 ck

Astende: ende3
        1 , 1 ,  ," ]"   0 ,            ende ,

Blatt: blatt3
        1 , 1 ,  ," ]"   1 ,            ende ,

Art: art3
        vor3 verzweig3 blatt3 ende3 ;

: arbeite5      ( -- )    " A]B]" arbeite generation5 ;
: zeichne5      ( -- )    arbeite5  320 370 -90 art3 zeichne ;

: graphtale     ( -- )
   zeichne1 zeichne2 zeichne3 zeichne4 zeichne5 ;
```

Neuer Stringstack mit Heap

Konrad Scheller, Forchheim

Stichworte

- Stringstack,
- Heap,
- Pointer

Quelltext Service

Klaus Schleisiek hat in seiner 'Vierten Dimension' ein Beispiel seiner Programmierkunst gegeben. Sein Stringstack ist ein Vorbild für alle, die gut FORTH programmieren wollen. Kaum eine Definition ist länger als eine Zeile, und er hat überall gut Leerräume verteilt, um die Programmierstruktur deutlich zu machen.

Gleichwohl hat die Implementation dieses Stringstacks einige Schwächen. Zum einen ist sie ziemlich langsam. Bei einem *"DUP* oder einem *"SWAP* wird jedesmal ein neuer String gebildet. Außerdem ist dieses Konzept Speicherplatz-fressend.

Ein weiterer Punkt ist, daß nach wie vor die String-Variablen nicht wirklich in den Stack eingebunden sind. Sie müssen ebenso wie vorher mit einer maximalen Länge angemeldet werden, die sie später nicht überschreiten dürfen.

Aus diesem Grund habe ich den Stringstack von Klaus noch einmal überarbeitet und einen sogenannten HEAP eingeführt. Auf Deutsch hiesse das soviel wie »Halde« oder »Haufen« und das ist es auch. Kurz gesagt, ich werfe einfach alle Strings auf einen Haufen. Natürlich muß ich mir dabei merken, wo jeder einzelne String liegt, sonst gibt's Chaos. Und wie es sich für einen anständigen Haufen gehört, muß auch er manchmal aufgeräumt werden.

Um es klarer zu machen, ist hier eine Skizze.

Betrachten Sie bitte Abbildung 1. Der Haufen ist einfach ein freier Speicherbereich, n Bytes groß. Der Anfang des Haufens ist durch die Variable *BOTTOM* bestimmt, das Ende durch die Variable *TOP*. Dieser Haufen wird jetzt von oben nach unten mit Strings angefüllt. Die Variable *LOWEST* zeigt hierbei immer auf den Anfang des untersten Strings. So weit, so gut, aber wie finde ich die Strings? Klaus' Lösung war einfach: Da es sich um 'gecountete' Strings handelt, die die Länge im ersten Byte stehen haben, kann man sich ja von oben nach unten vortasten. Meine Lösung ist etwas komplizierter.

Betrachten Sie nun Abbildung 2. Ich lege mir hier einen Stack an. Dieser Stack ist *nur für Stringadressen* gedacht. Diese Adresse »zeigt« dann auf einen String im Haufen. Deshalb nennt man auch solche Adressen Zeiger, oder auf Deutsch Pointer. Diese Zeiger sind nun als Stack organisiert.

Auf Bild 3 können Sie nun sehen, wie ein String abgelegt wird. Der String selbst wird von 'top' ab in den Speicher gelegt. 'lowest' zeigt nun auf den Anfang des untersten, in diesem Falle einzigen Strings. Gleichzeitig enthält nun der erste Zeiger im Stack diese Adresse.

Auf Bild 4 sehen Sie, wie ein weiterer String dazukommt. Der zweite Zeiger des Stacks zeigt nun auf den zweiten String. Lowest wurde entsprechend korrigiert. Wie man nun auch leicht erkennt, gibt die Differenz zwischen 'lowest' und 'bottom' den freien Speicherplatz auf den Haufen an.

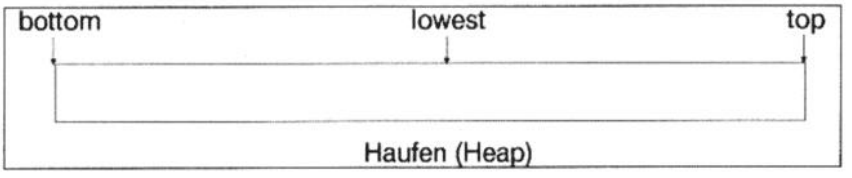


Bild 1

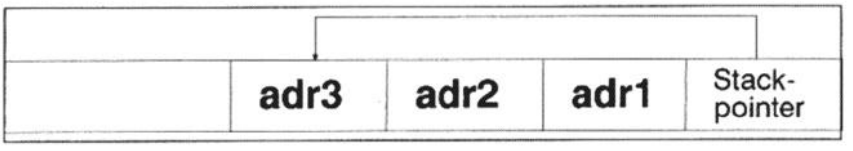


Bild 2

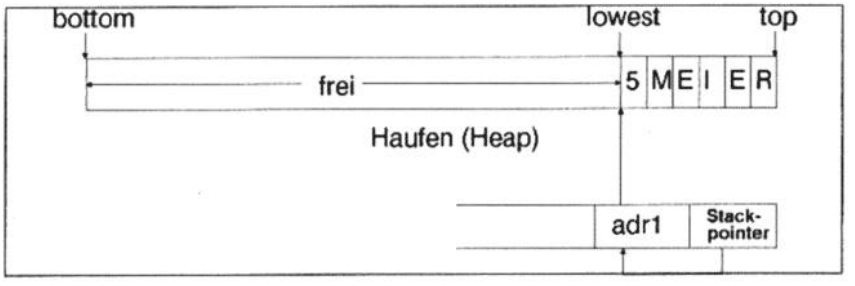


Bild 3

Nun lassen wir einmal ein *"SWAP* ausführen. Auf Bild 5 können Sie das Ergebnis ansehen: Die Strings selbst bleiben unberührt! Lediglich die Adressen der Zeiger werden vertauscht. Nun offenbart sich ganz eindeutig der Vorteil dieses Konzepts: Viel schneller!

Aber das Ganze hat noch einen weiteren Vorteil. Betrachten Sie Bild 6. Es zeigt den Stack und den Heap nach Ausführung von *"DUP*. Obwohl der String "Schulze" zweimal auf dem Stack liegt, existiert er nur einmal im Speicher! Das ist nicht gefährlich, obwohl es so aussieht. Wenn man nur Stringoperationen nutzt und nicht direkt im Haufen rumwühlt, passiert kein Unglück. Auf alle Fälle spart dieses Konzept eine Menge Speicherplatz und Rechenzeit.

Bleiben wir noch einen Augenblick bei Bild 4 (nur 2 Strings). Was geschieht jetzt, wenn wir einen String mit *"DROP* wegnehmen? Die Antwort gibt uns Bild 7. Der Stringheap bleibt wieder unverändert! Nur der Zeiger auf den String "Schulze" wurde vom Stack entfernt, alles andere blieb, wie es war. Auch 'lowest' wurde nicht korrigiert!

Wenn man nun genau hinschaut, entdeckt man, daß der String "Schulze" vollständig überflüssig geworden ist. Es handelt sich um sogenannten »Stringmüll«. Im Laufe der Zeit kann sich auf einem Haufen beträchtlich viel Müll ansammeln.

Irgendwanneinmal kommt dann die Stunde der Wahrheit: Auf dem Haufen ist kein Platz mehr! Jetzt tritt die sogenannte »Garbage Collection« auf den Plan. Diese »Müllabfuhr« sucht alle gültigen Strings auf dem Haufen und schiebt sie zusammen an das obere Ende des Haufens. Das schafft im allgemeinen genug Platz. Sollte der Platz immer noch nicht ausreichen, war der Haufen also zu klein angelegt. Das führt dann zu einer Fehlermeldung "String Memory full".

In diesem Programmstück habe ich noch die Stringvariable auf den Haufen geworfen. Das hat den Vorteil, daß eine Stringvariable jetzt wirklich eine variable Länge haben kann, von Null bis 255.

Trotzdem verbraucht sie nur soviel Speicherplatz, wie es ihrer tatsächlichen Länge entspricht!

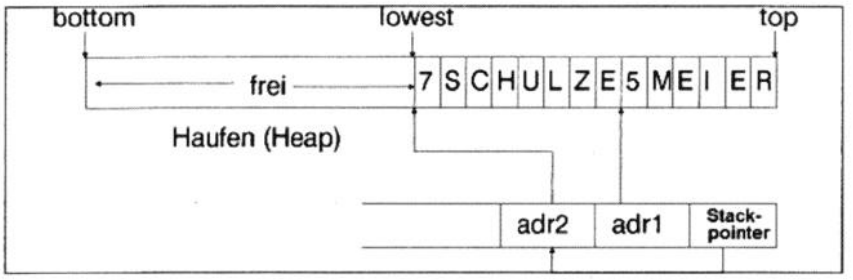


Bild 4

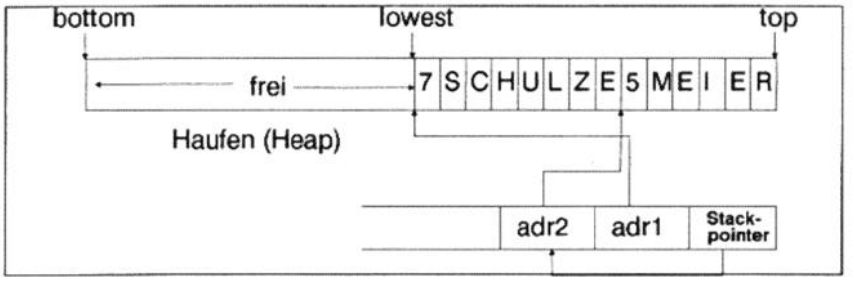


Bild 5

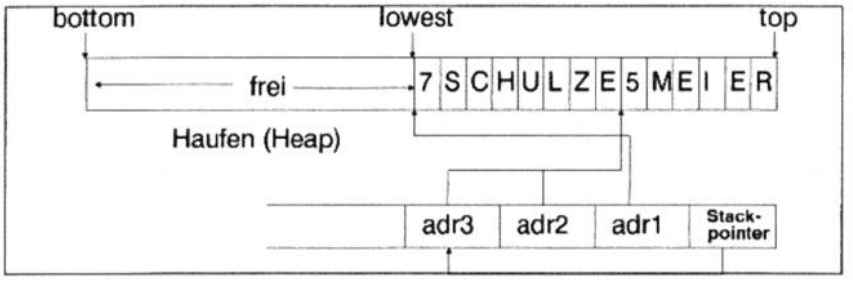


Bild 6

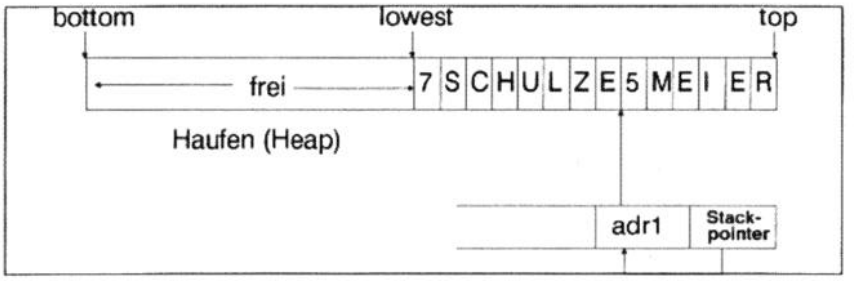


Bild 7

Wer genauer in den Code hineinschaut, wird entdecken, daß trotz des zusätzlichen Speicherzugriffs gegenüber der Lösung von Klaus vieles einfacher geworden ist. Zugriff auf Adressen ist eben einfacher als auf Strings!

Wer nun Lust bekommen hat, sich diesen Stringheap einzutippen, den möchte ich noch einmal ausdrücklich darauf hinweisen, daß ich das Wort *"join* verändert habe. Es fügt nun den zweiten String an den ersten, so daß

```
" Herr" " Meier" "join
```

nun auch

```
" Herr Meier"
```

ergibt und nicht

```
" Meier Herr"
```

Das aber ist wie vieles Geschmackssache....

Mit so einem Haufen kann man aber noch viel mehr anfangen. Von Uwe Kloss stammt der Vorschlag, das ganze FORTH-Wörterbuch als ein Heap anzulegen. Das hätte dann den Riesenvorteil, daß man einzelne Definitionen abändern und vergessen kann, ohne daß man immer alles neu laden muß. Was ist Ihre Meinung dazu?

Möget Ihr nie von FORTH genug haben!

Beschreibung der einzelnen Worte

Auf Screen 10 befinden sich im wesentlichen Stackworte u.a. die die Klaus auch verwendet hat und hoffentlich bald Standard werden.

Screen 11 legt den Haufen an. *Bottom* und *Top* zeigen auf Anfang und Ende des Haufens, und *Lowest* auf den untersten String, wie gehabt. Die Variable *"stack* hält die Adresse des Stringzeigerstacks. *"ptr* und *"top* arbeiten wie bei Klaus. Das Wort *?"stack* checkt den Zeigerstack auf Über- und Unterlauf. Schließlich dient das Wort *"heap* zum Anlegen des Haufens.

Hierbei muß man noch angeben, wieviele Strings man auf den Stack zu werfen gedenkt und wie groß der Gesamtspeicherraum sein soll. Also, wenn ich z.B. maximal 30 Strings auf dem Stack habe, und die - mit *"variable* - einen Gesamtspeicherraum von 1000 Bytes - optimistisch gerechnet - brauchen, dann definiere ich mir das mit 30 1000 "heap.

Screen 12 und 13 zeigen die "Garbage Collection". Es wird zuerst der am höchsten liegende String gesucht, indem einfach die Zeiger verglichen werden. *Move"* schiebt diesen String dann ans oberste Ende des Haufens. Beim nächsten Durchlauf wird der zweithöchste gesucht, usw. Wenn alle Strings nach oben geschoben worden sind, ist der Stringmüll automatisch verschwunden.

Screen 14 enthält die elementaren Worte um den Heap zu erweitern oder zu verkleinern. *?enough* testet, ob noch genug Platz ist.

"push legt einen durch »adresse« und »count« bezeichneten String auf den Heap und auf den Stack. *"pop* holt den obersten String runter und stellt »adresse« und »count« zur Verfügung. Diese sollten aber sofort - zumindest vor dem nächsten *"push* - weiterverarbeitet werden.

Die Screens 15 bis 21 enthalten Worte, die alle von Klaus schon beschrieben wurden. Achtung bei *"join*!!

Auf Screen 22 bis 23 habe ich noch ein paar Operatoren hinzugefügt, von denen ich mir einbilde, daß sie ganz nützlich sind. *"find* durchsucht s1 und s2 und hinterläßt die Position - oder eine Null, falls nicht gefunden - auf dem Stack.

"insert setzt den String s1 in s2 ab der Position n ein und hinterläßt das Ergebnis als s3 auf dem Stringstack.

"discard nimmt aus den String s1 n2 Zeichen ab der Stelle n1 heraus und legt den resultierenden String wieder ab.

"replace ist komplizierter. Es durchsucht s1 nach s2. Wenn der String gefunden wird, wird er durch s3 ersetzt. Falls er nicht gefunden wird, passiert nichts. Auf alle Fälle ist s4 das Ergebnis.

Screen # 10

```
 0) // Neuer Stringstack mit Heap
 1) // Stackoperationen und so weiter
 2) : under ( n1 n2 -- n2 n1 n2 ) swap over;
 3) : -rot  ( n1 n2 n3 -- n3 n1 n2 ) rot rot ;
 4) : u>    ( u1 u2 -- f ) swap u< ;
 5) : on    ( addr -- ) true swap ! ;
 6) : off   ( addr -- ) false swap ! ;
 7)
 8) : move  (from to cnt -- )
 9)   >r 2dup u< if r> cmove exit then r> cmove ;
10) : place (addr cnt to -- )
11)   over >r rot over 1+ r> move c! ;
12)
13) : <"    [compile] < ; immediate
14)
15) : ,"    ascii " word count here over 1+ allot place ;
```

Screen # 11

```
 0) // Stringheap anlegen & Errorchecking
 1) variable bottom variable top (Anfang & Ende Heap)
 2) variable lowest              (Niedrigster String)
 3) variable "stack              (Adresse vom Stringpointerstack)
 4) : "ptr ( -- addr )    "stack @ ;
 5) : "top ( -- addr )    "ptr @ ;
 6) : "clear ( -- )  (" wsn..sl -- ")
 7) "ptr dup ! ;
 8)
 9) : ?"stack ( -- )
10)   "ptr dup @ u< abort" Stringstack empty"
11)   "top "ptr 2+ @ u< abort" Stringstack full" ;
12)
13) : "heap   ( n len -- )
14)   here bottom ! allot here dup top ! lowest !
15)   2* here swap allot  here dup , "stack ! , ;
```

Screen # 12

```
 0) // Stringheap aufräumen: Grabage collection
 1) variable ?end variable "link 0 " link !
 2) : higher (maxptr -- greatestptr )
 3)   dup @  lowest @ u< not if drop exit then
 4)   over @ over @ u<  if swap then  drop ;
 5)
 6) : ready? (ptr -- ptr)
 7)   dup bottom 0 if ?end on then ;
 8)
 9) : stack-highest (ptr1 -- ptr2)
10)   "top "ptr 2- do i higher -2 +loop ;
11)
12) : var-highest   (ptr2 -- ptr3)
13)   "link begin @ ?dup while dup >r 2+ higher r> repeat ;
14)
15) : highest   bottom stack-highest var-highest ready? ;
```

Screen # 13

```
 0) // Stringheap aufräumen: Grabbage collection
 1) : same?     (ptr1 ptr2 -- ptr1 ptr2)
 2)   2dup = not if over @ over @ = if lowest @ over ! then then ;
 3)   correct-ptr's (ptr -- ptr)
 4)   "top "ptr 2- do i drop +loop
 5)   "link begin @ ?dup while 2+ same? 2- repeat
 6)   lowest @ swap ! ;
 7)
 8) : "place      (addr cnt --)
 9)   dup 1+ lowest @ swap - dup lowest ! place ;
10) : move"       (--)
11)   highest ?end @ not
12)   if dup count "place correct-ptr's  else drop then ;
13) : collect     (--)
14)   ?end off   top @ lowest !
15)   begin move" ?end @ until ;
```

Stringstack mit Heap

Screen # 14

```
 0) // Stringheapfunktionen
 1)
 2) : ?enough @   (n -- flag)
 3)   lowest @ bottom @ - > not ;
 4)
 5) : ("push)     (addr cnt -)  (" -- s 2)
 6)   "place -2 "ptr +! ?"stack lowest @ "top ! ;
 7)
 8) : "push       (addr cnt --) (" -- s ")
 9)   dup 1+ ?enough not
10)   if collect dup 1+ ?enough not abort" String Memory full"
11)   then ("push) ;
12)
13) : "pop        (-- addr cnt) (" s -- ")
14)   "top @ count 2 "ptr +! ?"stack ;
15)
```

Screen # 15

```
 0) // Stringstackoperationen
 1)
 2) : "skip    (addr -- addr2)
 3)   count + ;
 4) : th"      (n -- ptradr)     ("sn...s1 -- sn...s1")
 5)   "top swap 2* + ;
 6) : "depth   (-- n)            ("sn...s1 -- sn...s1")
 7)   "ptr "top - 2/ ;
 8) : "drop    (--)              ("s -- ")
 9)   "pop 2drop ;
10) : "pick    (n --)            ("sn...s1 -- sn...s1 sn")
11)   th" @  -2 "ptr +!  "top ! ;
12) : "roll    (n --)            ("sn...s1 -- sn...s1 sn")
13)   "top over 2* + @ >r "top dup 2+ rot 2* cmove>
14)   r> "top ! ;
15)
```

Screen # 16

```
 0) // Stringstackoperationen
 1)
 2) : "dup    (" s -- s s ")              0 "pick ;
 3) : "over   (" s1 s2 -- s1 s2 s1 ")     1 "pick ;
 4) : "swap   (" s1 s2 -- s2 s1 ")        1 "roll ;
 5) : "rot    (" s1 s2 s3 -- s2 s3 s1 ") 2 "roll;
 6)
 7) : (")     (" -- s ")
 8)   r> dup "skip >r count "push ;
 9)
10) : "       (" -- s ")
11)   State @ IF  compile (") ,"
12)           ELSE Ascii " Word count "push
13)           THEN ; IMMEDIATE
14)
15)
```

Screen # 14

```
 0) // Stringoperationen
 1)
 2) : ""       (--)      (" -- 0s ")   0 0 "push ;
 3) : >string  (c --)    ("-- s ")
 4)   0 1 "push   0 th" @ 1+ c! ;
 5)
 6) : "join    (--)      (" s1 s2 -- s3 ")
 7)   "swap  "pop  pad place   "pop  pad c@ 2dup + pad c!
 8)   pad + 1+ swap cmove   pad count  "push ;
 9)
10) : "append  (c --)    (" s1 -- s2 ")
11)   >string "join ;
12)
13) : "split (n --)      (" s1 -- s2 s3 ")
14)   "pop pad place  pad 1+  over pad c@ min "push
15)   pad c@ over - 0 max swap pad + 1+ swap  "push ;
```

Screen # 18

```
 0) // Stringoperationen und -vergleiche
 1)
 2) : "extract (n1 n2 --)    (" s1 -- s2 ")
 3)   "pop rot min rot over min  rot over +
 4)    -rot - "push ;
 5)
 6) : compare (addr1 cnt1 addr2 cnt2 -- n)
 7)   rot 2dup -> min
 8)   BEGIN ?dup WHILE 1- >r count rot count rot - ?dup
 9)         IF r> r> 2drop -rot 2drop negate EXIT THEN
10)   swap r> REPEAT 2drop r> ;
11)
12) ; "compare (-- n)   (" s1 s2 -- ")
13)   "pop "pop compare negate ;
14)
15)
```

Screen # 19

```
 0) // Stringvergleiche
 1)
 2) : "=   (-- f)    (" s1 s2 -- ")
 3)   "compare 0= ;
 4)
 5) : "<   (-- f)    (" s1 s2 -- "9
 6)   "compare 0< ;
 7)
 8) : ">   (-- f)    (" s1 s2 -- ")
 9)   "compare 0> ;
10)
11)
12)
13)
14)
15)
```

Screen # 20

```
 0) // String /o
 1)
 2) : ".   (--)      (" s -- ")
 3)   "pop  type ;
 4)
 5) : ".s  (--)      ("sn...s1 -- sn...s1 ")
 6)   "ptr "top 2dup > if
 7)   do cr i @ count type 2 +loop
 8)   else 2drop cr ." Stringstack empty " then ;
 9)
10) : "expect (n --)
11)   dup "push 0 th" @ count expect "pop drop span @ "push ;
12)
13)
14)
15)
```

Screen # 21

```
 0) // Stringvariable etc.
 1) : "@     (lnk --)    (" -- s ")
 2)   2+ @ -2 "ptr +! "top ! ;
 3)
 4) : "!     (lnk --)    (" s -- ")
 5)   2+ "top @ swap ! "drop ;
 6)
 7) : "variable (n --)  (" -- ")
 8)   create here "link @ , "link ! 0 , ;
 9)   <does> -- addr )
10)
11) : "constant (--)    (" s -- ")
12)   create "pop here over 1+ allot  place
13)   does> 2- "@   (" -- s ") ;
14)
15)
```

Screen # 22

```
 0) // Stringheapoperatoren
 1)
 2) : ("find)  ("addr1 addr2 -- f)
 3)   TRUE -rot
 4)   swap count ?dup IF 0 DO over I + c@ over I + c@
 5)   = not IF 2drop not 0. LEAVE THEN LOOP THEN 2drop ;
 6)
 7) : "find (-- 0/pos)  (" s1 s2 --)
 8)   FALSE  0 th" @ "drop "pop
 9)   ?dup IF 0 DO 2dup I + ("find)
10)   IF 2drop drop I 1+ 0. LEAVE THEN
11)   LOOP THEN 2drop
12)
13)
14)
15)
```

Screen # 23

```
 0) // Stringheapoperatoren
 1)
 2) : "insert  (n --)  (" s1 s2 -- ")
 3)   "swap  "split "rot "swap "join "join ;
 4)
 5) : "discard  (n1 n2 --) (" s1 -- s2 ")
 6)   over 0 swap "dup "extract "swap + 256 "extract "join ;
 7)
 8) : "replace (" s1 s2 s2 -- s4 ")
 9)   "rot "rot "over "over "find ?dup
10)   IF 1- dup "pop swap drop "discard
11)      "swap "insert
12)   ELSE "drop "swap "drop
13)   THEN ;
14)
15)
```

FORTH-Gesellschaft intern

Neuigkeiten und Bekanntes der FORTH-Gesellschaft e.V.

Entwicklung der FORTH-Gesellschaft e.V.: Die Mitgliederzahl hat sich sehr erfreulich entwickelt, sie konnte in Jahresfrist verdoppelt werden. Zwei erfolgreiche Tagungen haben stattgefunden: Die FORTH-Tagung '89 an der RWTH Aachen und die euroFORML '89 in Neunkirchen am Brand bei Nürnberg.

Die nächsten Veranstaltungen der FORTH-Gesellschaft e.V. sind: Die FORTH-Tagung '90 mit Mitgliederversammlung an der Johann-Wolfgang-von-Goethe-Universität in Frankfurt (20. bis 22. April) und der Kongreß Echtzeit '90 in Sindelfingen (19. bis 21. Juni). Die Echtzeit '90 wird unter Mitwirkung des PEARL e.V. organisiert und in Kooperation mit der Association for Computing Machinery (ACM) und der Gesellschaft für Informatik (Fachausschuß Informatik in der Echtzeit-Datenverarbeitung) veranstaltet. Studenten und Schüler haben die Möglichkeit, in begrenzter Anzahl, an diesem Kongreß für eine Gebühr von DM 90,00 teilzunehmen.

Bei der euroFORML 1989 wurde - wie schon lange erwartet - das German FIG-Chapter der FORTH Interest Group gegründet. Wir hoffen, daß dies neue und interessante Impulse für die FG bringt.

Die FORTH-Mailbox in München erfreut sich eines regen Zuspruchs und verfügt über Informationen, die für alle FORTH-Fans von großem Interesse sind; wir sehen es an dem breiten Echo, das die Box findet

Die FORTH-Gesellschaft hat von der Fa. Brühl EE GmbH zwei Akustikkoppler gespendet bekommen. Diese Akkustikkoppler können vom FORTH-Büro ausgeliehen werden, um erste Gehversuche in der DFÜ- Landschaft zu machen.

In den in Tabelle 1 aufgeführten Mailboxen kann man viel in Sachen FORTH erfahren. Auf Initiative unseres Mitglieds Friederich Prinz gibt es auch in einigen Boxen des Zerberus-Netzes einen FORTH-Anfängerkurs.

Administration

Mitgliedsbeiträge und Zahlungsmodus:

Bitte entrichten Sie Ihren Mitgliedsbeitrag bis Februar 1990 oder machen Sie Gebrauch von der Einzugsermächtigung. Als Erinnerung liegt dieser VIERTEN DIMENSION ein Überweisungsformular für 1990 und ein Antrag auf »Mitgliedschaft in der FG« bei. Auf der Mitgliederversammlung 1989 in Aachen wurden die Jahresbeiträge neu festgesetzt.

Beachten Sie bitte, daß die ermäßigten Beiträge nur gegen Nachweis gewährt werden. Bei Auslandsadressen ist wegen der erhöhten Versandkosten keine Ermäßigung möglich. Der Mitgliedsbeitrag gilt immer für das laufende Kalenderjahr. In diesem Beitrag ist der kostenlose Bezug der VIERTEN DIMENSION enthalten.

Mitgliedsbeiträge	
Schüler, Studenten, Rentner u. Arbeitslose	DM 48,00
Ordentliche Mitglieder, Auslandsadresse	DM 80,00
Fördernde Mitglieder, Firmen u. Institutionen	DM 160,00

Unsere Mitglieder mit Auslandsadresse bitten wir, Ihren Mitgliedsbeitrag in DM mittels Auslandspostanweisung oder durch Postgiroüberweisung an uns zu entrichten. Wurde der FG eine Einzugsermächtigung erteilt, werden wir den neuen Mitgliedsbeitrag Mitte Januar einziehen.

Spendenbescheinigungen:

Die FORTH-Gesellschaft e.V. ist laut Bescheid des Finanzamts für Körperschaften München 2 vom 15. März 1989 unter der Steuernummer 843/19531 als gemeinnützig anerkannt und nach Paragraph 5, Abs. 1, Nr. 9 des Körperschaftssteuergesetzes von der Körperschaftssteuer befreit. Bei

Name	Telefon	Betreiber	Mailboxtyp
MFB	089-7259625	Christoph Krinninger D-8000 München	PC-Board
FORTH-Box	07667-556	Klaus Flesch D-7814 Breisach	PC-Board
MHB	02841-57325	Hans Georg Lanzerath D-4130 Moers 1	Zerberus
IUS	0203-701806	Markus Warg D-4100 Duisburg	Zerberus
AME	09131-992998	Klaus Schmitz D-8520 Erlangen	Zerberus
BIONIC BOX	0521-171188	Rena Tangens & Padeluun D-4000 Bielefeld	Zerberus
INFINET	089-656632	R. Ruediger D-8000 München 90	Zerberus
FLM	00352-488088	Peter Burcek L-2430 Luxemburg (tägl. 6:00-23:00)	Zerberus
FLASH	02855-81258	Reinhard Breuer D-4223 Voerde/Spellen	Fido
UTS	02853-4597	Günther Henningsmeyer D-4235 Schermbeck	Fido
SoftProject	0208-55173	Rolf Rother D-4330 Mülheim/Ruhr	StandAloneBox
SAM*JEDI	0033-36431515 (1200/75,E,7,1)	M. Petremann F-75012 Paris	TRANSPACnetwork

Tabelle 1: Mailboxen

Spenden bis DM 100.00 genügt zur Geltendmachung der Zahlungsbeleg eines Kreditinstituts oder der Post als Spendennachweis. Wir stellen aber, bei Zuwendungen ab 30.00 DM, jeweils zum Jahresende Spendenbescheinigungen aus.

Lokale Gruppen und Fachgruppen:

Seit November 1989 gibt es in Stuttgart eine neue FORTH-Gruppe; Ansprechpartner sind Herr Wolf-Helge Neumann und Herr Ulf Katzenmaier.

Mitgliederliste:

Dieser Ausgabe liegt die aktuelle Mitgliederliste bei, sie ist nur für den privaten Gebrauch bestimmt. Das FORTH-Büro ist gerne bereit Berichtigungen und neue Interessengebiete der Mitglieder in die Liste aufzunehmen.

Für weitere Fragen stehen wir Ihnen gerne zur Verfügung. Telefonisch erreichen Sie uns unter 089-317 37 84.

Allen Mitgliedern und Lesern wünschen wir ein Frohes Weihnachtsfest und ein erfolgreiches, gesundes 1990.

Ulrike Schnitter

-FORTH-Büro-

Einladung

zur ordentlichen Mitgliederversammlung

der FORTH-Gesellschaft e.V.

am Sonntag, den 22. April 1990 ab 10:00 im Institut für Angewandte Physik, Robert-Mayer-Str.2-4 an der Johann-Wolfgang-von-Goethe-Universität 6000 Frankfurt/Main.

Ab 20.April findet das jährliche FORTH-Treffen statt. Auch in diesem Jahr gibt es wieder interessante Vorträge über FORTH-Anwendungen in Wissenschaft und Industrie.

Tagesordnung der Mitgliederversammlung

1. Rechenschaftsbericht des Direktoriums
2. Kassenbericht und Jahresbilanz 1989
3. Aussprache und Entlastung des Direktoriums
4. Wahl des Direktoriums
5. Mitgliederversammlung 1991
6. Verschiedenes

Ergänzungen zu den Tagesordnungspunkten können schriftlich bis März 1990 beim FORTH-Büro beantragt werden.

Ein Anmeldeformular für die FORTH-Tagung '90 finden Sie in dieser 'Vierten Dimension'.

Externer Editor auf der Basis eines allgemeinen Overlay-Treibers für das IBM-volksFORTH

von Frank Raschke, Erlenring 2,
6368 Bad Vilbel 3,
Tel.: 06101/47543 od. 42077
Mit Anregungen von: A. Soeder, D. Heid

Quelltext Service

Stichworte

- IBM-volksFORTH,
- externer Editor,
- Overlay-Programme

Bei meiner Arbeit mit FORTH hatte ich schon oft das Bedürfnis, gewisse Teile meiner Anwendungen aus dem System herauszulegen. Bisher bot zwar das Wort CALL eine bescheidene Möglichkeit dies zu realisieren, jedoch genügte das meist nicht, zumal bei häufigem Aufruf jedesmal das gesamte FORTH von Disk geholt werden mußte. Dies wäre spätestens bei der Realisierung eines externen Editors ein unangenehmer Effekt gewesen. Somit war die Idee, einen möglichst unkomplizierten Overlay-Treiber zu bauen, geboren.

Als ein Overlay-FORTH bezeichne ich hier ein komplettes FORTH, das in den Speicher geladen wird. Es kann dort direkt aus einem anderen FORTH ohne Diskzugriff aufgerufen werden (resident). Das Overlay hat ein eigenes Dictionary und intern einen eigenen Stack. Jedoch wird dieser über den OVL-Handler bei jedem Aufruf auf den Stand des Stacks des übergeordneten FORTH gebracht.

Verkettete Aufrufe über mehrere Ebenen sind möglich, jedoch könnte ein erneuter (bzw. rekursiver) Aufruf nicht korrekt abgearbeitet werden, da der statische Puffer des Handlers überschrieben würde.

Weitergehende Forderungen waren:

- der Kernel und der alte Editor sollten unverändert bleiben
- mehrere Overlays (FORTH-Systeme) müssen resident im Speicher liegen können
- der Zeitaufwand für einen Aufruf sollte akzeptabel sein
- die Möglichkeit der Übergabe einer kompletten Kommandozeile sollte erhalten bleiben
- die Handhabung sollte so einfach wie möglich sein
- ein Overlay-FORTH sollte prinzipiell auch bei einem Aufruf über die DOS-Funktion EXEC betreibbar sein

Auf eine genaue Erklärung der Worte im einzelnen möchte ich an dieser Stelle verzichten. Dazu sollte der Text in den Screen-Files und das Beispiel des EXDITORs dienen.

Das File EXDITOR.SYS liefert dem Anwender keine Words, da er hier überhaupt nicht in das System (Interpreterschleife: QUIT) gelangt, sondern nur den Editor mit seinen Funktionen bedienen kann (Editorschleife: FULL-QUIT). Bei Beendigung dieses Vorganges ist er dann

Worte des aufrufendes FORTH (OVERLAY.SCR):

Wort	Stackrelation	Erklärung
OVLFILE:	<name.ext> (--)	definiert ein word im Dictionary, dessen Name mit dem des aufzurufenden OVLs identisch ist (die Extension .COM wird automatisch. erzeugt). Bei Aufruf wird das entsprechende File als aktuelles OVL gekennzeichnet. Befindet es sich noch nicht im Speicher, so wird es geladen.
CALLOVL	<kommdoz.>	ruft das aktuelle OVL auf kopiert den Stack nach dort und bringt die Kommandozeile dort zur Ausführung.
"CALLOVL	(string --)	identisch mit CALLOVL jedoch wird die Kommandozeile als String übergeben.
KILLOVL	(--)	löscht das aktuelle OVL aus dem Speicher, jedoch nicht das zugehörige Word aus dem Dictionary.
.OVLNAME	(--)	gibt den Namen des aktuellen OVL aus.
>PREVOVL	(--)	rettet das aktuelle OVL (wenn z.B. ein anderes temporär ausgeführt werden soll).
PREVOVL>	(--)	setzt ein zuvor gerettetes OVL wieder als aktuell und lädt es falls es zwischenzeitlich entfernt wurde.

Tabelle 1

auch sofort wieder im aufrufenden FORTH, da nicht nach QUIT zurückgekehrt wird, sondern gleich OVLRET zur Ausführung gelangt.

Die Installation:

1. Installation des externen Editors

Das Exditor-Overlay wird einfach von DOS aus mit der Sequenz KERNEL INCLUDE EXDITOR.SYS installiert. Damit wird das File EXDITOR.COM erzeugt, wobei der Standard-Editor automatisch zugeladen wird. Etwaige Veränderungen bleiben hier also weiterhin aktiv. Dann wird im System-Loadscreen (z.B. VOLKS4TH.SYS) die Sequenz INCLUDE EDITOR.SCR durch INCLUDE EXDITOR.SCR ersetzt. Jetzt muß nur noch das Arbeitssystem von DOS aus neu hochgeladen werden (mit z.B. KERNEL INCLUDE VOLKS4TH.SYS).

2. Installation eines Overlays

In das System, das später das Overlay-Programm sein soll, wird einfach OVLFILE.SCR geladen und es dann nach Fertigstellung mit SAVESYSTEM <name>.COM abgelegt. Das aufrufende FORTH wird lediglich mit dem Screen OVERLAY.SCR erweitert. Mit OVLFILE: <name> können beliebig viele OVLs im Dictionary definiert werden. Erst bei Aufruf ihres Namens sind sie dann tatsächlich im Speicher verfügbar (über CALLOVL).

Das Overlay-FORTH kann auch mit BYE verlassen werden, welches dann auch die Ausführung des aufrufenden FORTH beendet. Dies ist sogar zwingend erforderlich, wenn es über DOS (EXEC) aufgerufen wurde (z.B. auch über CALL), d.h. wenn es als ganz normales FORTH aktiviert wurde.

Worte des aufgerufenen FORTH (OVLFILE.SCR):

Wort	Stackrelation	Erklärung
OVLRET	(--)	kopiert den Stack ins aufrufende FORTH und setzt die Ausführung dort fort.
RETURN SEG	(-- seg)	übergibt das Segment des aufrufenden FORTH.

Tabelle 2

Externer Editor (EXDITOR.SCR):

modifizierte Worte:

Wort	Stackrelation	Erklärung
USE & FROM		setzen jetzt screen und from-screen immer auf 0. Dies soll ein Entstehen von Wild-Screens verhindern.
L & EDIT		ermöglichen nur ein Editieren innerhalb von capacity (Grund s.o.).

neue Worte:

Wort	Stackrelation	Erklärung
SCR'	(-- addr)	identisch mit SCR jedoch. für das from-file (vgl. EDITOR.SCR).
r#'	(-- addr)	identisch mit r# jedoch. für das from-file (vgl. EDITOR.SCR).
ASSOCIATE	(--)	führt ein FSWAP aus, jedoch. werden auch der screen und r# vertauscht (vgl. EDITOR.SCR). dies soll wiederum das Entstehen von Wild-Screens verhindern.

Tabelle 3

SCR# 0

```
\                                                  lrd lrd 10sep89

                        OVERLAY.SCR

                    Verwaltung mehrerer
                       Overlay-Files
                  in Form von forth.COM-files

    Diese words sind für das aufrufende Programm
    bestimmt. Das aufgerufene Overlay-Programm muß
    dann mit den in OVLFILES.SCR definierten words
    ausgerüstet sein und mit OVLRET beendet werden.

  Frank Raschke Erlenring 2 6368 Bad Vilbel 06101/47543
```

SCR# 1

```
\   Loadscreen                                          lrd lr+ 19sep89

 3 | Constant  maxovl                    \ max. Anz. der ovl-Files
   | Variable  ovltab  here maxovl 2* \ Tab. für ovl. pfa's
                      dup allot erase
                      ovltab on
   | Variable  aktovl  aktovl on        \ akt. ovl. pfa
   | Variable  prevovl prevovl off     \ pfa d. geretteten ovl
   |         : aktseg  aktovl @ 2+ ;   \ addr hält seg des aktovl

   |     Defer ovlerr?  ' noop is ovlerr? \ Fehlerbehandlung

  Dos also Forth definitions 2 Capacity 1- thru Onlyforth

  cr .( Overlay-Treiber installiert )
```

Externer Editor mit Overlay-Treiber

SCR# 2

```
\ killovl allocovl                                  lrd lrd 12sep89

  : killovl ( -- )  \ löscht aktuelles ovl
       aktovl @ 1 ovlerr? ovltab 2+ maxovl 2* bounds 2dup
       DO I @ aktovl @ = IF 0 I ! leave THEN 2 +LOOP 2- swap 2-
       aktseg @ ?dup 0= not IF lfree aktseg ! THEN
       DO I @ ?dup 0= not IF aktovl ! leave THEN -2 +LOOP ;

 | : allocovl ( -- ) \ res. Sp.-Seg. für ovl & stored's in pfa
       $1041  lallocate 3 ovlerr? aktseg ! ;

\\ killovl  löscht akt. ovl-file aus RAM & Liste wenn vorhanden
            und setzt previous ovl-file als akt. wenn vorhanden.
   allocovl allok. 64k für 4th + 1k für INT-Tab. ( muss schon
            hier sein da sonst kein save-/restorevideo mögl.)
```

SCR# 3

```
\   setovl                                          lrd lrd 12sep89


 | : setovl ( pfa -- flag )  \ setzt ovl in ovltab & aktovl
       ovltab 2+ maxovl 2* bounds rot true 2over
       DO over I @ = xor 2 +LOOP          \ n. nicht geladen?
       IF false 2swap
            DO I @ 0= IF drop I leave THEN 2 +LOOP \ Tab. voll?
            2 ovlerr? stash ! true swap
       ELSE nip nip false swap THEN aktovl ! ;


\\ setzt ovl-file in die Liste der aktiven ovl-files falls noch
   nicht & Platz vorhanden. Dann wird es um aktuellen ovl-file
   erklärt.
```

SCR# 4

```
\  (ovlerr?                                         lrd lr+ 19sep89


  | : (ovlerr?  ( f, err# -- )
        1 case?
          IF not 0= abort" no OVL-FILE loaded" exit THEN
        2 case?
          IF dup 0= abort" too much OVL-FILES" exit THEN
        3 case?
          IF IF killovl  true abort" RAM full" THEN THEN ;


     ' (ovlerr? is ovlerr?
```

SCR# 5

```
\  ovlname> ovlmsg .ovlname                         lrd lrd 12sep89

 | : ovlname> ( -- )  aktovl @ 1 ovlerr? aktovl @
       body> >name count $1F and filename place ;

 | : msg> ( string -- ) aktseg @ ds@ over       \ übergebe S,S:
       dta 4- 1! sp@ 10 - swap dta 2- 1!        \ im PSP
       " ovlboot " count dta place count        \
       dta attach ds@ dta aktseg                \ code f. ovlay
       @ dta dup c@ 1+ lmove ;                  \ nach ovl-dta

   : .ovlname ovlname> filename count type ;

\\ ovlname> bringt den aktuellen ovl-namen nach filename
   ovlmsg>  bringt den im ovl-file auszuf. Code in dortige DTA
   .ovlname zeigt das aktive ovl-file an
```

SCR# 6

```
\  ((callovl  (callovl                              lrd lr+ 19sep89

  Label ((callovl
       A pop C: push A push           \ second = Returnsegment
       aktovl #) W mov 2 W d) C mov   \ C = Zielsegment
       C D: mov $80 # D mov $1A # A+  \ verlege DTA ins Zielseg.
       mov $21 int C E: mov C S: mov  \ und setze Seg-Register
       C: seg W ) far jmp  end-code   \ auf Zielseg., ovl-aufruf

  | Code (callovl  ( s0 -- )
       W push R push U push I push    \ rette Register
       D push ((callovl # call        \ TOS nach D
       D pop Next end-code

\\ das zus. CALL ist erforderl. um IP auf den hiesigen Stack
   zu sichern, dann erst wird S, S: nach Zielseg. verbogen.
```

SCR# 7

```
\  "callovl  callovl                                lrd lr+ 19sep89

  : "callovl ( string -- )
       aktovl @ 1 ovlerr? flush msg> s0 @
       (callovl ;

  : callovl ( --`| <msg> ) aktovl @ 1 ovlerr?
       source >in @ /string swap 1- under
       stash c! stash >r >r flush msg> r> r> 1+
       blank s0 @ (callovl  ;

\\ Aufruf des aktuellen ovl-files ( ohne save/restvid. ZEIT! )
   CALLOVL  : interaktiv  callovl name.ext word1 word2 ....
   "CALLOVL : inline      : "string " 4 3 * ovlret" ;
                          : test "string "callovl . ; => 7 ok
   "string bzw. wordn ist code der im ovl-file ausgeführt wird.
```

SCR# 8

```
\  ~ldovl                                           lrd lrd 12sep89


| Code ~ldovl ( tab, asciz -- f* )
     R C mov R pop $4B03 # A mov $21 int  \ lade als overlay
     C R mov CS ?[ A D mov ][ D D xor ]?  \ wenn fehler => err#
   Next end-code                         \ sonst ff


\\ tab muß folgendes Format haben:
   Byte 0-1 :  Zielsegment ( Achtung ! Es wird kein PSP angel.)
   Byte 2-3 :  Relokationsfaktor ( bei .COM-file immer 0 )
   Läd file als overlay nach Zielseg:0 dessen Name als 0-termin.
   String ab Addr asciz liegt.
```

SCR# 9

```
\  loadovl                                          lrd lrd 12sep89

 | : loadovl ( -- )
       ovlname> " .COM" count   filename count      \ wandle in
       Ascii . scan drop swap 2dup 1+ erase  move   \ COM-file
       $10 aktseg under +! filename 1+ ~ldovl       \ lade file
       -$10 aktseg +! ?diskerror                    \ nach $100
       ds@ 0 aktseg @ 0 $7F lmove                   \ kop. PSP
       aktseg @ dup $1041 + swap 2 1! ;             \ seg Prgend

\\ läd aktuelles ovl-file nach $100 im aktuellen ovl-seg,
   so daß das aufgerufene ovl-file dann quasi ein COM-file
   darstellt ( mit PSP ).
```

SCR# 10

```
\  Ovlfile:                                         lrd lrd 12sep89


   : Ovlfile:  Create $100 , 0 , 0 , ( -- )
               Does>  setovl          ( -- )
                      IF allocovl loadovl THEN ;


\\ Comp. : Erzeugt word name.ext das folgende Tab. enthält:
           word 1 : Einsprung-Offs. für far jmp
           word 2 : Seg. falls geladen für far jmp und ~ldovl
           word 3 : Relok.-Faktor (=0) für ~ldovl
   Runt. : falls name.ext n.nicht geladen wird es falls mögl.
           geladen und als aktuelles ovl-file in die Liste der
           aktiven ovl-files eingetragen.
```

SCR# 11

```
\

 : >prevovl aktovl @ dup not IF body> prevovl ! exit THEN drop ;

 : prevovl>  prevovl dup @ ?dup IF execute THEN off ;


\\ retten und restoren aktuelles Overlay für temporären
   Aufruf eines anderen.
```

Externer Editor mit Overlay-Treiber

SCR# 0

```
\                                                        lrd 14sep89


                        OVLFILE.SCR
                  stattet ein volks4th-file
                    als overlay-file aus.


              Wird es von einem Programm als Overlay
             aufgerufen, führt OVLRET zum Rücksprung,
                bei Aufruf über DOS das alte BYE.


      Frank Raschke Erlenring 2 6368 Bad Vilbel 06101/47543
```

SCR# 1

```
\                                                        rd 17sep89



 2 capacity 1-  thru

 cr .( System als Overlay modifiziert )
```

SCR# 2

```
\                                                        r+ 19sep89

  Label buffer 14 allot  : return_seg [ buffer 2+ ] Literal @ ;

  Code ovlboot ( W, R, U, I, s0, C:, IP -- n[0], ... n[m] )
     I A mov                                  \ rette I
     dta 2- #) I mov   dta 4- #) D: mov       \ ret_seg & ret_SP
     buffer # W mov 7 # C mov rep word movs   \ rette 7 wds in buf
     C: seg buffer 4 + #) C mov  I C sub      \ C=ret_depth
     0= not ?[ C S sub C shr S dec S dec      \ 0<> => kopiere d.
     S W mov rep word movs D pop ]?           \ Stack, TOS nach D
     C: I mov I D: mov A I mov                \ setze D: zurück
  Next end-code

\\ verlegt dta ins akt. seg. und kopiert den Stack.
```

SCR# 3

```
\                                                        rd 17sep89

  Label (#segs  ( -- )   Assembler
     cli   A A xor   A E: mov
     C: seg ' limit >body #) R mov   R R or  0= not
     ?[  4 # C- mov   R C* shr   R inc   ret  ]?
     $1000 # R mov C: D mov   D R add   R D: mov
     I I xor W W xor $200 # C mov  rep movs  sti ret
  end-code

\\ Dies ist aus dem Kernel
   Eine Symbiose aus dem ersten Teil von (bye und dem
   Label #segs. Es wird die Interupttabelle aus der
   segment-externen Sicherungskopie ( 1k auf limit folgend )
   zurückgesetzt.
```

SCR# 4

```
\                                                        lrd 12sep89

| Code (ovlret ( C:, IP, C:, depth -- )
     D push (#segs # call                  \ restore ints
     C pop D: pop                          \ C=depth*2 D:(mast.)
     $80 # D mov $1A # A+ mov $21 int      \ move dta
     D: A mov C: I mov I D: mov A E: mov   \ E:=D:(mast.) D:=C:
     S I mov C: seg buffer 4 + #) W mov    \ SI=SP get s0(mast.)
     C W sub W A mov C shr  rep word movs  \ ber. SP(mast.) kop.
     A S mov C: seg buffer 6 + #) I mov    \ restore SP,I
     C: seg buffer 8 + #) U mov C: seg     \ restore U,R,W
     buffer 10 + #) R mov C: seg buffer    \
     12 + #) W mov E: A mov A S: mov A D:  \ restore SS
     mov lret                              \ restore DS return
   end-code
```

SCR# 5

```
\                                                        rd 17sep89


| : buffer> ( -- C:, IP, C:, depth )
       buffer 2@ over depth 1- 2*  ;  \ hole Puffer auf stack

  : ovlret  ( -- )
       flush page buffer>  (ovlret ;  \ Rücksprung ins Hauptprg.
```

SCR# 6

```
\\                                                       r+ 19sep89
  : .ovlsp      ( n -- )  ."  OVL s" depth   swap 1+ - 2 .r ;

| : fstat  ( n -- )   .base .ovlsp
     .space .scr .dr file? 2 spaces order ;

| area: statusline  statusline invers terminal

  : (.nstatus  (at? area @ output @ over 7 + c@ statusline
               display area @ 7 + c!  c/col 1- dup window invers
               status @ IF page 4 fstat ELSE normal page THEN
               output ! area ! (at ;

  ' (.nstatus Is .status  status on

\\ neue Statuszeile: OVL wird beim Overlay angezeigt
```

SCR# 7

Lokale FORTH-Gruppen, die sich regelmäßig treffen:

1000 Berlin Claus Vogt, Tel.: 030/2168938. Treffen am letzten Donnerstag des Monats um 19.30 Uhr in der Technischen Universität Berlin, Mathematikgebäude, 6.Stock im Raum MA 621

2000 Hamburg 13 Karsten Roederer, Tel. 040/4104446, tagsüber 412 329 84, Treffen im Geomatikum Raum 1438 (14.Stock), Bundesstr. 55, am 28. Juni, 27. September, 25. Oktober jeweils um 19.30 Uhr

4130 Moers 1 Friederich Prinz, näheres Tel: 02841/583 98

Rhein-Ruhr Jörg Plewe, Tel: 0208/423514, Treffen nach Absprache. Der nächste Termin kann bei Jörg Plewe erreichbar unter obiger Telefonnummer erfragt werden.

6100 Darmstadt Andreas Soeder, Tel. 06257/2744. Treffen an der VHS an einem Mittwoch in der Mitte des Monats (Termine: 13.9, 11.10, 15.11 13.12 im alten Pädagog, Raum 3-1, 3.Stock).

6800 Mannheim Lokale Gruppe Rhein-Neckar, Thomas Prinz, Tel.: 06271/2830. Treffen jeden ersten Mittwoch im Monat im Vereinslokal des Segelflugvereins Mannheim e.V. Flugplatz, Mannheim-Neuostheim.

7000 Stuttgart Lokale Gruppe Stuttgart, Wolf-Helge Neumann Tel.: 0711/882638 und Ulf Katzenmaier, Tel.:0711/268293.

8000 München Heinz Schnitter, Tel. 089/3103385 oder Christoph Krinninger 089/7259382. Treffen jeden 4. Mittwoch im Monat 19 Uhr 30 im Vereinsraum 2 im Bürgerhaus Unterschleißheim am Rathausplatz (S-Bahnhaltepunkt S1 Unterschleißheim).

FORTH-Fachgruppen:

8000 München RTX 2000 Gruppe, Koordinator Max Diez, Treff- und Zeitpunkt wie oben bei der lokalen Münchner Gruppe.

6800 Mannheim FIS (FORTH Integriertes System) - Datenbank, Textverarbeitung, Kalkulation, Postadresse: Dr. med. Elemer Teshmar, Danziger Baumgang 97, 6800 Mannheim 31

Es möchten in ihrer Region eine Gruppe gründen:

3300 Braunschweig Martin Holzapfel, Bassestr.17.

8500 Nürnberg 20 Thomas G. Bauer, Fichtestr. 31, Tel. 0911/538321.

5000 Köln 60 Michael Heycke, Boltensternstr.

4830 Gütersloh 1 Ludwig Röver, Holzheide 145A

4900 Herford Andreas Findewirth, Im Großen Vorwerk 48, Tel.: 05221/23504

Eine Fachgruppe will gründen:

7000 Stuttgart 80 Grafik/Arithmetik, Jörg Tomes, Anweilerweg 56, Tel. 0711/7802293.

8000 München 70 Btx u. FORTH, Christian Schwarz, Lindenschmitstr.30, 8000 München 70

Hier kann man um Rat fragen:

02103/556 09 Jörg Staben, Dienstag und Freitag, 20.00 - 22.00 Uhr

06187/91503 Frank Stüss

02845/28951 Karl Schroer

Ansprechpartner zu bestimmten Interessengebieten:

volksFORTH/ultraFORTH:	Klaus Kohl, Tel.: 08233/30524 Bernd Pennemann, Tel. 0228/640979 und Klaus Schleisiek-Kern, Tel. 040/2202539.
32-Bit Systeme:	Robert Jones, Tel. 02434/4579
Künstliche Intelligenz:	Ulrich Hoffmann, Tel. 0431/678850
NC4000 Novix Chip:	Klaus Schleisiek, Tel. 040/6449412
Realtime & Petri-Netze:	Wigand Gawenda, Tel. 040/446941
Gleitkomma-Arithmetik:	Andreas Döring, Tel. 02631/52786
32FORTH	Rainer Aumiller, Tel. 089/6708355
PostScript/FORTHscript	Christoph Krinninger, Tel: 089/725 93 82
FORTH im Unterricht	Rolf Kretzschmar, Tel.:02401/4390
Objekt-orientiertes FORTH	Christop Krinninger, Tel.:089/725 93 82 Ulrich Hoffmann, Tel.:0431/678850

FORTH-Gesellschaft e.V. - Postfach 1110 - D-8044 Unterschleißheim
Tel.089/3173784, FORTH-Mailbox Tel.: 089/7259625
Postgiroamt Hamburg, Kontonr.: 563211-208 BLZ 20010020

Ergänzungen, Änderungen bitte dem Büro der FORTH-Gesellschaft e.V. mitteilen.

FORTH-SYSTEME

ANGELIKA FLESCH

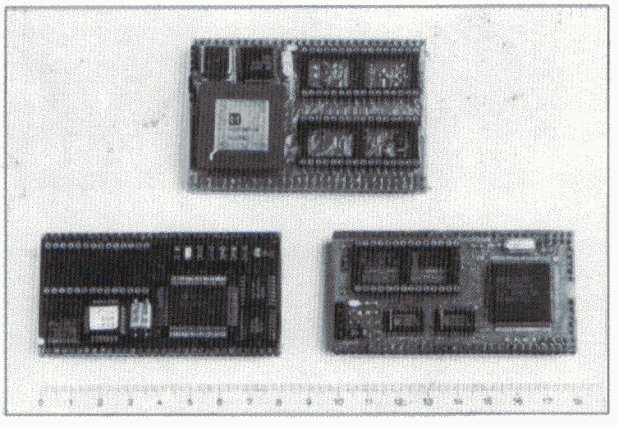
Postfach 1103
D-7814 Breisach

Telefon (0 76 67) 5 51
Telefax (0 76 67) 5 55

UR/FORTH

- Forth-83 Standard
- Für MS-DOS, OS/2, 80386
- Direkt gefädelte Code Implementationen mit dem obersten Stackwert im Register um größtmögliche Ausführungsgeschwindigkeit zu erreichen
- Segmentiertes Speichermodell mit Programm, Daten, Headers und Dictionary Hash Table jeweils in einem getrennten Segment
- Komplett gehashtes Dictionary führt zu extrem schneller Übersetzung
- Mächtige neue String Operatoren (Suche, Extraktion, Vergleich und Addition) sowie einen dynamischen String-, Speichermanager
- Kann mit Objektmodulen, die in Assembler oder anderen Hochsprachen erzeugt wurden, gelinkt werden
- Native Code Optimizer zur direkten Umsetzung in 80 x 86 Code im Lieferumfang

ModuNORM



CPU-Steck-Module im Scheckkartenformat:

- 8 Bit z. B. 6303
- 16 Bit z. B. V25
- Highspeed RTX-2000/1
- Softwareunterstützung durch SwissFORTH™
- Thermodrucker und Kontroller

Bitte fordern Sie unseren Produktkatalog und Preisliste an. FORTH-Gesellschaftsmitglieder erhalten bis zu 10 % Rabatt (artikelabhängig).

LMI FORTH-83 Metacompiler

Der LMI Forth Metacompiler wird mit komplettem Quellcode für ein ausführlich ausgetestetes, Hochgeschwindigkeits Forth 83 Kern ausgeliefert, wobei Sie die Auswahl aus folgenden Zielprozessoren haben:

8086/8088	8096/97
Z80	HD64180
8080/8085	8031/32/535
68000	6303
Z8	6502
1802	V25
6809	68HC11
65816/65802	RTX 2000

Sie erzeugen schnelle und kompakte Anwendungen, indem Sie Ihre Quellprogramme mit unserem Forth Nucleus zusammenstellen und ihn mit dem LMI Forth Metacompiler übersetzen.

Forth Programme, die mit einem LMI interaktiven Forth System z. B. PC/FORTH oder Z80 Forth geschrieben und getestet wurden, werden im Normalfall mit nur geringen Änderungen übersetzt.

Serieller ROM/RAM Simulator

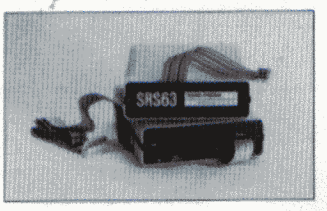
Entwickeln Sie romfähige Programme ?

Müssen Sie neu entwickelte Einplatinencomputer testen ?

Setzen Sie 2764, 27128, 27256, 27512 oder 4364, 43256 oder kompatible ROM/RAM-Bausteine ein ?

Wollen Sie diese Bausteine mit bis zu 38 400 Baud über die serielle Schnittstelle laden ?

Können Sie eine zusätzliche serielle Schnittstelle über den Speichersockel zum interaktiven Programmieren gebrauchen ?



Dann ist unser SRS63 die optimale Ergänzung Ihres Arbeitsplatzes.

Sie werden vom Preis-Leistungsverhältnis überrascht sein.

Unsere ROM-Compiler liefern direkt verwendbare Dateien, wir akzeptieren auch Intel-Hex oder Motorola-S-Formate.

mikrap

Mikrap
AG für Mikroelektronik-Applikation

CH-8840 Einsiedeln
Langrütistrasse 44
Postfach

Telefon 055 53 65 55
Telefax 055 53 61 23