VIERTE DIMENSION

Volume VI/Nr.4 - Dezember 1990

Themen

- **Lokale Variablen**
- **Modemanschluß**
- **Automatische Dokumentation**
- **Linked Actions II**
- **DFÜ: BBS.FORTH-eV.de**

FORTH MAGAZIN

7,50 DM

FORTH - Tagung'91
in München
12. - 14. April 1991

STRESS2 — Echtzeit-Erfassungs- und Berechnungs-System für Materialermüdung

Rhf Reilhofer KG

Das System für

✔ **die Erprobung von Konstruktionsteilen in der PKW- und Nutzfahrzeugindustrie.**
Nutzen: Reduzierung der Versuchszeiten durch frühzeitige Antworten aus der Erprobung.
Weit kürzere Testläufe bei der Simulation des Fahrbetriebs.

✔ **die Dauerüberwachung von stark beanspruchten Kraftwerkskomponenten oder Walzgerüsten in der Stahlindustrie.**
Nutzen: das Auswechseln von Komponenten kann auf die technisch notwendigen und richtigen Intervalle in Abhängigkeit der Ermüdung reduziert werden.

✔ **die Entwicklung.**
Nutzen: Einsparung von Material.
Einsatz von billigerem Material.
Zeit- und Qualitätsvorsprung.

REILHOFER KG, Frühlingsplatz 9, 8047 Karlsfeld, ☎ 08131/92059, FAX:08131/97447

EDV-Beratung – Software-Design – Goppold

Bgm. Germeierstr.4 – 8011 Poing – Tel.: 08121-82710

Wir haben das Forth Know-How:

❑ Consulting, Projekt-Management, Beratung, Schulung.

❑ FORTH unter UNIX (alle Systeme), SUN-Workstations (SPARC&68k), PC-Systeme, Forth-Prozessoren.

❑ Eigen-Entwicklung fortgeschrittener Software-Technologie: Objekt-Programmierung, Datenbanken, Hypertext.

❑ Distributor für MPE: Single Chip Targets&Boards, RTX 2000, Eprom-Emulatoren, Modular Forth, Power Forth.
(Katalog DM 5,- VK oder NN)
Vermittlung von: LMI, Harvard Softworks.

❑ Und natürlich Leibniz, das System nach Forth.
Demo-Disk: DM 20,- VK oder NN

EDITORIAL

Dies ist die letzte 'Vierte Dimension', die unter unserer Regie erschienen ist. Die frei werdende Zeit werden wir wieder FORTH genauer der Weiterentwicklung unseres GEM-Programmiersystems unter FORTH (Atari ST und PC) widmen. Wir möchten uns an dieser Stelle für das uns entgegengebrachte Vertrauen und die zahlreich eingesandten Artikel bedanken. Die neue Redaktion ist unter der Anschrift:

Peter Dinies,
Metzstr.38,
2300 Kiel

zu erreichen. Es war zu vernehmen, daß der neue Editor sich in der nächsten Ausgabe vorstellen wird.

Auch diese 'Vierte Dimension' enthält wieder viele interessante Artikel. Außer der Beschreibung des FORTH-Systems F68K, einem Tool zur automatischen Dokumentation und einer weiteren Version von lokalen Variablen steht in diesem Heft die DFÜ im Mittelpunkt. Neben einem äußerst grundlegenden Thema – dem Anschluß eines Modems an das Telefonnetz – erläutert Heinz Schnitter in dieser Ausgabe die Nutzung der Domain-Adressen der FORTH-Gesellschaft im EUnet.

Ein gutes neues Jahr wünscht Ihnen zum letzen Mal, Ihr Redaktionsteam

Rainer Aumiller
Denise Luda

FORTH-Tagung '91 im Hotel Olympia, W-8046 Hochbrück bei München, Ingolstädter Landstraße 110, Telefon: 089-31816-0.

IMPRESSUM

Titel:
FORTH MAGAZIN 'Vierte Dimension' ®
Zeitschrift der Mitglieder der FORTH-Gesellschaft e.V.
© 1990

Herausgeber:
FORTH-Gesellschaft e.V.

Redaktion und Satz©:
✉ D. LUDA Software©, Gustav-Heinemann-Ring 42, 8000 München 83, ☎ 089/670 83 55, FAX 089/679 22 71. Achtung ab 01.01.91 neue Redaktion; Anschrift siehe Editorial. Bitte keine Artikel mehr an D.LUDA Software übersenden.

Kontaktadresse:
Entweder direkt die Redaktion anrufen bzw. anschreiben, das FORTH-Büro in München, Postfach 1110, 8044 Unterschleißheim, ☎ 089/3173784 kontaktieren oder die FORTH-Mailbox (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 ✆ 08841/5880 8N1 zu finden.

Quelltext Service

Autoren dieser Ausgabe:
Jörg Staben, Jörg Plewe, Frank Stüss, Claus Vogt, Rafael Deliano, Rolf Lehnhardt, Ulrike Schnitter, Heinz Schnitter, Friederich Prinz, Arndt Klingelnberg.

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,-, bei Auslandsadresse DM 45,- inklusive Versand.

Für jedes eingesandte Manuskript sind wir sehr dankbar. Für die mit Namen oder Signatur des Verfassers gekennzeichneten Beiträge übernimmt die Redaktion lediglich die presserechtliche Verantwortung. Die in dieser Zeitschrift veröffentlichten Beiträge sind urheberrechtlich geschützt. Übersetzung, Vervielfältigung, Nachdruck sowie Speicherung auf beliebigen Medien ist allerdings auszugsweise mit genauer Quellenangabe erlaubt. Freie Mitarbeit ist erwünscht. Die Beiträge müssen frei von Ansprüchen Dritter sein. Veröffentlichte Programme gehen, sofern nicht anders vermerkt, in die Public Domain über. Für Fehler im Text, in Schaltbildern, Aufbauskizzen usw., die zum Nichtfunktionieren oder evtl. Schadhaftwerden von Bauelementen führen, kann keine Haftung übernommen werden. Sämtliche Veröffentlichungen erfolgen ohne Berücksichtigung eines eventuellen Patentschutzes, auch werden Warennamen ohne Gewährleistung einer freien Verwendung benutzt.

Vierte Dimension

I n h a l t

Leserbriefe und Zuschriften

Betrifft: Linked Actions

Eine kleine Anmerkung will ich nur loslassen zum Code-Sharing beim Ausführen und Anzeigen der Liste. Es geht, wie immer in FORTH noch anders (was der Artikel ja auch schön zeigt). Ich halte meine Lösung des Problems für wesentlich eleganter, da sie auf die Notwendigkeit eines »deferred« Wortes sowie auf das Aktionswort verzichtet. Nochmals zur Erinnerung: Es geht um ein Wort, das für jedes Element einer Liste eine gleichartige Aktion ausführt. Da das Hangeln durch die Liste immer gleich bleibt, kann man diesen Code teilen. In der Lösung von Jörg Plewe, Frank Stüss und Jörg Staben wird das mit einem deferred Wort erledigt, dem vor Aufruf des Listenbearbeiters per IS eine Aufgabe zugewiesen wird (eine nicht ganz saubere Sache – bei Multitasking können nicht zwei Listen gleichzeitig abgearbeitet werden). Nun meine Lösung:

```
| : >list ( element tail -- )
     >r ; restrict

: list> ( thread -- element )
  BEGIN @ dup WHILE dup r@ >list
  REPEAT drop rdrop ; restrict

\ In bigFORTH kann man >LIST
\ durch EXECUTE ersetzen

\ Als Beispiel: WORDS
: words ( -- ) context @
  LIST> ( element -- )
  ?cr cell+ .name space ;
```

Sicher ist diese Lösung etwas gewöhnungsbedürftig, da hier auf dem Returnstack implizit ein Funktionszeiger übergeben wird (der Rest des aufrufenden Wortes ist die auszuführende Funktion). Ebenso verkorkst ist dann der Einsatz des Wortes >R, das die ganze Ausführungsreihenfolge wirklich total umkehrt. Lernt man doch in jedem FORTH-Kurs, daß man >R und R> genau abzählen soll und schon gar keinen Wert auf dem Returnstack vergessen darf. Allenfalls wird mal erwähnt, daß RDROP auch zum Überspringen einer Aufrufebene genutzt werden kann, dieses Konstrukt wird aber gleich als »unschön« abgewertet.

LIST> zeigt auf sehr schöne Art und Weise, wie man den FORTH-Compiler auch erweitern kann; LIST> ist nämlich eine implizite Schleife, ein Konzept, das mir sonst nur noch von APL bekannt ist. Alle Befehle hinter LIST> werden auf jedes Listenelement angewendet.

Übrigens: TIB ist im ST-volksFORTH als : TIB >TIB @ ; definiert, zum Umlenken von TIB für EVAL ist also kein MULTIPUSH notwendig – nur >TIB muß gepusht werden und auf den neuen Zeiger gesetzt werden:

```
: eval ( <string> -- ?? >
  >tib push #tib push >in push
  blk push count #tib ! >tib !
  >in off blk off interpret ;
```

✉ Bernd Paysan
Stockmannstr.14
8000 München 71
✆ 089/798557

Fehler im ultraFORTH gesucht !

Wer hat im ultraFORTH 3.8 für C64 oder C16 einen Fehler entdeckt? Der oder die fühle sich hiermit aufgerufen, sofort zum nächsten Telephon zu stürzen oder hektisch ein Blatt Papier zu suchen, um mir eine kurze Fehlerbeschreibung zukommen zu lassen. Wir wollen vielleicht ein neues Update 'rausbringen, und da sollten dann doch möglichst alle bekannten Fehler verbessert sein. In der Hoffnung auf regen Rücklauf

✉ Philip Zembrod,
Kronenstr. 20 ,
W-7800 Freiburg
✆ 0761/702651

FORTecH Software

Wir haben ihn gewonnen,

... den Programmierwettbewerb zur Messe ECHTZEIT '90.

Und von den Kunden, für die wir seit fünf Jahren mit dem Siegersystem comFORTH arbeiten, hat sich (natürlich) keiner gewundert.

Auch für Sie zu haben: das Originalsystem comFORTH

wahlweise z. B. für:

- Totalkontrolle über Ihren PC
- Firmware-Entwicklung
- Echtzeitanwendungen in der Automation
- Programmierung verteilter Rechnersysteme
- KI-Probleme
- Numerische Aufgaben

Das ist akkumulierte akademische Brainpower - nicht von Informatikern, sondern von Ingenieuren für Ingenieure plus zehn Jahre praktische Automation potenziert mit zehn Jahren Forth-Know-how.

Automatisierung - unsere Spezialstrecke: Wir automatisieren a l l e s.

FORTecH Software GmbH
Albert-Einstein-Straße 2
O 2500 Rostock 6
Telefon 40 55 96
oder: **c/o Becker & Partner**
Bremer Straße 18
W 2100 Hamburg 90

Gratis-Coupon

Mich interessiert:

- ❑ comFORTH - Infos
- ❑ comFORTH - Preisliste
- ❑ Demo - Disk comFORTH
- ❑ Automatisierung

Mein Automatisierungsvorhaben:

Mein Name/Firma:

Straße/Nr.:

PLZ/Ort:

FORTH–GruppeMoers

BRIEF AUS DER PROVINZ

von Friederich Prinz

Liebe FORTH-Freunde

Der Brief unserer Gruppe basiert auf Diskussionen die wir vor dem Erscheinen des c't Artikels (*FORTHgeschritten F-PC 3.5*) geführt haben.

Inzwischen liegt dieses 'neue' FORTH von Tom Zimmer auch hier vor. Mir persönlich gefällt es sehr gut. Allerdings muß ich dazu gleich sagen, daß die 'Puristen' in unserer Gruppe mit F-PC ganz und gar nicht einverstanden sind. Ihnen ist es schon viel zu mächtig – und vor allem zu umfangreich. Unsere Anfänger habe ich bis jetzt noch gar nicht in F-PC hineinsehen lassen. Ich fürchte, daß die große Wortmenge ein bischen erschlagend wirkt.

Trotzdem möchte ich, um wieder an den Brief unserer Gruppe anzuknüpfen, gerade das F-PC 3.54 der FORTH-Gesellschaft empfehlen, bzw. zur Empfehlung vorschlagen. Nicht nur der Befehlsvorrat, sondern auch die Oberfläche, der Editor, Hypertext und diverse andere Werkzeuge tragen in F-PC dafür Sorge, daß dieses System keinem der kommerziellen Systeme von Borland oder Microsoft mehr nachsteht. Alle bisherigen Kritiken an FORTH, die sich vornehmlich auf die 'unfreundliche' Oberfläche der Entwicklungsumgebung gestützt haben, werden hier zunichte gemacht.

Die häufige Kritik, mit FORTH könne man keine kleinen Applikationen erstellen, stellt der in F-PC enthaltene Targetcompiler TCOM.EXE richtig. Um F-PC auch Anfängern zugänglich zu machen, bedarf dieses System allerdings einer gründlichen 'Umarbeitung'. F-PC ist kein System, in das man sich problemlos selbst einarbeiten kann. Die mitgelieferten Dokumentationen sollten übersetzt werden. Zu den einzelnen 'Teilen' des F-PC sollten versiertere FORTHler Arbeitsbeschreibungen und Beispiele schreiben und diese der FORTH-Gemeinde zu Verfügung stellen.

Das ist wiederum eine Arbeit, die von einer Einzelperson nicht in annehmbarer Zeit geleistet werden kann. Deshalb möchte ich vorschlagen, daß sich mehrere Mitglieder der FORTH-Gesellschaft in diese Arbeit teilen. Ich habe hierzu inzwischen mit Jörg Staben telefoniert, der wohl bereit wäre diese Arbeit zu koordinieren. Ich selbst habe mich bereits daran gesetzt, eine deutsche On-Line Hilfe zu TCOM und seinen Werkzeugen zu schreiben.

Ich halte F-PC für *das* FORTH-System der 90er Jahre. Sicher werden nicht nur die 'Puristen' diese Meinung ungeteilt übernehmen können. Auch F-PC ist nicht vollkommen. Die bisher aufgetretenen 'Macken' werden aber nur versierten FORTHlern auffallen und lassen sich leicht beheben. Grundsätzlich sehe ich F-PC als ein vollständiges Entwicklungssystem an, mit dessen Hilfe FORTH und die FORTH-Gesellschaft ein wenig mehr aus der Ecke der 'unverstandenen Elite' herauskommen und populärer werden könnten. Und das wünschen wir uns eigentlich wohl alle.

In diesem Sinne grüße ich recht herzlich vom Niederrhein

Euer

Friederich Prinz

Kleinanzeigen

Verkaufe **Minibee RTX 2000 FORTH-Board**, 128 KByte RAM, FG-FORTH + Netzteil + serielle Schnittstelle + Software + jede Menge Unterlagen, 1x benutzt, FP 700,-, Manfred Kraft ☎ 069/230138.

DRUMA-FORTH V.1.1 zu verkaufen, VB, Jörg Staben, ☎ 02103/55609.

EOS10 – Barcode – Programm, Heinz Lederer, 8013 Haar bei München, ☎ 089/466631.

KURZMITTEILUNG

von Arndt Klingelnberg

F-PC 3.50ak als überarbeitete Version mit Umlauten und weiter verbesserter Hilfe, korrigierten Bugs steht nun zur Verfügung. Distributionskosten inkl. Original Zimmer 3.50, Ergänzungen, ausgesuchten DOS- Utilities/Telemate 3.11, anderen FORTH- Systemen, komprimiert (!), 5*1M44 DM 80, 15*360k DM 90, bei Angabe der FORTH- Mitgliedsnummer DM 20 Nachlass, DEMO (alle PC- Formate) DM 20, nur V-Scheck. Ab Mitte Dezember soll die sehr umfangreiche Dokumentation in kompakter Form verfügbar sein (Bestellungen für die letzte Auflage spätestens Ende Dezember).

Arndt Klingelnberg,
Strassburgerstr. 12,
D- 5110 Alsdorf,
✆ 0 2404 - 6 16 48

Lokale Gruppe Rhein-Ruhr

von Jörg Plewe

Die Treffen der lokalen Gruppe Rhein-Ruhr können nun endlich zu festen Zeiten an festem Ort stattfinden. Dank der Initiative unseres Mitgliedes Albert Schäfer, der auch einem Schachverein angehört, stehen uns nun an jedem ersten Samstag im Monat die Räumlichkeiten dieses Vereins zur Verfügung. Wir bilden dort gleichzeitig den 'Workshop Schach und Computer'. Reinhard Scharnagl konnte uns hierbei schon in die Grundlagen der Programmierung von Schachcomputern einführen. Natürlich bleiben aber auch allgemeine Themen bei unserem monatlichen Kaffeeklatsch nicht auf der Strecke. So führte Horst Wilke ein FORTH-gesteuertes Echolot-Meßgerät vor, Jörg Plewe sucht Rat für seine Blockstreams und Jörg Staben berichtet über neue Geheimnisse, die er dem F-PC entringen konnte.

Im Oktober stellte sich die Gruppe auf einem Computerflohmarkt in Düsseldorf ins Licht der Öffentlichkeit, was sogar nachweislich eine Anmeldung bei der FORTH-GESELLSCHAFT und neue Teilnehmer bei unseren Treffen nach sich zog. Wer Lust hat, uns zu besuchen, sei herzlich eingeladen:

jeder 1. Samstag im Monat, 16.00 Uhr
Münsterstr. 199
4000 Düsseldorf

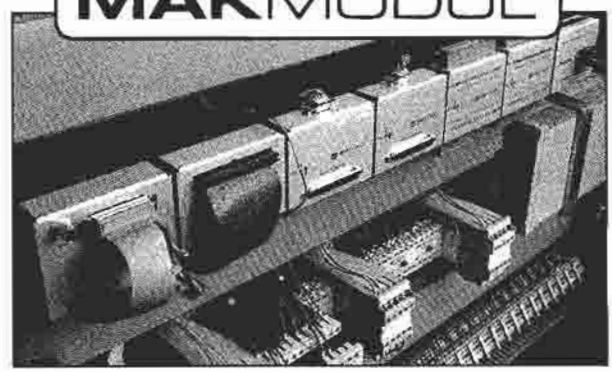
im Gebäude des S-Bahnhof Derendorf (nein, nicht in der Bahnhofskneipe).

(JP)

MICROPROCESS

Innovativ, leistungsstark, und einfach zu programmieren:

MAKMODUL® *



Das Prozeßrechner-Konzept für die Meß- und Regeltechnik.

- **Innovativ:** Durch extreme Präzision in der Meß- und Regeltechnik eröffnen MAKmodule Ihren Produktionsanlagen und Maschinen ungeahnte Möglichkeiten.
- **Leistungsstark:** Bei laufender Produktion sichern MAKmodule z. B. die Qualität durch Messungen mit einer Auflösung von unter 0,001 mm.
- **Einfach zu programmieren:** Klartext-Programmierung und die große Bibliothek der MAKmodule verkürzen die Programmierzeit entscheidend.

Namhafte Firmen in Europa nutzen den Vorsprung von MAKmodul.
Fordern Sie Informationen an!

MICROPROCESS GmbH
Vertriebspartner der Dr. Weiss GmbH
Division MAKmodul
Talstraße 136 Telefon (0 62 03) 67 81
D-6905 Schriesheim Telefax (0 62 03) 6 81 64

* eingetragenes Warenzeichen der Dr. Weiss GmbH

FORTH-Gesellschaft e.V. intern

von Ulrike Schnitter

Neuigkeiten und Bekanntes der FORTH-Gesellschaft e.V.

1990 fanden drei FORTH-Veranstaltungen statt. Bei der FORTH-Tagung '90 in Frankfurt am Main hatten wir die Möglichkeit Kontakte zu FORTH-Programmierern aus den neuen Bundesländern herzustellen. Auf dem Kongreß Echtzeit '90 in Sindelfingen konnte das FORTecH-Team aus Rostock Pfüller, Woitzel und Neuthe den Programmierwettbewerb für sich und FORTH entscheiden. Der FORTH-Workshop in Suhl war sehr gut besucht, und die die Tagung begleitende Ausstellung demonstrierte die Leistungsfähigkeit von FORTH.

Leider können wir den auf der FORTH-Tagung '90 gefaßten Beschluß, die Mitgliederversammlung 1991 in Rostock oder Leipzig abzuhalten, nicht verwirklichen. Die in den neuen Bundesländern ungeklärten Eigentumsverhältnisse erlauben vorerst keine langfristige Planung für unsere Jahrestagung. Im Einvernehmen mit allen Beteiligten haben wir uns nun entschlossen, die FORTH-Tagung '91 in München, im Hotel Olympia abzuhalten (bitte Termin vormerken: 12.-14. April 1991). Bei der Gestaltung der Tagung haben wir Anregungen und Wünsche von Mitgliedern zur Durchführung der Jahrestreffen aufgegriffen. Wir werden 1991 deshalb die Tagung und die Unterbringung unter einem Dach organisieren. Auch das Tagungsprogramm wollen wir lockerer gestalten, um Freiräume für Workshops und Diskussionen zu schaffen. Außerdem besteht die Möglichkeit, daß die Tagungsteilnehmer eine Begleitperson mitbringen können. Das Tagungshotel bietet für ca. 60 Personen Übernachtungsmöglichkeit in 2-Bett-Zimmern. Die Zimmer sind mit Dusche und WC, Telefon, Balkon und TV ausgestattet. Die Tagungsgebühren sind Komplettpreise und schließen Übernachtung, Vollpension und Tagungsgebühren sowie Tagungsband ein. Wir haben sehr knapp kalkuliert um möglichst vielen Mitgliedern die Teilnahme zu ermöglichen.

Die nächste 'Vierte Dimension' erscheint im März '91

Administration

Mitgliedsbeiträge und Zahlungsmodus

Bitte entrichten Sie Ihren Mitgliedsbeitrag bis Februar 1991, oder machen Sie Gebrauch von der Einzugsermächtigung. Als Erinnerung liegt dieser 'Vierten Dimension' ein Überweisungsformular für 1991 bei. Hier die gültigen Mitgliedsbeiträge:

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

Der Mitgliedsbeitrag gilt immer für das laufende Kalenderjahr. Beachten Sie bitte, daß die ermäßigten Beiträge nur gegen Nachweis gewährt werden, also bitte für 1991 die Studienbescheinigungen usw. in Kopie beilegen. Bei Auslandsadresse ist wegen der erhöhten Versandkosten keine Ermäßigung möglich. 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 abbuchen.

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 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.

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 Interessensgebiete der Mitglieder in die Liste aufzunehmen; bitte benützen Sie die vorgesehenen Abkürzungen:

BF	Beruflich mit FORTH
BV	Bildverarbeitung
DB	Datenbanken
DFÜ	DFÜ-fähig
DK	Datenkommunikation
FM	FORTH-Maschinen
GR	Grafik
IM	Implementationen
K	Kontakte gesucht
KI	Künstliche Intelligenz
NU	Numerik
OOF	Object Oriented FORTH
PZ	Prozeßtechnik
RTX	RTX-2000
TV	Textverarbeitung
VS	Verteilte FORTH Systeme
68k	68000-FORTH

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 erfolgreiches, gesundes 1991.

Ulrike Schnitter
FORTH-Büro

DFÜ-Aktivitäten

'FORTH-eV.de'

von Heinz Schnitter

Die FORTH-Gesellschaft e.V. hat noch eine zusätzliche Adresse 'FORTH-eV.de' erhalten. Diese Adresse ist eine sogenannte Domain-Adresse und wird an wichtige Institutionen wie Universitäten, Großfirmen und eben die FORTH-Gesellschaft e.V. vergeben. Das 'de' am Ende ist die Toplevel-Domain-Kennung für Deutschland. Davor steht der Domain-Name für die jeweilige Institution, in unserem Falle 'FORTH-eV'. Wieder durch einen Punkt getrennt schreibt man vor den Domain-Namen den Rechner-Namen. Unsere Mailbox BBS in Murnau hat folgenden vollen Namen 'BBS.FORTH-eV.de' und den Rechner der Verwaltung erreicht man mit 'Admin.FORTH-eV.de'.

Die neue Adresse hat mehrere Vorteile:

- ✗ Es entstehen keine zusätzlichen Kosten.
- ✗ Wir haben die Möglichkeit ein eigenes Rechnernetz aufzuziehen.
- ✗ Die neue Adresse ist aus fast allen Netzen heraus erreichbar

Die Gestaltung des USER-Namens wurde in der 'Vierten Dimension' Volume VI Nr 3 nicht deutlich genug dargestellt. Möchte ein User an allen Netzen teilnehmen, darf sein USER-Name nur Buchstaben des Alphabets (ohne Umlaute) und das Minuszeichen enthalten. Die richtige Adresse im EUnet wäre dann `d-duck@forthev.UUCP` und `d-duck@BBS.FORTH-eV.de` im Internet.

Einige wichtige Adressen

Das FORTH-Büro:

```
secretary@Admin.FORTH-eV.de
```

oder

```
secretary@forthev.UUCP
```

Sysop:

```
system@BBS.FORTH-eV.de
```

oder

```
system@forthev.UUCP
```

Ist man in der Mailbox eingeloggt, genügt es eine Mail an 'secretary' bzw. 'system' abzuschicken.

Um die ersten Kontakte mit unserer Mailbox zu erleichtern, ist diesem Heft ein 'TELIX-Capture-File' beigefügt (siehe Listing unten), und zwar für die Anmeldung und die ersten Gehversuche in unserer Mailbox. Versuchen Sie bitte den

zweiten Teil nachzuvollziehen, denn dann sind Sie im Besitz der Anleitung für die BBS-Software. Wenn Sie TELIX verwenden, schalten sie bitte die Capture-File ein, um bei Problemen ein 'Replay' des Vorgangs machen zu können.

In nebenstehender Tabelle sind alle Mailboxen aufgeführt, in denen Sie die vier Bretter der FORTH-Gesellschaft finden.

Heinz Schnitter

Name	Telefon	Ort	Mailboxtyp
MUFB	08841-5880	Johannes Teich W-8110 Murnau	Waffle
INFINET	089-656632	W-8000 München 90	Zerberus
NO_NAME	0209-598560	W-4650 Gelsenkirchen	Zerberus
MHB	02841-57325	W-4130 Moers 1	Zerberus
TBX	030-6930113	W-1000 Berlin	Zerberus
KBBS	0431-336038	W-2300 Kiel	Zerberus
BIONIC BOX	0521-171188	W-4000 Bielefeld	Zerberus
RAVENNA	089-8002993	W-8039 Puchheim	Zerberus
AME	09131-992998	W-8520 Erlangen	Zerberus

Tabelle 1: Mailboxen

```
 2400
,,,,,
BBS.FORTH-eV.de (forthev)

UUCP & Mail site,
FORTH-Gesellschaft e.V. Postfach 1110 W-8044 Unterschleissheim

Waffle version 1.63, out of the box.

Type "guest" or your special login handle,
or gain a handle by typing  new'.

Login or NEW: NEW[RETURN]

* the New User I.Q. Test..

Sending intelligent FEEDBACK after all this     | Bitte beantworten Sie
brouhaha will tend to expedite validation here. | die folgenden Fragen.

Enter your complete name or handle.             | Ihren vollen Namen, bitte.

Full name: Donald Duck[RETURN]

Enter a short, one word handle or nickname;     | Ihr Kürzel, mit dem Sie
this will be your permanent account name.       | sich eingloggen wollen, z.B.
For instance, "Nairobi Trio" might choose       | 'm-mue' für  Max Müller'.
either "nairobi",  trio , or invent something.  | Kein Punkt! Kein Umlaut!

Username (2-8 chars): d-duck[RETURN]

Your phone number...                            | Ihre Telefonnummer...

Phone: 089-317 37 84[RETURN]

Choose a password...                            | Ihr Passwort...

Password (4-8 chars): daisy[RETURN]
Tell the system how you prefer                  | Wie wollen Sie vom System
to be addressed.                                | angesprochen werden?

Your preferred (e.g. first) name is...? Donald[RETURN]

[voluntary]

If you are the sysop of another BBS, have an    | Besondere Angaben zu
Amateur Radio call sign or an UUCP address      | Ihrer Person, vor allem,
or a fax number or FG membership or something   | ob Sie FG-Mitglied sind
silly to say, please let us know here.          | (wegen der Privilegien).

Are you an FG member? (Number?) ja #4711[RETURN]

Be prepared that we will call back!             | Rechnen Sie mit einem Rückruf!
Answer "n" if you don't like that.              | Wenn unerwünscht, "n" eingeben.

Is the above information correct?               | Alles korrekt bis hierher?

Please enter 'y' or 'n': y[RETURN]

Wait [.] [.]

Login: d-duck
Password: daisy




WRITE THIS DOWN, or die man.                    | Benutzernamen  Password & merken,
                                                | oder Sie sind verratzt.
Press [RETURN] to continue: [RETURN]



Welcome, new user person --

Your account here will be restricted            | Der Sysop ist bemüht, Sie in
until validation, which should take less        | der Fülle der Antragsteller
that 24 hours. Feel free to explore,            | bevorzugt zu behandeln und
and if you find yourself lost, never fear.      | Ihren Account freizugeben,
The system was written to be intentionally      | d.h. Ihren level hochzusetzen.
confusing.                                      |

                    --  We will call back!      | Rechnen Sie mit einem Rueckruf!

If you are annoying,                            | Ansonsten sehen Sie sich mal
prepare to be 'removed.'                        | um, vielleicht entdecken Sie
If not, please come in.  Have a beer.           | irgendwelche Un/Gereimtheiten.
There's no one here but us.                     | Hier wird mit Wasser gekocht!


[more] [RETURN]
Hit return for BBS: [RETURN]

    Logging in: d-duck, #843
Previous login: 11-Dec-90 (0 new messages)
   Last caller: hfs (Heinz F. Schnitter)

  ,,,                      ,    ,,,
 ]] ']                    ]]    ]]
,]],,  ,]'''],  '],''], ''])''  ]],'],  ,]'''],  ]]   ]]
 ]]    ]]  ]]  ]]  ''   ]]     ]] ]]  ]]''''  '],  ,}'
,]],   '],,,]'  ,]],     '],]' ,]] ]]  '],,,]'   ']]'

UUCP-Knoten der FORTH-Gesellschaft e.V., Unterschleissheim

BBS_FORTHEV  D-8110 Murnau  47:40:30 North - 11:11:30 West
110 .. 2400 bps / 8n1 / 24 h / 08841-5680 / ++49-8841-5880

        FORTH gateway: Unonet/Ennet -- Zerberus
      Net addresses: forthev.UUCP, BBS.FORTH-eV.de

Journal: 10-Dec-90              FORTH articles: #152..491

No mail.

[Local Board #0: Info]
6:12p (?=help!) - boards[RETURN]

        L O C A L   F O R U M S

  0     Info hier das Neueste von der / über die FORTH-Gesellschaft e.V.
  1     Forum Diskussionen über Diskussionen

[Local Board #0: Info]
6:12p (?=help!) - read[RETURN]

[Local Board #0: Info]

(No new messages / highest is 4)




Post, List, [1-4] Quit: list 1 4[RETURN]

001: Die Programmierumgebung FORTH
002: Die FORTH-Gesellschaft e.V.
003: Die FORTHEV-Bretter
004: Mitgliedschaft

Post, List, [1-4] Quit: 3[RETURN]

* hier wird der Inhalt von "003: Die FORTHEV-Bretter" angezeigt *

[RET] 1-4, Q)uit: q[RETURN]

[Local Board #0: Info]
6:13p (?=help!) - 1[RETURN]

[Local Board #1: Forum]
```

```
6:13p (?=help!) - read[RETURN]

[Local Board #1: Forum]

(1 new message / begin reading at 33)

[RET] 1-33, Q)uit: list 20 33[RETURN]

020: 0=Info, 1=Forum
021: Buchtip fuer FFT, digitale Filter, ...
022: F-PC 3.54
023: Leitungsguete
024: F-PC in deutsch
025: Design-Wettbewerb RTX2001A (FORTH-Prozessor)
026: Messegefluester
027: Suche Artikel ueber Threading Techniken
028: Floating Point
029: Forth-Prozessoren
030: FORTH fuer C/PM 2.2
031: Re: Floating Point
032: (J.Henkel:) Re: Floating Point
033: Portfolio

[RET] 1-33, Q)uit: 33[RETURN]

| hier wird der Inhalt von "033: Portfolio" angezeigt |

Post, List, [1-33] Quit: q[RETURN]

[Local Board #1: Forum]

7:37p (?=help!) - bye[RETURN]
     d-duck   logged out   11-Dec-90 7:38p
NO CARRIER


Zwei Tage später mit höheren Privilegien.............


 2400
,,,,,
BBS.FORTH-eV.de (forthev)




UUCP & Mail Site,
FORTH-Gesellschaft e.V. Postfach 1110 W-8044 Unterschleisheim

Waffle version 1.63, out of the box.

Type 'guest' or your special login handle,
or gain a handle by typing 'new'.

Login or NEW: d-duck[RETURN]
Password: daisy[RETURN]

    Logging in: d-duck, #875
Previous login: 11-Dec-90 (0 new messages)
   Last caller: jgt (Johannes Teich)

  ,,,                       ,    ,,,
 ]] ']                     ]]    ]]
,]],,  ,]'''],  '],,''], ''']]''   ]],'],  ,]'''],  ]]    ]]
 ]]    ]]   ]]   ]]  ''   ]]     ]]  ]] ]]''''  '],  ,]'
,]],   '],,,]'  ,]],      '],]' ,]] ]]  '],,,]'    ']]'

UUCP-Knoten der FORTH-Gesellschaft e.V., Unterschleissheim

BBS_FORTHEV  D-8110 Murnau  47:40:30 North - 11:11:30 West
110 .. 2400 bps / 8n1 / 24 h / 08841-5880 / ++49-8841-5880

        FORTH gateway: Usenet/Eunet -- Zerberus
      Net addresses: forthev.UUCP, BBS.FORTH-eV.de

Journal: 10-Dec-90              FORTH articles: #152..491

No mail.

[Local Board #0: Info]
6:13p (?=help!) - usenet[RETURN]

alt.bbs.waffle              unsere Board-Software
comp.lang.forth             internationale News
control                     USENET-Informationen

[Local Board #0: Info]
6:14p (?=help!) - comp.lang.forth[RETURN]

[Newsgroup comp.lang.forth]
6:14p (?=help!) - read[RETURN]

[Newsgroup comp.lang.forth]

(340 new messages / begin reading at 152)

[RET] 152-491, Q)uit: list 460 491[RETURN]

460: F83 68000 Assembler Source ?
461: Re: Now that Harris is gone...
462: On-line updates and comments
463: FPC forth?????????
464: FORTH and UNIX
465: LMI Forth(s)
466: ANS FORTH TECHNICAL COMMITTEE
467: Now that Harris is gone...
468: FPGA Forth engines
469: Re: FOR US BEGINNERS?  HELP
470: What's WRONG with Forth?



471: 1990 FORML
472: Z80 fig-FORTH - binaries + sources
473: Re: BASIS 14... comments
474: CFAs as identifiers and Aliases
475: Basis On-line
476: Enhancements
477: Re: What makes Forth Forth
478: Need CHEAP bare board for F68HC11
479: Floating point stack
[more] [RETURN]
480: 1990 FORML
481: 1990 FORML
482: Forth Engines / Harris
483: Using FORTH for machine control
484: Re: What makes Forth Forth
485: Who am I...
486: F83 for Atari
487: Zilog Super8 & 4th
488: Re: What makes Forth Forth
489: F83 for the Mac
490: On-line updates and comments
491: 1990 FORML

[RET] 152-491, Q)uit: 491[RETURN]

| hier wird der Inhalt von "491: 1990 FORML" angezeigt |

Post, List, [152-491] Quit: q[RETURN]

[Newsgroup comp.lang.forth]
6:15p (?=help!) - files[RETURN]

* the FILES transfer section..


Protocols: Xmodem, Ymodem, Zmodem      | Mit "P" waehlen: X/Y/Zmodem oder
  and Kermit (X, Y, Z, or K)           |   Kermit (X, Y, Z oder K)
  - same letters as with Telix -       |   - Buchstaben wie Telix -

We hope you've read HELP TRANSFER ?    | Dort steht's genauer.

"?" shows the commands. Type one character only.

(Protocol: Z[RETURN]
Z protocol selected

(Directory: [RETURN]

docs          DIR    21-Nov-90   Anleitungen fuer WAFFLE
f83           DIR    21-Nov-90   FORTH-83-System von Laxen/Perry
fft           DIR    21-Nov-90   Errortrapping und Fourieranalyse in FORTH
miniterm      DIR    21-Nov-90   Fileinterface für Singleboarder, volksFORTH
z80asm        DIR    21-Nov-90   Z80-FORTH-Postfix-Assembler
vd            DIR    21-Nov-90   Source-Texte aus der 'Vierten Dimension'

(Log: docs[RETURN]

(Directory: [RETURN]

user.doc       49103   28-Aug-90

(Send: user.doc[RETURN]
384 blocks, 48k



* im TELIX RECEIVE und ZMODEM einstellen, dann wird die File user.doc gesendet *

(Quit..

[Newsgroup comp.lang.forth]
6:34p (?=help!) - bye[RETURN]
     d-duck   logged out   13-Dec-90 6:34p

NO CARRIER
```

Capture-File

Legende: Text in `fett kursiv` müssen vom Anwender eingegeben werden.
Text, die mit dem Zeichen »« abgetrennt sind, stellen Kommentare dar.

Einladung zur ordentlichen Mitgliederversammlung der FORTH-Gesellschaft e.V.

am Sonntag, den 14. April 1991 ab 10:00 Uhr im Hotel Olympia, W-8046 Hochbrück bei München, Ingolstädter Landstraße 110, Telefon: 089-31816-0.

Ab 12. 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 1990
3. Aussprache und Entlastung des Direktoriums
4. Wahl des Direktoriums
5. DFÜ-Versorgung der Mitglieder mit FORTH-Nachrichten
6. Verschiedenes

Ergänzungen zu den Tagesordnungspunkten können schriftlich bis März 1991 beim FORTH-Büro beantragt werden.

Ein Anmeldeformular für die FORTH-Tagung '91 finden Sie in dieser 'Vierten Dimension'.

FORTH in der BRD

oder

Wie Einsteiger FORTH und die FG sehen

von der FORTH-Gruppe Moers

Etwas Lebendiges, das wachsen und sich entwickeln soll, braucht das Gespräch. Das gilt für FORTH selbst ebenso wie für die FORTH-Gesellschaft und auch die lokalen Gruppen der FORTH-Gesellschaft. Deshalb setzen wir uns in Moers immer wieder zusammen, bunt gemischt, Anfänger und Fortgeschrittene und versuchen in Gesprächen festzustellen, wo die jeweiligen Interessen der einzelnen Gruppenmitglieder liegen. In einem dieser Gespräche haben wir das Thema »FORTH und die FORTH-Gesellschaft« diskutiert – und dabei einige Kritikpunkte aufgedeckt, die vermutlich zumindest von Einsteigern häufig so erlebt werden.

Daß 'unsere' Anfänger von FORTH begeistert sind, braucht hier wohl nicht besonders erwähnt zu werden. An FORTH selbst wird auch keine Kritik festgemacht. Was kritisiert wird, ist die FORTH-Welt. Da sind zum Beispiel die vielen, guten, aber zum Teil in der Performance sehr unterschiedlichen FORTH-Versionen, denen sich der Einsteiger recht hilflos gegenüber sieht. Wie soll er sich für eine der Versionen von ZF, F-PC 1.0 bis 3.5, F83, volksFORTH usw. entscheiden? Am Anfang des Interesses steht FORTH selbst, nicht aber das Sammeln verschiedener Systeme auf die man vermutlich ohnehin nie zurückgreifen wird. Und was sagen die Experten dazu? Nichts Genaues! Die Puristen schwören nach wie vor auf ein F83 mit Screen-Editor und Kommandozeilen. Die 'Künstler' schwärmen von den Systemen, die ihnen sequentielle Files und bequeme Editoren bieten, und damit allen erdenklichen Komfort....

Der Laie hörts und wundert sich. Gerade beginnt er zu erahnen, daß solche 'Features' für FORTH selbst völlig irrelevant sind, und gerät in eine unnötige Entscheidungsnot. Warum, so die Frage unserer Gruppe, empfiehlt nicht die FORTH-Gesellschaft eine Version aus dem PD oder Shareware Bereich? Sicher wäre es fraglich, wie viele Mitglieder der Gesellschaft dieser Empfehlung folgen würden. Die Moerser wären aber sicher dabei....

Und weil wir gerade bei PD und Shareware sind... daß sich ein Anfänger kein LMI-System für 1000,- Märker zulegt, daß ist sicher keine Frage. Dafür gibt es eben sehr leistungsstarke PD-Systeme. Aber wie sehen die oft aus?!? Wenn Dokumentationen vorhanden sind, dann immer in Englisch. Nun kommen gerade die 'privaten' Einsteiger nicht unbedingt aus Ingenieursberufen, und häufig genug sind es noch Schüler, also Menschen, die der englischen Sprache nicht unbedingt mächtig sind. Trotzdem gibt man nicht so schnell auf und beginnt, oft mühsam, die auf den Disketten vorhandenen Manuals zu übersetzen – und stellt dann fest, daß diese nicht vollständig sind. Meist beschränken sich die beiliegenden 'Handbücher' in Diskettenform auf ziemlich dürftige Beschreibungen einiger 'Highlights'. So wichtige Dinge wie FLOATING-POINT-Pakete, ASSEMBLER, VIDEO-Pakete usw. werden in diesen *.DOCs erst gar nicht angesprochen. Warum geht man hier offenbar davon aus, daß sich der wackere FORTHler schon von selbst darin zurecht finden wird? Wie gesagt, es kommt nicht jeder FORTHler aus einem Beruf, der eine entsprechende Vorbildung mit sich bringt und damit das 'tiefere Verständnis' erleichtert. Hier knüpft die Frage an die FORTH-Gesellschaft an. Wenn die Gesellschaft eine 'offizielle' Empfehlung zu einer FORTH-Version aussprechen wollte, wäre es dann nicht möglich, zu dieser Version einmal wirklich umfangreiche Handbücher und Dokumentationen *in Deutsch* zu liefern? Der ansonsten allein gelassene Anfänger hätte es sehr viel leichter über die erste 'Frustphase' des Einstiegs hinweg zu kommen. Sicherlich fänden sich in der FORTH-Gesellschaft erfahrene und routinierte FORTHler, die sich die Mühe machen würden, eine solchermaßen empfohlene Version einmal richtig 'auseinander' zu nehmen und vielleicht nach dem Vorbild der Borland Handbücher zu dokumentieren. Ein solches Handbuch wäre wohl nicht nur den Einsteigern einen entsprechenden 'Obulus' wert.

Bei Büchern zum Thema FORTH setzt ganz allgemein ebenfalls Kritik an. Es gibt zu wenige davon, und von diesen wenigen kann man noch weniger empfehlen. Das oft gelobte Buch von Brodie stößt zumindest teilweise auf Ablehnung. Die netten Bildchen des SWAP-Drachens erheitern beim ersten Hinschauen. Allerdings sind FORTHler, auch Anfänger, doch etwas ernsthafter an der Maschine Computer interessiert. Zuweilen fühlen sie sich durch Brodie 'veralbert'. Das andere 'Standardwerk' der deutschen FORTH-Landschaft, Zech's Bücher »FORTH83« und »Die Programmiersprache FORTH« sind für den Eleven einfach ungeeignet. Für den versierten FORTHler mögen sie unentbehrliche Nachschlagewerke sein, den Anfänger verwirren Zech's Hinweise wie »...wie allgemein bekannt ist..« nur. Mit anderen Worten, Zech setzt in seinen Werken einfach zu Vieles voraus, zumindest zuviel für den Anfänger. Gibt es überhaupt FORTH-Literatur zwischen diesen beiden Leveln? Die Buchbesprechungen der 'Vierten Dimension' bieten einen ersten Orientierungsansatz. Das sollte unbedingt ausgeweitet werden! Ein 'richtiges' Anfängerbuch müßte her. Ein regelrechter FORTH-Kurs wäre angebracht. Könnte man einen solchen Kurs nicht in der 'Vierten Dimension' abdrucken? Dazu müßte die Zeitschrift allerdings auch öfter erscheinen. Alle drei Monate einen kleinen Schritt weiter geführt zu werden ist ein wenig unbefriedigend.

Die 'Vierte Dimension' selbst stößt natürlich auch auf Kritik. Kurzgesagt: Das Niveau ist zu hoch. Was

nutzt dem Anfänger eine Zeitschrift, deren Buchstaben er zwar entziffern kann, deren Inhalt er aber nicht versteht ? Beiträge wie der 'BIBER' interessieren vielleicht Mathematiker und Liebhaber solcher Spielereien. Der FORTH-Eleve kann ein solches Programm höchstens abtippen – sofern er den passenden Compiler dazu hat. Ähnlich verhält es sich mit der Mehrzahl der sicher interessanten Beiträge in der 'Vierte Dimension'. Es sollte doch möglich sein, ohne größeren Niveauverlust, auch für den Einsteiger ein paar Seiten einzubauen, die dieser ganz einfach verstehen kann.

Überhaupt wird 'das Elitäre' der deutschen FORTH-Landschaft bemängelt. Sicher ist es 'schön' einer Elite aus wenigen Spezialisten zugerechnet zu werden. Aber der Weg dorthin ist recht hart. Der Weg bleibt auch unnötig hart, so lange es den Eleven nicht ein wenig einfacher gemacht wird FORTH ohne Frust zu erleben.

Eines der Ziele der FORTH-Gesellschaft e.V. ist es doch, sinngemäß, die Verbreitung von FORTH selbst, sowie die Verbreitung der FORTH-Philosophie zu fördern. Die heutigen, interessierten Einsteiger sind die FORTH-Experten von Morgen. Das gilt aber nur dann, wenn die 'alten' FORTHler ihren ohnehin noch spärlich nachrückenden 'Kollegen' den Weg ein wenig ebnen. Das Interesse an FORTH ist größer, als der allgemeine Bekanntheitsgrad vermuten läßt. Aber wer sich als Interessent allein gelassen fühlt, der wendet sich in den allermeisten Fällen, wie schon Unzählige vor ihm, an einen der vielen Basic, Pascal und Modula Clubs.

Die FORTHler in Moers sind dabei noch recht gut 'bedient'. Mit allwöchentlichen Treffen in 'eigenen' Clubräumen, mit privaten Kontakten der Gruppenmitglieder untereinander, die bei 'dringenden' Problemen auch spätabendliche Telefonate nicht übel nehmen, mit einem eigenen Kursangebot und engagierten Mitglieder die einen großen Teil ihrer Freizeit in die Gruppe einbringen, lassen sich viele der aufgezeigten 'Probleme' meistens so weit abschwächen, daß allgemein der Spaß an der Maschine Computer und an FORTH erhalten bleibt.

Wir wissen, daß das nicht überall so ist, und daß noch viel zu wenig Gruppen existieren. Gerade deshalb werden vor allem jüngere Mitglieder der FORTH-Gesellschaft unsere Kritik verstehen und nachvollziehen können. Um wieder auf das eingangs erwähnte Gespräch zurück zu kommen, möchten wir an dieser Stelle anregen, diesen 'Bericht' in der nächsten 'Vierte Dimension' zu veröffentlichen und damit eine Diskussion zwischen *allen* FORTHlern zu beginnen. Wir grüßen ganz herzlich vom linken Niederrhein

Martin Pannenbecker	Martin Wissusek
Thomas Peters	Ulrich Prinz
Bernd Feuerer	Michael Major
Friederich Prinz	Hans Chrapia
Karl Schroer	Christopher Weber
Karl-Peter Purmann	Ulrich Feek

sowie die Mitglieder der Moerser Gruppe, die (noch) nicht in der FORTH-Gesellschaft organisiert sind.

Volker Götiger	Dirk Wissusek
Willi Wissusek	Hans Büren
Detlef Malbrecht	Tobias Malbrecht
Udo Sattler	

Das FWD-Team Ludwig Richter bietet:

Kommunikation Automation Meßwerterfassung Industriesteuerungen
Hard/Soft-Lösungen, auch unorthodoxe
Vollständige Lösungen vom Sensor über die Elektronik, bis zum Aktor

Unsere Effizienz und Flexibilität beruht auf den Faktoren:

- Wir sind ein Team aus Spezialisten verschiedener Richtungen
- Wir nutzen das Know-How auf Ihre Anwendungen bezogen
- Wir regen Ihre Mitarbeiter und Maschinen an
- 10 Jahre Erfahrung Assembler — PC 68000 Z80 6809 6502
- 10 Jahre Erfahrung physikalische Meßtechnik / Nachrichtentechnik
- 7 Jahre Erfahrung Forth — Metacompiler UR/FORTH PC/FORTH CFORTH
- 5 Jahre Erfahrung analoge und digitale Schaltungsentwicklung
- 3 Jahre Erfahrung C — PC AMIGA ATARI Crosscompiler

FWD-Team
Bernhard Emese Reinhard Fenger Jürgen Gerhard Ludwig Richter

Ludwig Richter, Essenheimerstr.94, D-6500 MZ-Bretzenheim, Tel.06131-368274

volksFORTH-83

Lokale Variablen

von Rolf Lehnhardt

Quelltext Service

Inspiriert von den zwei Versionen von lokalen Variablen in der Vierten Dimension Juni 1990 entwickelte ich eine Version für das volksFORTH-83. Leider ist sie 9 Screens lang geworden, aber Luxus und Sicherheit haben ihren Preis.

Diese Realisierung von lokalen Variablen wurde in der IBM-PC Version entwickelt, sollte aber auch auf den anderen Versionen lauffähig sein.

Die drei Worte die während Run-Time aufgerufen werden sind kurz und können wohl recht einfach in den jeweiligen Code übertragen werden.

Nur drei Worte verbleiben nach der Kompilation von LOCAL.SCR aufrufbar.

Dieses sind:

LOCAL

Verhalten während Compile-Time

Der Zustand der momentane Kompilationsumgebung (DP, CURRENT, CONTEXT) wird gespeichert. Danach wird eine neue Kompilationsumgebung gesetzt. Sämtliche Worte die nun zwischen LOCAL und ENDLOCAL angelegt werden, landen automatisch im Vokabular VARVOC und in einem Speicherbereich, der durch LVDIC definiert ist. Des weiteren wird (LOCAL ins Wort kompiliert und mit dem Wert Null initialisiert. Es wird eine erweiterte Fehlerroutine installiert, um bei einem Fehler die alte Kompilationsumgebung einrichten zu können (sonst droht das Nirwana).

Verhalten während Run-Time

(LOCAL erledigt während Run-Time zweierlei: Zuerst allokiert es einen Speicherbereich (und meldet sich, wenn zuwenig Speicher vorhanden ist) für die im Wort vorkommenden lokalen Variablen. Des weiteren manipuliert es den Returnstack in der Form, daß bei der Beendigung des Wortes ein »Aufräum-Wort« aufgerufen wird, welches den belegten Speicherbereich wieder freigibt.

Über den Autor

Anschrift:	Rolf Lehnhardt, 5427 Bad Ems, Pfingstwiese 14,
Tel.:	(02603) 124 61
Alter:	33 Jahre
Familie:	verheiratet, eine Tochter
Beruf:	Krankenpfleger
Hobby:	FORTH

LOCVAR

Verhalten während Compile-Time

Es wird ein Wort in die durch LOCAL bestimmte Kompilationsumgebung kompiliert. Dieses Wort ist immediate und kompiliert, wenn es außerhalb von LOCAL/ENDLOCAL aufgerufen wird, (LOCVAR ins Wort und gibt ihm einen Offset mit auf den Weg.

Verhalten während Run-Time

(LOCVAR ermittelt anhand des aktuellen Speicherbereich für lokale Variablen und dem Offset (auf den er ähnlich wie LIT zugreift) die Variablenadresse und legt sie auf den TOSS.

INSERENTENVERZEICHNIS

Firma	Seite der Anzeige
Reilhofer KG, Karlsfeld	2
EDV-Beratung - Software-Design - Goppold, Poing	2
FORTecH Software, O-Rostock	5
MICROPROCESS GmbH, Schriesheim	7
FWD-Team, Ludwig Richter, MZ-Bretzenheim	14
Bernd Paysan, 8000 München 71	21
Angelika Flesch, FORTH-Systeme, Breisach	40

Lokale Variablen

ENDLOCAL

ENDLOCAL setzt wieder die normale Kompilationsumgebung und teilt (LOCAL mit, wieviel Speicherplatz es während Run-Time allokieren soll. Zur Sicherheit wird (wie z.B. bei IF THEN) geprüft, ob zuvor auch ein LOCAL aufgerufen wurde. Die Worte, die zwischen LOCAL und ENDLOCAL definiert wurde werden »ausgelinkt« und fallen dem Vergessen anheim.

Lokalen Variablen sind recht schnell und schränken Schleifenstrukturen wie z.B. DO/LOOP nicht ein.

Sicherlich ist manches Wort mit lokalen Variablen schneller entwikkelt, getestet und ist wohl auch besser lesbar als undurchsichtiges hin- und herge"DUP"e und ge"ROT"e.

Quellen

- Vierte Dimension, Ausgabe Juni 1990
- Einführung in FORTH 83, Manfred Mader
- Sourcecode des volksFORTH-83

Rolf Lehnhardt

```
\ Lokale Variablen                                          RL 06Sep90

Lokale Variablen können zwischen den " Kernwörtern "
LOKAL  -  ENDLOCAL definiert werden, sind innerhalb
des Wortes namentlich aufrufbar , und geben den von
ihnen belegten Speicherbereich nach Beeendigung des
Wortes frei.

Worte mit lokalen Variablen sind uneingeschränkt
schachtelbar , es sind dem Programm nur die aktuellen
lokalen Variablen bekannt.
```

```
\ LOKALE VARIABLEN                                          RL 06Sep90


empty
decimal
onlyFORTH

2 11 thru
```

```
\   VARIABLEN                                               RL 09Sep90

1 ?head !  \ alle Header auf den Heap

Variable lvdp
Variable lvtop
Variable lvbot
Variable lvpnt
Variable lvlast
Variable (lverror
Variable (concur 2 allot

Vocabulary VARVOC
```

```
\  HILFWÖRTER LVALLOT,DICSWAP,LVDIC                         RL 09Sep90

   50 hallot here heap dp !
\  dieses Wort wird nur während der Kompilation gebraucht

: lvallot ( 16b -- )
     here dup lvbot ! lvdp ! allot here 2- dup lvpnt ! lvtop !
;
   dp !


: dicswap ( -- )
     dp lvdp  @ swap @ lvdp ! dp !
;

Variable lvdic  $100 lvallot
```

```
\  LVPNT< , (LOCAL  ( HIGH-LEVEL )                         RL 09Sep90


: lvpnt<  lvpnt @ @ lvpnt ! ; immediate

: lvspace?
     dup lvbot @ < IF
                   abort" Lokale Variablen : Speicher voll "
                 THEN
;
: (local  ( -- )
     R> dup @ swap 2+ ['] lvpnt< >body >R >R
     lvpnt @ dup rot - dup lvspace?
     lvpnt !  !
;
```

```
\   LVPNT< , (LOCAL  (  8086 Code  )                        RL 09Sep90
\\
code lvpnt<
  here , here ,
  lvpnt #) W mov  W ) W mov  W lvpnt #) mov
  R ) I mov   R inc   R inc  Next
end-code

code (local
   1 ) A mov I inc I inc
   R dec R dec ' lvpnt< >body #  R )  mov
   lvpnt #) W mov  W C mov  A W sub
   lvbot #) W cmp cs ?[ ;c:
   abort" lokale Variablen : Speicher voll " ; Assembler ]?
   W lvpnt #) mov   C W ) mov Next
end-code
```

```
\  LVOFF , LVERROR , ERRORINIT                              RL 09Sep90
: lvoff  ( -- )
    lvlast @ last ! (concur 2@ current ! context !
    (lverror @ errorhandler !
;

: lverror  ( -- )
    lvoff  hide 00 ['] varvoc >body !
    lvbot @ lvdp !  errorhandler perform
;

: errorinit ( -- )
     errorhandler @ ['] lverror - not
       IF errorhandler @ (lverror ! ['] lverror errorhandler !
       THEN
;
```

```
\  LVON , LVINIT , LVERROR                                  RL 06Sep90

: lvon  ( -- )
   last @ lvlast !  reveal  context @ current @ (concur 2!
;

: lvinit  ( -- )
   ['] varvoc >body context ! 00 ['] varvoc >body !
   lvbot @ lvdp ! errorinit
;

: lverror? ( -- )
   errorhandler @ ['] lverror -
     IF abort" : Aufruf innerhalb von Local\Endlocal "
     THEN
;
```

```
\  (LOCVAR (HIGH-LEVEL/CODE) , LVCREATE                     RL 09Sep90

: (locvar  ( -- ) R> dup @ swap 2+ >R lvpnt @ + ;

\  code (locvar    \\ 8088 Code
\  D push   I ) D mov  I inc  I inc  lvpnt #) D add  next
\  end-code


: Lvcreate (    )
   dicswap create immediate 2+ dup , dicswap
;
```

```
\  LOCVAR , LOCAL , ENDLOCAL                                RL 09Sep90

?head off

: Locvar  ( -- )
    lvcreate  does> lverror? ['] (locvar , @ ,
; immediate restrict


: local  ( -- )
   lvon lvinit  &05  ['] (local , here $0000 $0000 ,
;  immediate restrict

: endlocal  ( -- )
   rot 5  ?pairs  lvoff 2+ swap !
;  immediate restrict
```

LINKED ACTIONS

Teil II

Listen, Listen und nochmals Listen

von Jörg Plewe, Frank Stüss, Jörg Staben

Quelltext Service

Die allgemeine Lösung für FORTH-Worte in Listen (siehe Quelltext 2)

Die neue Syntax

Die bisherige Lösung ist zwar ein brillantes Feuerwerk der FORTH-Programmierung, aber es werden für *eine* neu angelegte Liste *drei* zugehörige Prozeduren ins Dictionary eingetragen; da muß es doch möglich sein, einen allgemeinen Listentyp einzuführen, der dann mit drei, vier allgemeinen Zugriffsprozeduren auskommt. Damit ändert sich auch die Syntax, deren neue Form wir hier an einem Beispiel vorstellen möchten: Wie oben beschrieben, führen Programme beim Booten oftmals Initialisierungen aus. So könnte es notwendig sein, beim Betriebssystem zusätzlichen Speicher anzufordern. Also wird den Aktionen, den *BOOT-ACTIONS*, ein Wort `ALLOCATE_MEMORY` hinzugefügt:

```
List: boot-actions
boot-actions add: allocate_memory
```

Das gleiche gilt für die Aktionen beim Verlassen des Programms: Hier muß, zusätzlich zu `BYE`, der angeforderte Speicher wieder an das Betriebssystem zurückgegeben werden:

```
List: bye-actions
bye-actions add: free_memory
bye-actions add: bye
```

Hier werden zwei Lösungen – eine statische und eine dynamische – vorgestellt mit geringfügig unterschiedlicher Syntax. Grundsätzlich gilt, leider im Gegensatz zur UPN:

```
<list name> <operation:> <wort name>
```

Der Bezugspunkt für die statische Liste ist das definierende Wort `LIST:`, das eine Größenangabe (max. 255) übernimmt und einen Namen erwartet.

`<size> List: <name>`
zum Neuanlegen einer statischen Liste

Dagegen ist der Bezugspunkt für die dynamische Liste eine simple Variable, die dadurch den Anker der Liste darstellt.

`Variable <name>`
zum Neuanlegen einer dynamischen Liste

Hier kann man die Syntax angleichen durch ein

```
' Variable Alias List:
```

Es stehen folgende Worte zum Handhaben von Listen zur Verfügung:

`<liste> add: <name>`
zum Hinzufügen eines Wortes

`<liste> act`
zum Ausführen eines Wortes

`<liste> show`
zum Anzeigen eines Wortes

`<liste> kill: <name>`
zum Löschen eines Wortes

`ACT` als Name für das Ausführen gefällt keinem der Autoren, aber `PERFORM` ist schon vergeben. Vorschläge dazu werden gerne entgegen genommen.

Leider wird bei diesem zweiten Listenkonzept eine erklärte Eigenschaft der FORTH-Worte geopfert – ihr implizites Ausführen durch Aufrufen des Namens. Um kompa-

Stichworte

- ✗ objektorientierte Programmierung,
- ✗ Listen,
- ✗ Rekursion

tibel zu bestehendem Code zu bleiben, müßte man im Fall von (DECODE dann

```
: (decode   (decode act ;
```

definieren.

Die statische Liste

Die Struktur

Um die Gliederungsmöglichkeiten von FORTH zu nutzen, werden – zumindest für die Testphase – Vokabulare angelegt, da sich die Zugriffe für statische Listen und dynamische Listen inhaltlich stark unterscheiden. Auf weitergehende Ordnungsmöglichkeiten innerhalb der Sprache werden wir später noch detailliert eingehen.

Die statische Liste entspricht einem Feld, einem Vektor mit den dafür üblichen Zugriffsmethoden. Die Grösse des Feldes wird beim Definieren der Liste festgelegt und in ein führendes Countbyte eingetragen; damit ist die Anzahl der Worte in einer solchen Liste begrenzt.

Dieses Konzept ist insofern einfacher als das der dynamischen verketteten Liste, da sich das Feld (= die Liste) viel einfacher manuell handhaben läßt. DUMP, @ und ! lassen sich in gewohnter Form einsetzen; als Zugriffsmethode reicht eine schlichte DO . . . LOOP-Schleife.

Die Liste ist mit NOOP's vorbesetzt, um alle Feldelemente auf jeden Fall (mit PERFORM) ausführen zu können. So kennzeichnet ein NOOP einen leeren Listeneintrag. Eine andere Möglichkeit wäre es gewesen, das Feld mit 0-Bytes vorzubesetzen. In diesem Fall hätte das PERFORMende Wort prüfen müssen, ob das jeweilige Listenelement leer oder gültig ist.

Die Zugriffsmöglichkeiten

Hinzufügen eines Elementes ADD:

Bei dieser Listenstruktur ist es notwendig, *vor* einem neuen Eintrag in die Liste diese zu verdichten. Damit wird verhindert, daß eine neue Aktion zu einem undefinierten Zeitpunkt ausgeführt wird. Dies ist wichtig, wenn Argumente an die einzelnen Listenelemente übergeben werden sollen, z.B.:

```
: status  status off  ... ;
```

Bei diesem Zusammenschieben erinnert eine rekursive Funktion daran, daß die Rekursion eine der Stärken von FORTH ist. Diese Funktion schiebt das gesamte Feld um eine Position nach unten, wenn das angetroffene Listenelement ein NOOP ist. Ist dies der Fall, wird die oben freiwerdende Position mit NOOP aufgefüllt. Damit ist sichergestellt, daß die Worte in der Reihenfolge des Einhängens ausgeführt werden.

```
: (compact   ( pfa[liste] size --)
   recursive
   dup 1 > IF  2dup
      get_parmsrecursion (compact
      over dup cell+ @ ['] noop <>
      swap          @ ['] noop = and
      IF shift1 ELSE 2drop THEN
   ELSE 2drop THEN ;
: compact ( pfa[liste] -- )
   count (compact ;
   \ Größe ist als Rekursionsgrenze
   \ nötig
```

Das Hinzufügen eines Elementes mit ADD: bedingt die Verwaltung der Listenlänge auf dem Returnstack mangels lokaler Variablen in FORTH:

```
: add:  ( listadr |  <name> --)
   ...
   count >r
   BEGIN  ...
   WHILE  ...
          ... counter @ r@ = ...
   REPEAT ... rdrop ;
```

Hier könnte man die LOCALS des ANSI-FORTH schön einsetzen, damit diese Schreibweise möglich wäre:

```
: add:
   count  local size ...
```

Noch schöner wäre natürlich **count LOCAL: size**, um den Vorwärtsbezug von LOCAL auszudrükken. Ach ja, die Syntax einer Sprache!

ACT als Assemblerprogramm

Alles in allem ist der Einsatz dieser neuen Listenwörter zeitunkritisch. Lediglich das Ausführen selbst sollte den völlig unbegründeten Ruf vom schnellen FORTH weiter fördern; und weil kaum eine andere Sprache eine so schöne Schnittstelle zur Maschinensprache hat, wurde die freundliche FORTH-Form von ACT durch eine Assembler-Routine namens ACTLOOP ersetzt, die Tabellen von CFAs ausführt:

```
also
Code ACTLOOP ( addr len --)
     D C mov  \ Anzahl der Adressen
     I D mov
     I pop    \ Basisadresse des
              \ auszuführenden Feldes
     [[
        lods     A W mov
          \ Lade Adresse
        I push   C push  D push
Label returnjmp
        $5555 # I mov \ wird gepatcht
        W ) jmp
          \ ...irgendwann ein UNNEST
        here returnjmp 1+ !
          \ welches hierhin springt
        here 2+ ,  here 2+ ,
          \ cfa = here
        D pop    C pop    I pop
          \ Register auf alten Stand
     c0= ?]
     D I mov    D pop
Next end-code
previous
```

Das Assemblerwort ACTLOOP könnte sofort in der dargestellten Form für die Ausführung der statischen Liste eingesetzt werden, aber es wäre doch viel zu schade, sich nicht den Schmelz kühler Assemblerprogrammierung auf der Zunge zergehen zu lassen. Denn so ohne weiteres ist es nicht möglich, ein Execute aus reinem Maschinencode heraus zu machen. Um den hier notwendigen Übergang von Highlevel zu Lowlevel zu ermöglichen, war ein kleiner Kunstgriff nötig: Zuerst wird der Instruction Pointer IP mit einer Listen-Adresse geladen, die nur einen Eintrag hat, nämlich die darauffolgende Adresse. Diese dient eben dann als CFA und hat abermals die darauffolgende Adresse als Inhalt. Dort beginnt dann der weitere Maschinencode. Es wird auf den weiteren Maschinencode genestet, jedoch niemals zurückgekehrt:

```
here returnjmp 1+ !   here 2+ ,
here 2+ , ...
```

Das Grundprinzip dieser Lösung entspricht also dem des *inneren Interpreters*, der FORTH-Worte ausführt, indem er die in ihnen enthalten CFAs exekutiert. Die einzelnen Worte werden dann durch NEST angesprungen und durch UNNEST zurückgekehrt. Deshalb der zweifache Verweis auf die nächstfolgende Adresse: Einmal als Quasi-CFA, der andere als Quasi-NEST. Die hier vorgestellte Struktur ähnelt einem Assemblerwort, einem Primitiv, in dem die CFA direkt *hinter sich* auf ausführbaren Code zeigt. Das hier vorgestellte Verfahren kann auch

benutzt werden, um anderweitig FORTH-Worte aufzurufen, etwa in dieser Struktur:

```
Code <NAME>
   (Assemblercode...)
 High-level:
      \ Reg retten,IP mit dummy
      \ setzen, Compiler anschalten
   (Forth-Code...)
 ;Low-level
      \ Compiler ausschalten, dummy
      \ korrigieren
   (Assemblercode...)
 Next
end-code
```

Diese ganze Trickserei ist nur bei FORTH-Systemen nötig, die mit den einander sehr ähnlichen Aufbau-Formen [5] indirect-threaded Code ITC und direct-threaded Code DTC arbeiten, weil die CFA hierbei nicht direkt die Adresse von ausführbarem Maschinencode ist. Dagegen können Systeme, die mit subroutine-threaded Code STC arbeiten, direkt solche Tabellen ausführen, da solche STC-Systeme prinzipiell Lowlevel arbeiten.

Beim Einsatz von ACTLOOP trat ein kleines Problem im Umgang mit den Vokabularen heraus: Der Übersichtlichkeit halber sollten alle Worte im Vokabular STATIC definiert werden. Dadurch aber, daß Code den CONTEXT auf Assembler umschaltet, sind die Worte des Vokabular STATIC nicht mehr sichtbar und FORTH fragt ganz vorsichtig nach, wo denn das Label RETURNJMP aufzufinden sei. Daher die Überlegung, ob CODE nicht vielleicht ein ALSO und END-CODE ein TOSS/PREVIOUS machen sollte ?

Löschen eines Elementes
KILL:

Das Löschen einer Aktion ist ausgesprochen einfach, da die statische Liste als lineares Feld eine einfache Datenstruktur ist. Mit der hier eingesetzten sequentielle Suche nach einem Element werden die Operationen abgeschlossen, die bei einem Feld aus Prozeduren sinnvoll sind. Die vor allem bei großen Datenmengen wesentlich effizientere binäre Suche kommt hier glücklicherweise nicht in Frage, weil dafür unsere Aktionsliste sortiert vorliegen müßte. Und BYE OFF STATUS ist mit Sicherheit nicht das, was wir möchten. Somit ist das hier vorgestellte lineare Datenfeld die ideale Struktur für unseren Einsatzzweck; die Nachteile werden erst deutlich, wenn viele Einfüge- und Löschoperationen in der Liste erforderlich sind, abgesehen von der einmal festgelegten Größe.

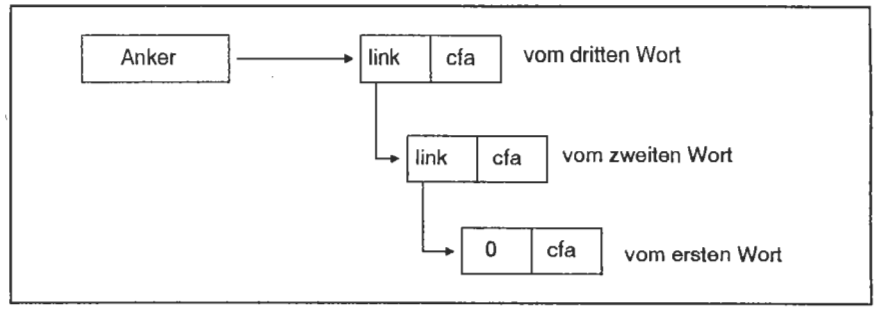
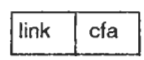


Bild 1: Graphische Darstellung

Die dynamische verkettete Liste DLL

Die Struktur

Ein Knoten der hier eingesetzten DLL (dynamic linked list) besteht aus dem Zeigerfeld (link) und dem Informationsteil, der in unserer Anwendung die CFA des auszuführenden Wortes enthält. Die Struktur eines Listenelementes stellt sich grafisch so dar:



Bezugspunkt für eine solche Liste und damit ihre *Identifikation* ist immer ihr Anker – womit die Liste dann wie in Bild 1 gezeigt aussieht.

Einhängen eines Elementes
ADD:

In diese Liste werden die neuen Worte immer vorne eingehängt, der Anker enthält immer den Zeiger auf das zuletzt eingefügte Wort. Da ein solcher Knoten der Liste wie ein Mini-Record aussieht, hätten wir gerne mit **node link** oder **node data** auf die jeweiligen Informationen zugegriffen, um das *forthige* **cell+** zu ersetzen. Aber da es ADD: gelingt, einen Knoten namenlos ins Dictionary einzutragen, bleibt es halt beim **cell+**. Sollte sich dennoch jemand an dieses Thema heranpirschen wollen, die Mini-Records findet man in letzter Zeit häufig als **Field:**, die dann so eingesetzt werden:

```
: Field:     ( offs1 len -- offs2)
    Create  over , +
    Does>    ( addr field )  @ + ;
 0    2 Field: link
      2 Field: data
  Create NODE   allot
 0    2 Field: link
      2 Field: data
  Constant size
  Create NODE  size allot
```

ACT - vorwärts und rückwärts

Zum Abarbeiten der Liste kann man nun den ersten Link aus dem Anker holen und die Liste der Links absuchen, solange der Link nicht 0 ist - denn die Null ist das Zeichen für das Ende der Liste. Ist der Link nicht 0, dann wird zur zweiten Zelle (cell+) des Listenelementes gesprungen und die darin enthaltene CFA ausgeführt. Aber würde man die Liste in dieser Form ausführen, so käme das zuletzt einhängte Wort zuerst zur Ausführung. Dieses Verhalten hätte bei der Wortfolge STATUS OFF BYE bestimmt großes Erstaunen, aber keine Begeisterung für die hier vorgestellten Konzepte zur Folge. So ist für dieses Problem eine Rekursion die natürliche Lösung und damit schimmert fast ein Hauch von Eleganz durch den Code:

```
| : (act    ( firstlink -- ) recursive
    dup >r  @ IF      r@ @ (act
              THEN    r> cell+
                      perform ;
  : act     ( *firstlink --)
  @ (act ;
```

BACK arbeitet, wie bereits ausgeführt, die Liste von hinten nach vorne ab und verhält sich damit LiFo-typisch:

```
: back    ( *firstlink -- )
    @  BEGIN
          ?dup      \ enthält der
                    \ Zeiger 0 = nil?
       WHILE
          dup >r  cell+ ( *cfa)
          perform \ perform
                  \ 'link-action'
          r>  @   \ holt Adresse
                  \ des nächsten
                  \ Zeigers
       REPEAT
;
```

Die zwanglos eingestreuten to-r's erwiesen sich als nötig, um weiterhin die Parameterübergabe zwischen den FORTH-Worten zu ermöglichen, denn wir wollen ja nicht auf `STATUS OFF` verzichten! Von der Syntax her schön wäre eine Lösung, in der **`backward list`** als definierendes Wort einer Liste die Ausführungs-Methode **`backact`** einträgt, wogegen **`forward list`** natürlich **`act`** einträgt. Wir werden schauen, ob wir so etwas noch anbieten können.

Die Rekursion

Die Berechnung der Fakultät ist ein traditionelles Beispiel für den Selbstaufruf einer Prozedur in FORTH. Da dieses Beispiel auch in Lehrbüchern anderer Sprachen häufig angeführt wird, sei es hier in Listing 1 noch kurz wiedergegeben.

Wir haben noch mehr zu vererben! (siehe Quelltext 3)

Abstrakte Datentypen (ADT)

Ganz zu Beginn unserer großen FORTH-Rundreise haben wir die Ordnungsmöglichkeiten eines FORTH-Systems über Vokabulare kurz gestreift.

Denn normalerweise sind alle FORTH-Worte, gleich ob als Prozeduren oder Daten, im Dictionary sichtbar und damit einem globalen Zugriff ausgesetzt. Durch den Einsatz von Vokabularen erfährt dieser Zugriff die erste Einschränkung. Durch die Nennung des Vokabulars `DYNAMIC` wird der `CONTEXT` auf die dazugehörigen Worte gesetzt. Damit sind alle Worte sichtbar, die in diesen Zusammenhang gehören. Dennoch ist über eine Verkettung von Vokabularen ein falscher Kontext für Zugriffe denkbar, so daß ein STATIC ADD: auf eine DLL angewendet wird und der Rechner FORTH-typisch mit Stillstand reagiert.

Ein sicherer Weg, den globalen Zugriff auf Worte zu begrenzen, ist die Einführung von abstrakten Datentypen (ADT), die den Komfort strukturierter Daten mit größerer Zugriffs-Sicherheit verbinden. Ein ADT enthält nicht nur die Struktur der Daten, die manipuliert werden, sondern auch die Operationen, die auf diese Struktur angewendet werden können. Mehr noch – bei abstrakten Datentypen bleibt die physikalische Darstellung der Daten vollständig verborgen; auf die Daten kann nur über die zugeordneten Operationen zugegriffen werden. Bei den hier vorgestellten Listen wären nur die Operationen `KILL:`, `ADD:` und `SHOW` zugelassen, andere Operationen auf eine solche Liste werden abgewiesen. Auch die Art, wie die Liste angelegt ist, ist verborgen. Die Liste kann ein statisches Datenfeld sein oder eine dynamisch verkettete Liste – alles was man über eine Liste wissen muß, sind die darauf anwendbaren Operationen.

So bieten ADTs eine Reihe von Vorteilen:

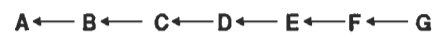
- ✗ Änderungen im Programm bringen keine Nebeneffekte mit sich.
- ✗ ADTs vermitteln Sicherheit. Durch die Beschränkung auf Datentyp-spezifische Operationen werden die meisten Fehler schon während des Kompilierens, zur Compilezeit, aufgedeckt.
- ✗ umfassendere Möglichkeiten des Programmentwurfs. Instanzen eines ADTs werden oftmals als Objekte angesehen, die über bestimmte Attribute (Daten) und ein bestimmtes Verhalten (Operationen) verfügen.

Kapselung

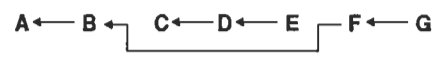
Programmieren in FORTH ist von Grund auf objekt-orientiert, weil die Unterprogrammeinheiten, die Worte, sowohl Daten als auch Anweisungen enthalten können und eigenständig ausführbar sind.

Darüberhinaus verfügt FORTH mit dem Vokabularkonzept über einen Mechanismus des Verbergen von Informationen, Information-Hiding.

Allgemein bedeutet die *Kapselung* einer Struktur in FORTH, daß sie im Wörterbuch nicht mehr sichtbar ist. So geht dieser Ansatz davon aus, daß die Listen von einem eigenen Vokabular *wissen*, das ihnen angehört. Wenn sie aufgerufen werden, wird dieses Vokabular automatisch geöffnet, die Zugriffsmöglichkeiten damit sichtbar und das Vokabular nach dem Zugriff wieder geschlossen. So werden *private Dictionaries* geschaffen, die in der normalen Suchreihenfolge unsichtbar sind. Ein Diagramm macht das deutlich:



Die Worte A und G sind verbunden. Denn das Link-Feld von G zeigt auf F; das Link-Feld von F zeigt auf E usw.




```
   : .last     last ( *nfa)  @ ( nfa)  .name  ;
\ : myself    last @  name> , ; immediate
   : myself              last' , ; immediate
: fakultät  ( +n -- n! )
    dup 0< IF  ." keine negativen Argumente erlaubt! "
           ELSE
            ?dup 0= IF  1                          \ Spezialfall: 0
                   ELSE  dup 1-  myself *
                   THEN
           THEN ;
cr  5 fakultät .    .( Muss 120 sein! ) cr
\ RECURSIVE                                              jrg 09jan89
: fakultät  ( +n -- n! )
   recursive
     dup 0< IF drop ." keine negativen Argumente! " exit  THEN
    ?dup 0= IF 1                          \ Spezialfall: 0
           ELSE  dup 1-  fakultät *
           THEN ;
```

Listing 1

Biegt man nun das Linkfeld von F auf B um, sind C, D und E versiegelt und können im Dictionary nicht mehr gefunden werden. Damit ist auch kein Zugriff auf diese Worte mehr möglich.

Das neue TYPE>

Entsprechend dem Beispiel aus dem Artikel in der letzten 'Vierten Dimension', kann man mit unserem `<Type:` die komplexe Zahl mit Realteil/Imaginärteil und den Zugriffen Einspeichern/Auslesen als *Typ* einführen, deren Instanzen schon ziemlich gekapselte Objekte sind.

```
<Type: COMPLEX:
2 Var: real
2 Var: imag
: !!  ( real imag --)
   dup imag rot swap !  real !  ;
: @@  ( -- real imag)
   dup real @  swap imag @  ;

Type> COMPLEX:
Complex: x
       \ X ist eine Instanz vom Typ
       \ COMPLEX:
50 100 x !!
x u?   \ gibt: "Operator unbekannt"
x @@ . .
```

Soweit, so gut. Aber trotz aller Vorzüge (Laufzeiteffizienz, Speichereffizienz, transparente Syntax) gibt es noch eine Reihe von Nachteilen:

- ✗ Die Kapselung ist nicht vollständig. Dadurch, daß die Instanzvariablen `REAL` und `IMAG` im privaten Wörterbuch sichtbar sind, kann man global mit `x real .` direkt auf die physikalischen Adressen zugreifen. Dieses Manko verletzt zwar das Prinzip des »Information Hiding«, ist aber durch den Headerless-Mechanismus leicht zu beheben. Ein `clear` und es hat sich ausgegriffen.
- ✗ `@@` und `!!` wirken durch `dup rot swap` unschön und unleserlich, weil die Basisadresse des Objektes über den Stack übergeben wird. Bedenken Sie, es gibt noch viel komplexere Objekt-Typen als `COMPLEX:` !
- ✗ Objekt-Typen, die auf die hier vorgestellte Weise definiert worden sind, lassen sich nicht verschachteln. Sie können also keinen neuen Typ deklarieren, der seinerseits `COMPLEX:` enthält.

wenn forth zu klein ist:

(für alle Atari ST)

bigFORTH hat eine lange Entstehungsgeschichte hinter sich. Seinen Ursprung hat es im PD-System volksFORTH. Ziel der Entwicklung war ein FORTH-System, das in seiner Leistungsfähigkeit nicht hinter modernen und professionellen Compilern für "konventionelle" Sprachen zurücksteht. Dieses Ziel kann man als erreicht ansehen, wie ein Blick auf einige der Features sicher zeigt:

- **Mächtiger Compiler:**
  32-Bit-Stacks, erzeugt optimierten 68000-Native-Code, relokatibles System (ohne Einschränkungen!), weitgehend FORTH-83-kompatibel, unbeschränkter Adreßraum, nutzt Screen- und Stream-Files, relokatible Turnkey-Applikationen (denen man FORTH nicht mehr ansieht) sind problemlos machbar...

- **Vielseitige Tools:**
  Assembler, Disassembler, Multitasker, sourcefähiger Decompiler (decompiliert alle Optimierungen), Source-Level-Debugger (Single-Step und Trace, beliebig viele Breakpoints), Post-Mortem Dump und Returnstack-Trace bei allen Fehlern (Ausgang über ABORT"), resetfest (für den Ausstieg aus der Endlosschleife), Multiwindow-Editor (natürlich unter GEM), beliebiger anderer Editor nutzbar...

- **Umfangreiche Libraries:**
  Komfortables Fileinterface, Druckertreiber, sämtliche TOS-Funktionen (GEMDOS, BIOS, XBIOS, GEM-AES, GEM-VDI und Line-A-Grafik), Floating-Point-Arithmetik, leistungsfähiges Memory Management, High-Level-GEM-Libraries, Turtle Graphic und mehr...

- **Transparenz:**
  Sämtliche Sourcen (einschließlich Kernal) auf Diskette, Target-Compiler wird mitgegeben...

Die ausführliche deutsche Dokumentation (über 250 Seiten Handbuch) wird Anfängern sicher ihre ersten Schluckbeschwerden bei dem großen Bissen beseitigen, aber auch alte Hasen werden ihren Nutzen daraus ziehen...

Und das alles nur für
DM 200.- (incl. Mehrwertsteuer und Versand)

Für Vorsichtige gibt es eine frei kopierbare Demoversion (ohne Sourcen&Handbuch), für eine Schutzgebür von
DM 10.- (incl. Versand)

Bestellungen bitte schriftlich an:
Bernd Paysan
Stockmannstr. 14
D-8000 München 71

Und zum guten(?) Schluß: Sollte Ihnen die Syntax aus [6] besser gefallen, so benennen Sie bitte `Type>` in ein `End-type>` um. Aber das ist ja gerade das Schöne in FORTH: Hier kann jede(r) machen, was er/sie will – denn der Compiler ist Wachs in unseren Händen!

Schöne Aussichten: Immer weitere Probleme

Handarbeit in FORTH

Wie viel weiter oben im Leben von `(DECODE` dargestellt, hat der Anwender von FORTH sehr viel harte Arbeit, wenn ein Wort geändert wird, das in vielen anderen Worten benutzt wird. So führte die Redefinition von `(DECODE` zugleich auch zur Redefiniton von `(EXPECT` und zur Redefinition von `KEYBOARD`.

Dieses Problem wird z.T. dadurch gemildert, daß in der Listenlösung von `(DECODE` immer die Liste mit dem aktuellen Inhalt ausgeführt wird.

Lösungsansatz:

```
<list> replace: <act_old> <act_new>
```

Die Vorgehensweise dürfte beim `KILL:` der statischen Liste zu finden sein ...

Kann man das FORGET vergessen?

Was ist, wenn durch ein unüberlegtes `FORGET` Listenelemente einer Liste vergessen werden? Eine gute Frage – ist `custom-remove` die Antwort?

Es geht immer weiter

Dieser Ansatz kann auch zu einem neuen Konzept der `METHODS>` führen. Die Methoden wären dann kein Feld mehr, sondern auch eine Liste, um späteres Hinzufügen (und Löschen) von Methoden zu ermöglichen. Dies ist ja zur Zeit nicht möglich; allerdings gibt's da ein kleines Handikap, weil wir gegen die UPN verstoßen haben:

Bei `METHODS>` ist die Syntax:

```
<value>   <destination>   <operation>
5000      members         put
```

Die Syntax für die Listen sieht (zur Zeit) so aus:

```
<destination>   <operation>  <value>
members         put          5000

<list>          <operation>  <value>
cold            add:         logo
```

Sonst müßte man ja wieder mit `'` und `[']` und wer weiß was allem arbeiten.

Ein anderes Einsatzgebiet wären die User-Variablen eines Systems, die man auf diese Listen-weise dynamisch handhaben könnte.

Die Literaturhinweise entnehmen Sie bitte dem Artikel in der letzten 'Vierten Dimension'. Außerdem konnte aus Platzgründen der Quelltext zu diesem Artikel nur unvollständig abgedruckt werden. Interessenten wenden sich bitte an den Quelltextservice.

Autoren
✉ Jörg Plewe,
Großenbaumer Str.27
4330 Mülheim/Ruhr,
☎ 0208-423514
✉ Frank Stüss,
An der Turnhalle 6
6369 Schöneck 2,
☎ 06187-91503
✉ Jörg Staben,
Hagelkreuzstr.23
4010 Hilden,
☎ 02103-55609

SCR 01

```
\ loadscreen für statische Listen                          jrg 25Jul90

  System-on
  empty
  2 load
    3  12 thru
\    13 19 thru
  save
```

SCR 02

```
\ dummy actions  1NC                                       jrg 25Jul90

  OnlyForth  also definitions


  : inc   ( adr --)  1 swap +! ;

\ So macht's ein 4TH-Hobbyist:
  : eins        ." Action 1   "  ?cr ;
  : zwei        ." Action 2   "  ?cr ;
  : drei        ." Action 3   "  ?cr ;
  : vier        ." Action 4   "  ?cr ;


\ Wie macht es jemand mit 4TH-Erfahrung?
    : cw:
      Create ( n -- n+1 )
       dup c, 1+
      Does> ( -- )  c@ .'  Dies ist Wort Nr. " . ?cr
    ;

    0  cw: w1   cw: w2  cw: w3  cw: w4  cw: w5  ( )
                                                ( )
    ( Jetzt liegt COUNT auf dem Stack !! )

    Create ctable ( count) c, ] w1 w2 w3 w4 w5 [
    Create  table             ] w1 w2 w3 w4 w5 [
    Create: :table              w1 w2 w3 w4 w5 ;
```

SCR 03

```
\ Liste          **** Datenstruktur: Liste ****            jrg 25Jul90

\ Die "Testumgebung"
  Vocabulary Static
  Static definitions

  Variable counter  counter off

\ Max. Listengröße ist 255
: List:
      Create ( size | <name> --)    1 arguments     \ Aufrufsicherheit
        dup   c,                                    \ Größe eintragen
      ( size) 0  DO   ['] noop ,   LOOP ;
```

SCR 04

```
\ verdichten     **** leere Listenpostionen verschieben ****   jrg 11Jun90

] : shift1  ( adr size -- )
      1- cells 2dup
      >r dup ( adr) cell+ swap r> cmove   + ['] noop swap ! ;

| : got_parms>recursion  ( adr size -- adr+2 size-1 )
      1- swap 2+ swap ;


- *** das eigentliche Zusammenschieben ***
~

  : <>  ( n1 n2 -- f)  = not ;
```

Quelltext 3

```
| : (compact  ( pfa[liste] size --)  recursive
     dup 1 > IF  2dup
                 get_parms>recursion (compact
                 over dup cell+ @ ['] noop <>
                 swap          @ ['] noop = and IF shift1
                                                ELSE 2drop THEN
            ELSE 2drop THEN ;

: compact ( pfa[liste] -- )
    count (compact ;    \ Größe ist nötig als Rekursionsgrenze
```

SCR 05

```
\ add:          **** Hinzufügen eines Elementes ****            jrg 03Jul90

: add:  ( listadr | <name> --)     counter off
    dup compact
    count >r
    BEGIN  dup @  ['] noop = not
    WHILE  cell+  ( nächste listaddr)
           counter inc  counter @ r@ = Abort" Liste vollständig ausgefüllt! "
    REPEAT
    ' \ cfa von action
    swap !  rdrop ;

.( ADD:  )
```

SCR 06

```
\ Zugriff und Darstellung der Liste über eine DO...LOOP - Schleife jrg 03Jul90

  | Defer action
  |     : lister  ( listadr --)
                  count cells bounds  DO  I action   2 +LOOP ;
  |     : .id     ( addr --)  @ >name .name  2 spaces ;

        : act     ['] perform Is action   lister ;
        : show    ['] .id     Is action   lister ;


.( ACT und SHOW  )

\\
Die obige Lösung vermeidet die Duplizität von Code, denn 1x Code ist nur
halb so fehleranfällig wie zweimal Code.

: act       ( listadr --)
    count cells bounds  DO  I perform                       2 +LOOP ;

: show      ( listadr --) cr
    count cells bounds  DO  I @ >name .name   2 spaces  ?cr   2 +LOOP ;
```

SCR 07

```
\ ACTLOOP  führt eine Tabelle von cfas aus                      jrg 03Jul90

 also          \ Sollte CODE nicht vielleicht ein ALSO machen ??
 Code actloop ( addresse länge -- )
      D C mov  \ Anzahl der Adressen
      I D mov
      I pop    \ Basisadresse des auszuführenden Feldes
      [[
         lods     A W mov                  \ Lade Adresse
         I push   C push  D push
         Label returnjmp    $5555 # I mov \ wird gepatcht
         W ) jmp                          \ .....  irgendwann ein UNNEST
         here returnjmp 1+ !              \ welches hierhin springt
         here 2+ ,  here 2+ , ( cfa = here )
         D pop    C pop    I pop          \ Register auf alten Stand
      c0= ?]
      D I mov    D pop
 Next end-code
 previous

.( ACTLOOP  )
```

SCR 08

```
\ kill:  *** Löschen eines Listenelementes ***                  jrg 25Jul90

: kill: ( listadr | <name>)   counter off
    ' >r      \ cfa von <name> zum Returnstack
    count  swap
    BEGIN   dup @  r@ = not
    WHILE   cell+  ( nächste listaddr)
            counter inc  counter @ over = Abort" Liste ist leer! "
    REPEAT
    |'] noop  swap !
    drop      \ count der Liste
    rdrop ;   \ cfa von <name> vom Returnstack löschen


.( KILL:  )
```

SCR 09

```
\ Beispiel I                                                    jrg 10Jun90
cls

10 List: ex


ex add: eins
ex add: zwei
ex add: drei
ex add: vier

cr  ex act
cr  ex show
cr
```

```
ex add: drei
ex add: drei

cr  ex act
cr  ex show
cr
```

SCR 10

```
\ Beispiel II                                                   jrg 10Jun90

ex kill: zwei
ex kill: drei
ex kill: vier
ex act
ex show


ex add: zwei
ex add: drei
ex act
ex show
```

SCR 11

```
\ Beispiel III  -  ein FORTH-Wort                               jrg 24Jun90

10 List: get


: ask   cr ." Bitte Taste drücken: " ;
: answer    ." Danke!" ;

get add: ask
get add: key

get add: dup
get add: .
get add: space
get add: emit

get add: answer

get act
```

SCR 12

```
\  Beispiel der Anwendung von ACTLOOP              fnk 20jun90    jrg 03Jul90

   : asmtab    table  5 actloop ;
   : test      ctable count cellw bounds  DO  I perform  2 +LOOP ;
   : asmctab   ctable count actloop ;
```

SCR 13

```
\\ loadscreen für dynamische Listen                             jrg 25Jul90

  System-on
  cls
   2  3 thru
  14 20 thru
```

SCR 14

```
\      *** eigene Umgebung ***                                  jrg 25Jul90

  OnlyForth  also definitions

  Vocabulary Dynamic
  Dynamic definitions

cr .( Vokabular für dynamische Listen angelegt)
```

SCR 15

```
\ List:                                                         jrg 03Jul90

Variable counter  counter off

' Variable Alias List:          \ Damit ist der Anker mit 0 vorbesetzt
```

SCR 16

```
\ add:          *** Worte in die Wort-Liste einfügen ***        jrg 25Jul90

   : add:  ( *firstlink  -- )
        dup @             \ holt die Adresse des 1. links aus dem Anker
        here  rot !       \ Anker zeigt auf den DP
        ( l.link) ,       \ l.link wird bei DP eingetragen
                          \ beim ersten Mal wird die 0  aus dem Anker eingetragen
        ' ( cfa) ,        \ das Wort, das kommt, in die nächste Zelle eintragen
   ;

cr .( add: fügt dem ListenWort eine Aktion hinzu)
```

Quelltext 3 - Fortsetzung

SCR 17

```
\ act / show        *** Wort-Liste ausführen / anzeigen ***      jrg jrg 03Jul90

| : (act   ( firstlink -- )  recursive
      dup >r  @ IF     r@ @ (act
              THEN     r> cell+ perform ;

  : act     ( *firstlink --)   @ (act ;


\ back  arbeitet die Liste von hinten nach vorne ab !! LiFo-type
: back    ( *firstlink -- )
    @  BEGIN
         ?dup                              \ enthält der Zeiger 0 = nil?
       WHILE
         dup >r  cell+ ( *cfa) perform     \ perform 'link-action'
         r>  @                             \ holt Adresse des nachsten Zeigers
       REPEAT
   ;

-       *** show  funktioniert genauso ***
-

 : show   ( *firstlink --) cr  ." Listeninhalt von "
     dup body> >name .name space ." ist:" cr
     @ BEGIN   ?dup WHILE  dup cell+ @  >name .name  2 spaces  @ REPEAT ;


cr .( ausführen/zeigen ist nun möglich)
```

SCR 18

```
\ kill:          *** Wort aus Wortliste löschen ***                jrg 10Jun90

\ wollen wir den ersten löschen?

 : lst?  ( *lstlink cfa --)
    over
    @ cell+ @ = IF        \ cfa's sind gleich, wenn es der letzte Eintrag ist
                dup @ @   \ holt aus lstlink die Adresse von 2ndlink
                swap !    \ speichert diese in lstlink
              ELSE  drop THEN ;
```

```
: kill:          ( *firstlink <name> --)
    '   \ nimmt name
    2dup lst?
    >r                          \ cfa auf den Returnstack, weil oft gebraucht
    ( *lstlink) @               \ adr of firstlink
    BEGIN ?dup WHILE
                  dup  @        \ für nächsten link
                  cell+ @ \ für cfa
                  r@ = IF  dup dup @ @  swap !  THEN
    @  REPEAT  rdrop ;


cr .( kill:  löscht Aktionen aus der WortListe)
```

SCR 19

```
\ Beispiele                                                          jrg 25Jul90
  Dynamic definitions
  List: ex            \ Anker zeigt auf den 1.link


  ex add: eins
  ex add: zwei
  ex add: drei
  ex add: vier

cls    ex act
       ex show

  ex add: drei
  ex add: drei

  cr   ex act
       ex show

  ex kill: drei
  ex kill: drei   ex show
```

Quelltext 3 - Fortsetzung

```
\ Variablendeklaration                                  jrg 25jul90

 | Variable size
 | Variable public
 | Variable last_private
 | Variable trash

 : cell+                    2+ ;

 | : >link  ( cfa -- lfa) >name 2- ;
 | : n>link ( nfa -- lfa)      2- ;
 | : l>name ( lfa -- nfa)      2+ ;

 | : latest ( -- lfa)     context @ @ ;
   : .last  ( lfa --)     latest l>name .name ;

cr .( Vars deklariert)

------------------------- Scr 2 ---------------------------------

\ Var:   lock  unlock                                   jrg 25jul90

\ Instanzlokale Variable
    : Var:        Create   size @  cell+ ,
                           size +!  immediate
                  Does>    @  [compile] Literal
                           compile +
    ;

  | : unlock      context @  dup @  trash !  !  ;
  | : lock        trash @  context @  !   ;

cr .( lock/unlock )

------------------------- Scr 3 ---------------------------------

\ do.operation   make.instance                          jrg 25jul90

| : context.find       vp push   vp 4+ push
                       context @ vp 4+ !  2 vp +!   find ;
| : find.op            bl word swap
                       unlock context.find lock
                0= Abort" als Operator ist unzulässig! " ;
| : do.op              find.op execute ;
| : compile.op         find.op swap  [compile] Literal  ,  ;
| : do.or.comp         state @ IF    compile.op
                       ELSE  do.op
                       ENDIF ;
| : make.instance      Create  dup @ ,
                              cell+ @ allot   immediate
                       Does>   dup @  do.or.comp ;

cr .( operation/instance)

------------------------- Scr 4 ---------------------------------

 \ Type> End-Type> seal                                 jrg 25jul90

   : <Type:      latent public !
                 Create
                 here cell+  last_private !   size off ;

 | : seal        latent
                 public @ swap !
                 0  last_private @  !  ;

   : Type>       warning off  latent
                 Create   seal ,  size @ ,
                 Does>    make.instance  ;

clear

cr .( type/endtype )

------------------------- Scr 5 ---------------------------------

 \ Validition                                           jrg 25jul90

   <Type: COMPLEX:
    2 Var: real
    2 Var: imag
         : !!        dup imag rot swap !  real ! ;
         : @@        dup real @  swap imag @  ;
   Type> COMPLEX:

  Complex: x
  50 100 x  !!
         x  @@  . .
         x  u?
         x  @ .
```

Quelltext 3

DFÜ

Der Modemanschluß

von Rafael Dellano

In Abstimmung auf die DFÜ-Aktivitäten der FORTH-Gesellschaft e.V. ist dies ein reiner Hardwareartikel zum Thema Modems. Damit das Ganze nicht schon in ISO-Schicht 1 scheitert.

Das V24-Kabel ...

... ist natürlich nur bei externen Modems nötig. Es sollten mindestens die in Tabelle 1 aufgeführten 9 Verbindungen vorhanden sein. Kreuzungen sind nicht nötig. Die Verbindung PC an Modem ist der problemloseste Fall einer V24-Verbindung. Für diesen Fall ist die V24-Verbindung gedacht.

Die Amtsleitung

Aus der Wand kommen üblicherweise nur die beiden Drähte *La* und *Lb*. Sie sind polungsunabhängig. Über sie läuft der ganze Betrieb ab. Vorsicht bei Messungen: sie führen eine unangenehm hohe Spannung. Insbesondere bei ankommendem Ruf.

In Nebenstellenanlagen (PBX) kommt häufig eine dritte Leitung hinzu: *E* wie Erde. Diese Leitung

Stichworte

- ✗ Modem
- ✗ TAE-Dosen
- ✗ US-Norm

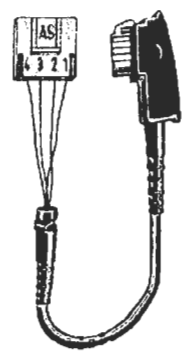


Bild 2: AS-Stecker

dient nur zur Signalisierung. Sie darf nicht zur Schirmung von Metallgehäusen verwendet werden. Oder gar mit »Erde«/Nulleiter des Lichtnetzes verbunden werden.

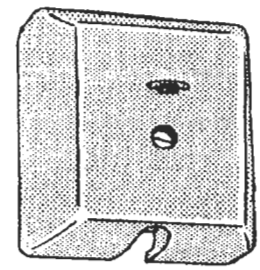


Bild 1: Dose VDo

Ein Drücken der »Erdtaste« am Telefon schließt *La*, *Lb* und *E* kurz. Die Nebenstellenanlage erkennt das und schaltet das Amt durch. Probleme treten durch Sparversionen auf: Das Endgerät schaltet nur *La* auf *E*, die Nebenstellenanlage erkennt nur *Lb* auf *E*. Lösung: *La* und *Lb* in der Zuleitung vertauschen.

Die Dosen an der Wand

Es gibt da mehrere Möglichkeiten. Die ADo8 wird hier nicht behandelt, sie war Vorläufer der TAE. Am verbreitesten ist immer noch die »alte« Dose VDo (siehe Bild 1). Nach Abschrauben der Abdeckkappe wird der 4polige AS-Stecker (siehe Bild 2) des Telefons zugänglich. Tabelle 2 zeigt die Belegung.

Die Leitung *W* wie Wecker ist eine deutsche Eigenheit. Der Gabelschalter ist als Umschalter ausgeführt und nimmt beim Abheben die Klingel des Telefons aus der Leitung. Die ist ziemlich kapazitiv und würde die Sprachübertragung stören. Diese geschaltete Leitung wird

SubD 25pol	SubD 9pol	Modem DCE-Belegung Buchse	PC DTE-Belegung Stecker	Name
Pin 2	Pin 3	in	out	TXD
Pin 3	Pin 2	out	in	RXD
Pin 4	Pin 7	in	out	RTS
Pin 5	Pin 8	out	in	CTS
Pin 6	Pin 6	out	in	DSR
Pin 7	Pin 5	GND	GND	SG
Pin 8	Pin 1	out	in	DCD
Pin 20	Pin 4	in	out	DTR
Pin 22	Pin 9	out	in	RI

Tabelle 1: Belegung V24

Pin	Farbe	Bezeichnung
1	weiß	La
2	braun	Lb
3	grün	W
4	gelb	E

Tabelle 2: Belegung AS4

wieder aus dem Telefon herausgeführt und kann für externe Wecker oder AWADOs verwendet werden.

Rund 25% der Anschlüsse haben bereits die »neue« Dose TAE (siehe Bild 8; Belegung vgl Tabelle 3; Stekker siehe Bild 3). Es gibt 3 Kodierungen:

- ✗ Nase unten: "F" = Telefon
- ✗ Nase mitte: "Z" = gibt es nicht
- ✗ Nase oben: "N" = ROW also Modem, Anrufbeantworter, Fax

Für ca. 65 DM installiert sie die Post und wechselt gleich das Telefon mit aus. Wesentlich ist, daß man sich nicht eine simple F-Dose einbauen läßt. Die ist für ein Modem kaum besser als eine VDo. Mindestens eine NF oder eine NFN (siehe Bild 5 - 8). Aus der Dosenbeschaltung sieht man, daß diese Dose weiterschaltet, wenn der Stecker gezogen wird. Das Telefon führt allerdings *a2/b2* nicht weiter.

Die Telefonbuchse am Modem

Postmodems bevorzugen den besonders exotischen FKS8-Stecker (siehe Bild 4; Belegung Tabelle 7; entsprechendes TAE-Kabel siehe Tabelle 4).

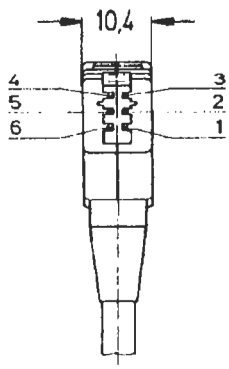


Bild 3: TAE 6N-Stecker

US-Modems haben stattdessen zwei RJ11-Buchsen. Eine für *La/Lb*=»Line«. Die andere für *a2/b2*=»Phone« (Belegung siehe Tabelle 6) oder nur »Line«. Eine Erdtaste ist meist nicht vorhanden. Das entsprechende TAE-Kabel zeigt Tabelle 5. Die Buchse RJ13 wird hier nicht behandelt. Auch in der 6-poligen Variante RJ12 sind *La/Lb* meist mittig.

Die Verdrahtung

Das Modem wird zwischen Amt und Telefon eingeschleift (siehe Bild 9). Im Ruhezustand ist am Modem *La* auf *a2* und *Lb* auf *b2* durchgeschaltet. Belegt das Modem die Leitung sind diese Verbindungen getrennt.

Man sollte nicht versuchen, beliebig viele Modems, Faxe usw zu kaskadieren. Deren Ruferkennungen sind zwar nicht so kapazitiv wie die eines Telefons, sie bleiben jedoch dauernd in der Leitung und verschlechtern die Übertragungsverhältnisse. Aus diesem Grund ist auch von simplem Parallelschalten an den Lüsterklemmen der Dose abzuraten. Der Telefonwecker beeinträchtigt die Funktion des Modems.

Die US-Modems

Rund 30% der heute in Deutschland betriebenen Modems entsprechen nicht deutschen Vorschriften. Durch die steigende Zahl privater Anwender wird sich der Trend noch verstärken. Skrupel sind nur bedingt angebracht, da Schwarzmarktzustände meist auf einem Fehler des Systems und weniger auf der Unmoral der Bürger beruhen. Was ist ein US-Modem? Ein Modem das von der FCC (= US-FTZ) zugelassen ist. Am Boden des Ge-

Pin	Farbe	Bezeichung
1	weiß	La
2	braun	Lb
3	grün	W
4	gelb	E
5	grau	b2
6	rosa	a2

Tabelle 3: Belegung TAE6-N

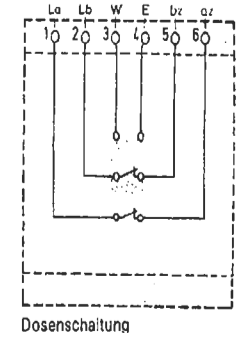


Bild 5: TAE-F-Dose

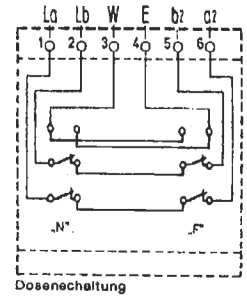


Bild 6: TAE-NF-Dose

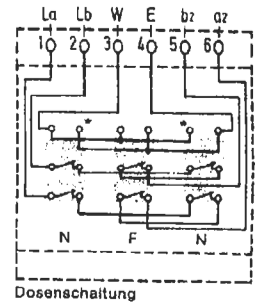


Bild 7: TAE-NFN-Dose

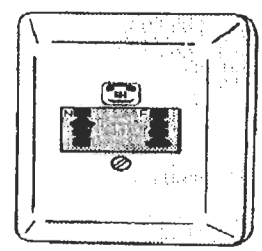


Bild 8: Außenansicht TAE-NF-Dose

räts befindet sich ein Aufkleber mit auf dem u.a. steht: FCC-Reg.No. xxxxxxxxx. Ein Gutteil der Nicht-FTZ-Modems hat allerdings auch keine US-Zulassung, orientiert sich jedoch an den dortigen technischen Gegebenheiten. Qualität und Funktionsfähigkeit sind zweifelhaft, sie werden über den Preis verkauft. Hier die wesentlichen technischen Unterschiede DBP/US:

- *Gleichstromwiderstand*: US: < 200Ohm, DBP: 300 - 470Ohm. Unkritisch, da eine sichere Auslösung der Schleifenstromerkennung erfolgt.
- *AC-Sendepegel*: US: -9dBm, DBP: einstellbar. Sollte ohnehin auf -9dBm eingestellt werden.
- *AC-Widerstand*: US: 600Ohm , DBP: Zr = 220 Ohm + 820 Ohm // 120nF. Zr ist technisch günstiger.
- *MVF-Wahl*: kaum Unterschiede. DBP-Soll ist 80msec Puls und 80msec Pause.
- *IWF-Wahl*: DBP-Soll ist Make/Break 40/60. Hayes erlaubt Einstellung auf US 39/61 und Hongkong 33/67.

Man kann davon ausgehen, daß bei US-Modems die Hörtonerkennung nicht sonderlich funktioniert. Vollautomatischer Verbindungsaufbau ohne Anwesenheit des Benutzers scheidet damit aus. Haben sie nur eine »Line«-RJ11 können sie nicht vor dem Telefon eingeschleift werden. Umstöpseln an der TAE ist denkbar. Eine Umschaltdose WS80 ist besser. Der Betrieb ist in beiden Fällen unbefriedigend. Man sollte auch überprüfen, ob sie nach Verbindungsabbau aus der Leitung gehen, sonst tickt der Gebührenzähler munter weiter.

Marginalien

Ein weiteres Problem kann der 16kHz-Gebührenimpuls sein. Wer ein Telefon mit Gebührenzähler besitzt hat ihn auf der Leitung. Modems sind nicht dagegen geschützt. Nominell schaltet ihn der Guardton (1800Hz) des Answer-Modems (Mailbox) ab. Es sei noch angemerkt, daß die Post dumme Wählgeräte haßt. Z.B. Modems die laufend eine besetzte Nummer anwählen. In allen anderen Ländern bestehen da keine Bedenken, weil der Gebührenzähler nach Abheben des Hörers sofort läuft. Ein Vorteil der DBP ist, daß man nur zustandegekommene Gespräche bezahlt. Ein dummer Automat, der ständig eine Verbindung München-Hamburg aufbaut verursacht Betriebskosten.Für DBP-Dosen und Kabel gibt es mehrere Hersteller. Empfohlen sei Ackermann/Gummersbach. Hübschere Dosen in modernem Design hat Rutenbeck unter dem Namen »RUBIK«.

✉ Rafael Deliano
Steinbergstr.37
8034 Germering

Pin	Farbe	Bezeichung
1	weiß	La
2	braun	E
3	grün	W
4	gelb	Lb
5	grau	b2
6	rosa	Schirm
7	blau	
8	rot	a2

Tabelle 7: Belegung FKS8

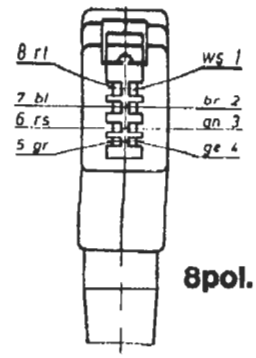


Bild 4: FKS8-Stecker

TAE6-N	Farbe	RJ11	Bezeichung
1	rot	RJ11-Line	Lb
6	rot	RJ11-Phone	b2
2	grün	RJ11-Line	La
5	grün	RJ11-Phone	a2
3			
4			

Tabelle 5: Kabel TAE6/RJ11/RJ11

TAE6-N Stecker	Farbe	FKS8 (Stecker wird mit Kabel geliefert)	Bezeichung
1	weiß	1	La
6	rot	8	a2
2	gelb	4	Lb
5	grau	5	b2
3	grün	3	W
4	braun	2	E

Tabelle 4: Kabel TAE6/FKS8

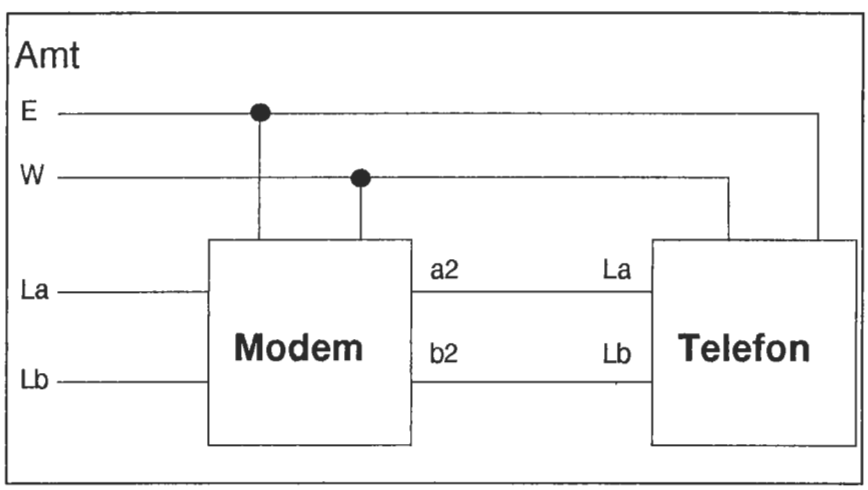


Bild 9: Schaltskizze

F68K

Ein schnelles Chamäleon

Jörg Plewe, Großenbaumer Str. 27,
4330 Mülheim/Ruhr, 0208/423514

Quelltext Service

Eine Computerzeitschrift ist immer dann eine schlechte Computerzeitschrift, wenn darin enthaltene Testberichte über Programme von den Autoren der Programme selbst oder deren Vertriebsfirmen geschrieben und zum Preise einer ganzseitigen Werbung auch veröffentlicht werden. Deshalb kann ich hier mein Programm nicht testen, denn die 'Vierte Dimension' ist ohne Zweifel ein gute Zeitschrift. »Das kann ja auch ein anderer tun«, dachte ich mir, aber vorstellen möchte ich mein System doch am liebsten selbst; zudem auch kein kommerzielles Interesse den objektiven Blick trüben kann.

Das erklärte Ziel dieses Beitrages ist es, Interesse für ein System zu wecken, das noch tief in den Kinderschuhen (Größe 19-20) steckt, und in das es noch viel Arbeit zu investieren gilt, wenn es einmal wirklich große, schöne Schuhe (Größe 50-56) bekommen soll, wie ich es mir wünsche. Jeder, der will, darf etwas zum Erreichen dieses Zieles beitragen.

Stichworte

- ✗ MC68000,
- ✗ Quelltextverwaltung,
- ✗ Fileinterface

Ein Problem ...

FORTH-Entwicklungssysteme für MC68000-Computer wie den Atari ST, Amiga oder Macintosh gibt es inzwischen zur Genüge. Diese Systeme sind jedoch meist untereinander inkompatibel und kosten zudem noch eine gute Stange Geld (sind dieses aber auch meist Wert), so daß sie manchmal für Schüler oder Studenten nicht verfügbar sind. Auf Nicht-Standard-Computern wie Einplatinenrechnern oder Prototypen, wo FORTH ja eine seiner größten Stärken hat, ist das Angebot nicht so reichhaltig, so daß oft nur der Verzicht auf FORTH oder das Selbstschreiben bleibt, was sicher nicht jedermanns Sache und ein ständiges Neuerfinden des Rades ist.

... und eine Lösung

Auf der Basis meines FFORTH-Kerns wurde innerhalb weniger Monate F68K zu einem eigenständigen, lauffähigen FORTH-Kern entwickelt. Die Hauptzielsetzung war es, ein System zu entwerfen, das auf allen 68000-Systemen lauffähig ist und zudem natürlich eine höchstmögliche Kompatibilität zwischen verschiedenen Rechnern bieten soll. Es sollte so einfach wie möglich auf fremde Rechnertypen portierbar sein, so daß ein großer Anreiz entsteht, auf einem neuen Rechner doch zunächst einmal FORTH zu benutzen, um im Handumdrehen ein leeres Board mit einem interaktiven Programmiersystem auszustatten, das alle weiteren Entwicklungen vereinfacht. Außerdem sollte es in guter FORTH-Tradition für private Anwender kostenlos sein, was den erwähnten Anreiz nur noch steigern kann. Zusätzlich sollten einige Ideen verwirklicht werden, die in anderen Systemen nicht zu finden sind, welche wiederum für die scheinbar nötigen Inkompatibilitäten sorgen, aber hoffentlich auch durch Nützlichkeit den Anreiz so weit steigern, daß FORTH am Ende als einzige Alternative übrig bleibt – auf daß die FORTH-Gesellschaft blühe.

Der Kern

F68K reiht sich bei den 'schnellen' Systemen wie 4xFORTH, MACH2, FFORTH oder BIGFORTH ein. Anstelle des traditionellen, virtuellen Fadencodes wird direkt Maschinencode für den MC68000 erzeugt, was in einer hohen Ausführungsgeschwindigkeit der erzeugten Programme resultiert, da der innere Interpreter durch die 68000er CPU sehr gut dargestellt werden kann [1]. Viele schöne Eigenschaften der gefädelten Implementation gehen dafür leider verloren. Man braucht das Für und Wider hier nicht zu diskutieren, es ist eine Sache des Geschmacks und vorrangig natürlich auch der Zielsetzung! Über die Technik der Codeerzeugung in nichtgefädelten Systemen habe ich mich bereits in zwei Ausgaben der 'Vierte Dimension' [1,2] langatmig ausgelassen, so daß ich hier nur die wichtigsten und charakteristischen Punkte herausstellen möchte.

- ✗ MC68000 Maschinencode, Primitive werden als Makros eingesetzt
- ✗ Trennung von Code und Daten (Header)
- ✗ Adressierung nur relativ über zwei Basisregister

Um den Kern maschinenunabhängig gestalten zu können, wurde einfach alles, was eben maschinenabhängig ist, aus dem Kern verbannt. Dies sind genaugenommen die vier E/A-Funktionen KEY, KEY?, EMIT und R/W. Doch verbannt heißt nicht verschwunden!

Der Lader

Diese vier Funktionen werden von einem Lader bereitgestellt, der für jedes System neu erstellt werden muß. F68K zeichnet sich nun dadurch aus, daß es die Erstellung eines wirklich einfachen Laders ermöglicht und unterstützt.

Wegen der oben bereits erwähnten durchgehend relativen Adressierung kommen im Bitimage des Kerns keine absoluten Adressen vor. Das System kann von dem Lader einfach in den Speicher geladen und angesprungen werden. Es sind keine Reloziierungen o.ä. notwendig. Dies macht den Lader zu einem sehr einfachen kleinen Programm, das keine Schwierigkeiten bereiten sollte.

Da der Kern nun keine Systemabhängigkeiten beherbergt und auch nicht mehr systemabhängig geladen werden muß, folgert man richtig, daß der Kern für die Portierung auf ein neues System nicht einmal neu kompiliert bzw. assembliert werden muß. Das Bitimage des Kerns ist selbst übertragbar. Auf jedem Rechner läuft letztendlich der exakt gleiche Code. Eigentlich wird dadurch die Portierung des in Assembler geschriebenen Kerns trotz der totalen Unabhängigkeit erst möglich, denn Assembler ist noch lange nicht Assembler. Man versuche nur einmal, unter GEMDOS assemblierbare 5000 Zeilen durch einen OS9- oder RTOS/UH-Assembler zu bekommen!

Für eine Portierung genügt es also, einen Lader zu schreiben. Er stellt neben den besagten vier Funktionen eine weitere Funktion für ein standardisiertes SAVESYSTEM sowie den Speicher für Code- und Datensegment, die Stacks und den TIB zur Verfügung. Diese Werte werden auf dem Stack (A7) abgelegt und dann wird zum Anfang des Codesegments gesprungen. F68K richtet es sich dann selbst mit den gegebenen Parametern ein. So einfach ist das!

Zu den E/A-Funktionen selber gibt es zunächst nicht viel zu sagen. Ihr Verhalten ist durch das bekannte Verhalten der entsprechende FORTH-Worte klar definiert und wohl jedem bekannt. Die Parameterübergabe geschieht über den normalen F68K-Datenstack, der durch das Adreßregister A6 des M68000 indiziert wird. Um es möglichst einfach zu halten, sichert F68K vor dem Aufruf von Laderfunktionen alle Register selbst und stellt auch nach dem Aufruf den Stackzeiger auf die richtige Position. Auf dieser Ebene ist bei den E/A-Funktionen also fast kein Fehler möglich.

Möglichkeiten des Laders

Eine weitere Differenzierung der Ladertätigkeit zeigt einige Möglichkeiten auf, die man dem einfachen Ansatz des Laders zunächst nicht ansieht. Diese Möglichkeiten erlauben ein breites Spektrum für den Entwurf des Laders. Es reicht von Minimallösungen, die nicht mehr als ein paar hundert Byte umfassen, bis zum ausgeklügelten 100K-Programm.

Doch von vorn: die E/A-Funktionen, d.h. ihre Einsprungadressen, werden F68K nicht selbst übergeben. Sie werden vielmehr in jeweils eine Tabelle eingetragen. Diese Tabellen erhält F68K. Die Tabellen können nun mehr als nur eine E/A-Funktion enthalten. Damit ist es möglich, F68K z.B. EMIT-Prozeduren für die Ausgabe auf dem Bildschirm, dem Drucker, der seriellen Schnittstelle und einem File bereitzustellen. Dazu werden diese vier Prozeduradressen einfach in eine kleine Tabelle von Langworten (4 Byte) eingetragen:

```
4      /* Anzahl der Funktion */
CONEMIT
PRTEMIT
AUXEMIT
FILEMIT
```

Die Adresse dieser Tabelle liegt beim Start von F68K auf dem Stack. Der erste Eintrag wird von F68K als Standardeinstellung angenommen. Für jede der vier E/A-Funktionen gibt es nun solch eine Tabelle.

Auf F68K-Ebene sind diese Tabellen ebenfalls zugänglich. Die obige heißt einfach EMITS als der Plural von EMIT. Der Zugriff auf diese Tabelle ermöglicht nun auf einfachste Weise eine Umlenkung von Ein- und Ausgabe:

```
: >PRINTER ( -- )
     EMITS 8+ @ (EMIT) ! ;
```

oder

```
: TERMINAL ( -- )
   EMITS $C +  @  (EMIT) !
   KEYS  $C +  @  (KEY)  ! ;
```

Oder man erzeugt ein Ausgabeprotokoll in eine Datei:

```
: PROT-EMIT ( char -- )
   DUP
   EMITS 4    +  @  EXECUTE
   EMITS $10 +  @  EXECUTE ;

' PROT-EMIT  (EMIT) !
```

Die Möglichkeiten für die Gestaltung der E/A sind allein auf dieser Ebene unerschöpflich. Aber auch die Funktionen selber können sehr unterschiedliches Verhalten an den Tag legen und damit das Erscheinungsbild von F68K erheblich beeinflussen. Natürlich genügt es, wenn die Prozedur für KEY darauf wartet, daß auf der Tastatur eine Taste gedrückt wird und das entsprechende Zeichen zurückliefert. Das ist die Minimalvariante. Auf der anderen Seite kann die Funktion für KEY unter einem Betriebssystem wie GEM eine Menüleiste mit allen dazugehörigen Maus- und sonstigen Ereignissen verwalten, und im Bedarfsfall ganze Zeichenketten an F68K durchpipen, so daß der Benutzer z.B. den Menüpunkt 'BYE' mit der Maus auswählt und KEY eben in vier aufeinanderfolgenden Aufrufen die Zeichen 'B', 'Y', 'E' und <cr> liefert, ohne daß tatsächlich auch nur eine Taste berührt wurde. Passend dazu landen die durch EMIT ausgegebenen Zeichen in einem GEM-Fenster und über vereinbarte Escape-Sequenzen kann F68K dann zum Editieren sogar eigene Fenster öffnen. Obwohl F68K nichts von alledem ahnt, was dort geschieht, ist es GEM-fühlig wie eine richtige GEM-Applikation.

Für den Atari eine Maximallösung (und in Vorbereitung).

Ich glaube, diese Andeutungen alleine reichen bereits, um die ernormen Möglichkeiten, mit Hilfe dieses einfachen Ladeprogramms auf F68K einzuwirken, klarzumachen. Da braucht man von intelligenten Komprimierungs- und Cacheverfahren bei R/W oder dem Nachrichtenaustausch verschiedener F68K Systeme in Multitasking/Multiusersystemen und und und... nichts mehr erwähnen.

Auch die technische Erstellung des Laders läßt den Portierer verschiedene Wege gehen. Bevorzugt und klar favorisiert ist hier allerdings die Benutzung eines Assemblers, besonders wenn das 'bring es zum Laufen' im Vordergrund steht. Mein assemblererzeugter GEMDOS-Lader unterhält schon einigen Komfort und ist trotzdem nicht größer als 1 KByte. Für die Herstellung umfangreicherer Lader lassen sich aber auch andere höhere Programmiersprachen einsetzen. Man muß nur dafür Sorge tragen, daß sich die geschriebene Funktionen auf das vorgeschriebene Parameterformat trimmen lassen, indem man sie z.B in kleine Assemblermodule einbindet. Ein zweiter Lader ist bei mir mit Hilfe eines Freeware-C-Compilers (SOZOBON) entstanden. Dort genügen zwei bis drei Zeilen In-line-Assembler, um das Programm 'F68K-tüchtig' zu machen. Eine ideale Basis für Maximallader. Natürlich darf man auch FORTH-Systeme benutzen. Wenn schon ein Lader vorhanden ist, kann man sogar F68K zum Schreiben das Laders verwenden (Lader lädt F68K-Applikation, diese lädt F68K).

Die letzte Eigenschaft des Laders, die hier noch Erwähnung finden soll, sind zwei zusätzliche Funktion, die F68K übergeben werden und dem System einen systemabhängigen Schreib/Lesekanal eröffnen. Diese Funktionen schreiben/lesen z.B. unter GEMDOS lediglich die übergebenen Daten in/aus ein(em) File, das beim ersten Aufruf erzeugt/geöffnet wird. Dieser Kanal wird von F68K mit der Funktion

```
WRITESYS ( addr count -- flag )
```

beschrieben und mit

```
READSYS ( addr count -- flag )
```

gelesen.

WRITESYS ist vielseitig einsetzbar. Die Hauptanwendung sicher liegt beim Abspeichern des Systemimages, weithin als SAVE-SYSTEM bekannt. Da das Systemimage nicht innerhalb des F68K-Massenspeicherzugriffs abgelegt werden kann, sondern nur auf dem Massenspeicher des Hostsystems, auf den dann auch der Lader zugreifen kann, ermöglicht dieser Kanal trotz der damit verbunden Systemabhängigkeit ein systemunabhängiges, einheitliches SAVE-SYSTEM. Weiterhin kann dieses Funktionspaar z.B. für Im- und Export von Quelltexten dienen.

F68K

Nachdem der Lader erstellt und gestartet wurde, sollte sich F68K mit dem wohlvertrauten 'ok' zur Arbeit melden. F68K gehorcht in den meisten Punkten dem weitverbreiteten FORTH-83 Standard. Einige Abweichungen davon sind jedoch erwähnenswert. Zum ersten ist F68K ein 32 Bit-System, was dem Standard natürlich entgegenläuft. Vielleicht sollte man auch dazu übergehen, generell 32 Bit-Systeme, die sich an diesen Standard anlehnen, als 32FORTH-83 o.ä. bezeichnen, um Mißverständnissen vorzubeugen. F68K richtet sich aber in machen Dingen nicht mehr nach dem FORTH-83, sondern bezieht zum Teil schon den dämmernden ANSI-Standard mit ein. So wurde auf COMPILE und [COMPILE] verzichtet und statt dessen das etwas smartere POSTPONE eingesetzt. Gefällt mir persönlich nicht so gut, aber wenn's denn sein muß...

In BEGIN-WHILE-REPEAT Schleifen sind mehrere WHILEs erlaubt. Für jedes WHILE verlangt F68K ein THEN hinter dem REPEAT, um die Sprungreferenz von WHILE aufzulösen, auch wenn nur ein WHILE vorhanden ist. Diese Lösung ist einfach und sehr konsequent. Andere Implementationen existieren, in denen z.B. das REPEAT die erste WHILE-Referenz selbst auflöst und man bei fehlendem WHILE deshalb AGAIN statt REPEAT schreiben muß (z.B. ZEN-FORTH); oder bei denen REPEAT so lange auf dem Stack herumwühlt, bis jedes WHILE (falls vorhanden) weiß, wo es hinspringen soll (volksFORTH, FFORTH). Diese Lösungen sind zwar manchmal syntaktisch kürzer, aber nicht so FORTHy. Außerdem eröffnet diese F68K-Schleife mehr Möglichkeiten, denn zwischen REPEAT und THEN darf natürlich auch noch Code stehen! Einige Features des zugrunde liegenden FFORTH-Kernels wie das gehashte Vokabular, die DO-LOOP-Indizes in Registern oder Kontrollwerte für Strukturen wurden zu Gunsten der kompatibleren Handhabung von Tricks besonders in der Vokabularstruktur gestrichen. Pountain's 'Object Oriented FORTH' ist mit gehashten oder gar baumstrukturierten Vokabularen eine Katastrophe (siehe Beitrag 'Linked Actions').

Dadurch wurde F68K besonders beim Kompilieren etwas langsamer, aber auch deutlich überschaubarer. Wie meine Erfahrung bisher gezeigt hat, ist auch das langsamere Laden noch recht flott und verlangt eigentlich nur wenig Geduld.

BLOCKSTREAM-Interface

Nachdem ich nun über ein Jahr mit dem auf Streamfiles gestützten FFORTH gearbeitet hatte, erwischte ich mich schon gelegentlich dabei, wie ich Jörg Staben beim Hantieren mit seinen volksFORTH-Blöcken neidisch über die Schulter schaute. Da zudem Streamfiles ein Filesystem vorraussetzen, was im Zielgebiet von F68K nicht vorhanden sein muß, ließ ich mich von dieser Tatsache gern zur reumütigen Rückkehr zu Blocks zwingen. Nur so, wie sie sind, sollten sie nicht hingenommen werden. Der von Ulrich Hoffman in Frankfurt so eindrucksvoll beschriebene 'Druck' von oben und unten, der die Quelltexte manchmal unlesbar komprimiert, wurde verringert, indem einem Block einfach der gesamte heute übliche Bildschirm von 80x25-Zeichen zur Verfügung gestellt wird. Heute kann

ich sagen, daß daraus ein sehr angenehmes Schreibgefühl erwächst, da zumeist auch noch ausreichend Platz für zeilenweises kommentieren vorhanden ist. Die Blockgröße wurde also schlicht auf 2K eingestellt. Ein stechender Schmerz sollte jedem Systemprogrammierer an dieser Stelle durch den Körper fahren! 2K sind 2048 Bytes; 80x25 ergeben gerade 2000! 48 ungenutzte Byte!! Während ich noch mich krümmend auf dem Boden wandt, kam die Idee, diese 48 Byte zu nutzen, um darin eine Art Fileinterface zu organisieren. Der Name dafür kam mir als nächstes:

FIFI – FORTH Internes File-Interface

Eines der größten Probleme von blockgestützten Systemen, die nicht das Glück haben, auf ein durch ein Betriebssystem vorgegebenes Fileinterface aufsetzen zu dürfen, ist die fehlende Möglichkeit, Blocks irgendwo auf einfache Weise einzufügen oder zu entfernen. Hier setzt FIFI an: zusammengehörige Blocks sollen nicht mehr auf dem Gerät physikalisch hintereinander stehen müssen, sondern sind durch eine Zeigerkette vorwärts und rückwärts miteinander Verbunden. Für diese Zeiger findet sich in den überschüssigen 48 Bytes ausreichend Platz. Wenn man nun noch einige Blocks als Verzeichnisse abstellt, so kann man solch einer Zeigerkette von Blöcken einen Namen geben und diesen in einem Verzeichnisblock eintragen. Jeder dieser Blockketten kann für sich etwa wie eine Diskette behandelt werden. '1 BLOCK' liest eben den ersten Block dieser Kette, '10 LOAD' lädt den zehnten. Welche physikalischen Blöcke dabei angesprochen werden, ist unerheblich. Diese Ketten haben nun den Vorteil, daß man sie an beliebigen Stellen verlängern oder verkürzen kann, einfach durch ein- oder auslinken von Blöcken. Wird ein physikalisches Gerät auf die Benutzung mit FIFI vorbereitet, so werden einfach alle Blöcke bis auf den 0-ten zusammengelinkt. Die so erhaltene Kette aller freien Blöcke wird 'Free' getauft, und in den Verzeichnisblock 0 eingetragen. Man könnte diesen Vorgang 'formatieren' nennen. Will der Benutzer nun ein File, ich sage lieber einen BLOCKSTREAM, erzeugen, um darin einen Quelltext zu erstellen, so werden einfach die gewünschte Anzahl von Blöcken vom Anfang der 'Free'-Kette abgehängt und mit einem Namen versehen in das Verzeichnis eingetragen. Erweist sich dieser Blockstream später als zu kurz, so werden weitere Blocks von der 'Free'-Kette ab- und dem Blockstream angehängt. Überschüssige Blocks werden wieder in die 'Free'-Kette eingehängt und stehen so anderen Blockstreams zu Verfügung.

Da jeder Block nur selbst 'weiß', welcher physikalische Block sein logischer Nachfolger und sein logischer Vorgänger ist, erspart man sich die in Filesystemen sonst übliche Verwaltung einer FAT (File Allocation Table), die angibt, an welchen physikalischen Stellen die Teile eines Files zu suchen sind. Dadurch wird das gesamte System ziemlich unanfällig gegen Störungen und die meisten Fehler können repariert werden, selbst wenn sie das Wurzelverzeichnis betreffen.

Mit diesem einfachen Konzept ist es möglich, dem Benutzer ein DOS-Feeling zu vermitteln. Blockstreams können erzeugt, gelöscht oder kopiert werden, wie es auch mit DOS-Files möglich ist. Da ein Verzeichnis selbst nur aus einem oder mehreren Blöcken besteht, die ebenso als Kette organisiert sind, können auch Unterverzeichnisse erstellt werden. Damit ist es möglich, viele Vorteile der von manchen bevorzugten Streamfiles mit den unbestrittenen Vorteilen von Blocks zu verbinden. Hat man einen Blockstream selektiert, arbeitet man wie man es mit Blökken gewohnt ist. '1 LOAD' oder '3 LIST' sind hier häufige Sequenzen. Zusätzlich existiert jetzt aber eine Struktur, mit der sich ein Blockgerät nach logischen Gesichtspunkten unterteilen läßt.

Ich möchte an dieser Stelle nicht auf die technischen Details der Implementation eingehen und auch kein vollständiges Glossar liefern. Ich möchte nur ein paar Worte auf höchster Ebene vorstellen:

MAKE (--)
MAKE <name>

erzeugt einen Blockstream mit der Bezeichnung <name>. Dieser umfaßt zunächst nur einen Block. <name> kann auch eine komplette Pfadangabe enthalten. Beispiel:

```
MAKE 2:USR/LOCAL/SRC/TEST
```

erzeugt auf dem Gerät mit der Nummer zwei (wird durch den Lader vorgegeben) den Blockstream TEST im Verzeichnis SRC. SRC ist ein Unterverzeichnis von LOCAL und dieses ein Unterverzeichnis von USR.

USE (--)
USE <name>

selektiert einen Blockstream für die Benutzung. Worte wie LOAD, BLOCK oder BUFFER beziehen sich jetzt auf diesen Stream. Gleichzeitig wird ein FORTH-Wort <name> angelegt, in dem einige Informationen über diesen Stream in einer Struktur gespeichert werden. Die Adresse dieser Struktur wird in der USER-Variablen STREAM abgelegt. BLOCK, BUFFER und LOAD beziehen sich grundsätzlich immer auf den Blockstream, dessen Informationen bei der Adresse in STREAM abgelegt sind. Für <name> gilt das gleiche wie bei MAKE

DEL (--)
DEL <name>

löscht den angegebenen Blockstream. Seine Blocks werden wieder in die 'Free'-Liste eingehängt.

MORE (wieviel --)

erweitert den mit USE selektierten Stream um 'wieviel' Blocks. Beispiel:

```
MAKE TEST
USE TEST
10 MORE
```

erzeugt einen Blockstream mit Namen TEST, der 11 Blöcke umfasst und für die Benutzung selektiert ist.

LESS (wieviel --)

verringert den Blockstream um 'wieviel' Blöcke. Diese werden vom Ende des Streams entfernt.

```
INSERT ( wo wieviel -- )
```

fügt an der Stelle 'wo' 'wieviel' Blöcke ein.

```
EXPEL ( wo wieviel -- )
```

entfernt an der Stelle 'wo' 'wieviel Blöcke. Der Block 'wo' selbst wird nicht entfernt. Beispiel:

```
4 2 EXPEL
```

entfernt die Blöcke 5 und 6

```
MAKEDIR ( -- )
MAKEDIR <name>
```

erzeugt ein Verzeichnis (Directory) mit dem angegebenen Namen. Dieses Verzeichnis enthält zunächst als einzigen Eintrag das Verzeichnis '..', womit das übergeordnete Verzeichnis selektiert werden kann.

```
DELDIR ( -- )
DELDIR <name>
```

entfernt das angegebene Verzeichnis, sofern dieses leer ist. Andernfalls müssen zunächst alle Einträge entfernt werden.

```
CD ( -- )
CD <name>
```

macht das angegebene Verzeichnis zum aktuellen. Wird <name> nicht angegeben, so gibt CD den aktuellen Verzeichnispfad aus.

```
DIR ( -- )
```

listet alle Einträge des aktuellen Verzeichnisses.

Neben diesen hier aufgeführten Worten gibt es eine ganze Reihe, die mehr den Charakter von Betriebssystemaufrufen haben und deutlich flexibler sind. Sie heißen BCREATE, BDELETE, BUSE, DCREATE, NEXTBLOCK, FORMATTED?, ?EXISTSTREAM, ... Eine vollständige Aufzählung wäre hier müßig.

Zusammen mit FIFI traten einige Schwierigkeiten auf, die inzwischen gelöst sind. Man denke nur an VIEW. Mit jedem FORTH-Wort wird die Nummer des physikalisches Blocks, in dem dieses Wort definiert wurde, gespeichert. Nun nutzt es dem Anwender nicht viel, wenn man ihm den Block anzeigt, in dem das betreffende Wort definiert wurde. Ihn interessiert auch der Kontext, d.h. Pfad und Name des entsprechenden Blockstreams. Zu einer physikalischen Blocknummer muß also der Stream gefunden werden, der diesen Block enthält. Damit ist ein Blockstream-VIEW doch um einiges komplexer als das konventionelle. Das Prinzip ist jedoch ganz einfach: zunächst hangelt man sich von dem physikalischen Block über die Links zum ersten Block des betreffenden Blockstreams. Dann stellt man anhand einer Tabelle, in die die Startblöcke und Kapazitäten der einzelnen Geräte eingetragen und die vom Lader gestellt wird, fest, auf welchem Gerät dieser Stream zu suchen ist, was auch noch sehr simpel ist. Dann beginnt man einfach beim Block 0 dieses Gerätes, der ja ein Verzeichnis ist, mit der Suche nach einem Blockstream, dessen erster Block mit dem gefundenen übereinstimmt. Dabei werden rekursiv auch alle Unterverzeichnisse durchsucht, bis man ihn endlich gefunden hat. Um den Benutzer nicht allzu sehr zu langweilen, hat man währenddessen den physikalischen Block bereits auf dem Bildschirm ausgegeben. Dann noch ein Tastendruck und der Benutzer erfährt Gerät, Name des Streams und die logische Blocknummer des gesuchten Wortes. Zur Veranschaulichung dieser Programmierung in den tiefen des Blockstreamsystems weit unterhalb der Oberfläche sei VIEW hier als Quelltext aufgeführt (siehe Listing 1).

Das Blockstreamsystem bietet sich noch für viele Erweiterungen an: für Erstellungsdaten oder Zugriffsrechte ist noch ausreichend Platz in den 48 Bytes vorhanden. Es bietet sich auch an, hierin Dokumentationssysteme zu verwirklichen, die die Nützlichkeit von Shadow-Screens o.ä. bei weiten übersteigen. Informationen für ein VIEW auf noch nicht definierte Worte haben auch noch Platz und und und ...

Dokumentation

Da die deutsche Sprache allmählich verfällt (man denke an die neuen Trennungsregeln in Berlin) und um eine weite Verbreitung zu unterstützen, wird die Dokumentation in radebrechendem Touristenenglisch verfasst. Sie ist noch nicht, und wird es wohl auch niemals werden, fertig. Sie wächst jedoch von Tag zu Tag, so daß man jetzt bereits einiges mit ihr anfangen kann.

Zusammenfassung und Ausblick

Während sich begnadete FORTH-Entwickler auf der Intel-Seite der Computerwelt austoben und dort phantastische System erstellen [3], macht sich auf der 68er-Seite Einsamkeit breit. Dies mag wohl an der breiteren Rechnervielfalt liegen. Zwischen diesen Rechnern soll F68K eine preiswerte Brücke schlagen. Gegen die für spezielle Rechner geschriebenen Systeme kann und will es nicht antreten. F68K wird niemals fertig werden. Es ist als solide Basis eines großen Systems gedacht, das sich bei den Anwendern entwickeln soll. Das Blockstreamsystem stellt die notwendige Quelltextverwaltung dar, um dem System sein noch recht rüdes und kantiges Auftreten unter den formenden Händen engagierter Entwickler abzugewöhnen. Der Idealismus des Autors wird für die nötige Unterstützung bis zur Verselbständigung à la volksFORTH auch noch reichen.

Quellen

[1] "Schnelles FORTH für den MC68000", Jörg Plewe, Vierte Dimension VI, Nr. 1, 1990

[2] "Schnelles FORTH für den MC68000, Teil 2", Jörg Plewe, Vierte Dimension VI, Nr. 2, 1990

[3] "FORTHgeschritten", Jörg Staben, J. Plewe, c't 11/90

✉ Jörg Plewe,
Großenbaumer Str. 27,
4330 Mülheim/Ruhr,
✆ 0208/423514

```
\ blockstreams: finding phys. blocks in streams                    JPS 10-05-90

: (phys>log ( dir fblk -- entr1 dir1 entr2 dir2 ... )  [ reveal ]
    2dup containsstream? dup -1 <>
    IF nip swap exit ELSE drop THEN          \ finish recursion
    >r 0 swap
    BEGIN
        nextdir dup -1 <>
    WHILE
        2dup blk>first @  r@  (phys>log dup -1 =
        IF not ELSE -1 THEN
    UNTIL THEN r> drop ;



: phys>log ( blk -- entry1 dir1 entry2 dir2 ....  n  lognr )
  sp@ 4 + >r
  dup finddevice (devRoot  swap StreamFirst >r (phys>log
  sp@ r> r> rot -  8 / swap ;




\ blockstreams user words: change directory                        JPS 10-06-90
Variable whatdevice

: (pwd ( entr1 dir1 entr2 dir2 ... n -- )   [ reveal ]
    ?dup 0= IF whatdevice @ <# #s #> type ascii : emit  exit THEN
    whatdevice @ >r
    over finddevice whatdevice !
    1- -rot >r >r   (pwd
    r> r> ascii / emit blk>name count0 type
    r> whatdevice ! ;

: pwd ( -- )
    directory @ dup devRoot =
    IF finddevice <# #s #> type ascii : emit
    ELSE directory @ phys>log drop (pwd
    THEN ;

: cd ( -- )     \ cd <name>
    name nullstr?
    IF pwd
    ELSE changeDir abort" *** no such directory ***"
    THEN ;



\                                                                  JPS 10-16-90

     also blockstream
: viewlist ( blocknr -- )
    (block dup &2000 +  swap  DO i &80 -trailing cr type  &80 +LOOP ;

: view ( -- )   \ view <name>
    ' 4- code@ dup 0= not
    IF
        dup viewlist dup finddevice formatted?
        IF  phys>log key drop cr ." found in block " .  ." in " (pwd
        ELSE  key drop  cr
            ." found in block " dup dup finddevice (devroot - .
            ." on the unformatted device " finddevice .
        THEN
    ELSE ." no source exists" drop THEN ;


toss
```

(Diese Blocks wurden übrigens mit der oben bereits erwähnten Funktion WRITESYS vom F68K ins GEM-DOS übertragen)

Listing 1: Quelltext von VIEW

PC-FORTH-Plus 3.2

Automatische Dokumentation von FORTH-Programmen

von Claus Vogt, Bülowstr.67, D-1000 Berlin 30

Quelltext Service

Der Artikel stellt ein kurzes Programm vor, das während des Ladens eine Liste aller definierten Worte erstellt und sie zusammen mit ihrem Ort im Quelltext ausgibt. Die Unterstützung der Arbeit eines Programmierers wurde bereits an einem Programm mit ca. 1 MByte Quelle in 50 Screen- und ASCII-Files unter Beweis gestellt. Das Programm wurde unter PC-FORTH-Plus 3.2 entwikkelt, die Anpassung an andere Systeme dürfte aber keine großen Probleme aufwerfen.

Gründe gibt es genug ...

...sich über die automatische – oder vorsichtiger formuliert halbautomatische – Dokumentation Gedanken zu machen. Die Dokumentation wird oft erst nach dem eigentlichen Programmierakt verfertigt, meist widerwillig, im schlimmsten Fall von anderen Personen als den Programmautoren. Oft unterbleibt sie ganz, vielleicht aus Zeitgründen, vielleicht weil die verwendete Programmiersprache bekannterweise selbstdokumentierend ist. So blüht denn grade in der FORTH-Szene haufenweise Code in verborgenen Diskettenkästchen, weil außer dem Autor und seinem besten Freund keiner versteht, was das Programm eigentlich wie macht. Der Rest der Welt vertritt mit guten Argumenten, die These, daß das Schreiben eines eigenen Programms immer noch leichter ist, als das Lesen eines fremden. Die Einfachheit der Bedienung von FORTH war bis 1983 sicher ein Clou, seitdem Borland-Turbo-Derivate Stand der Technik sind, hat FORTH an Reiz verloren.

An welchen Stellen können Dokumentations-Tools ansetzen?

Mögliche Orte/Zeitpunkte zum Ansetzen von Werkzeugen sind:

- ✗ *Das Kompilat*: Der Informationsgehalt eines FORTH-Kompilat ist hoch, das demonstrieren Instrumente wie `WORDS` (bzw. `VLIST`), und Werkzeuge wie Decompiler und Debugger. Am Kompilat setzt Kim R. Harris seinen Cross-Reference-Generator [1] an, der zu jedem Wort jene Worte angibt, in denen es verwendet wird.
- ✗ *Das laufende Programm*: Vielleicht die einzige Stelle, an der über übliche und mögliche Wertebereiche von Variablen Auskunft erhalten werden kann.
- ✗ *Der Quelltext*: Eine Art Filter könnte aus dem Text wichtige Informationen extrahieren. Disziplin im Layouten des Textes und bei der Namensgebung vereinfacht das Schreiben des Filters ungemein. Das Leibnizsystem von Andreas Goppold, aber auch andere Systeme wie comFORTH demonstrieren das eindrucksvoll. Ein Nachteil: Ein Quelltext, der sich zuviel Freiheiten herausnimmt, erfordert einen hochintelligenten Filter. Dieser würde bald Ähnlichkeit mit einem FORTH-Compiler bekommen.
- ✗ *Der Akt des Kompilierens*: Während des Ladens eines FORTH-Programms ist die Informationsdichte sehr hoch. Der Quelltext liegt vor, das Kompilat gleichfalls, die Zuordnung ist vorhanden. Es fehlen lediglich Informationen über die latente Potenz des Programms: mögliche Wertebereiche einer Variablen oder verschiedene Möglichkeiten einen Quelltext abhängig vom Systemzustand verschieden zu laden. Diese Variante soll im weiteren verfolgt werden.

Stichworte

✗ Dokumentation
✗ Patchen

Wie funktioniert der FORTH-Interpreter-Compile?

Während des Ladens eines Quelltextes soll sinnvolle Information über das geladene Programm abgespeichert werden.

Dazu soll kurz die prinzipielle Arbeitsweise eines FORTH-Systems beschrieben werden. Fast jedes FORTH-83-System und auch die meisten anderen arbeiten etwa folgendermaßen:

Mit `WORD` wird ein Wort aus dem Quelltext entnommen. Mit `FIND` wird gesucht, ob es im Wörterbuch ist. Wenn es vorhanden ist, wird es

```
<DOC                                \ Ausgabe auf File umleiten
CR  >IN @ .WORD >IN !  SPACE        \ Name des zu definierenden Wortes ausgeben
.SRC .SRCLIN                        \ Stelle im Quelltext, Quelltextzeile ausgeben
DOC>                                \ Ausgabe auf Console zurückleiten
CREATE                              \ das normale FORTH-Create aufrufen.
```

Listing 1: zentrale Programmsequenz

mit EXECUTE ausgeführt. Wenn es sich innerhalb einer Colondefinition befindet wird es nicht ausgeführt, sondern im Wörterbuch abgelegt, oft mit dem Wort Komma (','). Wenn das Wort mittels FIND nicht gefunden werden konnte, wird das Laden mit einer Fehlermeldung abgebrochen, es sei denn es läßt sich als Zahl interpretieren, was von NUMBER? oder ähnlichen Wörtern geleistet wird. Ein definierendes Wort (wie der Doppelpunkt) ruft CREATE auf, das wiederum mittels WORD das nächste Wort aus dem Quelltext entnimmt und ins aktuelle Definitionvokabular einträgt.

Nur einige wenige Teile des Quelltextes lösen andere Aktionen aus. Das sind jene, die als Kommentar zwischen Klammern oder hinter dem Backslash '\' stehen und jene, die als Zeichenketten abgelegt werden sollen. Sie werden in einigen Systemen auch mittels WORD gelesen, dies ist aber nicht zwingend. Der doppelte Backslash in volksFORTH und PC-FORTH '\\' , der den Rest eines Screens als Kommentar auffaßt (im F83 '\s') wird Block eher aufs Ende positionieren.

Wie läßt sich der Interpreter/Compiler verändern?

Durch Patchen. Patchen heißt auf deutsch »Flicken« und bezeichnet den Vorgang, durch gezielte Veränderung eines Speicherbereichs diesem ein (sinnvolles) Verhalten zu entlocken, das vom Hersteller nicht intendiert war.

Für das Patchen von FORTH-Worten wird im allgemeinen der PFA einer Colondefinition verbogen. Die Definition:

```
: TuWas
   TuerstWAS TuNochWas ;
```

kann durch die Sequenz:

```
: TuNix
  rdrop
  ." ätsch, nix getan" ;
' TuNix ' TuWas >body !
```

dazu so verändert werden, daß bei Aufruf von TuWas nur der Text ausgegeben wird. Würde das 'rdrop' fehlen, so würde nach dem Text noch TuNochWas ausgeführt.

Diese Art des Patchens hat einen Nachteil: Sie arbeitet nur mit Doppelpunktdefinitionen und nur auf einigen (gängigen) Systemen korrekt.

Die folgende Art des Patchens verändert den CFA des Wortes. Die Definition:

```
: TuWas
   TuerstWAS TuNochWas ;
```

kann durch die Sequenz:

```
: CreateTuNix
  Create does>
  drop
  ." ätsch, nix getan" ;
CreateTunNix TuNix
' TuNix ' TuWas cfamove
```

zum selben Verhalten wie oben genötigt werden. Bis auf das Wort CFAMOVE stehen übrigens alle verwendeten Worte im FORTH-83-Standard. Diese Methode funktioniert auch für CODE-Definitionen, Variablen etc., denn jedes Wort hat einen CFA und zwar in jedem FORTH-System. CFAMOVE bewegt eine gewisse Anzahl von Bytes von einer Adresse zur anderen. Bei 16-Bit-Systemen sind es 2 Byte, im PC-FORTH+ sind es 4 Byte, auf exotischen Systemen kann diese Zahl variabel sein. Grundsatz ist: Besser zuviel als zuwenig, denn zumindest Doppelpunkt und Code-Definitionen haben genug Platz. Diese Patch-Methode ist meiner Ansicht nach portabler und sicherer als die erste. Den zusätzlichen Schreibaufwand nehme ich gerne in Kauf.

Was verbiegen wir: CREATE

Das Wort CREATE bietet sich zum Patchen an. Wenn wir voraussetzen, daß jegliche Neudefinition eines Wortes CREATE aufruft, können wir so das Wort, das definiert werden soll und seine Position im Quelltext ausgeben. Damit würde bereits eine VIEW-Funktion möglich. Praktischerweise wird weiterhin die komplette Zeile des Quelltextes ausgeben. Hier befindet sich meist der Stackkommentar oder eine Kurzbeschreibung. Die Ausgabe erfolgt als Liste und kann anschließend alphabetisch sortiert oder anderweitig weiterverarbeitet werden.

Die zentrale Programmsequenz finden Sie in Listing 1. Der Rest des Programmlistings beschäftigt sich mit dem Patchvorgang, dem Eröffnen und Schließen von Files, dem Umleiten der Standardausgabe, und dem Erkennen der Herkunft des Quelltextes: PC-FORTH+ kann außer Tastatur und Screenfiles auch normale Textfiles lesen und das auf eine Art, die etwas aufgesetzt erscheint. Diese Fragen sind systemspezifisch und sollen hier nicht weiter behandelt werden.

Die Bedienung

Nach Laden des Dokumentations-Hilfsmittels können die Worte DOC <filename> und NODOC benutzt werden. Alle zwischen DOC und NODOC definierten Worte ergeben eine Zeile auf dem File <filename>. Die Zeile enthält Name des Wortes, Ort im Quelltext und den Anfang der Zeile, in der es definiert wurde. An das File wird angehängt, es ist also ggf. vorher zu löschen. Das Programm arbeitet mit Screenfiles, ASCII-Files und Tastatureingaben. Bei Screenfiles wird Blocknummer und Offset ausgegeben, bei ASCII-Files nur der Offset, da eine Ausgabe von Zeilennummern

etwas kompliziert ist und besser bei der Nachbereitung geschehen sollte.

Ausblick

Der Artikel stellte eine einfache Methode dar, während des Ladens alle Definitionen von neuen Worten zu erkennen und sie auf einem Dokumentationsfile festzuhalten. Dies wurde durch Patchen des Wortes CREATE erreicht. Ein restriktiveres Textformat der Quelle würde ein leichtes Extrahieren von Handkommentaren ermöglichen, das Konzept läßt sich in wenigen Zeilen darauf erweitern. Mit der vorgestellten Methode läßt sich das Compilerverhalten beliebig verändern. Durch Verändern anderer Worte (WORD EXECUTE , IMMEDIATE COMPILE ...) wäre es möglich, Aufrufhierarchien und Import-Exportlisten zu erstellen oder all jene Worte aufzulisten, die in der fertigen Anwendung nicht benutzt werden. Dies setzt genaue Kenntnisse des verwendeten FORTH-Systems voraus. Es wäre weiter denkbar, Eigenschaften wie: 'definierendes Wort' oder 'verändert den Returnstack' automatisch (mit einer gewissen Trefferquote) zuzuordnen. Hierzu gilt es allerdings vorher eine Vererbungstheorie für diese Eigenschaften aufzustellen (was nicht trivial ist!) und einem Grundwortschatz von Assemblerworten diese Eigenschaften zuzuschreiben. Weitere Anwendungsgebiete sind denkbar.

Literatur

[1] Kim R. Harris: Using the Structure-Tool ... . In: Dr.Dobbs Toolbox of FORTH, Vol II. M&T Publishing: Redward City, California, USA, 1987.

✉ Claus Vogt,
Bülowstr.67,
D-1000 Berlin 30

```
TYPE.L          DOC.4TH     456   : TYPE.L  ( adr count minlen -- )      \ linksbündig in einem
DOCFIL          DOC.4TH     583   CREATE DOCFIL  64 ALLOT  DOCFIL OFF    \ room for DOC-Filenam
<DOC            DOC.4TH     629   : <DOC ( -- )  \ switches output to DOC-FILE
DOC>            DOC.4TH     697   : DOC> ( -- )  \ ends DOC-FILE-output
vWORD           DOC.4TH     855   71 ORIGIN+ CONSTANT vWORD                          \ VECTORED
SRC?            DOC.4TH     929   :  SRC?    ( -- 0 | 1 | 2 )                      \ CONSOLE,
ASCPOS@         DOC.4TH    1267   : ASCPOS@ ( -- D_POSITION )       \ GET FILEPOSITION
ASCPOS!         DOC.4TH    1373   : ASCPOS! ( D_POSITION -- )       \ SET FILEPOSITION
.SRCFIL         DOC.4TH    1473   : .SRCFIL ( -- )                   \ OUTPUT FILENAME
.SRC            DOC.4TH    1588   : .SRC ( -- )                      \ OUTPUTS CURRENT SOURCE-
.SRCLIN         DOC.4TH    1847   : .SRCLIN  ( -- ) \ outputs the actual inputline
.WORD           DOC.4TH    2134   : .WORD ( -- ) \ outputs following word of inputstream
CFAMOVE         DOC.4TH    2246   : CFAMOVE ( cfa1 cfa2 -- ) \ moves cfa1 to patch cfa2
oCREATE         DOC.4TH    2355   CREATE oCREATE    ' CREATE ' oCREATE CFAMOVE   \ save old cfa
nCREATE         DOC.4TH    2434   CREATE nCREATE                                 \ later filled
paCREATE        DOC.4TH    2486   : paCREATE ( -- )    \ patches CREATE with nCREATE
upCREATE        DOC.4TH    2579   : upCREATE ( -- )    \ restores original CREATE
cCREATE         DOC.4TH    2650   : cCREATE ( -- ; name )
pCREATE         DOC.4TH    2890    cCREATE pCREATE  ' pCREATE ' nCREATE CFAMOVE
DOC             DOC.4TH    3000   : DOC  ( -- ; docfile )   \ subsequent loading produces docum
NODOC           DOC.4TH    3117   : NODOC ( -- )             \ ends documentation mode
```

Diese Liste wurde durch Verändern des Wortes CREATE erreicht. Zum Erstellen der Liste wurde die am Ende vom Programmlisting befindliche Befehlssequenz benutzt.

Programmlisting

```
\
\  DOC.4TH 1.Version - Documentation-tool for PC-FORTH+3.2
\
\  Usage:
\
\  DOC filename    - enables doc-output during subsequent loading
\                    appends doc-output to <filename> if present
\  NODOC           - returns to normal mode
\
\
\  Author: Claus Vogt, Bülowstr.67, D-1000 Berlin 30
\

DECIMAL

   \ output primitives

   : TYPE.L  ( adr count minlen -- )      \ linksbündig in einem minlen langen Feld ausgeben
      OVER - >R
      TYPE
      R> SPACES
    ;

   CREATE DOCFIL  64 ALLOT  DOCFIL OFF     \ room for DOC-Filename

   : <DOC ( -- )  \ switches output to DOC-FILE
      DOCFIL ->>FILE ;

   : DOC> ( -- )  \ ends DOC-FILE-output
      CONSOLE ;

  \ WHICH IS THE PRESENT INPUT STREAM

  71 ORIGIN+ CONSTANT vWORD                    \ VECTORED WORD OF PC+3.2

  :  SRC?    ( -- 0 | 1 | 2 )                 \ CONSOLE, BLOCK, OR ASCII-LOADING
```

```
      vWORD @ [ vWORD @ ] LITERAL =  IF 2 EXIT THEN   \ WORD patched like now --> Ascii
      BLK @                          IF 1 EXIT THEN   \ BLK not 0 --> Screenfile
      0 ;                                             \ else console

\ SAVE AND OUTPUT SOURCE-POSITION

: ASCPOS@ ( -- D_POSITION )        \ GET FILEPOSITION
   SCRHCB HCB>H  0.0  1 lseek  >IN @ SOURCE NIP _ "_ S>D D+ ;

: ASCPOS! ( D_POSITION -- )        \ SET FILEPOSITION
   SCRHCB HCB>H  -ROT 0 lseek  2DROP ;

: .SRCFIL ( -- )                   \ OUTPUT FILENAME
    SCRHCB HCB>N COUNT 13 TYPE.L ;

 : .SRC ( -- )                     \ OUTPUTS CURRENT SOURCE-POSITION
   BASE @ DEZIMAL
   SRC?
   CASE
   0  OF ." ** from console **" >IN @ 5 U.R ENDOF
   1  OF .SRCFIL BLK @ 5 U.R ." /" >IN @ 5 U.R ENDOF
   2  OF .SRCFIL ASCPOS@ 10 D.R ENDOF
   ENDCASE
   SPACE
   BASE !
;

: .SRCLIN  ( -- ) \ outputs the actual inputline
    SOURCE
    SRC? 1 =
    IF DROP                        \ Blocks have lines of 64 characters
       >IN @ [ HEX ] FFFFFFC0 [ DECIMAL ] AND
       +
       64
    THEN
    64 MIN
    TYPE
 ;

: .WORD ( -- ) \ outputs following word of inputstream
   BL WORD COUNT 32 TYPE.L ;


\ patching

: CFAMOVE ( cfa1 cfa2 -- ) \ moves cfa1 to patch cfa2
   4 CMOVE ;
\ patch CREATE

CREATE oCREATE    ' CREATE ' oCREATE CFAMOVE   \ save old cfa
CREATE nCREATE                                 \ later filled with new cfa

: paCREATE ( -- )    \ patches CREATE with nCREATE
   ['] nCREATE  ['] CREATE CFAMOVE ;
: upCREATE ( -- )    \ restores original CREATE
   ['] oCREATE  ['] CREATE CFAMOVE ;

: cCREATE ( -- ; name )
   CREATE DOES> DROP upCREATE

   \ this is what happens:

   <DOC
   CR  >IN @ .WORD >IN !  SPACE
   .SRC .SRCLIN
   DOC>
   CREATE

   paCREATE ;
 cCREATE pCREATE  ' pCREATE ' nCREATE CFAMOVE

\  EXCISE cCREATE pCREATE

: DOC  ( -- ; docfile )   \ subsequent loading produces documentatuion on docfile
   FEED DUP DOCFIL OVER C@ 1+ CMOVE
   paCREATE ;

: NODOC ( -- )             \ ends documentation mode
   upCREATE
   DOCFIL OFF
;

\ Vorsicht: PC-Forth+ kann offenbar nichtmal verschachtelte INCLUDEs


\ Ein Beispiel: Im Test das sich selbst dokumentierte Programm

\  INCLUDE DOC.4TH
\  DOC DOC.DOC
\  INCLUDE DOC.4TH
\  NODOC
```

BÜCHERECKE

von Arndt Klingelnberg

Zöller, Horst
FORTH in der Automatisierung: Einsatzmöglichkeiten und Anwendung
Horst Zöller und Heiko Loewe
Düsseldorf: VDI-Verlag 1990
274 Seiten 21*14.5cm ISBN 3-18-401056-2 DM 68.-

Das handliche Buch kommt von schulungserfahrenen Praktikern (VDI-Seminare und {RW}TH) und wendet sich an Techniker, um ihnen ihre Labor-/Versuchsarbeiten zu vereinfachen oder die Automatisierung technischer Prozesse zu ermöglichen.

Die Autoren bringen fundierte Kenntnisse aus anderen Programmierumgebungen mit, so daß sie nicht über die Mauer schauen müssen, um auch einmal über C, ADA, Smalltalk, SPS oder objektorientierte Ansätze zu diskutieren. Sie können FORTH als eine (wenn auch oft sehr sinnvolle) Möglichkeit unter vielen bewerten. In anderen Büchern kaum gefundene Einblicke in Ansteuerung von A/D-Wandlern, Multitasking, verteilte Systeme, Laboranwendungen geben dem Praktiker wichtige Anregungen und Hilfen.

Negative Kritikpunkte ergeben sich dort, wo ungenau bzw. falsch OR NOT ... in boolschem Kontext angewendet werden, wo FORTH doch gerade aufgrund seiner Bit-Operationen für den Unerfahrenen eindeutige Hilfe fordert. Auch wird der Nicht-Informatiker z.B. auf S. 72 mit dem Begriff 'Nibble' konfrontiert, ohne ihn entsprechend aufzuklären.

Der wesentlichste Schwachpunkt ist jedoch wie in so vielen Computerbüchern der Textsatz, der bei Quelltexten das wirklich Profihafte vermissen läßt. Durch ' BL WORD ' potenziert FORTH dieses Problem. So gut der allgemeine Text lesbar ist, so ist FORTH in Proportionalschrift ein Unding. Blanks, Blanks, Blanks ! muß man dem hier angesprochenen Newcomer-Leserkreis entgegenschreien, jeder noch so kleine Zwischenraum ist zu erahnen und zu expandieren, sonst wird FORTH zu FRUST.

Sinnvolle FORTH-Beispiele, einleuchtende Schemata für Stack, Speichernutzung, Ablaufstrukturen und Auswahltabellen runden das Buch positiv ab.

Kein Lehrbuch und auch nichts für die abgehobenen Spären der Systemdiskussionen, aber eine Basis-Lektüre für Anwender, die eine Programmiersprache für Bereiche suchen, in denen und für die FORTH ursprünglich erarbeitet wurde: Forschungslabors, dann insbesondere die Prozeßsteuerung (damals: Radio-Teleskope) inklusive der Erfassung und Bewertung von Rohdaten (back to the roots).

Vack,
Programmieren mit FORTH
Ergänzung, Korrektur zu VD V Nr.3 September 1990

Der Preis hat sich im Zuge der Wiedervereinigung um 20% auf DM 42.- erhöht. Die Lesbarkeit des Textes leidet etwas unter dem Matrixdrucker, verwirrend ist der dominierende Anteil an fig-FORTH. Lobenswert ist allgemein die systematische Aufbereitung, das mehrfache Register, das sehr umfangreiche Literaturverzeichnis, die Hinführung zu professionell fundiertem Programmieren (um einmal von den hobbyistischen Sourcen wegzukommen).

Auch bei 42 DM immer noch ein sicherlich empfehlenswertes Basiswerk für Leute, die unter FORTH auch Anwendungen mit professionellem Anspruch erstellen möchten, nur eben nicht (als alleiniges Buch) für den Anfänger. Wird auf (sauberes) Deutsch Wert gelegt, ist dieses Buch bei der sehr eingeschränkten Konkurrenz sicherlich ein 'Muß'.

Arndt Klingelnberg

ANZEIGEN

Da auch wir nicht allein von Luft und Liebe existieren können, ist es möglich, Anzeigen in der 'Vierten Dimension' zu plazieren. Ist der Leserkreis vergleichsweise nicht sehr umfangreich, so werden doch im Gegensatz zu anderen Zeitschriften nur wirklich Interessierte und Fachkundige angesprochen. Deshalb lohnt es sich auf alle Fälle eine Anzeige in der 'Vierten Dimension' aufzugeben. Über Preise und alle weiteren Modalitäten können Sie sich bei der Redaktion informieren.

Lokale FORTH-Gruppen, die sich regelmäßig treffen:

1000 Berlin	Claus Vogt, ☎ 030/2168938. Treffen am letzten Donnerstag des Monats um 19.30 Uhr in der Technischen Universität Berlin, Mathematikgebäude, 6.Stock im Raum MA 621
4130 Moers 1 Rhein-Ruhr	Friederich Prinz, näheres Tel: 02841/583 98 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, ☎ 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, ☎ 06271/2830, Ewald Rieger, ☎ 06239/8632. Treffen jeden ersten Mittwoch im Monat im Vereinslokal des Segelflugvereins Mannheim e.V. Flugplatz, Mannheim-Neuostheim.
7000 Stuttgart	Lokale Gruppe Stuttgart, Wolf-Helge Neumann ☎ 0711/882638 und Ulf Katzenmaier, ☎0711/268293.
8000 München	Heinz Schnitter, ☎ 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).
O-Leipzig	FORTH-Gruppe Leipzig, Michael Balig, Lützner Plan 17, O-7033 Leipzig oder Dr. Jürgen Hesse, Lieselotte-Herrmann-Str. 40, O-7050 Leipzig, ☎ 041-69 56 02

FORTH-Fachgruppen:

8000 München	RTX 2000 Gruppe, Koordinator Herr Krämer, 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.
5000 Köln 60	Michael Heycke, Boltensternstr.
4830 Gütersloh 1	Ludwig Röver, Holzheide 145A
5110 Alsdorf	Arndt Klingelnberg, ☎ 02404/61648, voraussichtlich zusammen mit der Computergruppe RWTH Aachen

Eine Fachgruppe will gründen:

7000 Stuttgart 80	Grafik/Arithmetik, Jörg Tomes, Anweilerweg 56, ☎ 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
05221/23504	Andreas Findewirth, Im Großen Vorwerk 48, 4900 Herford
030/3965227	Andreas Jennen, Berlin, UUCP

Ansprechpartner zu bestimmten Interessengebieten:

volksFORTH/ultraFORTH:	Klaus Kohl,	☎ 08233/30524
	Bernd Pennemann,	☎ 0228/640979
	Klaus Schleisiek-Kern,	☎ 040/2202539.
32-Bit Systeme:	Robert Jones,	☎ 02434/4579
Künstliche Intelligenz:	Ulrich Hoffmann,	☎ 0431/678850
NC4000 Novix Chip:	Klaus Schleisiek,	☎ 040/6449412
Realtime & Petri-Netze:	Wigand Gawenda,	☎ 040/446941
Gleitkomma-Arithmetik:	Andreas Döring,	☎ 02631/52786
32FORTH	Rainer Aumiller,	☎ 089/6708355
PostScript/FORTHscript	Christoph Krinninger,	☎ 089/725 93 82
FORTH im Unterricht	Rolf Kretzschmar,	☎ 02401/4390
Objekt-orientiertes FORTH	Christoph Krinninger,	☎ 089/725 93 82
	Ulrich Hoffmann,	☎ 0431/678850
Amiga - MULTI-FORTH	Rafael Deliano,	☎ 089/841 83 17
F-PC Zimmer FORTH, ASYST, Meßtechnik	Arndt Klingelnberg,	☎ 02404/61648, box:geo1:klingelnberg

FORTH-Gesellschaft e.V. - Postfach 1110 - D-8044 Unterschleißheim
☎ 089/3173784, FORTH-Mailbox ☎ 08841/5880
Postgiroamt Hamburg, Kontonr.: 563211-208 BLZ 20010020

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

FORTH-SYSTEME

ANGELIKA FLESCH

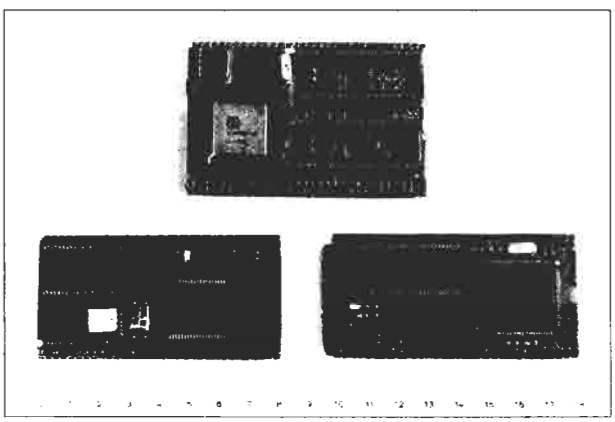
Postfach 1103
D-7814 Breisach

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