Volume V/Nr.1 - März 1989

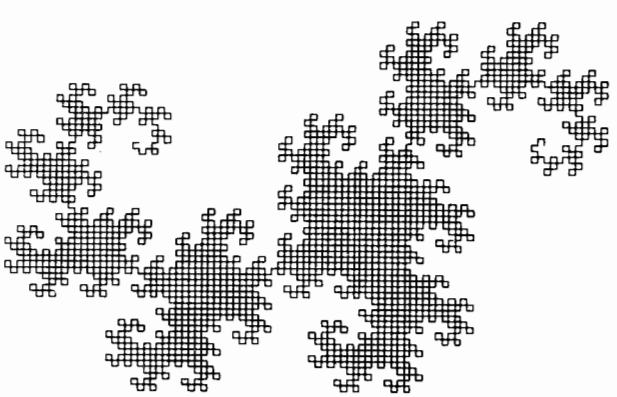
FORTH FM Magazin

VIERTE DIMENSION

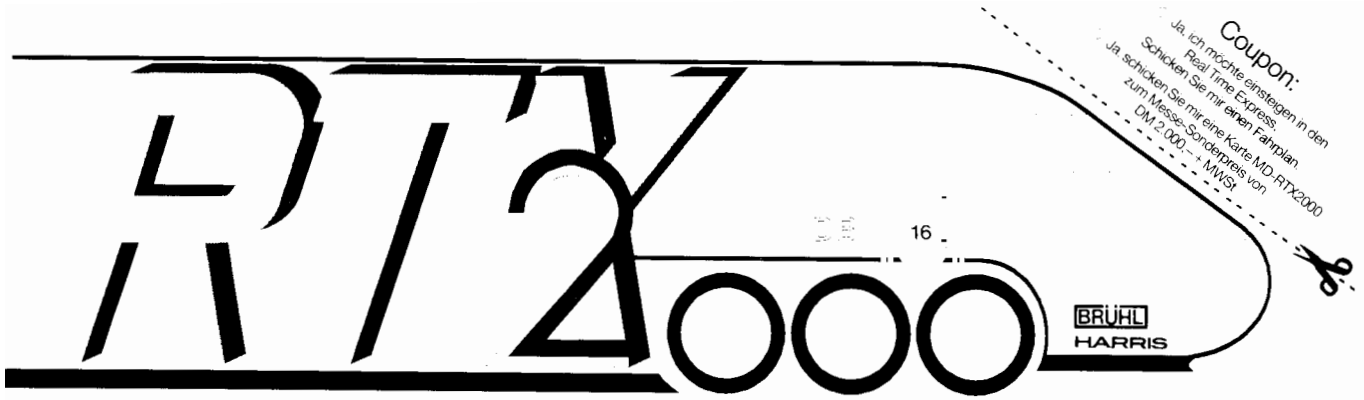
THEMEN

- **SWOPILER**
- **Datenstrukturen in FORTH**
- **WISC und das FORTH-Dilemma**
- **DRAGON - fraktale Graphik**
- **Das Mini-BEE-2000 System**

Einzelpreis DM 7,50



FORTH FG Gesellschaft e.V.



Coupon:
Ja, ich möchte einsteigen in den Real Time Express.
Schicken Sie mir eine Karte MD-RTX2000
Ja, schicken Sie mir einen Fahrplan zum Messe-Sonderpreis von DM 2.000.– + MWSt.

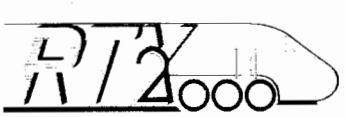
Die Zukunft hat schon begonnen ....

Warten Sie nicht auf Transkarbid oder Grande Tristesse!
Steigen Sie jetzt ein in den Real Time Express.
Nur mit dem Real Time Express gewinnen Sie Echt Zeit.
Sie brauchen Ihr Fahrzeug nicht mehr mit Assembler zu tunen.
Genießen Sie das Hochgefühl mit Hochsprache und Echt Zeit Ihr Ziel zu erreichen!

FORTH

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

REAL TIME EXPRESS und RTX ist TradeMark der HARRIS CORPORATION, Palm Bay, Florida MINIREF und QuadCard sind Trademark der BRÜHL ET GmbH, Nürnberg



Karten zum Einsteigen erhalten Sie ab sofort bei:
BRÜHL ELEKTRONIK ENTWICKLUNGS-GESELLSCHAFT mbH, Hegelstraße 10, 8500 Nürnberg 10, Tel. 0911/359088
QuadCard 100×100 mit RTX2000, 128 KB RAM, 64 KB PROM,
40 MHz Quarztakt: DM 2.000.– + MWSt (electronica Sonderpreis incl. FG-FORTH-Compiler)

DELTA t

Die Firma mit dem FORTH - KNOW - HOW

Seitdem es in Echtzeit geht wird der Zufall immer greifbarer.

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

5. 4. – 12. 4. 1989
HANNOVER MESSE INDUSTRIE 89
HALLE 7 STAND A 46/B 45

Ulrich Hoffmann Marina Kern Klaus Schleisiek
DELTA t Entwicklungsgesellschaft für computergesteuerte Systeme mbH
Telefon 040 / 644 57 82 · Roter Hahn 42 · D - 2000 Hamburg 72

Anzeige

EDITORIAL - IMPRESSUM

EDITORIAL

Sie halten das erste Exemplar der 'Vierten Dimension' des Jahres 1989 in Händen. In diesem Frühjahr stehen einige empfehlenswerten Messen und Tagungen auf dem Programm. Vom 08. bis 15. März findet die CeBIT in Hannover statt. Im April und zwar vom 05.04. bis 12.04. ist die Hannover Messe. Und natürlich nicht zu vergessen die Mitgliederversammlung der FORTH-Gesellschaft e.V. am 09. April in Aachen, sowie die FORTH-Tagung vom 07.04. bis 09.04. in Aachen. Bei diesen Gelegenheiten können sicherlich viele Anregungen gesammelt werden, die hoffentlich in vielen Artikeln für die VD ihren Niederschlag finden.

FORTH FM Magazin

In diesem Heft finden Sie u.a. Artikel über ':DOES> -- Datenstrukturen in FORTH', 'Floating-Point-Einbindungen für's IBM-volksFORTH' und 'SWOPILER - Ein Generator für Stackoperator-Worte'. Aus den FORTH DIMENSIONS haben wir einen interessanten Artikel über 'WISC und das FORTH-Dilemma' übersetzt. Und auch sonst bietet die vorliegende VD allerhand anregenden und unterhaltsamen Lesestoff.

Denise Luda

Rainer Aumiller

IMPRESSUM

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

Herausgeber:
FORTH-Gesellschaft e.V.

Redaktion:
D. LUDA Software, Staudingerstr. 65, 8000 München 83, Tel. 089/670 83 55

Kontaktadresse:
Entweder direkt die Redaktion anrufen bzw. anschreiben oder das FORTH-Büro in München, Postfach 1110, 8044 Unterschleißheim, Tel.: 089/3173784 kontaktieren.

Autoren dieser Ausgabe:
Jörg Staben, Klaus Schleisiek, Ulrike Schnitter, Christoph Krinninger, Frank Raschke, Frank Stüss, Andreas Findewirth, Ulrich Paul, Andreas Zilker, Ulrich Hoffmann. Übersetzungen: D. Luda. Graphiken teilweise Rolf Kretzschmar.

Erscheinungsweise:
Vierteljährlich

Redaktionsschluß:
Die zweite Woche im mittleren Quartalsmonat

Auflage:
ca. 1000 Stück

Druck:
Druckservice Ost GmbH, Pfanzeltplatz 4, 8 München 83

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

Vierte Dimension

Inhalt

LESERBRIEFE und Zuschriften

Betrifft: volksFORTH

Zunächst mal herzlichen Glückwunsch zu neuer Form und neuem Inhalt der Zeitschrift. Hoffentlich haltet Ihr das hohe Niveau durch. Sowas kostet ja wahrscheinlich auch eine Menge Geld.

Nun aber zum eigentlichen Thema meines Briefes. Meine Freude war groß, als ich Eure (kleine) Ankündigung las, Ihr wollt das volksFORTH gründlich überarbeiten und in FG-FORTH umbenennen. Bei meiner Anpassung des volksFORTH an den C16 fielen mir auch einige überarbeitungswerte Dinge auf (später schlugen Benutzer/innen weitere vor). Leider befand sich die Autorengruppe zu diesem Zeitpunkt in jenem Zentrifugal-Loch in das wohl alle unbezahlte Projekttätigkeit nach all' zu langem Kreisen um ein Thema fällt. Das Interesse an weiterer volksFORTH-Arbeit war gering.

Mein Hauptinteresse geht in Richtung auf einen einheitlichen, dokumentierten und wartbaren Stand der vier volksFORTH-Versionen, der insbesondere Einsteigern und Umsteigern den Umgang vereinfacht. Änderungen in der Grundphilosophie halte ich nur in solchen Bereichen für nötig, wo sie wichtige neue Anwendungsgebiete eröffnen (z.B. ROM-Fähigkeit).

Einige Vorschläge:

- Wiederholt merkten Benutzer/innen an, eine Erklärung oft benutzter Begriffe, wie 'Loadscreen', 'Targetcompiler' etc. fehle.
- In Assemblerquellen sollten die Mnemonics - wie oft in Lehrbüchern üblich - ein Komma am Ende des Namens haben. Das erleichtert Einsteigern das Leben ungemein, auch wenn es den Autoren etwas Tipparbeit macht.
- Umsteiger würden sich - wie ich aus eigener Erfahrung weiß - freuen, wenn die Editoren der vier Versionen einheitlich zu bedienen wären. Ebenso sollten die Programmnamen gleich sein, um einen einheitlichen Aufruf zu haben. Derartige Eigenbrötelei einzelner Autoren kann man nur als ärgerlich bezeichnen.
- Die Dokumentation sollte für Einsteiger lesbarer werden. Vielleicht hilft auch eine Kennzeichnung aller Worte, die beim Einstieg überlesen werden können.
- die ROM-Fähigkeit ist bei professioneller Anwendung dringend nötig.
- Aufsätze für spezielle Anwendungen, wie z.B. ein Floating-Point-Paket oder Schnittstellen zu anderen Programmiersprachen werden immer wieder gewünscht.
- In Hinsicht auf sehr kleine und sehr große Anwendungen sollte etwas geschehen. Erstere würden sich eine Entschlackung des (mit 12-18 KB) doch etwas verfetteten Kerns wünschen. Für letztere wäre ein einfaches Modulkonzept als Aufsatz denkbar. (Wer weiß ein Konzept?)
- Last not least, sollte das Copyright geklärt werden. Meiner Ansicht nach, sollte (analog zur FIG-USA) die freie Verwendung gestattet werden. Die nötigen Geldmengen für Support und Weiterentwicklung könnten ggf. aus dem Vertrieb einzelner Aufsätze und von Spezialsystemen (die HP-PC's schreien geradezu danach) kommen, falls das Spendenaufkommen der begeisterten Industrie nicht ausreichen sollte.

Ich hoffe, hiermit ein wenig zur 'Standard'-Diskussion innerhalb der FORTH-Gesellschaft e.V. beigetragen zu haben. Vielleicht habe ich auch den einen oder die andere dazu angespornt eigene Vorschläge ebenfalls niederzuschreiben.

Claus Vogt
Bülowstr. 67
1000 Berlin 30

Gründung einer FORTH-Gruppe im Raum Herford

Ich beschäftige mich seit einiger Zeit mit FORTH und bin seit einigen Monaten Mitglied der FORTH-Gesellschaft e.V.

Ich würde gerne mit einigen Leuten aus meiner Gegend (Bielefeld-Herford-Minden-Lippe) eine lokale FORTH-Gruppe gründen und bin bereit, mich dafür als Ansprechpartner zur Verfügung zu stellen.

Spezielle Schwerpunkte habe ich (noch) nicht. Ich studiere Informatik mit Nebenfach BWL an der Fernuni in Hagen. Als FORTH-System benutze ich volksFORTH auf dem Atari ST.

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

FORTH-POSTSCRIPT

Als Graphik-Designer und Dozent an der FH Augsburg/FB-Gestaltung arbeite ich häufig mit Postscript. Dabei wäre es sehr nützlich, die Interpretierung zwecks Kontrolle nicht immer auf dem PS-Laserdrucker ausgeben zu müssen, sondern am Bildschirm.

Mit anderen Worten: Display-Postscript in volksFORTH auf dem Atari ST täte gut! Da ja Postscript wohl ein Abkömmling von FORTH ist, sollte das, mindestens was die gängigen Graphik-Operationen betrifft, kein allzu großes Problem für Könner sein. Leider bin ich (noch) keiner. Zwar kann ich 'swap' in 'exch', 'drop' in 'pop' verwandeln, doch spätestens bei den Prozeduren und Kontrollstrukturen verlassen den einfachen FORTH-Fußgänger die Kräfte ...

Dabei könnte ich mir vorstellen, daß so ein Programm in graphischen Bereichen (auch in der MS-DOS-Welt) durchaus verkäuflich wäre. Oder vielleicht könnte das ein Beitrag in ST-Computer oder c't werden (FORTH täte da auch mal gut).

Noch ein paar Vorschläge zum Ganzen:

- Schriftausgabe ist natürlich schwierig, könnte aber der Einfachheit halber auch als Tonflächen erfolgen.
- Als Anflugsmaßstab würde sich 1 Point = 1 Pixel anbieten, damit ließe sich das amerikanische Legal-Format (612 * 792 Points) auf 2 Atari-Bildschirmen darstellen.

Literatur:

1. Adobe Systems, Postscript, 3 Bände Addison-Wesley, Bonn 1988.

Prof. Ulrich Harsch
Lilienthalstr. 5
8900 Augsburg

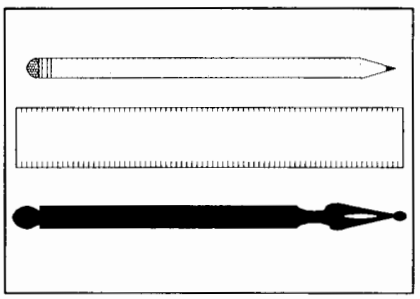
Hinweise für Autoren

Auch in Zukunft möchten wir Beiträge veröffentlichen, die Sie uns hoffentlich in großer Zahl liefern werden. Schicken Sie Ihre Manuskripte bitte an die Redaktion der 'Vierten Dimension' D.LUDA Software Staudingerstr. 65, 8000 München 83, Tel. 089/6708355.

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

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

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



FORTH-PD-Pool aufgelöst

Leider muß ich mitteilen, daß ich den FORTH-PD-Pool nicht weiterführen kann.

Sven Lütkemeier
Kölner Heerweg 11
5140 Erkelenz 1

Inserentenverzeichnis:

FORTH-Workstation

von Dirk Brühl

Diesen Arbeitstitel (FORTH-Workstation) möchte ich einem neuen Projekt geben, das dem Einsatz und der Demonstration der Leistungsfähigkeit von FORTH weiteren Auftrieb geben wird.

Die Vorgeschichte:

Christoph Krinninger hat das Wort von der 'FORTH-Workstation' beim 'Mittwochstreff' der Münchner FORTH-Gruppe in die Runde geworfen, nachdem er in einem Vortrag PostScript vorgestellt hatte. Diese Anregung ist auf fruchtbaren Boden gefallen. Nachdem es schon Heinz Schnitter gelungen ist, eine Kooperation zwischen drei unabhängigen Ingenieurbüros, allesamt Mitglieder der FORTH-Gesellschaft e.V. (die sich bis vor kurzem nicht kannten), zu initiieren, aus der die MB-RTX2000-Prozessorkarte als Singleboardcomputer der deutschen FORTH-Gesellschaft e.V. entstanden ist, soll nun dieser Synergie-Effekt genutzt werden und ein Projekt in Angriff genommen werden, an dem alle Interessierten teilzunehmen aufgerufen sind.

Das Projekt:

Die FORTH-Workstation soll alle Hardware-Bestandteile einer modernen Workstation beinhalten: Farbbildschirm mit High-Resolution Farbgrafik, Festplatte mit XX Megabyte, Volltastatur, Maus-, Drucker- und Kommunikations-Schnittstellen. Als Betriebssystem ist FORTH vorgesehen (was sonst), Gateways zu anderen Betriebssystemen (z.B. MS-DOS) sollen jedoch zur Erleichterung des Diskettenaustausches entwickelt werden. Als Bus-System für die Workstation sind bisher VME-Bus (wie bei der SUN-Workstation) und der PC-XT/AT-Bus (wie bei PC-Workstations) im Gespräch. Für den PC-XT/AT-Bus spricht, daß äußerst preisgünstige Peripheriekarten (Harddisk-Controller, Farbgrafik-Controller) für diesen Bus auf dem Markt angeboten werden.

Zur FORTH-Tagung 1989 (7.-9. April in Aachen) soll Näheres zum Projekt besprochen werden. Der Prozessor, mit dem die Workstation arbeiten soll, steht allerdings schon fest: Natürlich ein FORTH-Prozessor!

Der Prozessor:

Der Kern der FORTH-Workstation existiert bereits: Die RISC-Prozessorkarte MB-RTX2000-MINI (siehe VIERTE DIMENSION Nr.2/3 September 1988, Seite 7 'Fachgruppe RTX-2000', Seite 47 'Der RTX 2000™' und VIERTE DIMENSION Nr.4 Dezember 1988, Seite 9 'Die Zukunft hat schon begonnen ...'). Ich habe die Leistungsfähigkeit dieser Prozessorkarte im Wettbewerb mit einem 68030-Spezialisten geprüft und festgestellt, daß die MB-RTX2000 in ihrer Arbeitsgeschwindigkeit bei 6,7 MHz Prozessortakt in der Ausführung eines FORTH-Programms (ohne Assembler) einem 68030-Prozessor mit 24 MHz Prozessortakt bei der Ausführung eines in 68030-Assembler geschriebenen Maschinenprogramms programmsim Cache-Modus (ohne Memory-Zugriff) entspricht. Wenn man davon ausgeht, daß das Verhältnis der Laufzeiten von FORTH-Programmen zu Assembler-Programmen in der Regel etwa 10:1 ist, kann man mit gutem Gewissen sagen, daß der RTX2000 gegenüber dem oben angegebenen Prozessor in FORTH die zehnfache Leistungsfähigkeit hat. Damit zeigt es sich, daß es sich von dieser Seite her lohnt, den RTX2000 als Workstation-Prozessor einzusetzen. Wir haben auch andere Vergleiche gemacht: die MB-RTX2000 Prozessorplatine ist mit dem FG-FORTH etwa um den Faktor 200 schneller als der weitverbreitcte R6511Q (8Bit-65XX-Singlechip-Prozessor) mit 2 MHz-Systemtakt und RSC-FORTH.

Die Mitwirkenden:

Zur Mitwirkung am Projekt 'FORTH-Workstation' sind alle Mitglieder der FORTH-Gesellschaft e.V. herzlich eingeladen. Wie Eingangs schon erwähnt, soll der sich abzeichnende Synergie-Effekt in der FORTH-Gesellschaft e.V. genutzt werden, um einen weiteren Schritt vorwärts zu kommen.

Benötigt werden (FORTH-) Software- und Hardware-Spezialisten, die Interesse am Projekt 'FORTH-Workstation' haben und bereit sind, an diesem Projekt mitzuarbeiten und dieses Projekt mitzugestalten. Es wäre schön, wenn wir uns in Aachen zusammensetzen könnten, um Nägel mit Köpfen zu machen. Die ersten Schritte sind schon getan: Ein MSDOS-Interface für den RTX2000 und ein Anschluß für eine Harddisk über den PC-XT-AT-Bus (mit PC-OMTI-Controller) sind schon in Arbeit. Da die 'FORTH-Workstation' als virtuelle Workstation praktisch auf allen Rechnersystemen arbeiten kann, ist die Entwicklung in gewissen Grenzen hardwareunabhängig. Für die Hardwareansteuerung der FORTH-RISC-Workstation ist allerdings die Arbeit mit dem RTX2000 erforderlich.

Weitere Anregungen:

Weitere Anregungen sind herzlich willkommen. Es gibt viel zu tun. Packen wir's an! Die Mitwirkenden sollen in loser Folge in der 'VIERTE DIMENSION' mit ihrer Arbeit vorgestellt werden - zum einen, um mitzuteilen, wer am Projekt 'FORTH-Workstation' arbeitet, zum anderen, um über den Stand des Projekts zu berichten, und, last not least, weitere Mitglieder der FORTH-Gesellschaft e.V. zu animieren, am Workstation-Projekt teilzunehmen. Es gibt nichts Gutes - außer man tut es.

Lokale Gruppe Berlin im Aufbau begriffen

Die Notsitzung der lokalen Gruppe Berlin (im Aufbau) vom 25.1.89 war ein voller Erfolg. Die 14 Teilnehmer diskutierten lange Zeit über Vorstellungen und Ansprüche, die sie an eine solche Gruppe haben und ergriffen entsprechende Vorbereitungen. Handfeste Ergebnisse: Ein Anfängerkurs wird vorbereitet, der Austausch von Soft- und Paperware wird in Angriff genommen, drei Treffen (jeweils mit Kurzreferat) sind geplant und nicht zuletzt wurde die schlechte Kassensituation per Spontansammlung behoben.

Eingeladen hatten vier gefrustete FORTHler zur 'Auflösung' der lokalen Gruppe. Sie legten gleich ein 'fiktives' Jahresprogramm bei für den Fall, daß hinreichendes Interesse für eine gleich folgende Neugründung vorhanden sei. Das Interesse war vorhanden.

Nach dem obligatorischen Grußwort ('Wer ist die FORTH-Gesellschaft e.V., was ist eine lokale Gruppe?') stellten sich die Teilnehmer kurz vor. Sie kommen von Universitäten, der Physikalisch-Technischen Bundesanstalt, aus diversen Betrieben (wo sie teils beruflich mit FORTH befaßt sind, sogar ein einschlägiger Kleinunternehmer war anwesend) oder sind reine Hobbyisten.

Die teilweise recht kontroverse Diskussion, die folgte, drehte sich um die ganze Welt der FORTH-Programmierung: das Schreiben (oder Nichtschreiben) neuer Editoren, die Eignung von FORTH als Diagnoseinstrument (Expertensystem), die objektorientierte Programmierung, die (Nicht-)Notwendigkeit von Fließkommaarithmetik, FORTH-Prozessoren und das automatische Zählen von Münzen, die eine Schräge hinabrollen. Selbstverständlich kamen auch der Austausch von Produkt-News und die Probleme der Anfänger nicht zu kurz.

Die handfesten Ergebnisse:

- 45.- DM für die Kasse (handgezählt). Der letzte Rundbrief ist somit finanziert, der nächste wird vermutlich in 2-3 Monaten verschickt werden, falls bis dahin ein weitergehendes Konzept existiert.
- ein Anfängerkurs wird vorbereitet. Er kann vermutlich nur stattfinden, wenn sich weitere Anfänger bereitfinden. Eventuell wird er in Zusammenarbeit mit der Volkshochschule stattfinden, zumindest wird ein entsprechender Teilnehmerbeitrag erhoben. (Kontakt: Uwe Linker, 304 81 51)
- allgemein wurde das Bedürfnis nach einer Softwaretool-bibliothek und einer (Papier)-Artikelbibliothek laut. Konkrete Festlegungen wurden aber erstmal vertagt.
- die nächsten drei Termine werden Kurzreferate zu folgenden Themen enthalten:
  Do 23.2.1989: Integer-Sinus-Berechnung in FORTH (Neupert).
  Do 30.3.1989: 68HC11-Single-Chipper mit F83 (Blancke).
  Do 27.4.1989: FORTH in Hardware (Ulbrich).

Die Treffen finden jeweils am letzten Donnerstag des Monats um 19.30 Uhr in der Technischen Universität Berlin (Str.d.17.6.) im Mathematikgebäude im 6. Stock im Raum MA 621 statt.

Wir laden alle FORTH-Interessenten zu den Treffen der lokalen Gruppe Berlin (i.A.) ein, und würden uns über weitere Vorschläge und aktive Beteiligung freuen. Kontakt: Helge Horch (.....) Claus Vogt 216 89 38

Invitation & call for papers

euroFORML conference
on the FORTH programming language
and FORTH processors

from October 13th through 15th 1989
at Hotel Selau next to Forth near Nuremberg,
Federal Republic of Germany

sponsored by
"FORTH-Gesellschaft e.V., FRG"
and "Forth Interest Group Inc, USA"

euroFORML will be an international meeting of computer practitioners using FORTH as a problem solving tool. Lectures, workshops and presentations are planned to demonstrate techniques and problem solving strategies that have proved useful.

This years conference will specifically focus on FORTH in real time applications and its potential in the area of robotics and industrial control applications.

The conference will be held at Hotel Selau in Neunkirchen am Brand next to the village Forth 20km to the West of Nuremberg. Hotel Selau has all the usual facilities of a modern Hotel including a swimming pool and a sauna. The Hotel can accomodate 72 people in 33 double and 6 single rooms - additional single rooms will be available in the immediate vicinity.

The conference language will be English and FORTH, of course. The conference is supposed to be self organizing ie. there will not be a strict agenda prior to the conference. If you want to present your ideas you may choose one of the following formats:

Paper presentation

You will present a paper in a 10 minute talk in front of the whole group with the possibility to get immediate feedback.

Poster presentation

You will be assigned a "poster space" where you can present your ideas to a small group of people in a separate room. This is especially useful for demonstrations of hard- and software.

Workshops

You may organize or participate in workshops which will be organized at the beginning of the conference, depending on demand.

An English language proceedings will be published after the conference; papers to be included in the proceedings will be handed out at the beginning of the conference.

Registrations should be mailed by September 1st. A DM 250,- deposit per person is required. (Money order in German funds, eurocheques in foreign funds or transfer to Postgiro account# 5226 46 - 203, bank code 200 100 20). The full amount is due at the beginning of the conference. Space is limited and you are assigned on a first-come, first-serve basis.

We reserve 1/3 of the available space for students/low income at a reduced rate. Accomodation in a single room is DM 40,- extra.

Author instructions

Papers to be presented at the conference (to be included in the proceedings) will have to be mailed to FORTH-Gesellschaft eV no later then October 1st 1989 in camera ready form. The format is DIN A 4 (letter size) with a margin of 2,5 cm (1.5") on all sides. Every page should have a page number and the authors name. Papers should not exceed 15 pages. Code should be accompanied by shadow sereens.

For reservations and conference papers write or call

M.Kern / c/o DELTA t GmbH / Roter Hahn 42 / D-2000 Hamburg 72 / W-Germany / (+49) (40) 644 5782

Direct your questions which are not related to the conference to

FORTH-Gesellschaft eV / Postfach 1110 / D-8044 Unterschleiheim / W-Germany / (+49) (40) 317 3784

		FORTH- Gesellschaft e.V. member rate
Participant (accomodation, three meals per day and conference proceedings)	**DM 600,-**	**DM 560,-**
Guest (accomodation and meals)	**DM 400,-**	**DM 400,-**
Student (limited openings) (accomodation, meals, conference proceedings)	**DM 300,-**	**DM 280,-**
"external" Participant (meals, conference proceedings)	**DM 450,-**	**DM 420,-**

Accomodation in a single room DM 40,- extra.

:DOES> -- Datenstrukturen in FORTH

von Jörg Staben, Hilden und
Klaus Schleisiek, Hamburg

Nach dem geltenden Standard ist es in bestimmten Situationen notwendig, für das Compilezeit- und das Laufzeitverhalten eines Wortes zwei getrennte Worte zu definieren mit den daraus resultierenden Problemen. Ohne den Umweg über 'definierende Worte' zu gehen, soll es daher möglich sein, in einem Wort sowohl das Compilezeit- als auch das Laufzeitverhalten zusammenzufassen.

Die anzustrebende Syntax ist:

```
Create     <name>
           <compiletime>
:Does>     <runtime> ;
```

Damit fügt sich das neue Wort **:DOES>** nahtlos in die Reihe der Worte ein, die in volksFORTH für das Bearbeiten von Datenstrukturen zur Verfügung stehen:

```
Create    Create:
:Does>    Does>
```

Vorab zu den Begriffen:

Jedes Programm besteht aus einem Datenteil und einem Befehlsteil. Der Datenteil beschreibt den statischen Datenbereich, der die zu bearbeitenden Daten enthält. Datenbereiche sind durch die Interpretation ihres Inhaltes strukturiert, wobei Datenbereiche mit gleicher Strukturierung (= Datenstrukturen) zum gleichen Datentyp gehören. Der Befehlsteil enthält die zu Algorithmen angeordneten Befehle, wobei die Befehle selbst sich wiederum aus Operanden und Operatoren zusammensetzen.

Die Strukturierung und Verwaltung sowie die Interpretation des Dateninhaltes im Datenteil wird in PASCAL und seinen verwandten Sprachen am Programmanfang durch die Deklaration des Datentyps, in FORTH dagegen erst im Befehlsteil durch die eingesetzten Befehle bestimmt.

Da in FORTH die Typisierung über die Auswahl und den Einsatz von geeigneten Operatoren erfolgt, gehören in FORTH alle die Datenstrukturen dem gleichen Typ an, denen die Operatoren gemeinsam sind. Deshalb bezeichne ich ein 'definierendes Wort' mit Datenteil und Befehlsteil als Datentyp. Alle damit erzeugten Wörter gehören dem gleichen Typ an, weil ihnen die Operatoren gemeinsam sind.

Um das oben genannte Problem und seine Lösung zu verdeutlichen, möchte ich mich an der Aufgabenstellung von Wil Baden orientieren: Weil es oft vorkommt, daß innerhalb eines größeren Programmes überprüft werden soll, wie oft ein bestimmter Programmteil aufgerufen wird, ist für diese Aufgabe ein eigenes Wort zu definieren. Dieses Wort übergibt bei seinem Aufruf die Anzahl der bisherigen Aufrufe zur Auswertung. Es soll also als Zähler für seine eigenen Aufrufe arbeiten. Das Zählen beginnt sinnvollerweise bei NULL und als Name soll ZÄHLER gewählt werden. Dieses Wort würde man in FORTH so beschreiben:

```
Zähler ( -- Zählerstand )
```

Zur Lösung dieser Aufgabe wird man einen geeigneten Datenteil in der Standardgröße des Systems - hier auf dem PC 2 Byte - einrichten:

```
Variable Zähler Zähler off
```

Auf diesen Datenteil wird dann mit geeigneten Operatoren zugegriffen, um die genannte Aufgabenstellung zu erfüllen:

```
Zähler @ 1 Zähler +!

Zähler dup @ 1 rot +!
```

Zweckmäßigerweise faßt man diese sorgsam ausgetestete Zugriffsprozedur in einem einzelnen Operator @+ + zusammen:

```
: @+ + ( adr -- n )
   dup @ 1 rot +! ;

Variable Zähler Zähler off
Zähler @+ +
```

Immer noch sind Datenteil und Ausführungsteil voneinander getrennt. Dies bietet normalerweise die größere Flexibilität, weil der Operator @+ + für jede passende Datenstruktur eingesetzt werden kann. Ein Zusammenfassen der Datenteils und des Operators ist immer dann sinnvoll, wenn es sich um eine abgeschlossene Struktur handelt, auf die nicht von außen zugegriffen werden soll.

Stichworte:

» **Datenstrukturen,**
» **Laufzeitverhalten.**

Datenstrukturen in FORTH

```
Variable counter counter off

: Zähler  ( -- Zählerstand )
    counter @
    1 counter +!
;
```

Bei diesem Zähler handelt es sich um eine abgeschlossene Struktur mit einem genau festgelegten Verhalten. Dieses Verhalten wird durch das Wort @+ + bestimmt.

```
Variable counter counter off

: Zähler ( -- Zählerstand )
    counter
    @+ +
;
```

Allerdings ist der so eingerichtete Datenteil global; der Zugriff mit ungeeigneten Operatoren und aus anderen Wörtern heraus ist möglich, obwohl der Datenteil COUNTER nur für das Wort Zähler bestimmt ist. Nach [1] sollen "Objekte, die nur innerhalb einer Prozedur verwendet werden, prinzipiell lokal deklariert werden. Damit werden einerseits Objekte vor nicht zugriffsberechtigten Programmteilen versteckt. Andererseits können irrtümliche Verwendungen von (globalen) Objekten weitgehend vermieden werden."

Unbefriedigend ist an diesem letzten Ansatz auch, daß für EINE Datenstruktur ZWEI Wörter definiert werden müssen.

An diesem Punkt wird es wünschenswert, dem statischen Datenteil des Zählers den notwendigen Ausführungsteil fest zuordnen zu können.

Eine Lösung bietet R. ZECH in seinem Buch FORTH83 an:

```
Create Zähler
    0 ,              \ 2 Byte allot,
                     \ 0 init
    here ' : @ ,     \ compile NEST
                     \ für EXECUTE
    ]                \ compilemode
Does>                ( -- Zählerstand )
    @+ +             \ count und
                     \  increment
    exit
    [ execute        \ execute DOES>
```

Statt dieses etwas undurchsichtigen Verbiegens des **DOES>** führt die Notwendigkeit für die Kontinuität der Datentypen in FORTH zu einem definierenden Wort für einen einmaligen Anlaß **:DOES>** .

Hier die Definition für das volksFORTH der FORTH-Gesellschaft:

```
\  :Does>  für  Create  <name>  :Does>  ... ;    ks 25 aug 88

|  :  (does>  here  >r  [compile]  Does>  ;

   :  :Does>  last  @  0=  Abort" without reference"
      (does>  current  @  context  !  hide  0  ]  ;

\  :Does> leitet den Befehlsteil für den zuletzt geschaffenen
\  Datenteil ein.
```

Damit werden die anfänglichen Versuche zur Lösung der Aufgabenstellung so umgesetzt:

```
Create Zähler
    0 ,          \ mit 0 initialisieren
:Does>           \ pfa auf den stack
    dup @        \ akt. Zählerstand
    1 rot +!     \  inkrementieren
;
```

Darüberhinaus kann **:DOES>** bereits vorhandenen Datentypen ein neues Laufzeitverhalten geben, indem der Ausführungsteil umgesetzt wird. Dies führt zu der klareren Definition:

```
Variable Zähler Zähler off
            :Does> @+ + ;
            ( -- Zählerstand )
```

Ebenso kann einem Datenbereich ein Ausführungsteil zugewiesen werden:

```
: >Var: Create , ;
0 >Var:  Zähler
    :Does>   @+ +   ;
```

Die nächste Stufe der Datenstrukturen, die allerdings von der Aufgabenstellung nicht mehr gefordert wird, ist die Schaffung des neuen Datentyps **ZÄHLER:** als definierendes Wort:

```
: Zähler:
Create 0 ,          \ Initialisierung
Does>               ( -- Zählerstand )
    dup @           \ count
    rot +! ;        \ increment
```

Oder man faßt die ausgearbeitete Problemlösung in dieser Form zusammen, indem man unkorrekterweise einen Nebeneffekt im volksFORTH ausnutzt:

```
: Zähler:
    Variable
    Does> @+ + ;

Zähler: Eingabe
Zähler: Auswertung
```

EINGABE und **AUSWERTUNG** sind beides Datenstrukturen vom gleichen Typ, weil sie eine gemeinsame Zugriffsprozedur haben.

Wie sinnvoll das neue Wort **:DOES>** ist, wird deutlich, wenn mit Tabellen gearbeitet werden soll. So gibt es z.B. nur eine SINUS-Tabelle, als Datentyp ist die SINUS-TABELLE unsinnig.

Auch wenn nur ein Byte-Ringpuffer [2] im Programm benötigt wird, bewährt sich **:Does>** :

```
Create Queue 0 c, 255 allot
    :Does> exit q@ q! q? ;

Queue put Queue get
Queue show
Queue count type
```

Wird dagegen der Datentyp RINGPUFFER erforderlich, greift die wieder die Konstruktion mit

```
: Queue:
    Create 0 c, 255 allot
    Methods> q@ q! q? ;

Queue: Tastatur
Tastatur put Tastatur get Tastatur
show
Tastatur 10 >expect
```

So kann **:Does>** auch den **METHODS>** Ansatz von Terry Rayburn [3] um ein **:METHODS>** ergänzen. Damit wird das Folgende möglich, denn es kann in einem Programm vielleicht mehrere Datenstrukturen vom Typ BYTE-RINGPUFFER geben, aber doch nur eine ZEIT:

```
: :Methods>  [compile]
   :Does> compile exit ;
immediate

Create Time
   :Methods> time@ time! time?
;

Time put Time get Time show
```

Die oben vorgestellte Implementierung von **:DOES>** gilt ausschließlich für's volksFORTH ; es ist interessant, diese Lösung auch in anderen FORTH-Systemen verwirklicht zu sehen. Schickt deshalb bitte weitere Implementierungen für andere FORTH-Systeme an die 'Vierte Dimension' !

Literatur:

1. Die PASCAL-Fibel, Strukturierte Programmierung in PASCAL,S.96; Witschital u. Kühme.
2. VD87 III/1, S.15; Klaus Schleisiek.
3. VD87 III/3, S.26 ff; Terry Rayburn.
4. Wil Baden, FORML84, NONCE DEFINING WORDS

Screen # 1

```
\ Die Queue - Struktur & Operatoren                jrg 13okt88

\ Create Queue   0 c,  255 allot


: more?    ( adr -- n )  c@ ;

: q@   ( -- char )
dup more? IF  detract exit
ELSE  ." Ringpuffer leer! "  drop
THEN ;

: q!   ( char -- )  1 arguments
append ;

: q.   count type ;
```

Screen # 2

```
\ Integer: 2Byte Variable  Queue: 255Byte Ringpuffer jrg 13okt88

: Integer:
Variable
Methods>   @ ! u? ;

Integer: Wert


: Queue:
Create   0 c,  255 allot
Methods>   q@ q! q. ;

Queue: Tastatur
```

Screen # 3

```
\ Datentyp ZEIT   Zugriffsprozeduren                jrg 12okt88

: time@  drop time@ ;

dos
|      dos: time! time "          \ Zeit setzen
forth
: time!  drop time! ;

: time.
time@
3 0 DO
2 .r Ascii : emit
LOOP
;
clear
```

Screen # 4

```
\ Der abstrakte Datentyp ZEIT                       jrg 12okt88

Create Time    :Methods>   time@ time! time.  ;
```

Floating-Point-Einbindung für's IBM-volksFORTH

von Frank Raschke, Frank Stüss

Basierend auf dem Standard der FORTH vendors group haben wir eine Floating-Point-Einbindung für's IBM-volksFORTH entwickelt. Sie bedient sich des 80(2)87-Prozessors.

Um das volksFORTH auch für flp-Anwender zu erschließen, und da die mathematischen Koprozessoren 80(2)87 mittlerweile relativ preisgünstig geworden sind, lag es nahe diese Möglichkeit für das IBM-volksFORTH zu nutzen. Mit dem unten beschriebenen Paket lassen sich leicht numerische Probleme in FORTH realisieren. Diese Version benutzt für FP-Zahlen den Standard-FORTH-Stack, um größere Geschwindigkeitsverluste zu vermeiden. Dabei wurde durchweg das 64-Bit Format genutzt. Der 80(2)87 rechnet zwar intern mit 80-Bit Genauigkeit, jedoch geht diese bei der Ein-/Ausgabe verloren. Der Prozessor-Stack wird nur intern benutzt, da eine Erweiterung auf den Speicher laut Entwickler des Chips [1] sehr schwierig und ineffizient ist.

In Fehlersituationen kann der 80(2)87 einen Interrupt auslösen. Bei XT-Kompatiblen liegt dieser fast immer auf dem NMI der CPU. Dies stellt aber kein Problem dar, da die Interrupt-Ursachen im 8087 einzeln maskierbar sind. Beim 80287 liegt der Interrupt auf dem IRQ13 des 2. Interrupt-Controllers des AT. Hier ist es zusätzlich nötig, die Interrupts der Int.-Controller zu demaskieren und den beiden Controllern nach jedem Interrupt ein EOI zu senden. Zur Unterscheidung dieser Fälle existiert ein Flag IBMAT, welches - je nach System (XT/AT) - vor dem Einladen entsprechend gesetzt werden muß. Sowohl die Fehlerbehandlung (FPERROR), als auch die Fehlerausgabe (FP.ERROR) sind als DEFERRED-Words implementiert.

Als Zusatz zum Standard haben wir das Datenformat ARRAY eingeführt, das ein eindimensionales Feld darstellt. Da sich bei sinnvollen Anwendungen das restliche 'Dünn'--tionary doch recht schnell erschöpft, haben wir die Arrays in den erweiterten Adreßraum jenseits des FORTH-Segments gelegt. Dadurch sind, je nach Speicherkonfiguration, 50000 Einträge und mehr möglich. Um das Laden und Speichern größerer Arrays effektiv zu gestalten, wurde mit den Worten LOADARRAY / SAVEARRAY die Möglichkeit geschaffen, sie als Speicherblock direkt auf Disk zu schreiben. Es ist so z.B. möglich, eine 200X200-Einheitsmatrix als 'Filekonstante' zu definieren. Weitere Zugriffe auf den erweiterten Adreßraum bieten die Worte FL!/FL@.

Beschreibung des Befehlssatzes des Floating-Point-Pakets

1. Arithmetische Funktionen:

Alle arithmetischen Funktionen erwarten ein reelles Argument (64 Bit) auf dem FORTH-Stack, eine Ausnahme bilden die Fakultät FNFAC, welche einen positiven Integer erwartet.

F+, F-, F*, F/ tun genau das, was man erwartet

FSQRT zieht die Quadratwurzel

FABS bildet den Absolutwert

FNEGATE negiert

1/F bildet das Reziprok

FLOG Zehner-Logarithmus

FLN Logarithmus naturalis

FRP2 bildet das Quadrat

FRP3 bildet den Kubus

FALN berechnet exp(r)

Stichworte:

» Floating Point,
» volksFORTH,
» 80(2)87

Floating-Point-Paket für's IBM-volksFORTH

FALOG berechnet 10^r

F** (r1 r2 --> r1^r2) potenziert

FROOT (r1 r2 --> r1^(1/r2)) zieht die r2-te Wurzel

FNFAC (n --> r) bildet die Fakultät (integer !)

FMOD (r1 r2 --> r1 mod r2) Modulo-Division. (Bem.: Diese ist GENAU)

FMIN, FMAX Minimal- oder Maximalwert

2. Trigonometrische Funktionen:

Bei den Trigonometrischen Funktionen werden je nach Einstellung von TRIG-MODE die Argumente in Radianten oder Grad verstanden. Ebenso erfolgt die Ausgabe bei den Arcusfunktionen dementsprechend.

FSIN, FCOS, FTAN, FCOT, FASIN, FACOS, FATAN tun wieder genau das, was man erwartet.

DEG schaltet den Grad-Modus an.

RAD schaltet den Radiant-Modus an.

DEG->RAD wandelt Grad-Zahl in Radiant-Zahl.

RAD->RAD wandelt Radiant-Zahl in Grad-Zahl.

TRIG-MODE Flag, ob Grad oder Radiant (TRUE = Radiant)

3. Hyperbolische Funktionen:

FSINH, FCOSH, FTANH, FCOTH Sinus,Kosinus,Tangens und Kotangens hyperbolicus

FSECH, FCOSECH Hyperbelsecans und -cosecans

FASINH, FACOSH, FATANH Hyperbolische Arcusfunktionen.

4. Stackmanipulation:

Folgende Worte sind implementiert:

FSWAP, FDUP, FDROP, FROT, FNIP, FTUCK, FPICK, FOVER, F-ROT

5. Umwandlung zwischen Zahlenformaten:

R>S wandelt Realzahl in Integerword; Fehlermeldung wenn Realzahl zu groß.

S>R umgekehrt.

R>D wandelt Realzahl in Doubleword; Fehlermeldung, wenn Realzahl zu groß.

D>R umgekehrt.

Dabei wird die Rundungsart verwendet, welche in 87MODE (siehe unten) eingestellt wurde.

FLOAT wandelt Doubleword in Realzahl, jedoch wird der Dezimalpunkt in DPL berücksichtigt.

6. Conditionals:

F=, F<, F>, F<=, F>=, F<>, F0=, F0<, F0>, F0<=, F0>=

7. Rundung:

FFIX rundet 'normal' (zur nächsten ganzen Zahl).

FINT rundet 'floored' zur nächsten kleineren ganzen Zahl.

FRND-INF rundet nach -

FRND+INF rundet nach +

8. Konstanten:

Es sind verschiedene Hardware-Konstanten (im 8087 eingebaut) vorhanden:

FPI, F0, F1

FLN(2) logarithmus naturalis von 2

FLOG(2) Zehnerlogarithmus von 2

FLOG2(10) Zweierlogarithmus von 10

FLOG2(E) Zweierlog. von e

Folgende Konstanten wurden softwaremäßig eingebaut:

F2, F3, F4, F10, F-1, FE (= Euler-Zahl)

INFINITY + unendlich, vielmehr die größte darstellbare positive Zahl.

2PI, PI/4

9. Datentypen:

FCONSTANT, FVARIABLE, ARRAY

10. Speicherzugriff:

F! (r addr -->)

F@ (addr --> r)

FL! (r seg:addr -->)

FL@ (seg:addr --> r)

11. Systemvariablen & -konstanten:

variable FPSTAT (siehe Fehlerbehandlung)

variable 87MODE (ebenfalls)

constant F#BYTES enthält Anzahl der Bytes für eine FP-Zahl (8).

variable F#PLACES enthält die Anzahl signifikanter Stellen bei der Ausgabe

variable #PLACES enthält die Anzahl der Dezimalstellen, welche mit F. ausgegeben werden sollen.

variable TRIG-MODE Flag, ob die Argumente trigonometrischer Funktionen in Radianten oder Grad angegeben sind

12. Eingabe von Zahlen:

FNUMBER (string --> r) wie Number, jedoch für Realzahlen.

FIN# (--> r) holt interaktiv eine Realzahl.

Floating-Point-Paket für's IBM-volksFORTH

FLITERAL (r -- >) compile-time bzw. (-- > r) run-time

FEXTENSION (string -- > r) wird an die Stelle von NOT-FOUND vektorisiert, um fp-Zahlen aus dem Input-Strcam zu filtern.

13. Ausgabe von Zahlen:

E. gibt die Zahl auf dem TOS in 'wissenschaftlicher' Notation, d.h. mit allen signifikanten Stellen und Exponent aus. (z.B.: -.123456789098765432e-98)

E.R (r dig col -- >) tut dies rechtsbündig mit (dig) Dezimalstellen.

F. gibt eine Zahl in 'normaler' Schreibweise ohne Exponent und mit Dezimalpunkt an der entsprechenden Stelle aus. Es werden dabei soviele Dezimalstellen berücksichtigt, wie #PLACES angibt. (z.B.: 3145345770000000000000000000000000000000.0000 oder -98798978.4445 bei #PLACES = 4)

F.R wie e.r jedoch ohne Exponent.

PLACES (n -- >) initialisiert #PLACES. Etwa 4 Places bewirkt z.B. obige Ausgabe.

14. Sonstige numerische Befehle:

FMAG (r1 -- > n) übergibt den Exponenten.

FPARSE (r -- > vorkomma nachkomma) Zerlegt eine Realzahl in Vorkomma- und Nachkommaanteil.

FRNDFRC (r -- > r) rundet eine Realzahl auf #PLACES Stellen.

PACK (d n -- > d ^ n) 'packt' einen double und einen integer zu einer Realzahl

UNPACK (r -- > d n) umgekehrt

15. Fehlerbehandlung:

Laut Standard (FVG) sollten folgende Worte vorhanden sein: Variable FPSTAT, welche den Implementationsabhängigen Fehlerstatus des letzten Fehlers beinhaltet, und deferred FPERROR welches FPSTAT setzt und die Realzahl auf dem Stack hinterläßt, welche zum Fehler führte. Wir haben hier noch eine vektorisierte Fehlerausgabe (FP.ERROR) angebunden. Wie oben erwähnt erfolgt der Aufruf von FPERROR entweder durch den 8087-Interrupt, oder explizit durch Aufruf z.B. bei der Argumentenkontrolle.

16. Steuerungsworte:

FPCW > holt Controlword vom 8087 zum FORTH-TOS

> FPCW umgekehrt

FPSW > holt 8087-Statusword

CLRFP Löscht die Interrupt-Anforderung im 8087

FP > R holt 8087-Tos zum FORTH-Tos

87MODE ist eine Variable, welche den Inhalt des 8087-Steuerregisters darstellt.

(initfp ist ein Assembler-Makro, welches in fast allen FP-Befehlen benutzt wird. Dabei wird der Inhalt der Variable 87MODE in das Steuerregister des 8087 übertragen.

INITFP Das ganze in High-Level-Form.

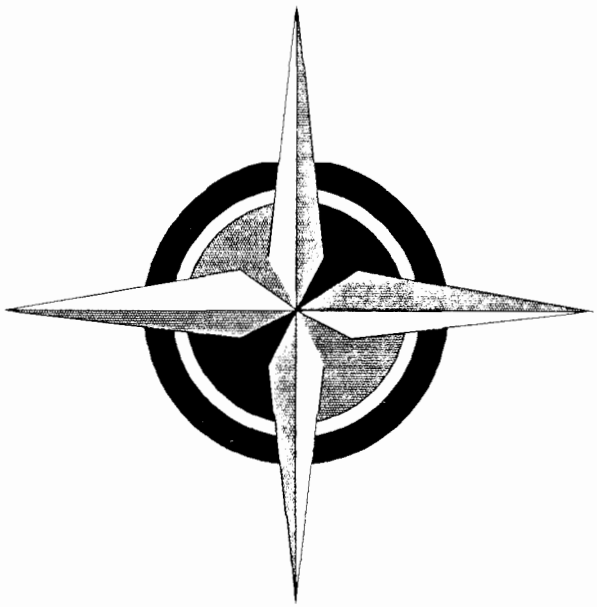
FLOATING ist einerseits das Vokabular, andererseits der Einschaltbefehl. Er initialisiert den 8087 und verbiegt den Interrupt. (ähnlich ASSEMBLER)

Quellen:

[1] Palmer,Morse: "Die mathematischen Grundlagen der Prozessoren 8087, 80287" TEWI-Verlag

[2] Steve Pollack: "8087 Floating Point Support" Orange County FIG 28.11.84

[3] Bronstein - Semendjajew "Taschenbuch d. Mathematik" Harri Deutsch, Frankfurt



FORTH - Der Richtungsweiser

ADMINISTRATION

Neues und Altes vom FORTH-Büro:

Anfragen nach früheren Ausgaben des FORTH-Magazins VIERTE DIMENSION

In Unterschleißheim haben wir noch folgende Ausgaben der VIERTEN DIMENSION vorrätig:

- Ausgabe 1985 Volume I Nummer 2 à DM 5,00
- Ausgabe 1986 Volume II Nummer 2, 3 und 4 à DM 5,00
- Ausgabe 1987 Volume III Nummer 1, 2 und 3 à DM 5,00
- Ausgabe 1988 Volume IV Nummer 1 pro Heft DM 5.00
- Volume IV Nummer 2/3 Doppelheft DM 10.00 und Nummer 4 neuer Preis DM 7.50
- Ausgabe 1989 Volume V Nummer 1 DM 7.50

Jedes Exemplar kann zum jeweiligen Preis plus Porto bei uns angefordert werden. Weitere VD's gibt es beim Kopierservice: Klaus Schleisiek, Roter Hahn 42, 2000 Hamburg 72

Lokale Gruppen und Fachgruppen:

Wir freuen uns über die folgenden Neugründungen!

*** Lokale Gruppe Rhein-Neckar ***

Am 07.12.1988 fand die Gründungsversammlung der Rhein-Neckar - Gruppe statt.

- Lokaler Koordinator: Thomas Prinz, Telefon: 06271-28 30
- Stellvertreter: Ewald Rieger, Telefon: 06239-68 32
- Treffpunkt: Jeden ersten Mittwoch im Monat im Vereinslokal des Segelflugvereins Mannheim e.V. Flugplatz 6800 Mannheim-Neuostheim

*** Lokale Gruppe Berlin ***

am 25.01.1989 fand die Gründungsversammlung der Berliner Gruppe statt.

- Lokaler Koordinator: Claus Vogt, Telefon: 030-216 89 38
- Stellvertreter: Helge Horch, Telefon:
- Treffpunkt: Jeweils am letzten Donnerstag des Monats um 19:30 Uhr in der Technischen Universität Berlin, Mathematikgebäude, 6. Stock im Raum MA 621.

Wir wünschen den neuen Gruppen viel Erfolg!

Mitgliedsbeiträge:

Auch für 1989 benötigen wir von Mitgliedern, die den ermäßigten Beitrag entrichten, einen Nachweis für diese Vergünstigung.

Mitgliederliste:

Dieser Ausgabe liegt eine Mitgliederliste der FORTH-Gesellschaft e.V. bei. Sie ist ausschließlich zum privaten Gebrauch bestimmt.

Ergänzung der Tagesordnung der Mitgliederversammlung 1989 in Aachen

Bei einer Überprüfung der laufenden Ausgaben der FORTH-Gesellschaft e.V. haben wir festgestellt, daß die Erstellung und der Versand des FORTH-Magazines VIERTE DIMENSION pro Mitlgied und Jahr ca. DM 40.00 beträgt. Das Direktorium ergänzt deshalb die Tagesordnung der Mitgliederversammlung 1989 um den folgenden Punkt:

- Erhöhung der Mitgliedsbeiträge ab 1990 und für neue Mitglieder ab Mai 1989.

Unser Vorschlag:

- ermäßigter Beitrag von HEX 20 auf 30 DM DECIMAL **DM 48,-**
- ordentliche Mitglieder von HEX 40 auf 50 DM DECIMAL **DM 80,-**
- fördernde Mitglieder von HEX 80 auf 0A0 DM DECIMAL **DM 160,-**

ok

Ch. Krinninger J. Reilhofer H.F. Schnitter

Leserbriefe und Kleinanzeigen:

Zuschriften für unsere Leserbriefecke fördern als offene Diskussionsbeiträge die Kommunikation der Mitglieder in der FORTH-Gesellschaft, deshalb freuen wir uns über jede Zuschrift.

In der VIERTEN DIMENSION können ab sofort Mitglieder kostenlos Kleinanzeigen aufgeben. Für Nichtmitglieder berechnen wir eine Mindestgebühr von DM 5,00 für die ersten 5 Zeilen, jede weitere Zeile DM 1,00.

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

Ulrike Schnitter
-FORTH-Büro-

Einladung

zur ordentlichen Mitgliederversammlung

der FORTH-Gesellschaft e.V.

am Sonntag, den 9. April 1989 ab 10:00
an der RWTH Aachen, Steinbachstr. 54, 5100 Aachen

Ab 7. April findet das jährliche FORTH-Treffen statt. Auch in diesem Jahr gibt es wieder interessante Vorträge über FORTH-Anwendungen in Wissenschaft und Industrie.

Tagesordnung:

1. Rechenschaftsbericht des Direktoriums
2. Kassenbericht und Jahresbilanz 1987 und 1988
3. Aussprache und Entlastung des Direktoriums
4. Berichte aus den lokalen - und Fachgruppen
5. Mitgliederversammlung 1990
6. Erhöhung der Mitgliedsbeiträge ab 1990 und für neue Mitglieder ab Mai 1989
7. Verschiedenes

Ergänzungen zu den Tagesordnungspunkten können schriftlich bis März 1989 beim FORTH-Büro beantragt werden.

WISC und das FORTH-Dilemma

Glen B. Haydon - La Honda, Kalifornien
Übersetzung aus "FORTH Dimensions" Vol. X , Nr. 3: D.Luda

Die FORTH-Philosophie lautet: Die Dinge so einfach wie möglich halten. Sie wissen ja, daß etwas Elegantes sehr unkompliziert sein kann. Einfachheit bedeutet nicht gleich Primitivität. Die Händler fühlen sich aber leider nicht verpflichtet dem Käufer alle Utilities anzubieten, an die er sich gewöhnt hat. Es liegen also unvereinbare Voraussetzungen vor. Das Dilemma für die FORTH-Gesellschaft besteht darin, daß sie nicht weiß wem sie es recht machen soll. Auch für die Firma WISC Technologies bestand die Schwierigkeit zu entscheiden, wie sie ihre neue Computer-Architektur gestalten sollte.

Einfachheit ist Trumpf

Wie die meisten von Ihnen wissen, entstand FORTH bei der Entwicklung einer Echtzeit-Applikation der "rug factory". Das Problem bestand damals im zugrunde liegenden Betriebssystem. Was fehlte war ein einfaches kleines Programm, welches das Betriebssystem umgehen sollte. Ganz allgemein kann gesagt werden: Je einfacher ein Programm ist, um so leichter kann es erstellt und bedient werden. Falls 64Kb Speicherplatz nicht ausreichen, haben Sie das Problem nicht verstanden. Es ist sicherlich nicht effizient Programme zu schreiben, die den gesamten Speicherplatz in Anspruch nehmen.

Das Problem

Die mangelnde Akzeptanz von FORTH unter den Programmierern, liegt vielleicht an deren Forderung alle Tools zur Verfügung zu haben, die sie von anderen Programmiersprachen her kennen. Obwohl die WISC Architektur an einem Punkt angelangt war, an dem sie in der Lage war, FORTH zu 'verstehen' und durch das leicht zu schreibende Instruction-Set zu einem leistungsfähigen allgemeinen Prozessor wurde, bestand nur wenig Neigung es zu übernehmen, da die Beziehung zu FORTH hätte aufgezeigt werden müssen und eine Entwicklungsumgebung hätte aufgebaut werden müssen, um die Forderungen der Programmierer zu erfüllen.

WISC hatte immer das Ziel im Auge, die Hardware so einfach wie möglich zu halten. So weit wie möglich wurde die Verbindung der Hardware-Komponenten mit Instruktionen dem Programmierer überlassen. Deshalb wurde ein Software-gebundenes System benutzt.

Durch die Entwicklung unserer Produkte büßte das Design an Einfachheit ein. So wuchs z.B., als wir vom System CPU/16 zum System CPU/32 übergingen, notwendigerweise die Komplexität des Designs. Es ist natürlich viel einfacher einen einzelnen Bus zu leiten, als mehrere Busse, besonders, wenn jeder Bus 32-Bit groß ist. Als wir zur Maschinenimplementation in Silizium übergingen, wurden andere Fähigkeiten hinzugefügt. Bei jeder Ergänzung mußte die Balance zwischen gewachsener Komplexität und weitestgehender Einfachheit gewährleistet sein.

Ein neuer Ansatz

Ist ein Projekt so weit fortgeschritten, daß es anderen zugänglich gemacht werden kann, werden auch andere Anforderungen gestellt. Das System muß den Anforderungen derer gerecht werden, die es benutzen. Genau wie bei jedem anderen Lehrprogramm, muß man immer auf den schon vorhandenen Kenntnissen und den Erwartungen der Lernenden aufbauen.

In diesen Fall ist der Lernende ein Programmierer, der sich mit Computersystemen schon auskennt. Er erwartet ein Betriebssystem. Er erwartet einen Editor oder eine Art Wortprozessor. Er erwartet eine umfangreiche Library mit nützlichen Hilfsfunktionen. Er erwartet einen leistungsfähigen Debugger. Wie sollte er auch ein Programm schreiben können, das nicht gedebuggt werden muß? Wie aus einigen Referaten der Konferenz ersichtlich ist, können einige Programmierer ganz gut mit der Tastatur umgehen und benutzen diese am liebsten, andere arbeiten lieber mit Icons und der Maus und wieder andere arbeiten sowohl mit der Maus, als auch mit der Tastatur. Jedes Entwicklungssystem das all' diese Anforderungen erfüllt, kann

WISC und das FORTH-Dilemma

niemals mit den Zielen der FORTH-Philosophie in Einklang gebracht werden.

Programmier-Psychologie

Das Betriebssystem kann als symbiotische Sprache aus Programmiersprache und der Sprache des Programmierers verstanden werden. Probleme tauchen auf, falls der Programmierer sein System nicht beherrscht. So ist z.B. jemand, der nicht gut mit der Tastatur umgehen kann, auch nicht in der Lage Keyboard-Befehle effektiv zu nutzen. Und jemand, der kein Geschick beim Umgang mit der Maus hat, wird diese immer als Behinderung empfinden.

Diese Einschränkungen ergeben sich größtenteils aus der individuellen Verknüpfung der Nerven und der persönlichen Geübtheit. Es gibt keine zwei identischen Individuen. Ich bin der Meinung, daß die persönliche Vorliebe für die Tastatur oder die Maus, sich aus den individuellen Fähigkeiten und Erfahrungen ergibt. Man kann selbstverständlich nicht alle Individuen gleich schalten, aber die psychologische Schlüsselstelle unserer Zielgruppe muß angesprochen werden.

Anforderungen der Realität

Die Realität erfordert ein Entwicklungssystem, das möglichst viele potentielle Anwender anspricht. Man muß ein solches Entwicklungssystem nicht FORTH nennen.

Die erste Bedingung, die erfüllt werden muß, ist, daß das System auf einer bereits existierenden Hardware mit einem schon definierten Betriebssystem laufen muß. WISC begann mit einem Verarbeitungsrechner auf der Basis des Intel Prozessors in Form eines IBM PC's, XT's oder AT's unter PC-DOS. Als Grundlage diente die Original-Implementation von FORTH auf diesen Prozessoren. Das PC-DOS Betriebssystem besitzt eine Dateistruktur für Massenspeicher, welche bei den neuesten Versionen Inhaltsverzeichnisse und Unterinhaltsverzeichnisse erlaubt. Zusätzlich zum schon vorhandenen FORTH, wird dann noch eine effektive Implementation benötigt, die den Zugriff auf das Filesystem ermöglicht.

Systemfiles bei PC-DOS sind unterschiedlich lang. Die meisten Programmierer kennen irgendeinen Editor, oft eine Variante des wohlbekannten WordStar. Aber FORTH besitzt in seiner ursprünglichen Form einen Zeileneditor und keine Files. Auf externe Speichergeräte wurde direkt mit sequentiellen auf 1KBytes Größe festgelegten Blöcke zugegriffen. Es gab keine festgeschriebene Filestruktur. Das führte zu Konflikten mit heute aktuellen Betriebssystemen. Eine Möglichkeit diese Konflikte zu beseitigen besteht darin auf 1K Blöcke innerhalb der Systemfiles zuzugreifen, obwohl das natürlich zu größerer Komplexität führt. Aber die Verwendung von FORTH-Blocks beeinflußt den Denkprozess des Programmierers positiv. Es führt zu einer Aufschlüsselung der einzelnen Faktoren eines Problemes.

Das ist jedoch mehr eine Frage des Programmierstils. Bei offenen frei gestalteten Textfiles gibt es keine vorgeschriebene Vorgehensweise, d.h. keinen zwingenden Programmierstil. Was gebraucht wird, ist ein freier Programmierstil mit Standard-Textfiles. Bei Textfiles von variabler Länge, ist keine Trennung von Quellcode und Hilfscreens oder Shadowscreens erforderlich. Im Gegenteil, eine Reihe von Testvektoren könnte mit jeder neuen Funktion dazukommen. Zusammengehörige Sachen könnten gemeinsam untergebracht werden. So würde der Programmierer durch die Übernahme eines Programmierstils der Kommentare, Quellcode und Testvektoren als Einheit betrachtet, sehen, daß die Dokumentation vereinfacht. Es ist dann sehr einfach den Quellcode direkt aus solchen Files zu kompilieren und die Dokumentationsdaten zu überspringen. Dieser Ansatz unterscheidet sich völlig von den Erfahrungen vieler FORTH-Programmierer. Andere Programmierer sind an einen eigenen Programmierstil für jede Sprache, die sie benutzen, gewöhnt. Deshalb ist ein neuer Programmierstil auch keine Überraschung für sie.

Standardbibliotheken

FORTH hat für umfangreiche Funktionen keine Standardbibliothek. Teils liegt das daran, daß es keine einflußreiche treibende Kraft wie AT&T oder IBM gab, die auf der Entwicklung der notwendigen Bibliotheken bestand. Aber es darf nicht vergessen werden, daß es andere Programmierer gab, die diese Bibliotheken erwarteten.

Eine wichtige Überlegung an dieser Stelle ist, wie diese Bibliotheken benutzt werden. Da das Kompilieren bei FORTH einfach und schnell von statten geht, besteht wenig Notwendigkeit Objektcode abzuspeichern. Quellcode in der oben vorgestellten Form beinhaltet die ganze notwendige Dokumentation und kann nach Belieben kompiliert werden. Das macht ein Programm überflüssig, das Objektmodule relok-tabel macht.

Was für Tools sind notwendig?

Die Programmiersprache FORTH flößt vielen Angst ein, die schon über Programmiererfahrung verfügen. Das Erziehungssystem in diesem Land hat die Schüler und Studenten schon mit BASIC, C, Pascal, FORTRAN usw. indoktriniert. Leider wird die Programmiersprache FORTH nicht als Alternative angeboten. FORTH ist aber auch nicht komplizierter als BASIC oder C. Es ist nur anders und die meisten Menschen wollen ihre gewohnte Programmiersprache nicht wechseln.

Bei der Entwicklung eines marktfähigen Systems müssen wir uns am potentiellen Anwender orientieren. Wollen wir überleben, müssen wir seine Bedürfnisse befriedigen. Wenn wir Glück haben, benutzt er nicht mehr allzuviele Tools, die wir ihm zur Verfügung stellen, nachdem er gelernt hat mit dem Entwicklungssystem umzugehen. Aber damit sich das Produkt verkauft, müssen wir seine Erwartungen erfüllen. Hat der Anwender einmal mit dem Entwicklungssystem gearbeitet, wird er beispielsweise feststellen, daß durch den Gebrauch der Zerlegungstech-

niken von FORTH und einiger dazu passender Textvektoren, ein Debugger so gut wie überflüssig wird. Aber vorher kann er sich so etwas überhaupt nicht vorstellen. Das Problem bei der Sache ist aber, wer schreibt ein solches gut verkäufliches Programm mit vielen Tools, die nachher überhaupt nicht benötigt werden?

Von anderen lernen

Im Gegensatz zur weit verbreiteten Meinung von FORTH-Programmierern, daß Nicht-FORTH-Programmierer viel lernen müßten, wäre es vielleicht nicht das Schlechteste von diesen etwas zu lernen. Das heißt nicht, daß jeder FORTH-Programmierer sein FORTH so gestaltet, daß es wir irgendeine andere Programmiersprache aussieht. Die Erfordernisse der Compiler bei anderen Programmiersprachen, müssen nicht auf Programme die in FORTH geschrieben werden übertragen werden.

Es ist vielmehr so, daß der FORTH-Programmierer neue Konzepte aus dem Verständnis anderer Programmiersprachen entwickeln sollte. Die meisten Hochsprachen wurden für bestimmte Applikationen im Zusammenhang mit spezifischer Hardware entwickelt. Einige dieser Sprachen wurden auf andere Rechner die mit anderen Applikationen arbeiten übertragen. Um so umfassender die Fähigkeiten eines Compilers werden, desto komplizierter wird er. FORTH ist in der Lage, im Verlauf eines Programmes intern dem Compiler neue Compilerinstruktionen hinzuzufügen. Das ist für viele, die FORTH noch nicht kennen, neu.

Der C-Kern ist in der Tat fast so kompakt wie die meisten FORTH-Kerne. Dennoch müssen bei C noch viele zusätzliche Funktionen verwendet werden. D.h. der C-Kern ist so klein, daß er allein nicht existieren kann. Es muß mindestens ein Satz I/O-Funktionen vorhanden sein. Trotzdem gibt es mehr Gemeinsamkeiten, als man auf den ersten Blick erkennt.

Das aktuelle WISC Entwicklungssystem

WISC arbeitet unter Hochdruck an einem Entwicklungssystem, das mit unseren Hardwareprodukten zusammenarbeiten kann. Viele der vorstehenden Überlegungen, wurden in die schon verwirklichten Produkte eingebracht. Besonders beim CPU/32 müssen wir noch mit dem 16-Bit Adreßraum auskommen. Ein vollständiger linearer 32-Bit Adreßraum scheint unabdingbar zu sein. Systeme mit lediglich 16-Bit Adreßraum, werden bald überholt sein. Die neue Implementation soll den 32-Bit Adreßraum bzgl. der Segment- und Offsetanforderungen des Intel-Prozessors unterstützen, falls dieser benutzt wird.

Notwendige Bibliotheken sind u.a. ein vollständiger Satz von mathematischen und Fließkommafunktionen. Phil Koopman hat solch eine mathematische Bibliothek geschrieben und als Public Domain zur Verfügung gestellt. Sie enthält eine leistungsfähige Faktorisierungsfunktion und passende Näherungen für die transzendentalen Funktionen. Seine Bibliothek läuft schon seit über fünf Jahren in den verschiedensten Applikationen und es sieht so aus, als sei sie weitestgehend frei von Fehlern. Die Fließkommawerte basieren auf der IEEE Kurzform, einem in Computerkreisen üblichen Format. Große Teile dieser Bibliothek wurden schon in den "Forth Dimensions" veröffentlicht. Einige Händler vertreiben sie, andere weigern sich standhaft. Damit für die Zukunft ein gut ausgetestetes Standardset von mathematischen und Fließkommafunktionen zur Verfügung steht, hat WISC Kopien vom Quellcode auf Disketten, die im PC-DOS Format vorliegen, gemacht. Diese sind bei der 'Conference' erhältlich. Wir hoffen, daß das ungefähr den Wünschen derer nahe kommt, die möglicherweise einmal FORTH benutzen.

Ein anderes wichtiges Funktionstool wäre ein im Compiler integrierter Editor. Es sollte damit möglich sein, jeden beliebigen Editor oder Wortprozessor zur Erzeugung der Quelldateien einzusetzen. Ein im Entwicklungssystem integrierter Editor hätte viele Vorteile. Er könnte dazu benutzt werden, den Anfang eines neuen Kompilationsschrittes anzuzeigen, genauso wie die Aufteilung der alten Blockstruktur. Läuft die Kompilation schief, sollte der Grund dafür vom Editor angezeigt werden, genauso wie das die WHERE Funktion bei älteren FORTH Implementationen macht. Wie oben schon angedeutet wurde, kann ein auf der bekannten WordStar Funktion basierender Editor die Abneigung verringern, den Umgang mit einem neuen Editor zu lernen. WISC besitzt einen solchen Editor, der zwar schon betriebsbereit ist, dem aber noch der letzte Schliff fehlt.

Andere Bibliotheken könnten einen ansprechenden Debugger, ein gut zu handhabendes Set von String-Utilities und anderes mehr enthalten. Die Struktur des Entwicklungssystemes sollte eine einfache Verknüpfung mit anderen Bibliotheken ermöglichen, nachdem diese implementiert und getestet wurden.

Schlußfolgerungen

Die Dinge so einfach wie möglich zu halten, heißt nicht, daß sie primitiv sein sollen. Damit die WISC-Prozessoren auf dem Markt aktzeptiert werden, muß zuerst ein akzeptables Entwicklungssystem vorhanden sein, auch wenn dieses sich ziemlich von konventionellem FORTH unterscheidet. Wir sollten auf den Erfahrungen aufbauen, die wir bisher mit FORTH gesammelt haben. WISC ist momentan dabei, ein System zu entwickeln, das die in diesem Artikel gestellten Bedingungen erfüllt. Das macht überhaupt keine Schwierigkeiten. Das Ergebnis wird ein einfaches, elegantes System sein.

SWOPILER

Ein Generator für Stackoperator-Worte

von Andreas Findewirth

Stichworte

» Namenskonvention
» Stackoperatoren
» Lineare Grammatik
» Endlicher Automat

Zusammenfassung

Das Swopiler-Programm ermöglicht es, Worte zur Stackmanipulation zu deklarieren, die dann vom FORTH-System automatisch erzeugt werden. Die Stackoperatoren (Swopperatoren) werden zu diesem Zweck nach einer systematischen Namenskonvention benannt, die sich an den Stack-Kommentaren orientiert. Beispiel für eine Deklaration:

```
SWOPILER:
   (AB=BAB)
   4(ABC=CABC)
   3(12=212)
;
```

Die Syntax der systematischen Namen genügt einer rechtslinearen Grammatik, so daß zur Syntaxanalyse endliche Automaten herangezogen werden können.

Einleitung

Die hergebrachten und standardgemäßen FORTH-Worte für Stackoperatoren tragen sinnvolle, aussagekräftige Namen, wie beispielsweise DUP, SWAP, ROT, DROP. Wenn also im folgenden eine neue Namenskonvention für Stackoperatorworte ("Swopperatoren") hergeleitet und entwickelt wird, muß es dafür gute Gründe geben. Es wurde bereits in [Kre88] eine Alternative zu den sprechenden Namen vorgeschlagen, die eine systematische Namensbildung vorsieht. Um mit solchen Namen umzugehen, sind nicht in erster Linie sprachliche Kenntnisse und Fertigkeiten erforderlich, sondern visuelles Vorstellungsvermögen und systematisierendes Denken. Tabelle 1 enthält eine Übersicht der herkömmlichen Worte, ihrer Bezeichnung nach Kretzschmar und gemäß meinem Vorschlag.

Die neue Namenskonvention soll folgenden Anforderungen genügen, die Reihenfolge der Nennung entspricht der Bedeutung:

- Die Namen sollen einprägsam und leicht zu merken sein. Aufgrund ihres systematischen Aufbaus soll es möglich sein, unbekannten Swopperatoren ihre Funktion unmittelbar anzusehen bzw. für gewünschte Stackoperationen einen eindeutigen und treffenden Namen zu formulieren.
- Die Namen sollen so aufgebaut sein, daß der Code der Operatorworte vom FORTH-System automatisch generiert werden kann. Es soll also genügen, den Namen im System zu deklarieren, um ausführbare Worte zu erhalten.
- Die Worte sollen nicht zu lang und vom Schreibaufwand her vertretbar sein.

Namenskonvention

Zu jedem FORTH-Wort gehört üblicherweise ein Stackkommentar, der die Auswirkungen des Wortes auf die Stackparameter angibt. Stackoperatoren wirken nun

Standard	Kretzschmar	Vorschlag
DROP	1#X	(A=)
DUP	1#K	(A=AA)
NIP	2#X	(AB=B)
SWAP	2#P	(AB=BA)
OVER	2#K	(AB=ABA)
2DROP	2#2X	2(A=)
2DUP	2#2K	2(A=AA)
ROT	3#P	(12=21)
-ROT	3#2P	(21=12)
2SWAP	4#2K	2(AB=BA)
PICK	n#K	(A*=A*A)

Tabelle 1: Bezeichnungsweisen

ausschließlich auf den Stack und haben keinen sonstigen Effekt. Da ihre Bedeutung also durch den Stackkommentar vollständig gekennzeichnet ist, liegt es nahe, den Kommentar als Grundlage für die Namensbildung zu nehmen. Beispiele siehe Bild 1:

```
DUP  ( 16b -- 16b 16b )                          (A=AA)
SWAP ( 16b1 16b2 -- 16b2 16b1 )                  (AB=BA)
DROP ( 16b -- )                                  (A=)
ROT  ( 16b1 16b2 16b3 -- 16b2 16b3 16b1 )        (ABC=BCA)
```

Bild 1

Die einzelnen Stackpositionen sind durch Buchstaben gekennzeichnet, man sieht dem Wort unmittelbar an, was es tut, dazu gehört lediglich etwas Vorstellungsvermögen. Die Bezeichnungsweise kann noch etwas ausgebaut werden, da manche Worte mehrere Parameter verschieben, ohne deren interne Reihenfolge zu ändern. In diesem Fall genügt es, statt der Buchstaben die Zahl der Parameter anzugeben. Beispiele siehe Bild 2:

```
ROT  (ABC=BCA)  BC bleibt unberührt, also   (12=21)
-ROT (ABC=CAB)  AB bleibt unberührt, also   (21=12)
```

Bild 2

Diese Formulierung legt es geradezu nahe, ROT und -ROT nicht nur als Rotation von drei Werten anzusehen, sondern gewissermaßen als SWAP verschiedengroßer Werte, eines Wortes und eines Doppelwortes. Weitere Anpassungen sind leicht möglich:

```
DUP    -->  (A=AA)
2DUP   -->  2(A=AA)
?DUP   -->  ?(A=AA)
```

Selbst die Zugriffsoperatoren ROLL, -ROLL und PICK lassen sich angemessen formulieren, wenn man das Zeichen * als Symbol für eine Reihe von Stackparameterwerten auffaßt, deren Anzahl zuvor auf dem Stack übergeben wurde:

```
-ROLL  (*A=A*)
PICK   (A*=A*A)
ROLL   (A*=*A)
```

Die Wirkung dieser Operatoren kann so sehr leicht erkannt werden (Bild 3):

Die vorstehend entwickelte Art der Namensgebung genügt somit recht gut dem Kriterium 1. Die Kürze der Schreibweise ist nicht in allen Fällen befriedigend, ließe sich allerdings nur durch Verzicht auf Verständlichkeit und Systematik verbessern. Die automatische Umsetzung der Namen in ausführbare Worte ist Gegenstand der nächsten Abschnitte. Dabei wird allerdings auf die Einbeziehung der "*-Worte" (PICK, -ROLL, ROLL) und des bedingten Wortes ?(A=AA) verzichtet.

```
2  (*A=A*)   -->  (21=12)   -->  -ROT
0  (A*=A*A)  -->  (A=AA)    -->  DUP
1  (A*=A*A)  -->  (AB=ABA)  -->  OVER
```

Bild 3

Syntaxanalyse

Bild 4 enthält eine grafische Darstellung der Syntax gemäß der zuvor ad hoc entwickelten Namenskonvention. Die Syntax soll an dieser Stelle aber auch in BNF-ähnlicher Notation angegeben werden gemäß der Schreibweise in [Wir84] mit einigen kleinen Anderungen:

- Nichtterminalzeichen sind in < > eingeschlossen.
- Terminalzeichen sind in " " eingeschlossen.
- Das leere Wort ist e.
- Zeichen der Metasprache sind = | . [ ] { } ( )
- <digit> = "1" | "2" | "3" | "4" | "5" | "6" | "7" | "8" | "9" .
- <alfa> = "A" | "B" | "C" | "D" | "E" | "F" | "G" | "H" | "I" |

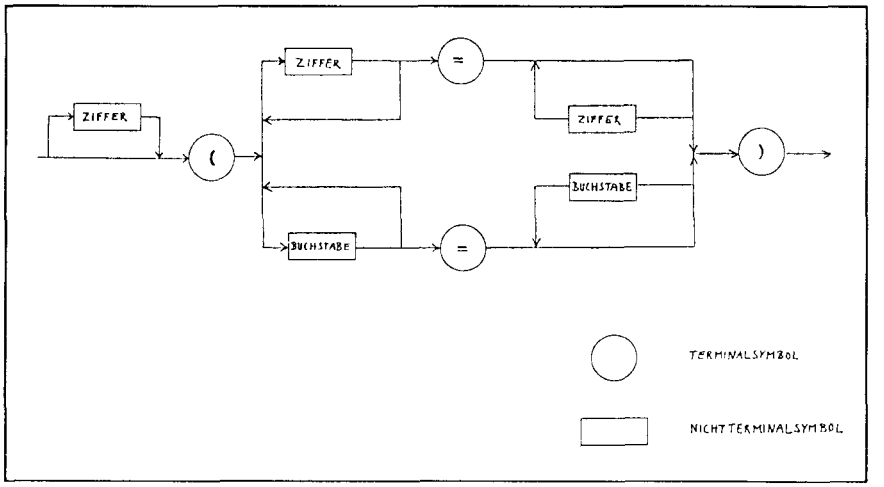


Bild 4: Swopperator-Syntax

```
<Z6> =  ")" .
<Z6> =  "1" <Z6> .    <Z6> = "2" <Z6> . ...   <Z6> = "9" <Z6> .
<Z4> =  "=" <Z6> .
<Z4> =  "1" <Z4> .    <Z4> = "2" <Z4> . ...   <Z4> = "9" <Z4> .
<Z3> =  "1" <Z3> .    <Z3> = "2" <Z3> . ...   <Z3> = "9" <Z3> .
<digitseq> = <Z3> .
```

Bild 5

"J" | "K" | "L" |
"M" | "N" | "O" |
"P" .

- <digitseq> =
  digit { digit } "=" { digit } .
- <alfaseq> =
  alfa { alfa } "=" { alfa } .
- <swopperator> =
  [<digit>] "(" (<digitseq> |
  <alfaseq>) ")" .

Die Syntax bedarf noch zweier Einschränkungen, die sich nur aufwendig im Rahmen der Regeln einer kontextfreien Grammatik formulieren ließen:

- Zwischen öffnender Klammer "(" und Trennzeichen "=" darf kein Buchstabe bzw. keine Ziffer mehr als einmal auftreten.
- Zwischen Trennzeichen "=" und schließender Klammer ")" darf keine Ziffer bzw. kein Buchstabe auftreten, die bzw. der nicht zuvor zwischen "(" und "=" bereits aufgetreten sind.

Andernfalls könnte dem Namen kein Stackoperator zugeordnet werden, z.B.:

(ABB=AB) ??? (ABC=AFC) ???

Im weiteren soll gezeigt werden, daß die Syntax der Swopperatoren einer rechtslinearen Grammatik genügt und damit determinierte endliche Automaten für die Erkennung der Swopperatoren herangezogen werden können.

Eine Grammatik heißt vom Typ Chomsky 0 oder rechtslinear, wenn alle in ihr auftretenden Regeln von einer der folgenden 4 Formen sind:

```
<A> = "a" <B> .
<A> = <B> .
<A> = "a" .
<A> = e .
```

Durch Auflösung der Alternativen in einzelne Regeln und Einführung weiterer Nichtterminalsymbole zeigt sich, daß die Umformung in Regeln unter den oben genannten Einschränkungen möglich ist (Bild 5).

Die Umformung für <alfaseq> verläuft analog (Bild 6).

Mit ein bißchen Übung hätte man die Rechtslinearität der Syntax-Grafik (Bild 4) direkt ansehen können. Nun kann man zu jedem Wort, das einer rechtslinearen Grammatik genügt, d.h. zu jedem regulären Ausdruck, einen endlichen Automaten konstruieren, der solche Worte akzeptiert [Alb83]. Zum einen kann man den

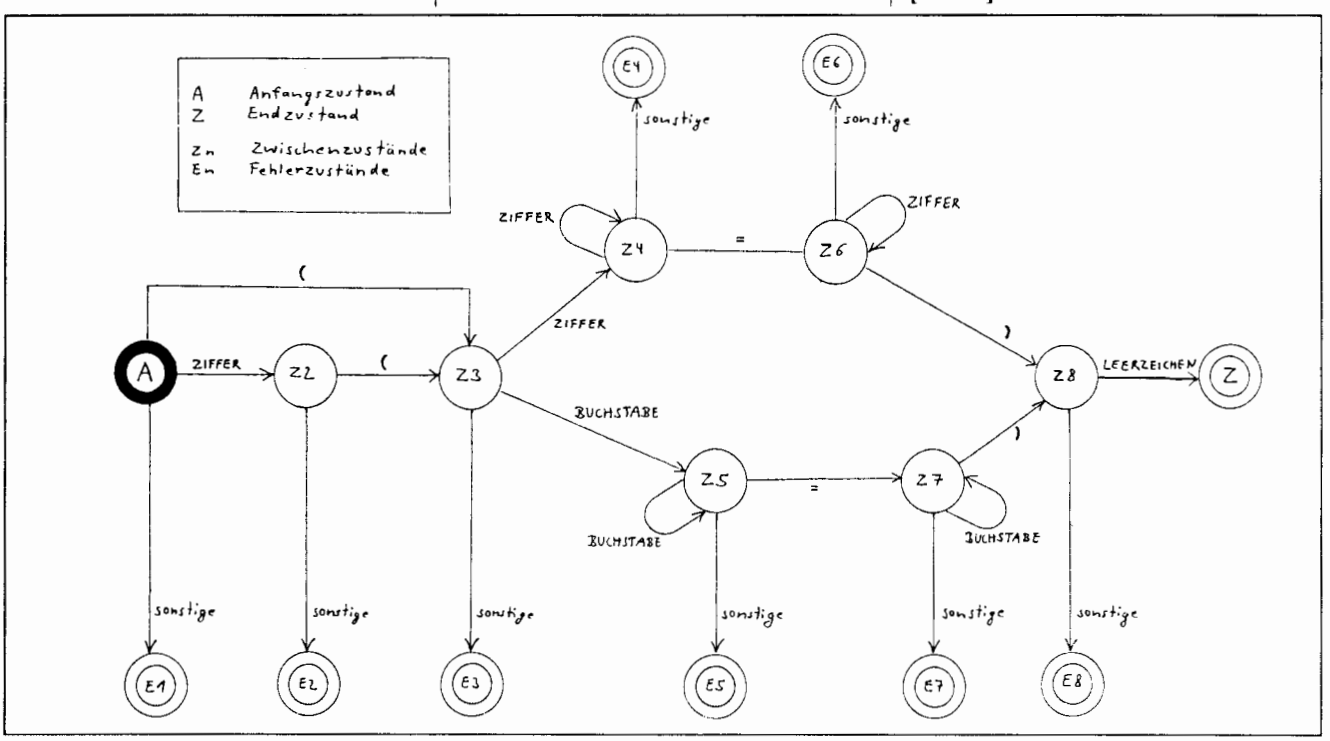


Bild 7: Endlicher Analyse-Automat

```
<Z2> =   "(" <digitseq> .
<Z2> =   "(" <alfaseq> .
<Z1> =   "1" <Z2> .    <Z1> = "2" <Z2> . ...    <Z1> = "9" <Z2> .
<swopperator> =  <Z2> .
<swopperator> =  <Z1> .
```

Bild 6

Automaten aus den grammatischen Regeln gewinnen, indem man die Nichtterminalsymbole (<Z1> usw.) als Zustände des endlichen Automaten interpretiert. Man kann auch direkt von der Syntax- Grafik (Abb. 1) ausgehen und dort an den Stellen, wo die gerichteten Strecken (Pfeile) auseinandergehen oder sich treffen, Zustände einfügen. Abbildung 2 zeigt die so entwickelte grafische Darstellung des endlichen Automaten, ergänzt um Übergänge für ein abschließendes Leerzeichen und Fehlerzustände. Bezüglich der Umsetzung dieses abstrakten Automatenmodells in FORTH-Code sei auf den beigefügten Quelltext verwiesen. Die Zustände <Z1> usw. können dort als die FORTH-Worte STAT1 usw. wiedergefunden werden.

Codeerzeugung

Die Swopperatoren sind mit High-Level-Worten erzeugbar, sie werden ausschließlich mit High-Level-Worten abgearbeitet. Der Verzicht auf Assemblerroutinen bedeutet zwar einen Verzicht auf Ausführungsgeschwindigkeit, macht den Swopiler aber portabel für FORTH-Systeme mit anderen Prozessoren.

Die Manipulation von Werten auf dem Stack mit Swopperatoren wird letztlich durch den Transport von Speicherbereichen realisiert. Um diese Transportbefehle zu erzeugen, werden die zwei Arrays CTRTAB und POSTAB verwendet. Die Swopperator-Worte werden nach der Syntaxanalyse ausgewertet, so daß schließlich die nötigen Transportbefehle aus den Eintragungen der Arrays hervorgehen. CTRTAB enthält dabei für jede Stackposition eine Angabe, ob und wie oft dieser Wert an andere Positionen transportiert werden muß. POSTAB enthält für jede Stackposition die Angabe, von welcher Position her ein Wert an diese Stelle zu kopieren ist. Beispiel (ABC = CAB) :

Position:	1	2	3	4	5	6	7	8
CTRTAB:	1	1	0	0	0	0	0	
POSTAB:	3	1	2	0	0	0	0	

Bei der Umsetzung in Transportbefehle ist zu beachten, daß nur an Stellen kopiert werden darf, die nicht mehr benötigt werden, bei denen also in CTRTAB eine 0 steht. Sie sind danach in POSTAB durch Eintragung von -1 als bearbeitet zu kennzeichnen, der zugehörige Zähler in CTRTAB ist um 1 zu dekrementieren. Kann keine Speicherzelle so ausgewertet werden, ist ein Wert in einer nicht verwendeten Zelle, d.h. einer Zelle, in die sowohl in CTRTAB wie in POSTAB eine 0 eingetragen ist, zwischenzuspeichern. Beispiel: 1 --> 4 :

Position:	1	2	3	4	5	6	7	8
CTRTAB:	0	1	1	1	0	0	0	0
POSTAB:	3	4	2	0	0	0	0	0

Danach kann ohne weiteres fertig ausgewertet werden:

Position:	1	2	3	4	5	6	7	8
CTRTAB:	0	0	0	0	0	0	0	0
POSTAB:	-1	-1	0	0	0	0	0	0

Erzeugt werden dabei folgende Transportbefehle für relative Stackpositionen:

1 --> 4 3 --> 1
2 --> 3 4 --> 2

Diese symbolischen Transportanweisungen werden in das Swopperatorwort kompiliert und zur Laufzeit des Swopperators vom SWOPRETER interpretiert.

Zeitbedarf für die Ausführung der Stackoperatorworte. Alle Angaben gemessen mit volksFORTH 3.80 auf Atari 1040 ST.

Wort	Forth-Primitiv (Mikrosekunden)	High-Level-Wort (Mikrosekunden)	Swopiler-Code (Millisekunden)
(a=aa)	6,7		0,89
2(a=aa)	7,7		0,9
4(a=aa)		33	0,9
(ab=ba)	8,7		1,5
(abc=bca)	11,3		3,6

Dabei war 4(a=aa) in High-Level definiert als 2(ab=aba) 2(ab=aba) .

Tabelle 2: Laufzeitverhalten

Effizienzbetrachtungen

Tabelle 2 enthält die Resultate von Laufzeitmessungen für Stack-Operatoren. Der Zeitbedarf für die Ausführung der mit dem Swopiler erzeugten Worte liegt ungefähr um den Faktor 100 - 200 über dem der Primitiv-Worte und 30 - 40 über dem regulärer High-Level-Worte. Trotz der nicht gerade berauschen-

den Geschwindigkeit sehe ich Anwendungsmöglichkeiten für den Swopiler. Unter didaktischen Gesichtspunkten kann er im Unterricht beim Erlernen der Sprache FORTH verwendet werden, ich verweise hier auf [Kre88]. FORTH-Neulinge können sich auf recht einfache Art schnell Worte für kompliziertere Stackmanipulationen erzeugen. Dies ermuntert zur verstärkten Verwendung der Parameterübergabe mittels des Stacks statt über Variablen. Auch der erfahrene Programmierer kann den Swopiler nutzen, um während der Entwurfs- und Entwicklungsphase eines Softwaresystems rasch und unkompliziert Stackoperatoren zu erzeugen, die später aus Geschwindigkeitsgründen durch High-Level-Worte ersetzt werden. Der Speicherplatz wird verhältnismäßig gut ausgenutzt, da die kompilierten Worte recht kurz sind. Der Swopiler wird zudem auf den Heap geladen, so daß im erstellten Programm nur der Code des Swopreters resident ist.

Außerdem ist die mit swopilierten Worten erreichbare Operationsgeschwindigkeit noch lange nicht ausgereizt. Die vorliegende Version 1.0 des Swopilers wurde in erster Linie nach der Devise "Bring es zum Laufen" erstellt. Aus Zeitgründen konnten bisher weder die Worte sauber faktorisiert noch Optimierungen, vor allem des Swopreters, vorgenommen werden. Wenn darüberhinaus unter Verzicht auf Portabilität ein möglicher weiterentwickelter Swopiler künftig direkt FORTH-Primitive (Maschinen-Code) erzeugt, dann ist noch eine Menge Tempo herauszuholen.

Quellenangaben

[Alb83] Albert, Jürgen; Ottmann, Thomas: Automaten, Sprachen und Maschinen für Anwender; Mannheim, Wien, Zürich: Bibliographisches Institut, 1983 (Reihe: Informatik).

[Kre88] Kretzschmar, Rolf: Swopperatoren - Stackoperatoren mit System; Vierte Dimension, Vol. IV, Nr.2/3, S.10ff.

[Wir84] Wirth, Niklaus: Compilerbau - Eine Einführung; Stuttgart: Teubner, 1984 (Reihe: Teubner Studienbücher Informatik).

Der Autor:

Ich bin 26 Jahre alt und befinde mich derzeit im dritten Semester des Diplomstudiengangs Informatik mit Nebenfach Betriebswirtschaftslehre an der Fernuniversität/ Gesamthochschule in Hagen. Meine Brötchen verdiene ich als Teilzeitbeamter in einer Kommunalverwaltung, wo u.a. die Koordination der technikunterstützten Informationsverarbeitung mit der gemeinsamen kommunalen Datenzentrale zu meinen Aufgaben gehört. Forth habe ich vor etwa neun Monaten kennengelernt und bin seit einem halben Jahr Mitglied der FORTH-Gesellschaft e.V.. Darüberhinaus bin ich in der evangelischen Jugendarbeit tätig und Mitglied einer Friedensinitiative.

Andreas Findewirth
Im Großen Vorwerk 48
D-4900 H E R F O R D
Telefon: 05221/23504

Screen # 0

```
*************************************************\ hf2feb89fi \
*
* Andreas Findewirth +-------------+ TARANTOGA SOFTWARE
* Im Großen Vorwerk 48 ! SWOPILER.SCR !
* D-4900 H E R F O R D ! Version 1.0 ! Herford, 02.02.89
* Telefon 05221/23504 +-------------+ (C) Public Domain
*
* Entwickelt mit dem 16-Bit-System volksFORTH-83 rev. 3.80
* auf Atari 1040 ST F ( 1 MB RAM ) unter GEM/TOS im ROM.
*
* Die Benutzung des Programms erfolgt auf eigene Gefahr, der
* Autor übernimmt für die Folgen von Programmfehlern keine
* Haftung. Für Verbesserungsvorschläge und Hinweise auf Feh-
* ler und Mängel bin ich jedoch selbstverständlich dankbar.
*
*************************************************************
```

Screen # 1

```
\ Loadscreen ...        SWOPILER.SCR        \ hf2feb89fi \

Vocabulary Swopiler

Onlyforth                   1 +load

1 ?head !                   \ Headerless laden
Swopiler also definitions     20 +load
0 ?head !
\ Auf den Heap laden
here $1400 hallot heap dp !  2 19 +thru
Onlyforth Swopiler also       21 +load      dp !

Onlyforth
```

Screen # 2

```
\ Umbenennung der vorhandenen Stackoperatoren   \ hf24jan89fi \

' drop   Alias  (a=)       ' 2drop  Alias  2(a=)
' dup    Alias  (a=aa)     ' 2dup   Alias  2(a=aa)
' nip    Alias  (ab=b)
' over   Alias  (ab=aba)   ' 2over  Alias  2(ab=aba)
' swap   Alias  (ab=ba)    ' 2swap  Alias  2(ab=ba)
' under  Alias  (ab=bab)

' -rot   Alias  (abc=cab)  ' -rot   Alias  (21=12)
' rot    Alias  (abc=bca)  ' rot    Alias  (12=21)

' -roll  Alias  (*a=a*)    ' roll   Alias  (a*=*a)
' pick   Alias  (a*=a*a)

' ?dup   Alias  ?(a=aa)
```

Screen # 3

```
\ Scanner für Buchstaben und Ziffern          \ hf25jan89fi \

Ascii 1 Constant firstdig      Ascii 9 Constant lastdig
Ascii A Constant firstalfa     Ascii P Constant lastalfa

: 2** ( u1 -- u2 )  1 (ab=ba)          \ Zweierpotenz
BEGIN  (a=aa) WHILE  1- (ab=ba) 2* (ab=ba)  REPEAT  (a=) ;

: filter ( char von bis  -- bmsk )     \ bmsk: Bitmaske
(ab=aba) >r  (abc=bca) (a=aa) >r (abc=cab)  1+ uwithin
IF  r> r> - 2** exit
THEN  rdrop rdrop 0 ;

: dig?  ( char -- bmsk )  firstdig  lastdig  filter ;
: alfa? ( char -- bmsk )  firstalfa lastalfa  filter ;
```

Screen # 4

```
\ Teste Erstauftreten und Wiederholungen      \ hf25jan89fi \

Variable charbag       : clearbag ( -- )  0 charbag ! ;

: >flag ( w -- flag )  (a=aa)  IF true or THEN ;
: inbag? ( bmsk -- flag )  charbag @ and  >flag ;
: setbag  ( bmsk -- )  charbag @  or  charbag ! ;

: newdig? ( char -- flag )
dig?  (a=aa) inbag? not and  (a=aa) setbag  >flag ;
: olddig? ( char -- flag )  dig? inbag? ;

: newalfa? ( char -- flag )
alfa?  (a=aa) inbag? not and  (a=aa) setbag  >flag ;
: oldalfa? ( char -- flag )  alfa? inbag? ;
```

Screen # 5

```
\ Fehlermeldungen                             \ hf28jan89fi \

: errmsg2     ." LEFT PARENTHESIS EXPECTED " ;
: errmsg3     ." DIGIT OR LETTER EXPECTED " ;
: errmsg4     ." SEPARATOR OR ALLOWED DIGIT EXPECTED " ;
: errmsg5     ." SEPARATOR OR ALLOWED LETTER EXPECTED " ;
: errmsg6     ." RIGHT PAREN' OR ALLOWED DIGIT EXPECTED " ;
: errmsg7     ." RIGHT PAREN' OR ALLOWED LETTER EXPECTED " ;
: errmsg8     ." BLANK EXPECTED TO TERMINATE SWOPERATOR " ;


: errpos ( w1 w2 -- )  ." AT POSITION "  - abs 1- u. cr
" SYNTAX ERROR " (error abort ;
```

Screen # 6

```
\ Diverses für Ein-, Ausgabe und Umwandlung   \ hf26jan89fi \

: say ( adr -- )  cr count type cr ;

: nextchar ( adr1 -- adr2 char )       \ adr2 := adr1 +1
(a=aa) 1+  (ab=ba) c@ ;

Variable preptr        \ counted string, Länge  #pre
Variable postptr       \ counted string, Länge  #post

\\ Verwendung der Variablen, Beispiel:
preptr @       #pre  = 3
adr0        = = >    !  postptr @   #post  = 4
!                 !  !
V                 V  V
(ABC=BCBA)          3ABC4BCBA)
```

Screen # 7

```
\ Det. endlicher Automat, Teil 1              \ hf26jan89fi \

: stat8 ( adr0 adr1 -- adr0 )  nextchar bl =
IF (a=)  ELSE (ab=aba) say  - errmsg8 errpos  THEN ;

: stat7 ( adr0 adr1 -- adr0 )
BEGIN  nextchar  (a=aa) oldalfa? WHILE
(a=)  postptr @ (a=aa) c@ 1+  (ab=ba) c!
REPEAT  Ascii )  =
IF stat8 ELSE (ab=aba) say  - errmsg7 errpos  THEN ;

: stat6 ( adr0 adr1 -- adr0 )
BEGIN  nextchar  (a=aa) olddig? WHILE
(a=)  postptr @ (a=aa) c@ 1+  (ab=ba) c!
REPEAT  Ascii )  =
IF stat8 ELSE (ab=aba) say  errmsg6 errpos  THEN ;
```

Screen # 8

```
\ Det. endlicher Automat, Teil 2                \ hf27jan89fi \

: stat5 ( adr0 adr1 -- adr0 )
BEGIN  nextchar  (a=aa) newalfa? WHILE
(a=)  preptr @ (a=aa) c@ 1+ (ab=ba) c!
REPEAT  Ascii = =
IF (a=aa) 1- (a=aa)  postptr !  0 (ab=ba) c!  stat7
ELSE (ab=aba) say  errmsg5 errpos  THEN ;

: stat4 ( adr0 adr1 -- adr0 )
BEGIN  nextchar  (a=aa) newdig? WHILE
(a=)  preptr @ (a=aa) c@ 1+  (ab=ba) c!
REPEAT  Ascii = =
IF (a=aa) 1- (a=aa)  postptr !  0 (ab=ba) c!  stat6
ELSE (ab=aba) say  errmsg4 errpos  THEN ;
```

Screen # 9

```
\ Det. endlicher Automat, Teil 3                \ hf27jan89fi \

: stat3 ( adr0 adr1 -- adr0 )  nextchar
(a=aa) newalfa?
IF    (a=)  preptr @ (a=aa) c@ 1+ (ab=ba) c!  stat5
ELSE  newdig?
IF  preptr @ (a=aa) c@ 1+ (ab=ba) c!  stat4
ELSE (ab=aba) say  errmsg3 errpos
THEN THEN ;

: stat2 ( adr0 adr1 -- adr0 )  nextchar Ascii ( =
IF  (a=aa) 1- (a=aa)  preptr !  0 (ab=ba) c!  stat3
ELSE (ab=aba) say  errmsg2 errpos
THEN ;
```

Screen # 10

```
\ Det. endlicher Automat, Teil 4                \ hf2feb89fi \

Variable size

: stat1 ( adr0 -- adr0 adr1 )  (a=aa) 1+ nextchar  (a=aa) dig?
IF Ascii 0 - 2* size ! ELSE (a=) 2 size ! 1- THEN stat2 ;

Variable protok              protok off

: log ( adr -- )  protok @ IF say ELSE (a=) THEN ;

: parsethru ( adr -- )  clearbag  stat1  log ;

: cont? ( adr -- flag )
(a=aa) c@ 1 = IF  1+ c@ Ascii ; = not THEN >flag ;
```

Screen # 11

```
\ Aufruf des Analyseautomaten                   \ hf28jan89fi \

: (analyse ( adr -- flag )  (a=aa) cont?  (a=aa)
IF (ab=ba) parsethru (ab=ba) THEN ;

: analyse ( -- adr flag )  >in push  name (a=aa) (analyse ;
```

\\ Bei korrekter Syntax werden die für die Codeerzeugung notwendigen Informationen nicht über den Stack, sondern mittels der Zeiger-Variablen PREPTR und POSTPTR übergeben, die auf einen entsprechend vorbearbeiteten String verweisen. Siehe auch Screen 6 .

Screen # 12

```
\ Übertragung in Arrays CTRTAB und POSTAB       \ hf31jan89fi \

$20 Constant tabsize
Create ctrtab  tabsize allot   Create postab  tabsize allot

: errsize ( flag -- )  cr abort" TOO MANY OPERANDS !" ;

: insize ( u -- u )  (a=aa) tabsize >  errsize ;

: str>tab ( -- )
ctrtab tabsize erase  postab tabsize erase
preptr  @  (a=aa) c@  (ab=ba) 1+ ctrtab (abc=bca) move
postptr @  (a=aa) c@  (ab=ba) 1+ postab (abc=bca) move ;
```

Screen # 13

```
\ Umcodierung Ziffern in Buchstaben, Teil 1     \ hf31jan89fi \

Variable 0adr        : 0adr@ ( -- adr )  0adr @ ;

: free? ( -- )  0adr@ tabsize 1- +  c@ 0< >  IF errsize THEN ;

: shiftup ( #tab -- )  free? tabsize 1-  (ab=aba) -
(ab=ba) 0adr@ +  (a=aa) 1+  (abc=bca) move ;

: shiftin ( char #tab -- )  (a=aa) shiftup  0adr@ +  c! ;

: replace ( char #tab -- char' )  (ab=aba) 1+ (abc=cab)
0adr@ + c! ;

: fillin ( char #ch #tab -- )  (abc=cab)  bounds
?DO  I (ab=aba) shiftin  LOOP (a=) ;
```

Screen # 14

```
\ Umcodierung Ziffern in Buchstaben, Teil 2      \ hf31jan89fi \

: posadapt ( char #ch -- )  postab 0adr !  0  tabsize 1-
DO  I postab  +  c@  Ascii 0 -  (ab=aba)  =
IF  2(a=aa)  1- (ab=ba) I replace (ab=ba) I fillin THEN
-1  +LOOP 2(a=) ;

: ctradapt ( -- )  Ascii A  0 tabsize 1-
DO  I ctrtab  +  c@  (a=aa) 0< >
IF  Ascii 0 -  2(a=aa) posadapt  ctrtab 0adr !
2(a=aa)  1- (ab=ba) I replace (ab=ba) I fillin  +
ELSE (a=) THEN
-1  +LOOP (a=) ;

: dig>alfa ( -- )  ctrtab c@ dig? IF ctradapt THEN ;
```

Screen # 15

```
\ Umrechnung in relative Stack-Positionen      \ hf31jan89fi \

: 0count ( #tab -- #tab char )  (a=aa) ctrtab  +  (a=aa) c@
(ab=ba)  0 (ab=ba) c! ;

: markpos ( #tab char -- )  tabsize 1-  0
DO I postab  +  c@ (ab=aba)  =
IF (ab=aba)  (a=aa) 1+  I postab  +  c!
ctrtab  +  (a=aa) c@ 1+  (ab=ba) c!  THEN  LOOP 2(a=) ;

: markall ( -- )  tabsize 1-  0
DO  I 0count  (a=aa) 0< >  IF markpos ELSE 2(a=) THEN  LOOP ;

: atplace ( -- )  tabsize 1-  0
DO  I postab  +  c@  I 1+  = IF  -1  I postab  +  c!
I ctrtab  +  (a=aa) c@ 1-  (ab=ba) c!  THEN  LOOP ;
```

Screen # 16

```
\ Compilierung des Swopreter-Codes             \ hf2feb89fi \

Variable #pre              Variable #post

: countpre ( -- )  0 #pre !  tabsize 1-  0
DO  I ctrtab  +  c@  0< >  IF  1 #pre +!  THEN  LOOP ;
: countpost ( -- )  0 #post !  tabsize 1-  0
DO  I postab  +  c@  0< >  IF  1 #post +!  THEN  LOOP ;

: prepost ( -- )  countpre countpost ;

: size, ( -- )  size @  c, ;
: pre,  ( -- )  #pre @  size @  *  c, ;
: post, ( -- )  #post @  size @  *  c, ;
: sum,  ( -- )  #post @  #pre @  +  size @  *  c, ;
```

Screen # 17

```
\ Abfrageworte für Kompilierungstabelle        \ hf2feb89fi \

: (0copy? ( #tab -- flag )  ctrtab  +  c@  0= ;
: (free?  ( #tab -- flag )  postab  +  c@  0= ;
: (finis? ( #tab -- flag )  postab  +  c@  &255  = ;
: (copy?  ( #tab -- flag )
(a=aa) (free?  (ab=ba) (finis?  or not ;

: free?  ( #tab -- flag )  (a=aa) (0copy? (ab=ba) (free? and ;
: finis? ( #tab -- flag )  (a=aa) (0copy? (ab=ba) (finis? and ;
: copy?  ( #tab -- flag )  (a=aa) (0copy? (ab=ba) (copy? and ;
: stay?  ( #tab -- flag )  (0copy? not ;

: #pos>ctradr ( #tab -- adr )  postab  +  c@  1- ctrtab  +  ;
```

Screen # 18

```
\ Such- und Kompilationsworte                  \ hf2feb89fi \

: size* ( u1 -- u2 )  size @ * ;

: (scopy, ( -- flag )  false  tabsize 1- 0
DO  I copy? or  I copy?
IF  I #pos>ctradr (a=aa) c@ 1- (ab=ba) c!  I postab  +
(a=aa) c@ size* c,  -1 (ab=ba) c!  I 1+ size* c,
THEN  LOOP ;
: scopy, ( -- )  BEGIN  (scopy, WHILE  REPEAT ;

: further? ( -- flag )  false tabsize 1- 0 DO I stay? or LOOP ;

: renumber ( u1 u2 -- )  tabsize 1- 0
DO  (ab=aba)  I postab  +  c@  =
IF  (a=aa)  I postab  +  c!  THEN  LOOP  2(a=) ;
```

Screen # 19

```
\ Zwischentransportbefehle erzeugen            \ hf2feb89fi \

Variable #stay             Variable #free

: (sfree ( -- )  -1 #stay !  -1 #free !  0 tabsize 1-
DO  I free?  IF I #free !  THEN
I stay?  IF I #stay !  THEN  -1  +LOOP
#free @ -1  =  IF errsize THEN ;

: sfree, ( -- )  (sfree #stay @ 1+ size* c, #free @ 1+ size* c,
#stay @ ctrtab  +  (a=aa) c@  #free @ ctrtab  +  c!
0 (ab=ba) c!  #stay @ 1+  #free @ 1+  renumber ;
```

Screen # 20

```
\ Swopretercode fertig kompilieren              \ hf2feb89fi \

: move, ( -- )
BEGIN
scopy,
further? WHILE
sfree,
REPEAT 0 , ;

\\
Format des Swopreter-Codes, z.B. (AB=BAB) :
02 04 06 A0 04 06 02 08 04 02 08 04 00 00   Alle Angaben und
! ! ! SUM! ! ! ! ! ! ! ! !       relative Adressen
! ! POST ! ! ! ! ! ! ! ! !       in Bytes.
! #PRE    ! TO ! TO ! TO ! TO END
SIZE       FROM  FROM  FROM  FROM
```

Screen # 21

```
\ SWOPRETER, Hauptroutinen                      \ hf2feb89fi \

Variable stadr                  Variable pgadr
: stadr! ( u -- )  stadr ! ;  : stadr@ ( -- u )  stadr @ ;
: pgadr! ( u -- )  pgadr ! ;  : pgadr@ ( -- u )  pgadr @ ;
: +char ( adr1 -- adr2 char )  (a=aa) 1+  (ab=ba) c@ ;
: 0stadr ( -- )  sp@ pgadr@ 1+ c@  + stadr! ;
: stackup ( -- )  stadr@ pgadr@ 3+ c@ negate +  sp! ;
: stackdown ( -- )  stadr@ pgadr@ 2+ c@ negate +  sp! ;

: swopreter ( adr -- )  pgadr! 0stadr stackup pgadr@ 4 +
BEGIN
+char (a=aa) WHILE
negate stadr@ + >r +char negate stadr@ + r> (ab=ba)
pgadr@ c@ cmove
REPEAT (a=) stackdown ;
```

Screen # 22

```
\ SWOPILER, Hauptroutinen                       \ hf2feb89fi \

: transform ( adr -- )  (a=)  \ oder SAY
str>tab dig>alfa prepost markall atplace ;

: makecode ( -- )  size, pre, post, sum, move, ;

: produce ( adr -- )  transform Create makecode
Does>  swopreter ;
: Swopiler: ( -- )
BEGIN
analyse
IF  produce
ELSE 2(a=) name say exit
THEN
REPEAT ;
```

Screen # 23

```
\\ Hinweise:                                    \ hf2feb89fi \
```

Wer sich die Mühe des Abtippens sparen will, kann das File mit diesem Quelltext gegen Kostenerstattung in Höhe von 10,-- DM für eine 3.25"-Diskette und 3,-- DM für Porto und Verpackung direkt bei mir beziehen. Wer eine leere Diskette zuschickt, den kostet es nur Porto und Verpackung. Wenn ein interessantes Programm drauf ist, kostet's gar nichts. Ich meine, das sind faire Bedingungen.

Das Geld bitte in Scheinen oder Briefmarken beilegen oder vorab überweisen:

Bankverbindung:
Kto.-Nr. 10000 20 626 Kreissparkasse Herford BLZ 494 501 20

DRAGON

von Christoph Krinninger

Stichworte:

- **Fraktale,**
- **Turtle-Grafik,**
- **Trigonometrie**

Dieses Programm zeichnet den sogenannten Harter-Hightway-Drachen. Es benutzt die hohe Auflösung des Atari ST und die hohe Geschwindigkeit des volksFORTH und bietet während des Aufbaus eine ganz besondere Ästhetik.

Für den Aufbau wird eine sogenannte Turtle verwendet, dieser muß nur eine Drehrichtung und eine Zeichenlänge mitgeteilt werden und ist dadurch eine besonders unkompliziertes Zeichenwerkzeug. Für die Umsetzung dieser Vektorgrafik in die Bildschirmkoordinaten werden elementare trigonometrische Funktionen wie Sinus und Cosinus benötigt. Wenn keine besonders hohe Auflösung benötigt wird, so liest man diese Werte am einfachsten in einer Tabelle ab, wobei man nicht unbedingt für jeden Winkel zwischen 0 und 360 Grad einen Wert abspeichern muß. Es genügt nach den bekannten Umformungsformeln durchaus der Bereich zwischen 0 und 90 Grad. Ferner kann man weiteren Speicherplatz einsparen, indem man nur die Werte für gerade Winkel abspeichert und für ungerade Winkel linear extrapoliert.

Die DRAGON-Kurve besteht aus rechtwinkligen Linien, die mit Hilfe der Rekursion in immer kleinere Abschnitte geteilt werden. Das Programm wird mit N DCURVE gestartet, N ist die Ordnungszahl. Mit der Auflösung des Atari ST kann man maximal einen Drachen 16. Ordnung zeichnen. Die Rechenzeit beträgt ca. 4 Minuten. Mit der Variablen STEPSIZE kann man die Größe der Grafik verändern.

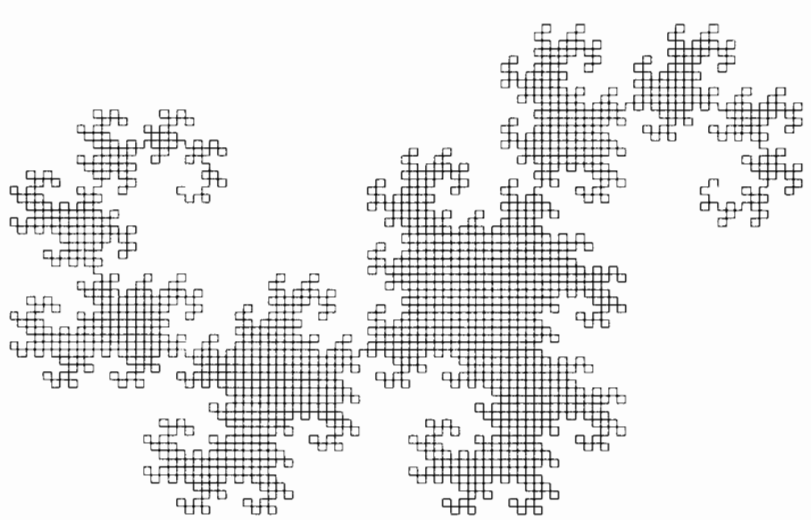
Diese Kolumne soll zu einer ständigen Einrichtung werden. Wer also eine FORTH-Grafik erstellt hat, die für das Titelbild geeignet ist, möge einen kurzen Artikel schreiben und diesen an die Redaktion senden.

Noch ein paar Worte zum Artikel in der letzten Ausgabe:

Durch einige Besonderheiten des Ventura Publisher sind bei diesem Artikel alle ">" und "<" unterdrückt worden. Dies ist besonders bei FORTH-Worten wie ">R", "R>" oder "->" verheerend. Wer am korrekten Sourcecode oder Artikel interessiert ist, kann sich mit mir über das FORTH-Büro in Verbindung setzen.

Bibliographie:

1. DRAGON, Bruce R. Land, BYTE 04/86, S. 137 ff
2. SIN, COS und 3D, Frank Schmidt, c't 10/85, S. 87ff
3. The Fractal Geometry of Nature, Mandelbrot B. B., Freeman 1982, S. 66ff



Beispiel 1

DRAGON

Screen # 1

```
\ Loadscreen                                    21feb88 ck

Onlyforth   gem also

\needs it       : it ;
forget it       : it ;

\needs pline    include vdi.scr

2 8 thru
```

Screen # 2

```
\ Sinustabelle                                  21feb88 ck

decimal

Create sintab

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

Screen # 3

```
\ Sinusberechnung                               21feb88 ck

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

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

Screen # 10

```
                                                21feb88 ck

Kleiner Trick, um das Programm wiederholt hintereinander laden
zu können.
```

Screen # 11

```
                                                21feb88 ck

SINTAB          Tabelle für Sinus-Berechnung
```

Screen # 12

```
                                                21feb88 ck

((SIN   Sinus-Berechnung für Winkel zwischen 0 und 90 Grad.

Gerade Winkel

Ungerade Winkel
Näherung nach sin(x) = ((sin(x-1) + sin(x + 1))/2

(SIN    Sinus-Berechnung für Winkel zwischen 0 und 360 Grad.
```

Screen # 4

```
\ Sinus, Cosinus                                21feb88 ck

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

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

Screen # 5

```
\ Turtle-Grafik                                 21feb88 ck

Variable angle
Variable xcood
Variable ycood
Variable stepsize

Variable #lines         25 Constant #maxlines

Create line-array       #maxlines 4 * allot

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

Screen # 6

```
\ Turtle-Grafik mit Akkumulation                21feb88 ck

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

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

Screen # 13

21feb88 ck

SIN Sinus-Berechnung für beliebige Winkel

COS Cosinus-Berechnung nach der Formel
$\cos(x) = \sin(90 + x)$

Screen # 14

21feb88 ck

ANGLE Einige Variable für die Turtle-Grafik
XCOOD
YCOOD
STEPSIZE Länge eines Turtle-Schrittes

#LINES #MAXLINES Zähler für die Akkumulation der Poly-linie.
LINE-ARRAY Zwischenspeicher für Linienkoordinaten

TURN Dreht die Turtle gegen den Uhrzeigersinn

Screen # 15

21feb88 ck

DRAW-LINE Zeichnet die zwischengespeicherte Poly-Linie

MOVE-TURTLE Bewegt die Turtle um den Betrag in STEPSIZE und speichert die Bewegung.

DRAGON

Screen # 7

```
\ Hauptroutine                                  21feb88 ck

: dragon        ( sign level -- ) recursive
key? IF key abort THEN
dup 0=
IF
drop drop move-turtle
ELSE
over 45 * turn
1
over 1- dragon
over -90 * turn
-1 ( +1 ergibt eine weitere hübsche Grafik )
over 1- dragon drop 45 * turn
THEN ;
```

Screen # 8

```
\ Initialisierung                               21feb88 ck

: dcurve        ( level -- )
page  200 xcood ! 150 ycood !  360 6 * angle !
2 stepsize !
xcood @ ycood @ 2dup 2 pline  1 swap
overwrite solid
xcood @ ycood @ line-array 2!
1 #lines !
dragon
#lines @ 0< > IF draw-line THEN ;
```

Screen # 16

21feb88 ck

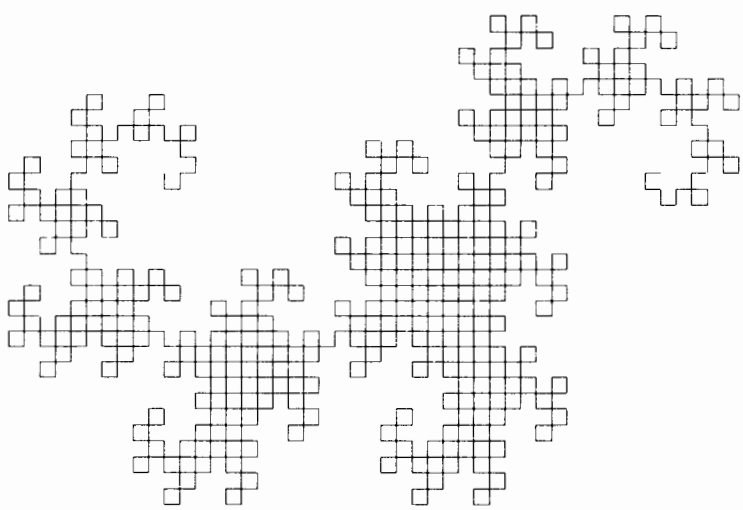
DRAGON Eigentliche Hauptroutine

Screen # 17

21feb88 ck

DCURVE Zeichnet die DRAGON-Kurve n-ter Ordnung.
N muß geradzahlig sein.

Maximal ist ein Drachen 16. Ordnung bei STEPSIZE = 1
auf dem Atari ST möglich.



Beispiel 2

Das Mini-BEE-2000 System.

Autor: Ulrich Paul

Mit diesem Artikel beginnt eine Serie über ein System, basierend auf dem RTX 2000 von Harris. Die beschriebene Hardware ist die des Mini-BEE-2000. Beginnend mit der CPU-Karte werden alle Komponenten in loser Folge näher vorgestellt.

Das ganze System besteht derzeit aus folgenden Komponenten: CPU-Karte, Digital-I/O-Karte, Analog-I/O-Karte und Floppy/SCSI-Karte. Sie werden über eine Backplane mit 96-poligen VG-Leisten miteinander verbunden. Dieser und der nächste Aufsatz befassen sich mit der CPU-Karte. Sie ist stand-alone als Development-Board betreibbar, da alle notwendigen Komponenten auf ihr enthalten sind. Nach einer generellen Beschreibung wird im nächsten Heft auf Details der Implementation eingegangen. Die anderen Karten folgen mit je einem Artikel in der oben genannten Reihenfolge.

Wie man dem Schaltbild entnehmen kann, sind 2x32kW RAM und 32kW ROM onboard untergebracht, was für die meisten Applikationen vollkommen ausreichend ist; vor allem, wenn man bedenkt, daß das FORTH-System (im Lieferumfang enthalten) nur eine Bank teilweise belegt. Die Glue-Logik besteht aus einem schnellen PAL. Weiterhin befindet sich eine serielle Schnittstelle auf der Karte. Zur flexiblen Erzeugung des Prozessortaktes wurde ein 40MHz Quarzoszillator und ein weiteres PAL verwendet. Der konsequente Einsatz von programmierbaren Logikbausteinen ermöglichte erst das Mini-Format von 100x100mm. Alle Signale der CPU und einige zusätzliche sind auf einer 96-poligen VG-Leiste herausgeführt.

In der Low-Cost-Version läuft der RTX 2000 mit ca. 6,7MHz PCLK, damit billige 70ns RAMs eingesetzt werden können. Allerdings war dazu noch ein kleiner Kunstgriff anzuwenden, der weiter unten näher beschrieben wird: Der Prozessor läuft mit einem unsymmetrischen Takt. Aber das allein hätte noch nicht ganz ausgereicht. So wurde auch der Adreßdekoder auf minimale Durchlaufzeit gebracht. Denn: Die Zugriffszeit auf den Speicher, vom RTX aus gesehen, setzt sich aus der Adreßdekodierzeit und der Zugriffszeit der RAM-Chips zusammen. Je schneller der Dekoder ist, desto langsamer dürfen die Speicher sein und ein flinker Dekoder ist billiger als vier schnelle RAMs.

Um den Speicher auf der Karte vollkommen abschalten zu können, wurde ein Eingang DESELECT vorgesehen; es wird dann nur auf den externen Bus zugegriffen. Entsprechend wird auch der Pin DBUFOE, der die Ausgänge der Datenpuffer freigibt, so bedient, daß nun jeder Speicherzugriff ihn aktiviert, während es sonst nur bei Adressen der Banks 2-31 erfolgt. Die Richtung für die Puffer ist durch das MRW-Signal bestimmt. Ein Signal, das eigentlich in das MDEC-PAL gehört, dort aber keinen Platz mehr hatte, kommt aus dem MTIM. Da die Funktion eines Adreßdekoders trivial ist, soll hier auch nicht näher darauf eingegangen werden. Erwähnenswert ist aber noch, daß, solange BOOT aktiv ist, in Bank 0 die EPROMs zum Lesen eingeblendet sind. Das Schreiben in Bank 0 und jeder Zugriff auf andere Bänke bleibt unverändert. Somit kann beim Booten der Inhalt der EPROMs sehr leicht ohne Adressierungseinschränkung ins schnelle RAM kopiert werden.

Das zweite schnelle PAL (MTIM) auf dem Board enthält die Logik zur flexiblen Erzeugung des ICLK für den RTX. Abgeleitet von einem 40MHz-Quarzoszillator werden, je nach Zustand der Eingänge am PAL, verschiedene Kurvenzüge für den ICLK ausgegeben. Warum den Aufwand? Nun, der RTX ist ein vollkommen statisch aufgebauter Chip. Das heißt einerseits, daß er nicht mehr Anforderungen an seinen Takt stellt als ein gewöhnlicher CMOS-Zähler, andererseits aber auch, daß alle Signalformen, intern und an den Pins, direkte Funktionen der Eingänge sind. Im RTX-Datenblatt spiegelt sich das z.B. in t26 wider. Der Maßpfeil ist zwar von der steigenden Flanke des PCLK weg gezeichnet (was irreführend ist!), die Zeit jedoch ist spezifiziert mit t2 + 20ns, d.h. 20ns nach der fallenden Flanke des ICLK.

Daraus folgt, daß eine kürzestmögliche Zeit für den H-Zustand von ICLK anzustreben ist, um möglichst frühzeitig gültige Signale an den RAMs zu haben. Die untere Grenze ist 25ns bei 8MHz-Chips und 20ns bei der 10MHz-Version. In einem Laboraufbau funktionierte ein 8MHz RTX noch mit unter 20ns Pulsbreite. Eine Ausschöpfung dieser, vom Hersteller eingebauten, Sicherheitsmarge sei aber Bastlern

Das Mini-BEE-2000 System

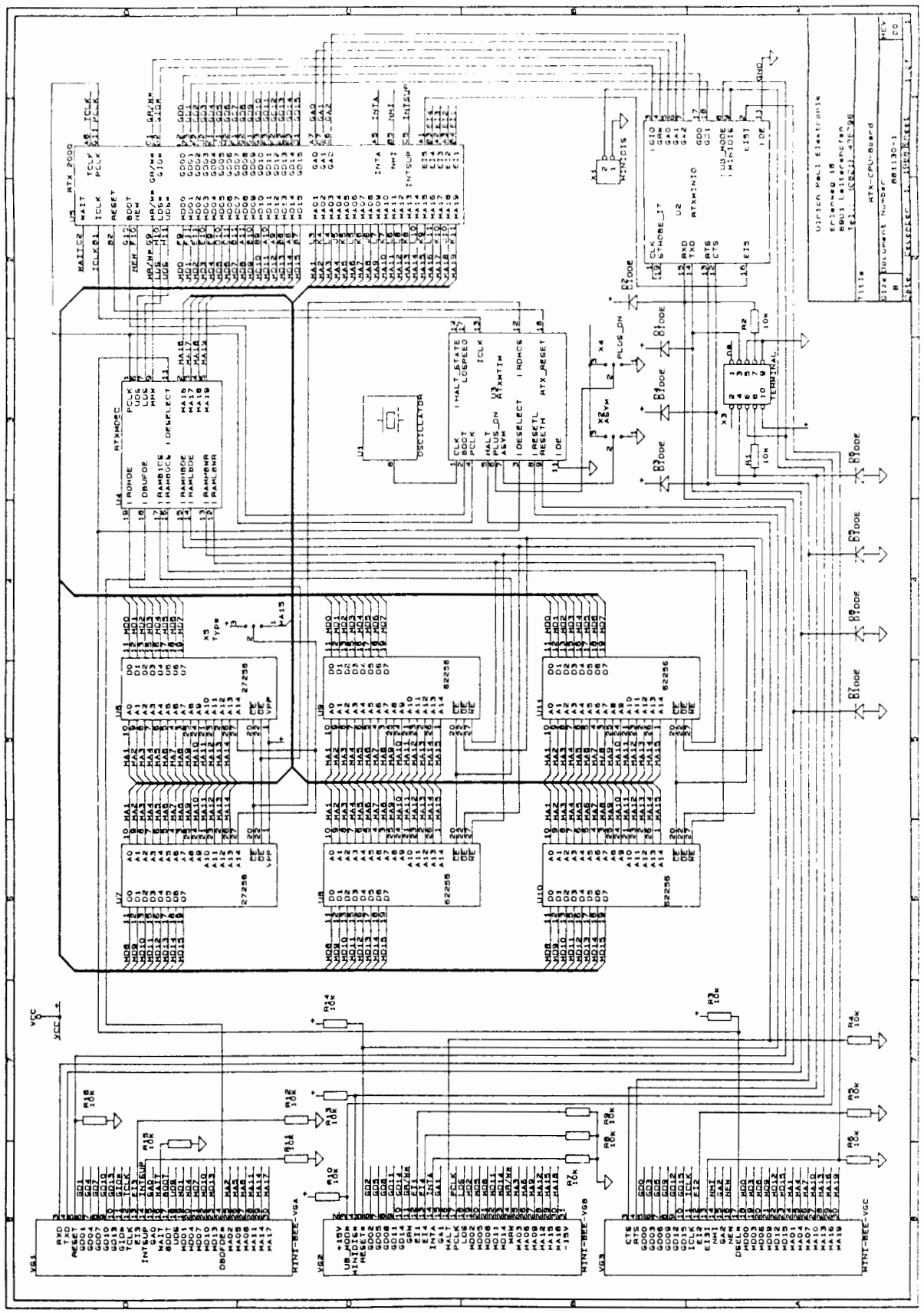


Abb.: 1 - Schaltbild

und Hobbyisten überlassen; für den professionellen Einsatz haben die Worst-Case-Werte Gültigkeit.

Weitere Signale, die der MTIM-PAL liefert, sind das RESET-Signal zum RTX und ein Kontrollsignal LOSPEED, das anzeigt, wenn der ICLK auf die niedrige Taktrate geschaltet ist. Dies ist nach einem Reset beim Booten der Fall, um mehr Zeit für die EPROMs zu haben. Die Funktion des PLUS_ON-Einganges, wie auch die möglichen Signalformen des ICLK werden in der nächsten Folge beschrieben.

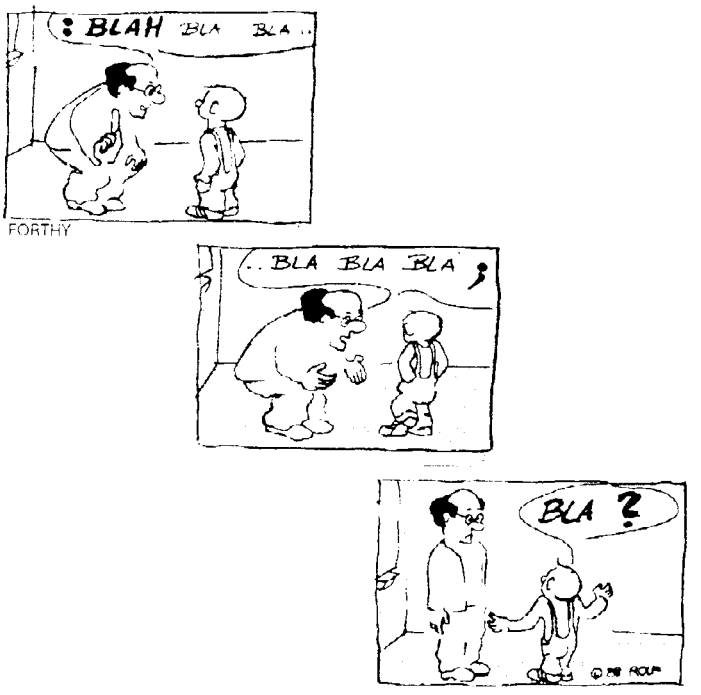
Die serielle Schnittstelle wurde durch ein GAL realisiert. Kompatibel zum Harris-Minimalsystem werden die zwei untersten Datenbits der ASIC-BUS-Adresse 7 verwendet. Die Besonderheit dieses Chips liegt in seinem Eingang UB_MODE. Liegt dieser auf 0V, werden die Signale auf der RS-232-Seite in der richtigen Polarität ausgegeben; bei einer "1" jedoch umgedreht, um die Inversion in den Treibern auszugleichen.

Auf den ersten Blick erscheint eine direkte Verbindung eines TTL-Chips mit der V24-Schnittstelle eines Terminals oder Computers eine unmögliche Sache. Doch die meisten RS-232-Empfänger besitzen ihre Umschaltschwelle bei ca. 1V. Bei nicht zu langem Kabel ist ein TTL-Ausgang also durchaus in der Lage alle Anforderungen zu erfüllen. Natürlich ist für den TTL-Baustein eine adäquate Schutzschaltung vorzusehen.

Damit die Bedienung nicht allzuviel CPU-Overhead erfordert (kein Polling!), löst der Ausgang EI5 des GALs bei jedem Startbit einen Interrupt aus. Wird über MIN-DIS die Schnittstelle abgeschaltet (Adresse 7 ist dann frei), ist auch der EI5-Ausgang mit EI5I verbunden und externe Interrupts der Ebene 5 sind zugelassen. Es gibt aber auch Versionen des MINIO-Chips, die die Verbindung EI5I-EI5 nie auftrennen, sondern nur mit den Daten verknüpfen.

Das Zusammensetzen der Zeichen aus dem Bitstrom geschieht in Software. Der dazu notwendige Overhead liegt unter 1% der CPU-Zeit. Als wesentlicher Vorteil ist die automatische Baudratenerkennung zu nennen. Nur das Format, also Anzahl der Datenbits, Parity und Anzahl der Stopbits ist einmal einzustellen; die aktuelle Übertragungsgeschwindigkeit wird dann beim Booten durch Eingabe von z.B. <CR> festgelegt. Es kann aber die Baudrate jederzeit durch ein Kommando neu gesetzt werden.

Wie weiter oben schon kurz erwähnt, kommt die Karte mit einem kompletten FORTH-System in den EPROMs. Es ist ein Superset des F83, erweitert um spezielle Worte für den RTX und mit einer geänderten Struktur des Dictionary. Darauf wird in einem eigenen Artikel detailliert eingegangen. Hier nur kurz ein paar Schlaglichter: Es gibt drei Bereiche im Speicher, je einen für den Code, die Variablen und die Namen (Header). Damit können ROM-fähige Versionen und solche in anderen Sprachen ohne neuerliche Kompilation erzeugt werden. Von der Möglichkeit eines echten RENAME ganz zu schweigen. Aber diese und andere Bonbons gibt es erst in einem der nächsten Hefte; Klaus Kohl wird dann aus seiner Trickkiste einiges mit Pep präsentieren.



Bücherecke

Ron Geere, FORTH: The Next Step, Addison-Wesley Publishers Ltd. 1986

Wie der Titel impliziert, ist dieses Buch nicht für die "ersten Schritte" in FORTH geeignet. Vielmehr beinhaltet es eine Sammlung von Erweiterungen und Utilities zum F79-Standard. Zu den einzelnen Kapiteln im Detail:

- **Getting Organized:** Die meisten, der hier vorgestellten, Worte (z.B \, ASCII, THRU etc.) sind bereits im F83-Standard oder im L&P F83 implementiert. Einige verwenden die nicht mehr standardgemäße Variable WIDTH (Zur Erinnerung: max. Namenslänge im Header). Die Vergleichstabelle F79 Vs. fig-FORTH ist ganz hilfreich.
- **Some FORTH Extensions:** Hier werden einige Stack- und Vergleichsoperatoren definiert (high level und 6502-Assembler), die meist ebenfalls durch F83 überholt sind bzw. in ihrer Verwendbarkeit auf die 6502 CPU beschränkt sind.
- **Double Number Definitions:** Dieses Kapitel enthält das normale double-number Vokabular allerdings um Multiplikation und Division (mixed size) erweitert.
- **Formatting:** Erwähnenswert sind einige Ausgabe-Worte für Stunden- bzw. Gradmaß
- **The Real World**: Ein FORTH-Interface zum 6522 VIA Chip und einige Bit-Operatoren sind hier definiert.
- **Mathematics:** Dies ist wohl das nützlichste Kapitel des Buches. Es werden Worte für sämtliche trigonometrischen und transzendenten Funktionen in 16- und teilweise 32-Bit Integer Format beschrieben (z.B sin, cos, arctan, exp, ln, etc.). Die Approximationen werden mittels Taylor-Reihe oder Chebyshev-Polynomen mit akzeptabler Genauigkeit berechnet.
- **Calendar Functions**: Hier findet man auch das berühmtberüchtigte JULIAN-DATE Wort aus dem Astrophysial J. 79/11 (ein typisches Beispiel für write-only FORTH Code) und ein "Kalender-Ausdruck-Wort".
- **Factors and Multiples** und **When 10 Digits are not enough:** Diese beiden Kapitel bringen die üblichen Primzahlen- und Fakultäts- Berechnungen, teilweise mit "unendlicher" Genauigkeit.
- **Pot-pourri:** Eine bunte Mischung aus Quick-Sort, Binary-Search und Screen-List Worten.

Insgesamt betrachtet ist das Buch trotz einiger "alter Hüte" recht nützlich. Viele der enthaltenen Definitionen hat man zwar schon irgendwo mal gesehen... Hier findet man sie alle schön beisammen, ergänzt durch umfangreiche Literaturangaben, ein Glossar und einen ausführlichen Index. Problematisch ist jedoch der Preis: DM 43.- für 89 Seiten Text !

James D. Terry, Library of FORTH Routines and Utilities, Shadow Lawn Press 1986

Der Autor erledigt in einem Rundumschlag auf 375 Seiten so ziemlich alles was man in FORTH gerne implementieren möchte. Die Themen reichen von CASE-Statements, einen Screen Editor, 8086 Macro Assembler mit 8087 Unterstützung, String Funktionen bis zum Experten System und der Verarbeitung natürlicher Sprache. Doch die Aufmachung des Buches (jede Menge Druckfehler) und der Programmierstil des Verfassers (frei nach L. Brodie: "Die Kunst Namen zu vergeben") sind nicht unbedingt dazu geeignet die Verbreitung von FORTH zu fördern. Der verwendete FORTH-Dialekt "Atila" gehört wohl eher zu den Exoten (scheint so etwas wie F79-Standard auf IBM-PC zu sein, für $69.95 beim Autor zu erwerben). Quellenangaben fehlen gänzlich (War der Autor wirklich so fleißig und hat alles selbst gemacht ???). Das Inhaltsverzeichnis (eine Seite !!!) ist ein schlechter Scherz. Fazit: Ein guter Ansatz mit massiven Schwächen in der Realisierung.

Andreas Zilker
Hauptstr. 64
8045 Ismaning
Tel. 0 89/ 96 31 73
88-12-15

"Object-Oriented FORTH", Dick Pountain, Academic Press, London 1987, ISBN 0-12-563570-2, Preis: $16 (ohne Gewähr).

Von Dick Pountain ist im Verlag Academic Press das Buch mit dem Titel "Object Oriented FORTH" erschienen. Eine Darstellung des im Buch verwendeten FORTH-dialekts ist dem Lehrteil des Buches vorgelagert, so daß Anpassungen an andere FORTH-Systeme keine großen Schwierigkeiten bereiten. Dick Pountain beschreibt in 4 Kapiteln, wie er komplexe Datenstrukturen in FORTH implementiert. Dabei ist das Kapitel 1 als Einführung gedacht. Kapitel 2 ist der Teil, der sich eigentlich mit objekt-orientierter Programmierung beschäftigt. Kapitel 3 und 4 stellen weitere Datenstrukturen vor, die zur Implementierung anderer objektorientierter Sprachen nützlich sind. Kapitel 1, überschrieben mit "Records", stellt Schritt für Schritt eine FORTH-typische Implementierung von Records à la Pascal vor. Sehr schön zeigt er, wie man von gleichartigen Definitionen zu CREATE DOES Konstrukten übergeht. Er beschreibt geschachtelte Records, erklärt die Fachworte "early-" und "latebinding", sowie deren Vorzüge, und geht auch auf das Auslagern von Daten auf Massenspeicher ein (virtueller Speicher). Der bis hierhin vorgestellte Code paßt noch auf eine Seite und würde etwa 3 Screens füllen. Er ist damit eine kompakte aber doch sehr leistungsfähige und flexible Implementierung von Records in FORTH.

Das folgende Kapitel ("Abstract Data Types") erläutert die Technik, Programme mit Hilfe abstrakter Datentypen, Sammlungen von

Datenstrukturen nebst zugehörigen Bearbeitungsmethoden, zu strukturieren. Sehr bald geht er auf die Schlüsselidee seiner Implementation für objekt-orientiertes FORTH ein: lokale Dictionaries. Zu denen werden im folgenden Verwaltungsworte entwickelt. Es folgt eine Erklärung, was "instance variables" sind, und welche Probleme in der vorgestellten Implementation mit ihnen entstehen. Als nächstes wird das Vererben ("inheritance") erörtert. Dieser Teil ist sehr knapp gehalten, obwohl die Vererbung eines der wesentlichen Elemente der objekt-orientierten Programmierung ist. Das erläuterte Vererbungs-Konzept erlaubt es lediglich Methoden von einer Superklasse zu ererben. Instanz-Variablen müssen per Hand vererbt werden, und ein Vererben von mehreren Superklassen ist überhaupt nicht möglich. Nach der anschließenden Auseinandersetzung mit Object-Arrays folgt ein Beispiel, wie die bisher vorgestellten Konzepte angewendet werden.

"Lists", das dritte Kapitel, veranschaulicht, wie Listen in FORTH implementiert werden. Dabei wird auf einfach- und doppeltgelinkte sowie zyklische Listen eingegangen.

Das letzte Kapitel ("Memory Management Using a Heap") setzt sich mit der Heap Datenstruktur auseinander, die eine grundlegende Basis für dynamische Speicherverwaltungen darstellt. Sie wird auch von vielen objekt-orientierten Sprachen verwendet.

Alles in allem ist das gut 100 Seiten starke Buch eine anregende und empfehlenswerte Lektüre. Die Schwierigkeiten, die bei der Implementierung von objekt-orientierten Sprachen auftreten und die interessanteren Probleme dabei (multiple Inheritance, static & dynamic binding) werden aber gar nicht behandelt. Das Buch wird daher eher seinem Untertitel "Implementation of Datastructures" gerecht, dies aber in der Qualität, die wir von Dick Pountain gewohnt sind.

Ulrich Hoffmann

FORTH-Tagung 1989
der FORTH-Gesellschaft e.V.
7. bis 9.April in Aachen

FORTH '89

Zwischenbericht zur Vorbereitung der FORTH-Tagung 1989

Die Vorbereitungen zur FORTH-Tagung 1989 laufen auf vollen Touren. In der Zwischenzeit liegen uns zahlreiche Anmeldungen sowohl aus dem Kreis der Vereinsmitglieder, als auch aus dem der unabhängigen Anwender vor. Die Anfragen und Anmeldungen beschränken sich dabei nicht nur auf den Bereich der BRD, sondern erstrecken sich auch auf verschiedene europäische Nachbarländer, wie etwa DDR, Frankreich und den Niederlanden, vielleicht nicht zuletzt eine Folge der Standortwahl Aachen als Euregio-Stadt.

Die bisher vorliegenden Vortragsanmeldungen decken ein weites Spektrum der Programmiersprache FORTH ab. Mehrere Vorschläge beziehen sich auf den RTX-2000, aber auch allgemeine Themen für Einsteiger der Sprache liegen vor. Ein weiteres Thema stellt die Anwendung von FORTH in verteilten Rechnersystemen dar. Vorträge mit hauptsächlich wissenschaftlichen Akzenten werden die Anwendung von FORTH unter den Gesichtspunkten des modernen Software-Engineerings betrachten. Vorschläge hierzu decken den Bereich der objekt-orientierten Programmierung unter FORTH ab, es gibt aber auch allgemeine Betrachtungen über die Einordnung dieser Sprache in das Gefüge moderner Hochsprachen.

Die diesjährige enge Verbindung der FORTH-Gesellschaft mit der Technischen Hochschule Aachen bietet die Chance, interessierten Studenten einen ersten Kontakt zur Programmiersprache FORTH und zu den Anwendern dieser Sprache herzustellen. Für diese Zielgruppe werden zur Zeit noch Vortragsvorschläge gesucht. Aber auch Beiträge zu den oben erwähnten Schwerpunktbereichen werden noch gerne berücksichtigt. Aus organisatorischen Gründen wird jedoch jetzt um möglichst baldige Anmeldung des Themas gebeten.

Die Technische Hochschule Aachen stellt für die Veranstaltung einen Hörsaal (Forum 3 des Karman-Auditoriums) sowie für die begleitende Ausstellung das Foyer des Karman-Auditoriums zur Verfügung. Das Tagungsbüro ist ab Freitag, den 07. April 1989, 15:00 Uhr im Foyer des Karman-Auditoriums geöffnet. Im Anschluß an die Begrüßung der Teilnehmer wird Herr Professor Pleßmann den Einführungsvortrag zur Veranstaltung halten. Beim anschließenden gemeinsamen Abendessen werden sich sicher erste interessante Diskussionen entwickeln. Für den zweiten Veranstaltungstag sind zwei Vortragsblöcke geplant, welche durch hoffentlich umfangreiche Diskussionen begleitet sein werden. Der Vormittag des letzten Veranstaltungstages bleibt der ordentlichen Mitgliederversammlung der deutschen FORTH-Gesellschaft e.V. vorbehalten. Im Anschluß an die Mittagspause ist eine öffentliche Podiumsdiskussion zur Problematik der Akzeptanz der Programmiersprache FORTH speziell im Raum der BRD geplant.

Die Aachener Organisatoren würde es freuen, wenn sich aus dem Kreis der Teilnehmer des Raumes Aachen Interessenten zur Gründung einer lokalen FORTH-Gruppe fänden.

Also:

```
: Motto          ( 89 -- )
        FORTH    <3>    IF
        Aachen meet     THEN
;
```

In diesem Sinne,

die Tagungsorganisatoren

FORTH-Bibliothek, Teil 2

Dies ist eine auszugsweise Übersicht der FORTH-Bibliothek der Münchner Gruppe. Bei Interesse an einem Artikel kann man sich an Christoph Krinninger wenden. Diese Serie wird in den folgenden 'Vierten Dimensionen' fortgesetzt werden. Diesesmal werden eine Reihe von Implementationen vorgestellt. (verw. Abkürzungen: DDToF = Dr. Dobb's Toolbook of FORTH, DD Journal = Dr. Dobb's Journal)

° 6809 fig-FORTH	Listing des 6809 fig-FORTH		
° 6809-Assembler in FORTH	FORTH-Assembler in fig-FORTH	Dr. Erich Küster	c't, 7/84
° A 68000 FORTH Assembler	Der Standard FORTH-Assembler für 68000	Michael A. Perry	DDToF
° A Fast FORTH for the 68000	Grundsätzliches zu einem subroutine-threaded FORTH auf 68000	Lori Chavez	DD Journal, 10/87
° A FORTH Assembler for the 6502	6502-Assembler in fig-FORTH	William F. Ragsdale	DDToF
° A FORTH Native-Code Cross-Compiler	Stand-alone Programme mit einem high-level Compiler für 68000	Raymond Buvel	DDT of 68000 Programming
° A Mini FORTH for the 68000	Artikel und Assembler Listing (FLINT)	G. Yates Fletcher	DD Journal, 1/87
° A simple Metacompiler	By accepting a set of limitations, it is possible to construct an interactive meDr.	Guy T. Grotke & Guy M. Kelly	DDToF, Vol. 2
° Aufbau des Vokabular im FORTH-83	Elementares über Vokabulare im volksFORTH	Klaus Kohl	
° chipFORTH		Datenblatt	Firmenschrift
° clusterFORTH, a High-Level Network Protocol	Beschreibung eines Netzwerk-FORTH	Greg Bailey	Rochester FORTH Conference 1984
° cmFORTH, mpFORTH, the FORTHchip and Optimizing Compilers	Diskussion von FORTH-Versionen auf NC4000	Marco Pauck	euroFORML '87
° Combine FORTH with other tools for rapid software development	Dies und Das über FORTH auf 8051	W. H. Payne	Electronic Design 01/88
° Comfort: A Faster FORTH	Tips für die Implementation eines subroutine-threaded FORTH auf 68000	Alexander Burger and Ronald Greene	DDT of 68000 Programming
° Controls from FORTH	Buttons, Check boxes etc. unter FORTH	Jörg Langowski	Best of MacTutor, Vol. 1
° Cross-Assembler unter FORTH	Grundlegendes über Cross-Assembler		65xx Micro Mag
° Der Schlüssel zur Maschine	Ein 68000-Assembler in FORTH	Dr. Erich Küster	c't, 11/86
° Design Considerations for a Microcontroller Development Design	Die Arbeit von FORTH auf Microcontroller	Andrew Haley	euroFORML '88
° Design of a FORTH Target Compiler	Implementation eines Recompilers für MVP-FORTH	Howard H. Robinsonm et al.	DD Journal, 10/85
° Die neuen Systemworte von SFORTH 3.1	Speicherresidente Targets und Interrupts in SFORTH		DOS Extra Nr. 2
° Disk-Interface für fig-FORTH	Disk-Interface für 3040, 4040 und 4031		65xx Micro Mag
° Ein-/Ausgabe im FORTH-83	Elementares über INPUT/OUTPUT und DEFERed words	Klaus Kohl	
° fig-FORTH 6502	Assembler Listing		fig
° fig-FORTH Installation Manual	Glossary, Model, Editor des fig-FORTH		fig
° FORT(H)schritt mit MACH2	Bericht über ein subroutine-threaded FORTH für 68000	Peter Glasmacher	c't, 7/87
° FORTH and the Motorola 68000	Mögliche Varianten eines inneren Interpreter auf 68000	Joe Barnhart	DDToF
° FORTH Anpassen, Teil 1: Die Systemanpassung	Anpassung des fig-FORTH an Apple II	Peter Glasmacher	c't, 3/84
° FORTH Anpassen, Teil 2: Kommunikation mit FORTH	Anpassung des fig-FORTH an Apple II	Peter Glasmacher	c't. 3/84
° FORTH Anpassen, Teil 3: Der Massenspeicher	Anpassung des fig-FORTH an Apple II	Peter Glasmacher	c't, 5/85
° FORTH Anpassen, Teil 4: Tips für Tapes	Anpassung des fig-FORTH an Apple II	Peter Glasmacher	c't, 7/84
° FORTH Anpassen, Teil 5: Diskettensimulation	Anpassung des fig-FORTH an Apple II	Peter Glasmacher	c't, 10/84
° FORTH für den QL	Diverse FORTH-Produkte für den QL	Klaus Gütter	Comp. Kontakt 2-3/88
° FORTH Goes to Sea	Implementation von FORTH-79 auf 6805 (ROM-based)	Everett Carter	DD Journal, 7/86

FORTH auf dem Atari ST

Ein Angebot an alle, die sich für 32FORTH (Atari ST-FORTH-System) interessieren. Sie können bei uns zu einem reduzierten Preis die beiden M&T-Bücher 'Programmieren in FORTH' (statt DM 49,- für DM 39,20), und '32FORTH-Compiler' (statt DM 69,- für DM 59,20) beziehen.

D.LUDA Software,
Staudingerstr.65,
8000 München 83,
Tel. 089/6708355

KLEINANZEIGEN

ISAM-Modul gesucht

Ich werde mit H/S-FORTH arbeiten und will mit diesem System eine Datenbankanwendung schreiben. Dazu benötige ich ein ISAM-Modul, möglichst ein B-Tree Modul. Wer weiß etwas über ein solches Produkt im Public Domain Bereich? Es muß nicht unbedingt für H/S-FORTH geschrieben sein, ich könnte es auch umschreiben. Volker Lendecke, Senner Hellweg 45, 4800 Bielefeld 12, Tel. 0521/491025

FORTH-68, ...

... ein 32-Bit FORTH für Hardware mit MOTOROLA M68000 CPU's. Target-compilersystem, verwendet PC als Hostrechner. Alle Quellen incl. Handbuch für DM 398,--. Info bei Stephan Klus, Tel. 0241/571431.

Telemodem 1200 zu verkaufen.

Das berühmte und bewährte Modem des FORTH-Tree, 300 Baud vdx und 1200 Baud hdx für Postmodem MDB 1200-03 oder MDB 1200-05. VB 450,- DM. Näheres vom FORTH-Büro Tel: 089/ 317 37 84

Preisliste vom 22.02.89: volksFORTH und ultraFORTH der FORTH-Gesellschaft e.V.

1. ultraFORTH 3.8 für Commodore C64, C16 und PLUS 4

Disketten (4*170K, 5 1/2")	DM 25.--
Handbuch	DM 65.--

2. volksFORTH 3.80 für ATARI ST

Disketten (3*360K, 3 1/2", einseitig)	DM 25.--
Handbuch	DM 65.--

3. volksFORTH 3.8 für CP/M2.2-Systeme

Diskette (Osborn-Format 5 1/4")	DM 15.--
Diskette (Kaypro 5 1/4")	DM 15.--
Disketten (Schneider CPC zwei 3")	DM 25.--
Handbuch	DM 70.--

4. volksFORTH 3.81.2 für IBM-PC und Kompatible

Diskette (1*360K, 5 1/4")	DM 15.--
Handbuch	DM 70.--
Floating-Point-Paket für 80(2)87	DM 30.--

5. F83 Laxen&Perry für IBM-PC und Kompatible

Disketten (3*360K, 5 1/4")	DM 50.--

6. Sourcecode der VD Vol IV Nr.1 bis Vol V Nr.1

Diskette	DM 15.--

Aufpreis für Verpackung und Versand:

Diskette(n)	DM 3.--
Handbuch (mit Disketten)	DM 4.50
Nachnahme	DM 1.70

Preisnachlaß für Mitglieder der FORTH-Gesellschaft e.V.:

Diskette	- DM 3.--
Handbuch	- DM 17.--

Bezugsquelle:Michael & Klaus Kohl , Pestalozzistr. 69, 8905 Mering
Tel.: 08233/30524
Bankverbindung:Postgirokonto München, BLZ 700 100 80 2835 06-800

Mit der Ausgabe dieser Liste verlieren alle bisher angegebenen Preise für volksFORTH und ultraFORTH ihre Gültigkeit. Mitglieder der FORTH-Gesellschaft e.V. erhalten Ermäßigungen sowohl für Disketten als auch für das Handbuch. Der Rechnungsbetrag kann entweder direkt auf das Konto überwiesen, per Nachnahme eingezogen oder als Verrechnungsscheck beigelegt werden. Bitte geben Sie unbedingt die gewünschte FORTH-Version an.

Möchten Sie ...,

... daß an dieser Stelle eine Kleinanzeige von Ihnen steht? Dann senden Sie uns einfach den Text zu. Für Mitglieder der FORTH-Gesellschaft e.V. ist dies kostenlos, andere zahlen DM 5,- für 5 Zeilen und für jede weitere Zeile zusätzlich eine Mark.

Die nächste 'Vierte Dimension' erscheint im Juni '89.

Gruppen

Lokale FORTH-Gruppen, die sich regelmäßig treffen:

1000 Berlin	Claus Vogt, Tel.: 030/2168938. Treffen am letzten Donnerstag des Monats um 19.30 Uhr in der Technischen Universität Berlin, Mathematikgebäude, 6.Stock im Raum MA 621
2000 Hamburg 13	Karsten Roederer, Tel. 040/4104446, tagsüber 412 329 84, Treffen im Geomatikum Raum 1438 (14. Stock), Bundesstr. 55, am 24. April, 24. Mai, 28. Juni, 27. September, 25. Oktober jeweils um 19.30 Uhr
4130 Moers 1	Friederich Prinz, näheres Tel: 02841/583 98
Rhein-Ruhr	Jörg Plewe, Tel: 0208/291 66, Treffen jeden 4. Freitag im Monat um 20.00 Uhr im Bahnhof Ottenbruch, Funckstraße, Wuppertal-Elberfeld
6100 Darmstadt	Andreas Soeder, Tel. 06257/2744. Treffen an der VHS an einem Mittwoch in der Mitte des Monats.
6800 Mannheim	Lokale Gruppe Rhein-Neckar, Thomas Prinz, Tel.: 06271/2830. Treffen jeden ersten Mittwoch im Monat im Vereinslokal des Segelflugvereins Mannheim e.V. Flugplatz, Mannheim-Neuostheim.
8000 München	Heinz Schnitter, Tel. 089/3103385 und Christoph Krinninger 089/7259382. Treffen jeden 4. Mittwoch im Monat 19 Uhr 30 im Vereinsraum 1 im Bürgerhaus Unterschleißheim am Rathausplatz (S-Bahnhaltepunkt S1 Unterschleißheim).

FORTH-Fachgruppen:

8000 München	RTX 2000 Gruppe, Koordinator Max Diez, Treff- und Zeitpunkt wie oben bei der lokalen Münchner Gruppe.
6800 Mannheim	FIS (FORTH Integriertes System) - Datenbank, Textverarbeitung, Kalkulation, Postadresse: Dr. med. Elemer Teshmar, Danziger Baumgang 97, 6800 Mannheim 31

Es möchten in ihrer Region eine Gruppe gründen:

7000 Stuttgart 31	Wolf-Helge Neumann, Huttenstr. 27, Tel. 0711/882638.
8500 Nürnberg 20	Thomas G. Bauer, Fichtestr. 31, Tel. 0911/538321.
5000 Köln 60	Michael Heycke, Boltensternstr.
4830 Gütersloh 1	Ludwig Röver, Holzheide 145A
4900 Herford	Andreas Findewirth, Im Großen Vorwerk 48, Tel.: 05221/23504

Eine Fachgruppe will gründen:

7000 Stuttgart 80	Grafik/Arithmetik, Jörg Tomes, Anweilerweg 56, Tel. 0711/7802293.
8000 München 70	Btx u. FORTH, Christian Schwarz, Lindenschmitstr.30, 8000 München 70

Hier kann man um Rat fragen:

02103/556 09	Jörg Staben, Dienstag und Freitag, 20.00 - 22.00 Uhr
02845/28951	Karl Schroer

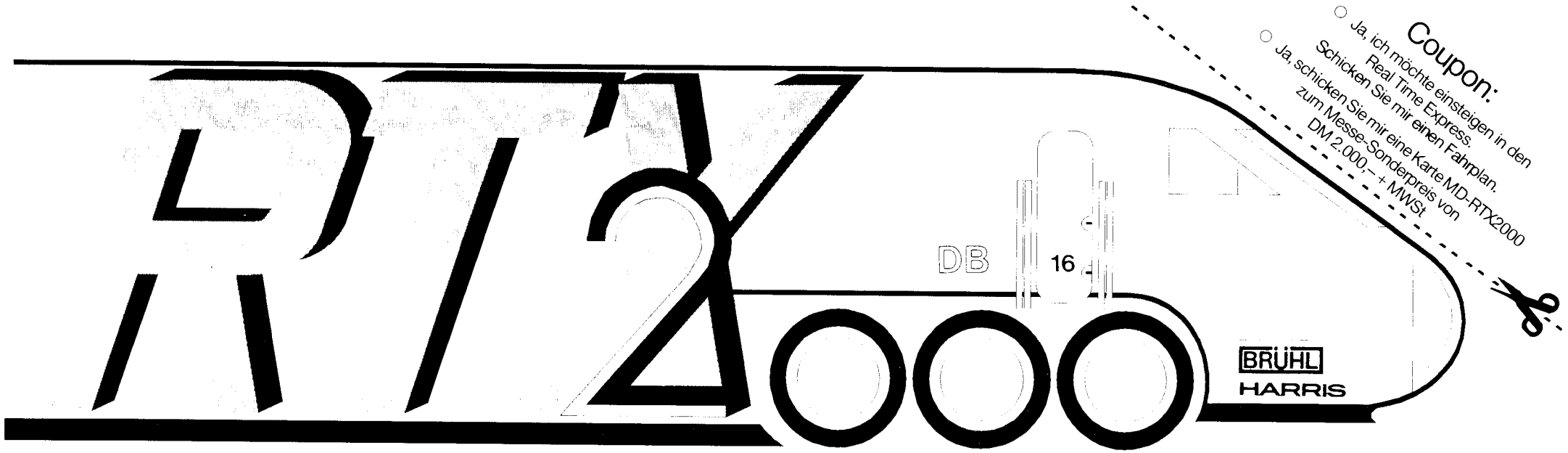
Ansprechpartner zu bestimmten Interessengebieten:

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

FORTH-Gesellschaft e.V. - Postfach 1110 - D-8044 Unterschleißheim

Postgiroamt Hamburg, Kontonr.: 563211-208 BLZ 20010020,Tel.089/3173784

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



Coupon:

○ Ja, ich möchte einsteigen in den Real Time Express.

○ Ja, schicken Sie mir einen Fahrplan.

Schicken Sie mir eine Karte MD-RTX2000 zum Messe-Sonderpreis von DM 2.000,– + MWSt

Die Zukunft hat schon begonnen ....

Warten Sie nicht auf Transkarbid oder Grande Tristesse!
Steigen Sie jetzt ein in den Real Time Express.
Nur mit dem Real Time Express gewinnen Sie Echt Zeit.
Sie brauchen Ihr Fahrzeug nicht mehr mit Assembler zu tunen.
Genießen Sie das Hochgefühl mit Hochsprache und Echt Zeit Ihr Ziel zu erreichen!

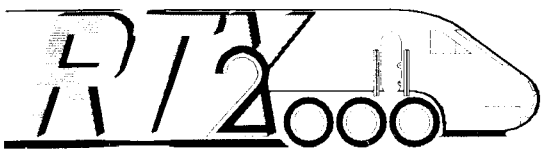
FORTH

SONDERPREIS für Mitglieder der FORTH-Gesellschaft:

DM 1.777,--

(incl. Versand, incl. MWSt)

Hochschulrabatt auf Anfrage!



Karten zum Einsteigen erhalten Sie ab sofort bei:
BRÜHL ELEKTRONIK ENTWICKLUNGS-GESELLSCHAFT mbH, Hegelstraße 10, 8500 Nürnberg 10, Tel. 0911/359088
QuadCard 100×100 mit RTX2000, 128 KB RAM, 64 KB PROM,
40 MHz Quarztakt: DM 2.000,– + MWSt (electronica Sonderpreis incl. FG-FORTH-Compiler)

FORTH-SYSTEME

Software-Vertrieb
Beratung und Schulung

ANGELIKA FLESCH

Software-Entwicklung
Hardware-Entwicklung

UR/FORTH

- Forth-83 Standard
- Für MS-DOS, OS/2, 80386, 68000 UNIX und XENIX
- 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

HARRIS RTX 2000

Informieren Sie sich über diesen Prozessor, der auch von uns unterstützt wird.

DSP APPLIKATIONEN

DSP Anwendungen mit dem AT & T DSP-32. Informieren Sie sich über unser Angebot.

FORTH MAIL BOX

Für alle FORTH-Interessierten hat unsere Firma eine Mailbox eröffnet. Sie ist unter der Nummer 0 76 67 556 zu erreichen und akzeptiert 300, 1200 und 2400 Baud, 8N1. Außer einer offenen Hauptkonferenz und einigen Fileareas enthält sie auch Supportkonferenzen für unsere komplette Produktlinie.

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	● 6802
● 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

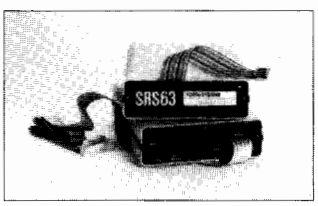
Entwickeln Sie romfähige Programme ?

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.

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

Kühnholmerstraße 21 · Postfach 1103 · D-7814 Breisach · Tel. (0 76 67) 5 51 · Fax 07667 555