VIERTE DIMENSION

3/1993

9. Jahrgang 1993 3. Quartal DM 10,--

DisCo

SI-Nuß

von BASIC zu Forth

Menüsteuerung

CAN-Bus

ZF-Vorstellung

F68KANS

UCS

FORTH MAGAZIN

Organ der Forth-Gesellschaft e.V.

IMPRESSUM

Name der Zeitschrift
VIERTE DIMENSION
FORTH MAGAZIN
Organ der Forth-Gesellschaft e.V.

Herausgeber
Forth-Gesellschaft e.V.
Postfach 1110
85701 Unterschleißheim
Tel.: 089-3173784 oder
Forth-Mailbox Tel. 089-8714548 8N1

Redaktionsleitung
Rolf Kretzschmar (rk), (verantwortlich)
Rote Gasse 7, 52499 Baesweiler
(Redaktionsadresse)
Tel/Fax: 02401-88891

Redaktion
Arndt Klingelnberg (akg), Alsdorf
Tel.: 02404-61648 Fax: 02404-63039
Klaus-Peter Schleisiek (kps), Aachen
Tel/Fax: 0241-873462

Layout, Satz, Herstellung
ORGA Sport, Rilkestr. 8, 52477 Alsdorf
Tel/Fax: 02404-61425

Grafik, Illustration, Layout
Rolf Kretzschmar (rolf)

Anzeigenverwaltung
Arndt Klingelnberg
Straßburger Str. 12
52477 Alsdorf
Tel.: 02404-61648 Fax: 02404-63039

Redaktionsschluß
Feb., Mai, Aug., Nov.

Erscheinungsweise
vierteljährlich

Auflage
1000

Preis
Einzelheft DM 10,-, Abonnementpreis DM 50,-, bei Auslandsadresse DM 55,- inklusive Versandkosten

Der Ärger

von Rolf Kretzschmar

Sie ärgern sich, wenn das Heft so spät kommt. Ich ärgere mich, wenn mal wieder Termine platzen oder dazwischenkommen. Die Teilnehmer an der Forth-Tagung ärgern sich, daß sie den Tagungsband noch nicht in Händen haben. Inserenten ärgern sich, wenn noch nicht die neue Postleitzahl in ihrer Anzeige steht. Das Forthbüro ärget sich, weil die Zahlungsmoral der Mitglieder zu wünschen läßt. Das Direktorium ärgert sich über das Finanzamt. Der Autor A. aus B. ärgert sich, weil bereits zum x-ten Mal sein Artikel verschoben wurde. Der Autor C. aus D. ärgert sich, weil er Zeichen vermißt (oder zuviele vorfindet). Wir alle ärgern uns, daß man Forth nicht den ihm gebührenden Platz in der Literatur einräumt (...wozu braucht man überhaupt andere Programmiersprachen...). Ich ärgere mich, daß sich akg ärgert, weil zu wenig für Profis im Heft steht. Ich muß dann entgegnen, daß doch von den Profis kaum etwas kommt. Und das ist ja dann wirklich ärgerlich. Ich will nicht, daß sich unsere Forth-Frischlinge ärgern. Darum nehme ich auch zukünftig in Kauf, daß sich Kenner, Künstler und Könner ärgern, weil mal wieder Triviales im Heft zu finden ist. Ärger hebt den Adrinalin-Spiegel! Wer sich ärgert, fühlt sich betroffen. Diese nützliche Begleiterscheinung des Ärgers sollten wir produktiv einsetzen: Greifen wir in die Schubladen, und füllen wir den Redaktionsstapel mit Meldungen und Meinungen, mit Beschreibungen und Berichten, mit Kniffen und Knobeleien, mit Briefen und Bildern. Das Heft kann nur so gut sein, wie die Beiträge, die von den Mitgliedern unseres Vereins kommen. Und wer sich zu sehr über die Unpünktlichkeit des Heftes ärgert, sollte den Finger heben und sagen: "Schluß mit all dem Ärger! Jetzt mach ich die Zeitung!" Was mich angeht, so will ich mich gerne noch eine Zeit lang weiter ärgern. Nur kann ich auch in Zukunft nicht versprechen, daß die VD wie eine professionelle Zeitschrift immer zu festen Zeiten erscheint. Vielleicht mildert das ja Ihren Ärger: Ich habe wie Sie (und Sie und Sie) einen Beruf, der wechselnden zeitlichen Einsatz erfordert. Und ich habe eine Familie. Die VD mache ich nebenher! Ohne die Unterstützung von *Klaus-Peter* (kps), *Arndt* (akg), *Boris Müller* und *Lothar Palm* hätte ich bereits passen müssen.

Ich wünsche mir, daß aus Ärger kein Frust wird...

Euer

R. Kretzschmar

Bericht des Direktoriums

von Thomas Beierlein, Ulrich Hoffmann, Jörg Staben

Die Sommermonate waren in diesem Jahr durch Urlaub und Ferien geprägt. Folgendes berichtenswertes hat sich ereignet: Dank der großen Initiative von *Jens Wilke* (SysOp der VD-Mailbox) steht uns seit August im *Internet* die Newsgroup "DE.COMP.LANG.FORTH" zur Verfügung. Zur Zeit gibt es an manchen Orten noch Schwierigkeiten, die Gruppe lokal zu beziehen. Wir hoffen, daß sich diese Probleme bald klären und die Gruppe aktiv genutzt wird. Wie in der letzten VD nachzulesen, ist die Echtzeit '93 für die Forth-Gesellschaft erfolgreich gewesen. Wir danken den MitstreiterINNEn *Hans Jörg Platz*, *Arndt Klingelnberg* und *Dorothea Knopf* für ihr Engagement und ihre Geduld. Auf der Echtzeit '94 in Hamburg sollten wir auf jeden Fall vertreten sein. Zur Vorbereitung suchen wir Ideen, Projekte und Anregungen. Die Auslieferung des Tagungsbandes der Jahrestagung in Nürnberg hat sich bedauerlicherweise stark verzögert. Dank der Initiative von *Heinz* und *Ulrike Schnitter* vom Forth-Büro wird er bald ausgeliefert (siehe Anmerkung der Redaktion). Als Austragungsort für die Tagung 1994 stehen z.Zt.

- abermals Nürnberg (*Dirk Brühl*) und
- der Raum Hamburg (*Jens Storjohann*, *Michael Kalus*) zur Debatte.

Wer sonst möchte die Tagung noch ausrichten?

Tagungsband erhältlich

Dirk Brühl faxte uns in letzter Minute die Themen und Autorenliste zum Tagungsband der Forth-Tagung '93 zu. Die Tagungs-Teilnehmer bekommen den Band bald zugeschickt. Es können aber auch zusätzliche Bestellungen an *Dirk Brühl* gerichtet werden. Siehe Seite 27.

Meldung aus dem Forth-Büro

Der VD-Redaktion kam zu Ohren, daß noch einige Sweat-Shirts mit dem SWAP-Drachen und Forth-Aufdruck zu haben sind. Siehe Anzeige im Heft 2/93.

❑

Termine

akg

PRODUCTRONICA (Electronic-Fertigung)
1993nov09--13 München

FORML Conference
1993nov26--28 Asilomar, Monterey, CA, USA

CeBIT
1994mar16--24 Hannover

Hannover (Industrie) Messe
1994apr20--27 Hannover

Forth Kongress94

Mitgliederversammlung
1994mar//apr exakter Termin und Ort liegen noch nicht fest

(bitte geben Sie uns Ihren Input zu für Forthler sinnvollen Veranstaltungen: Forth-Kurse, Forth-Anwendungen, Forth-Kunden, Forth-Konkurrenten, Hardware für Forthler)

Neue Ecke: FORTH/2

rk Neben der DisCo-Ecke, der F86K(ans)-Ecke, der Wanted-Ecke, der ....-Ecke nun auf Anregung von *Friederich Prinz* die Forth/2-OS/2-Ecke in der VD.

Das klingt jetzt etwas ironisch und ist auch so gemeint. Nicht, weil ich glaube, daß dies eine Ecke zuviel sei; nein, ich sehe nur, daß die Aktivisten, die mit Elan "ihre" Ecke bauen, dort oft alleine stehen. Wünschen wir *Friederich Prinz* eine bessere Resonanz. Zur Erinnerung:

In der VD2/93 stellte er dieses Forth-System vor und merkte an, daß es sich um Shareware handele. Nun erhielt ich von ihm folgendes Update:

"Mike Warot hat mir (Prinz, d.Red.) via INTERNET eine Nachricht zukommen lassen, daß er sein FORTH/2 für alle deutschen Interessenten 'frei' macht. Weder eine Registrierung ist zukünftig notwendig, noch irgendeine Bezahlung. FORTH/2 darf innerhalb der BRD (?, d.Red.) völlig frei an jeden Interessenten weitergegeben und beliebig genutzt werden. Diese Weitergabe bezieht sich ausdrücklich auch auf die Assemblerquelltexte, die Mister Warot seinen letzten Sendungen beigefügt hat."

Wie *F. Prinz* mir weiter meldet, stellt er sich für den deutschsprachigen Support zur Verfügung. Er hat bereits begonnen, einige Änderungen am Kern vorzunehmen. Es wäre daher ganz gut, wenn *F. Prinz* als Koordinator für Erweiterungen/Änderungen fungieren können. Wie ich den Fritz kenne, übernimmt er dieses Amt gerne. Wenn das wirklich funktioniert, daß sich Interessenten, Erweiterer und Verschlimmbesserer bei ihm melden, dann habe ich keine Sorge, daß eine Forth/2-OS/2-Ecke wächst und gedeiht!

Hier einige Anregungen und Fragen von Friederich Prinz für die zukünftigen Beschicker der Forth/2-OS/2-Ecke:

- ☐ Wie kann man unter Forth/2 die Systemaufrufe nutzen?
- ☐ Wer erweitert Forth/2 noch um die fehlenden Systemfunktionen?
- ☐ Wer schreibt erste Applikationen unter Forth/2?
  Wichtig wären zunächst:
  – Systemmonitore
  – Interprozess-Kommunikationen
- ☐ Wieviele Mitglieder der F.G. oder Leser der VD nutzen OS/2 und Forth/2?
- ☐ Wer ist bereit, beim Support zu helfen?
- ☐ Wer beschreibt empfehlenswerte Literatur zum Thema OS/2-Forth/2?
- ☐ Wer liefert Informationen über kommerzielle Produkte unter 'echten' 32-bit Forth-Systemen?

Inhalt

Verschiedenes

Object Code contra Metacode

von Andreas Goppold
Unterherrenhauserstr. 13, 82547 Eurasburg, Tel: 08179 1479

Um kurz zu definieren, wovon hier die Rede ist:

Object Code Compiliation ist das Software-Paradigma, unter dessen Anwendung die formale Beschreibung einer zu lösenden Aufgabe (das Computerprogramm) durch geeignete Werkzeuge (Compiler) in das Object-Code Format (oder den native Binary-Code) des Computers übersetzt wird, auf dem die Aufgabenlösung abgearbeitet werden soll.

Metacode Programmierung bedeutet, daß ein (native Object-Code) Kontrollprogramm im Arbeitsspeicher des Computers (permanent) resident ist, das die formale Beschreibung einer zu lösenden Aufgabe (das Computerprogramm) interpretierend abarbeitet.

Metacode Programme sind Zwitter in der VonNeumannschen Kategorisierung von Programm und Daten. Diese Zwitternatur widerspricht der sauberen dualen Trennung, die man eingeführt hat, um Programmierung wissenschaftlich und ingenieurmäßig zu behandeln. Die Tatsache, daß Metacode sowohl Programm als auch Daten ist, ermöglicht aber eine ganz andere Herangehensweise an das Thema der Programmierung.

In der Computertechnologie erleben wir heute einen Paradigmen-Streit von CISC contra RISC. Die Software-Technologie der Object Code Compiliation ist mit der Technologie der CISC Prozessoren verbunden.

Vergleichbar mit der Polarisierung RISC / CISC hat es in der Geschichte der Computerindustrie eine ebenso interessante Auseinandersetzung zwischen den Object-Code contra Meta-Code Verfechtern gegeben. Diese Auseinandersetzung war zwar interessant, aber sehr einseitig. Da bisher immer die reine Verarbeitungs-Geschwindigkeit eines Rechners, also die optimale Nutzung der Maschine im Vordergrund stand, gewannen die Objekt-Code Proponenten bisher immer. Einige Systeme sind dabei auf der Strecke geblieben, andere waren akademisch interessant und kommerziell unfruchtbar, und einige führen ein recht reges Leben in recht unbekannten Nischen: *UCSD-Pascal* mit der P-Code Maschine, *Smalltalk* mit der Bytecode-Maschine, das *PICK* System, *Forth* und einige Abkömmlinge wie *Actor* und *Amber. Lisp* kann auch dazugezählt werden, sowie *APL* und *Mumps*. Ebenfalls viele *BASIC*-Systeme, die incrementell zu einem Tokencode compilieren. Das heute bekannteste und am meisten verwendete Metacode System ist *Postscript.*

Warum fanden sich in der Vergangenheit immer wieder so engagierte Verfechter dieses Paradigmas, das ja von der Prozessor-Ökonomie gesehen keine Chance hatte? Es ist sehr einfach, wenn man einige solcher Sprachen kennt, zu sehen, warum Programmierer, die einmal mit einem solchen System zu tun hatten, so ungern wieder davon lassen (siehe auch: Die Gänseküken von Konrad Lorenz). Metacode-Systeme machen den Programmierern das Leben entscheidend leichter. Sie verlagern einen Großteil der Komplexität des Programmierer-Daseins dahin, wo sie hingehört, auf den Rechner. Erstaunlicherweise sind diese Ansätze so alt wie die Computerei selber, zum Beispiel APL oder LISP. Die guten Ideen sind sehr alt, aber heute ist endlich die Zeit gekommen, in der sie wieder ihre Auferstehung feiern dürfen. Der Grund ist - wieder mal - in der jetzigen Verschiebung der Technologie-Balance zu finden.

Seit ungefähr zehn Jahren hat sich die Balance insofern entscheidend verschoben, als daß nicht mehr die Maschine, sondern der Faktor Mensch der Grenzfaktor ist. Allerdings ist heute erst der größere Teil der EDV-Industrie mit den Schlagworten "Software-Krise" und "Downsizing" zu dieser Erkenntnis aufgewacht. Der *Macintosh Computer* war sozusagen das Fanal dieser Entwicklung. Mit der ersten Version dieser Maschine hatte man einen Rechner konstruiert, in dem 90% der Prozessor-Leistung auf das User-Interface konzentriert war. (Zumindest bei dem ersten "Toaster". Die 68030er Modelle haben ja noch etwas Power übrig) Es gab da noch nicht mal eine Programmier- Schnittstelle; dafür sollte man die Lisa nehmen.

Das war vor 20 Jahren undenkbar, und ist heute dadurch möglich geworden, daß man so etwas für 10.000 bis 20.000 DM bekommt (seit Apple die Preisstürze im PC-Bereich mitmachen mußte, auch für weniger). Ein weiterer Vertreter dieser Philosophie ist *NeXT* (von Steve Jobs, Vater des Macintosh, of course). Was dem Anwender recht ist, ist dem Programmierer billig. *NeXT* ist entscheidend auf dem Weg, denselben Quantensprung für den Programmierer zu liefern, wie der Mac für den "unbedarften" Benutzer. Und *NeXT* benützt als sein Display-Paradigma das Postscript-System (ein Metacode) im Gegensatz zum Industrie-Quasi-Standard X-Windows.

Die Chance

Ich sehe so etwas wie eine historische Chance. Die Computer-Industrie steckt in einer tiefen Krise. Der Rest der Welt anscheinend nicht minder. Lösungen werden dringend gesucht und teuer bezahlt. In der Forth Community existiert ein großes Potential an Know-How im Bereich Metacode, das in irgendeiner Weise organisiert ist. Sollte sich diese Gruppe koordiniert und konstruktiv zu den augenblicklichen Problemen der Computerindustrie äußern können, dann ist es möglich, ein Comeback der Forth Idee zu organisieren.

Dazu sind allerdings einige Vorbedingungen nötig. Es ist nötig, gewissen Ballast abzuwerfen, um neues Terrain zu gewinnen. Und eine solche

Entscheidung kann schwer fallen. Nicht alle wollen den Schritt machen. Um in der Computerwelt Eingang zu finden, müßte man die Kettung an die spezifische Erscheinung von Forth aufgeben. Was hier wichtig ist, sind die Strukturen, die unter der Oberfläche von Forth liegen. Der Tokenlisten- Interpreter. Die Möglichkeit, z.B. über das Token Threading einen Code mit minimaler Größe zu erstellen. *Postscript* ist der Beweis, daß Tokencode-Systeme eine Zukunft haben. *Postscript* ist selber alles andere als effizient oder elegant, aber es ist die beste Lösung für eine bestimmte Klasse von Problemen.

Wenn die Vierte Dimension wirklich für Leser außerhalb der Forth-Gesellschaft interessant sein will, muß die enge Bindung an die konventionelle Erscheinung von Forth aufgegeben werden. Das Thema "Metacode-Systeme" ist heiß und ein Markt mit Zukunft. Eine Zeitschrift, die sich auf dieses Thema konzentriert, wird eine breite Leserschaft finden. Wenn die Forther lernen, daß sie Experten für eine bestimmte Art von Metacode Systeme sind, dann werden sie plötzlich feststellen, daß ihr Wissen sehr begehrt ist.

Ich bin überzeugt, daß ich mit meinen Analysen in der Vergangenheit immer richtig lag, und daß ich die Fähigkeit habe, aus der historischen Perspektive Dinge zu erkennen, wenn sie noch hinter dem Konsensus-Realität-Horizont sind. Forth hat seine historische Chance vor fünzehn Jahren vertan. Nach allen Gesetzmäßigkeiten würde Forth den Weg aller abgestorbener Zweige der Evolution gehen. Hier ist eine weitere Möglichkeit. Unter dem Namen von Forth kann man sie nicht gehen, aber die Substanz kann erhalten bleiben.

❑

Gehaltvolles

kps

Inhaltsübersicht: Forth Dimensions der Forth Interest Group, USA

Heft 2, Vol. 15 Juli/August 1993

Heft 3, Vol. 15 Sept./Oktober 1993

Sessions der euroFORTH'93

In Mariánské Láznê / Marienbad (Tschechien) fand vom 15. bis 17 Oktober 1993 die euroFORTH '93 statt. Wer über Was berichtete, präsentieren wir ofenfrisch.

Session 1 : Formal FORTH
Friday 15.30
Moderator : *Sergei N. Baranoff*
Questions : How advanced is FORTH formalisation? Why does it take so long? Can FORTH be formalised at all? How do users benefit from formalisation?
P.J. Knaggs : Towards a Formal FORTH
J. Pöial : Some Ideas on Formal Specification of FORTH Programs
U. Hoffmann : Static Stack Effect Analysis
B. Stoddart : The Halting Problem in Forth
B. Stoddart : Self Reference and Type Inference in Forth
P.J. Knaggs : A Look at FORTH's Academic Standing. Genaral Discussion

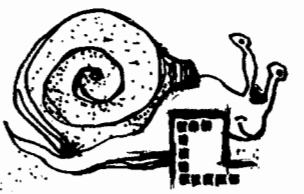
Session 2 : Evolution of FORTH
Saturday 9.00
Moderator : *Klaus Schleisiek-Kern*
Questions : Can we factor FORTH more? Which new capabilities are needed?
M.A. Ertl : A Portable FORTH Engine
M.L. Gassanenko : Context-Oriented Programming
M.L. Gassanenko : Multi-CFA DOES>
J.D. Carpenter : The Common Sense Pattern Classifier.
General Disussion

Session 3 : Software Engineering
Saturday 14.30
Moderator : *Rob Spruit*
Questions : What do we need to make FORTH into a good CASE tool? What is missing? Why is FORTH not on the forefront any longer? Which compromises are necessary? How do we programm across system boundaries? How do we interconnect host and target system? What are the Forth paradigms? What is the essence of FORTH w.r.t. other languages?
M. Dahm : A Graphical User Interface in natOOF
C. Lavarenne : Porting FORTH through C on Workstations
N.J. Nelson : Quick, Make a Weighing Machine!
R. Borrell : "Error-0 undefined" versus "C1000 Unknown Fatal Error..."
W. Wejgaard : Interactive Cross Platform Development
General Disussion

Session 4 : Products in Forth
Sunday 9.00
Moderator : *Kurt Heinz*
Questions : Does FORTH result in better products? What kinds of problems is FORTH good at?
V. Lolov : Increasing RTX-2001 Return Stack Depth
M. Bugler : AFORTHable ATE - Implementing Inexpensive Automatic Test Systems using FORTH
K. Schleisiek-Kern : IX: The First FORTH Single Chip Controller
General Disussion

Mit vielem Dank an das DELTA t - TEAM, insbesondere Marina Kern. akg

Schneckenrennen im DisCo



von Klaus-Peter Schleisiek
An den Finkenweiden 38, 52074 Aachen, Tel.: 0241/ 873462

Die von Michael Prümm vorgestellte Schnecke (VD 2/93) auf dem 8x8 LED-Feld - bewußt klar und einfach gehalten - nutzt die DisCo-Software kaum aus. Nehmen wir doch mal die Abfrage (mit GET) ins Spiel.

Michael Prümms Schnecke ist mittels Turtle-Grafik auf DisCo elegant programmiert und tut brav ihren Dienst, eine Art quadratischer Spiralen zu zeichnen.

Aber sie ist ein bißchen stur. Könnte man sie nicht mit Hindernissen auf ihrem Weg etwas necken? Die sensible Version ersetzt die abgezählten Schritte mit Drehung durch eine einfache Ausweichregel, die beim Anstoß an Hindernisse angewendet wird. Der alternative Schneck geht dann etwa so:

```
: schneck
  begin anstoß?
    if ausweichen
    else klecks
    then
     schritt frustriert?
  until ;
```

Ausweichen geschieht einfach durch das Hakenschlagen wie bei Michaels Schnecke. Wenn als Hindernis der Rand des Feldes erkannt wird, läuft der Schneck einmal den äußeren Rand ab. Werden bereits eingeschaltete Lampen ebenfalls als Hindernisse bewertet, gehen die folgenden Umläufe immer innen am vorhergehenden entlang. Wenn das Feld ganz gefüllt ist, versagt das Ausweichen. Unmittelbar aufeinander folgende Versager werden in der Variablen FRUST gezält und führen bei Unerträglicheit zum Abbruch.

Stichworte

DisCo
Lampenfeld
Turtle-Grafik

Unterschiedliche Startpunkte mit unterschiedlichen Startrichtungen führen zu verschiedenen Schnecken; starten Sie zur Abwechslung einmal in der Mitte! Oder necken Sie den Schneck, indem sie einzelne Plätze des Feldes schon vor dem Start besetzen.

Noch lebendiger wird das Tierchen, wenn es mit einer Vorliebe ausgestattet wird. Wenn es zum Beispiel mit Vorliebe einen Rand entlang läuft, kann es auch Hindernisse umrunden oder sogar einem Labyrinth folgen (falls die Pfade breit genug für den Rückweg sind). Die Vorliebe ist "deferred" - vergleichen Sie das Verhalten mit und ohne. Tierquälerei hier ausnahmsweise erlaubt.

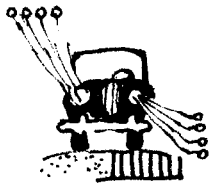
❑

Listing zu: Schneckenrennen in DisCo (Klaus-Peter Schleisiek)

```
ONLY FORTH ALSO DEFINITIONS
\ Ergänzung zu MiP's (MiP-Michael Prümm, Aachen) Turtle-Grafik
\ für die DisCo-Hardware (das Lampenfeld aus VD 4/92, 1/93)
\ aber ohne die DisCo-Software (spezielles Steuervokabular)

: -SCHRITT ( --)                        \ ein Schritt zurück
        HAKEN HAKEN                       \ umdrehen
        SCHRITT
        HAKEN HAKEN                       \ wieder umdrehen
;
\ sensiblere Schnecke im DisCo
\ unter Verwendung der speziellen DisCo-Worte

DISCO ALSO DEFINITIONS
: INNEN? ( col row -- f)                \ der Rand-Fühler der Schnecke
        #ROWS U SWAP #COLS U< AND
  ;
: ANSTOSS? ( col row -- f)                \ der Besetzt-Fühler der Schnecke
        2DUP INNEN?
  IF    .WEISE GET                        \ GET fragt den Zustand ab
  ELSE  2DROP TRUE
  THEN
  ;
: AUSWEICHEN ( --)                        \ im rechten Winkel
        -SCHRITT HAKEN                    \ RECHTER-WINKEL kann 90+ oder 90- sein
;
: AM-RAND ( --)                           \ eine mögliche Vorliebe
        RICHTUNG 2@                       \ gegenüber der Ausweich-Richtung..
        RECHTER-WINKEL RECHTER-WINKEL RECHTER-WINKEL
        2DUP STIFT 2@ VECTOR+             \ ..Fühler ausstrecken
        ANSTOSS?
  IF    2DROP                             \ hat Fühlung -> Richtung bleibt
  ELSE  RICHTUNG 2!                       \ sonst -> drehe in Fühlungs-Richtung
  THEN
  ;
VARIABLE FRUST                            \ Gemüts-Zustand des Schneck
DEFER VORLIEBE                            \ der Navigations-Instinkt

' AM-RAND IS VORLIEBE                     \ oder NOOP
: SCHNECK ( --)                           \ vorher Stift und Richtung geeignet setzen..
        FRUST OFF                         \ ..und das Feld aufräumen!
  BEGIN STIFT 2@ ANSTOSS?
        IF     AUSWEICHEN 1 FRUST +!
        ELSE   KLECKS       FRUST OFF VORLIEBE
        THEN   SCHRITT
        FRUST @ 1 >                       \ wenn der Schneck nicht im Frust stoppte,..
  UNTIL                                   \ ..dann tät's der Zuschauer.
  ;
```

❑

eForth CAN's



von Wolfgang Schemmert
Strahlenberger Str. 123, 63067 Offenbach Tel 069/800 12 08, Fax 069/64 82 59 57

Es wird ein Satz von Forth-Worten vorgestellt, mit dem a) ein einzelner Forth-Kern einen CAN-Bus bedienen kann b) ein verteiltes, CAN-vernetztes System von Forth-Kernen von einem beliebigen Knoten aus programmiert werden kann c) ein Forth-Kern als Gateway zwischen der seriellen Schnittstelle eines PC und einem CAN-Bus eingesetzt werden kann.

Gegenüber anderen Industriebussen hat der CAN-Bus den Vorteil, daß es preiswerte Controller zum Codieren, Decodieren und Verifizieren der Busnachrichten gibt, die die Arbeit der Busprogrammierung letztlich auf das Beschreiben der Senderegister, Lesen der Empfangsregister und Behandeln von Interrupt-Flags etc. reduzieren. Im Gegensatz zu anderen Industriebus-Controllern bekommt man aber damit weder Betriebssystem- noch Prozessoranbindung mitgeliefert, kann hierüber also frei verfügen. Wichtig zum Verständnis des CAN-Bus ist das Adressierungsverfahren: Abweichend von anderen Kommunikationssystemen wird eine Nachricht nicht mit Adressat und Absender gekennzeichnet, sondern mit einem 11-Bit "Identifier".

Jeder Identifier kennzeichnet ein Datenobjekt (oder etwas profaner ausgerdückt: eine Nachricht) mit 0 bis 8 Byte Datenumfang - irgendwo im CAN-Netzwerk. Die Bedeutung der Datenobjekte (einschließlich der Datenrichtung, ob der Identifier den Absender oder Adressaten bezeichnet) ist nicht vom CAN-System her vorgegeben, sondern wird durch die jeweilige Anwendungsprogrammierung bestimmt. Jeder CAN-Sender kann grundsätzlich jedes Datenpaket mit jedem gewünschten Identifier markieren. Im CAN-Empfänger ist (bei dem hier besprochenen Baustein; es gibt auch andere Selektionsverfahren) ein "Acceptance Filter" eingebaut, mit dem in gewissen Grenzen festgelegt werden kann, welche Nachrichten mit welchem Identifier vom Bus abgefangen und welche ignoriert werden sollen.

Stichworte

CAN-Bus
Fernprogrammierung in verteilten Systemen
eForth
H8/325 Mikrocontroller

In der neuesten Version 7.0 des UCASM Crossassemblers ist eine Mnemoniktabelle für den Hitachi H8/300 Mikrocontroller enthalten. Damit habe ich das eForth Modell, ähnlich wie ich es in einem früheren Artikel in der "4.Dimension" [1] für den 68HC11 Controller beschrieben habe, auf den H8/325 portiert. Dieser Microcontroller hat nur 64k Adreßraum, ist aber in seiner Rechenleistung etwa mit einem 68008 vergleichbar.

Durch die integrierten Peripheriefunktionen benötigt er weniger Chips zum Aufbau eines praktisch einsetzbaren Einplatinen-Systems und verbraucht vor allem nur einen Bruchteil des Stroms: ca. 35 mA für ein Minimalsystem mit Controller, Adreßdecoder, 32k RAM, 32k EPROM, RS-232 Schnittstelle. Durch die Summe dieser Eigenschaften läßt sich der H8/300 im Spektrum der 8-Bitter mit "easy und schnell" einordnen. Aus diesem Grunde auch wurde er als Controller für ein CAN-Bus-orientiertes Steuersystem ausgewählt-kombiniert mit dem wohl derzeit verbreitetesten CAN-Controller PCA 82C200. Damit war aber auch bereits das erste Problem geschaffen: Anders als die meisten Mikrocontroller hat der H8/300 keinen gemultiplexten Adreßbus während der 82C200 grundsätzlich gemultiplexte Daten und Adressen verlangt. Deshalb wurde die in Bild 1 skizzierte Hardware eingefügt, in Bild 2 ist das Listing der elementaren Routinen zur Bedienung des PCA 82C200 (Die Bussteuerung des H8/300 entspricht dem Intel-Konzept, während die Maschinensprache einem stark abgespeckten 68000-Befehlssatz ähnelt). Zum Listing ist anzumerken, daß das 16 Bit-Register R6 (R6L und R6H nach Hitachi-Mnemonik das untere bzw. obere Byte) der TOS der eForth- Implementierung ist, während R3 als temporäres Hilfsregister verwendet wird (hier zur Vorgabe

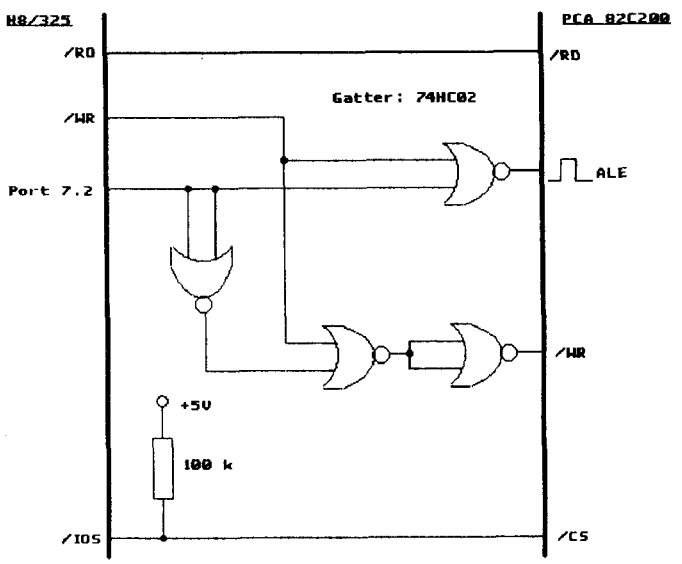


Abb. 1

der relativen Adresse im CAN-Controller). Wegen der UCASM-Notierung des eForth Codes siehe [1].

Nachdem die eher hardwaremäßigen Grundlagen angeschnitten sind, zum "informatorischen" Aspekt: Die Bedienung eines CAN-Bus von einem Forth-Kern aus (beliebige Nachrichten senden und abfangen) ist der

grundlegende Teil des hier vorgestellten Wort-Pakets: Zunächst gibt es mehrere User-Variable, die Parameter zur Konfiguration des CAN-Controllers speichern:

TXID

hält den im nächsten Datenpaket einzusetzenden Identifier. Für die Handhabung in Forth hat sich folgende Anordnung bewährt: LowerByte= Bits 3-10 des Identifiers, UpperByte=Bits 0-2 des Identifiers. Für die Details der nächsten 3 User-Variablen muß aus Platzgründen auf das Datenblatt des CAN-Controllers verwiesen werden:

CANACCEPT

(Einstellung des Acceptance Filters: oberes Byte= "Acceptance Code", unteres Byte ="Acceptance Mask")

CANSPEED

(oberes Byte = Baudrate, unteres Byte = Signalform)

CANOUT

(Konfiguration des Bustreibers, abhängig von der Bus-Hardware)

TXLEN

gibt die Länge des nächsten zu sendenden Datenpakets an (Längen von 0 bis 8 zulässig). Eine Sonderstellung nimmt der Wert $10 ein. Dann wird ein "Remote Command" gesendet, das den Inhaber des im Identifier beschriebenen Datenobjekts auffordert, den Inhalt dieses Datenobjekts auf dem Bus zu übertragen.

CANDAT

formal eine User-Variable, die aber inoffiziell den Kopf eines Arrays von 8 Bytes Länge markiert. Wird auch bei Empfang eines Datenpaketes benutzt (s.u.) Als nützlich hat sich eine programmierbare Verzögerung zwischen der Aussendung zweier CAN-Datenpakete erwiesen:

TXWAIT

hält als User-Variable die Verzögerungszeit in Milliskunden und wird automatisch von den unten beschriebenen "Sende-Worten" berücksichtigt. Diese Verzögerung kann notwendig sein, wenn als Empfänger der Nachrichten CAN-Knoten auf 8051-Prozessorbasis eingesetzt werden (z.B. 80C592, ein Mikrocontroller mit eingebautem CAN-Controller). Dann ist der Sender mit H8/300 schneller als der Emfänger-Interrupt mit 8051-Prozessorkern. Da der Überlauf bereits im Interrupt-Handler stattfinden würde, wäre auch ein Handshake-Verfahren nicht in der Lage, im Extremfall Datenverluste zu verhindern. Eine zweite sinnvolle Anwendung der Verzögerung wäre die Priorisierung von Nachrichten in Überlagerung des im CAN-System eingebauten Verfahrens, das grundsätzlich Nachrichten mit niedrigerem Identifier bevorzugt. Durch das künstliche Einbauen von Pausen bekommen in sehr stark ausgelasteten Bussen auch Nachrichten niedriger Priorität eine Chance. Die Zeitangabe in Millisekunden hat sich in der Praxis als recht grob erwiesen, 1 ms ist schon etwa das obere anzunehmende Maximum. Günstiger wären Zeiteinheiten von 50 oder 100 Mikrosekunden.

Erste Maßnahme um den Bus in Betrieb zu nehmen ist die Initialisierung des CAN-Controllers mit dem Forth-Wort CANOPEN:

CANOPEN (—)

Da das Wort (wie der überwiegende Teil des Pakets) als Primary geschrieben ist, hat es wenig Wert, hier ein Listing zu geben. Die Funktion ist jedenfalls folgende: zuerst werden alle Interrupts gesperrt, dann muß der CAN-Controller in den RESET-Status gebracht werden, sonst können verschiedene Voreinstellungs-Register nicht beschrieben werden. Nun werden die Voreinstellungen aus den User-Variablen in die Register des Controllers übertragen. Vor allem wird der Empfänger-Interrupt aktiviert. Dann wird der RESET-Status verlassen und die Interrupts wieder erlaubt. Eine Teilmenge von CANOPEN wird mit dem Wort FIL-

Elementare Routinen zur Bedienung des PCA 82C200

```
                    .MACRO @ADDRESS
                    bclr #2,@P7DRB        ;reset into address mode
                    .ENDM
                    .MACRO @WRDATA
                    bset #2,@P7DRB        ;switch into data write mode
                    .ENDM
muxwrite: ;*************************************************
; parameters: reladr = R3L
;                   byte = R6L
; start condition is always assumed Port7.2 = 0
;*************************************************************
                    mov.b R3L,@CANBASE
                    @WRDATA
                    mov.b R6L,@CANBASE
                    @ADDRESS
                    rts
muxread: ;*************************************************
; parameters: reladr = R3L
; return: byte = R6L
; start condition is always assumed Port7.2 = 0
;*************************************************************
                    mov.b R3L,@CANBASE
                    mov.b @CANBASE,R6L
                    rts

; CAN! ( b reladr - )
;                   write one byte into the CAN controller

                    @CODE CANST,LCANST,LCDAT,4,'CAN!'
                    mov.b R6L,R3L
                    pop R6
                    jsr @muxwrite
                    pop R6
                    @NEXT
; CAN@ ( reladr - b )
;                   read one byte from the CAN controller

                    @CODE CANAT,LCANAT,LCANST,4,'CAN@'
                    mov.b R6L,R3L
                    jsr @muxread
                    mov.b #0,R6H
                    @NEXT
```

Abb. 2

TER ausgeführt: Es wird das "Acceptance Filter" des CAN-Controllers neu beschrieben, d.h. die Empfänger-Charakteristik einer geänderten Aufgabenstellung angepaßt. Die Voreinstellungen in CANACCEPT bleiben unverändert.

```
FILTER ( bm bc — )
```

Dabei ist das Byte "bc" der "Acceptance Code", mit dem die Bits 3-10 jedes auf dem Bus bertragenen Identifiers verglichen werden, um zu entscheiden ob er in's Empfangsregister übernommen werden soll oder nicht. Das Byte "bm" ist die "Acceptance Mask", die hierin gleich "1" gesetzten Bits werden beim Vergleich von "bc" ignoriert. Mit "bm" kann gewissermaßen die Trennschärfe des "Acceptance Filters" reguliert werden. (Unabhängig von dieser quasi hardwareseitigen Filterung kann natürlich eine softwareseitige Fein-Filterung auf Forth-Anwendungs-Ebene durchgeführt werden.) Die Voreinstellung des CAN-Controllers wurde absichtlich wenig "automatisiert"; für den Preis eines etwas höheren Programmieraufwandes ist die Kontrolle über die Buskonfiguration vollständig dem Forth Hochsprachen-Zugriff übergeben. Das Gegenstück zu CANOPEN ist CANCLOSE:

```
CANCLOSE ( — )
```

setzt den CAN-Controller in den RESET-Status und macht damit diesen Knoten passiv. Mit dem bis jetzt aufgebauten Instrumentarium ist es kein Problem eine CAN-Nachricht zu versenden:

```
TXPACK ( — )
```

sendet das CAN-Datenpaket der Länge TXLEN mit dem Identifier TXID und dem ab CANDAT gespeicherten, vorproduzierten Datenkörper. Die Betriebsparameter des CAN-Bus hierzu wurden bei CANOPEN aktiviert.

Die Bereitstellung der Sendedaten als User-Variable mag zwar von Forth-Denkansatz her nicht so schön sein, macht aber weniger Probleme, wenn die Daten vorher z.B. aufbereitet werden müssen.

Nicht ganz so einfach gestaltet sich der gezielte Empfang von Daten: Jenseits von Forth befindet sich ein zentrales Element, nämlich der Interrupt-Handler des CAN-Empfängers, der im Hintergrund laufend alle vom "Acceptance Filter" durchgelassenen Datenpakete in einen 256 Byte tiefen FIFO-Puffer einreiht. Ein besonderes Problem entsteht daraus, daß in einem vernetzten System der Empfänger nur begrenzt Einfluß auf die Reihenfolge der eintreffenden Datenpakete hat. Zwischen einem "Remote Command" und der Antwort darauf kann z.B. eine Alarm-Meldung von einem angeschlossenen Sensor kommen. Außerdem beinhaltet die Pakethaftigkeit der Daten an sich schon eine gewisse Informationsmenge. Daher scheint es den Aufwand wert zu sein, den gesamten Datenempfang paketweise rekonstruierbar im FIFO festzuhalten. Dazu wird vor jedem empfangenen Byte ein Markierungsbyte in den Puffer geschrieben: Das erste Identifier-Byte wird mit gesetztem Bit7 markiert. Die unteren Bits weisen zyklisch zählend jedem Datenpaket eine Eingangsnummer zu.

Das zweite Identifier-Byte wird (etwas willkürlich aber unverwechselbar) mit $7F markiert. Die Datenbytes werden mit $0x markiert, wobei x (= 0 bis 7) jeweils die Zahl der danach noch folgenden Datenbytes angibt.

Auf dieser Basis läßt sich das "Empfangswort"

```
?RXPACK ( — T|F )
```

entwickeln. Wenn ein noch nicht entnommenenes Datenpaket im Puf-

Take CAR with CAN

akg

CAN steht für Computerized Automotive Network — so möchte man zumindest glauben, denn CAN hängt eng mit dem CAR Bereich zusammen. Richtig ist aber, daß CAN das Akronym für *Controller Area Network* ist. Genormt nach ISO/DIS 11898 und ISO/DIS 11519-1, deckt es die beiden unteren Schichten des ISO/OSI Referenzmodells ab.

CAN vernetzt als FeldBus Sensoren/Aktoren und verteilte Intelligenz, sprich embedded Controller, SPSen oder PCs. Es ist ein serielles, multimaster- und echtzeitfähiges BUSsystem. Es ist so schnell, daß auch noch bei einem 12 Zylinder 4-Takter zwischen den Zündfunken bei 10.000 min^{-1} zumindest *EIN* Daten-'Telegramm' störungsfrei übertragen werden kann, d.h. die Nachrichtendauer noch unter 500µs liegen muß. (Die Bemerkung eines Forthlers zu CAN: die beste Sache, an der BOSCH je beteiligt war).

Die objektorientierte Addressierung (um mal ein Modewort zu strapazieren) stellt eine interessante Besonderheit dar. Ähnlich wie im I^2C-Bus (Inter-Integrated Circuit Bus) werden priorisierte Nachrichten versendet, also ein 'Identifier' einer bestimmten Nachrichtenart zugeordnet, die damit gleichzeitig eine bestimmte Vorrangstellung bekommt. Die Kollisionsbearbeitung ist verlustfrei. Um beim Automobil zu bleiben: der Unterdruck im Saugrohr bekommt einen Identifier, bei dem sich dann die Einspritzpumpe, der Zündverteiler, die Katalysator-Steuerung, und die Fuzzy-GetriebeSteuerung einhängen, und eventuell auch noch der grüne Punkt im 'ECO-Bordcomputer', der eine SpritVerbrauchsAbschätzung versucht.

Mercedes Benz nutzt CAN in der S-Klasse genauso wie US-Nutzfahrzeughersteller (siehe auch: VAN). CAN wird aber auch in Schiffsanlagen, Aufzugssteuerungen, ErlebnisPark-'Geisterbahn'-MultiMedia, Medizintechnischen Geräten und in Teilbereichen des Maschinenbaus eingesetzt (großer Konkurrent Profibus und Interbus-S).

Auch auf dem iX-Controller (stackorientierter universeller FeldbusProcessor der Hamburger ForthSchmiede delta-T) ist CAN implementiert, wie auf der InterKama 92oct gezeigt wurde.

Vielleicht noch ein paar wichtige typische Daten zu Abrundung: Ein Telegramm ist 128 bit lang, nutzbar (netto) sind 64 bit oder 8 BYTE. Der große Overhead bietet bei einer Hamming-Distanz von 6 eine sehr hohe ÜbertragungsSicherheit. Es ist also kein Ethernet Konkurrenz, um MB-lange WindowsProgramme hin- und herzuschaufeln. Bei einer Zweidraht-CAN-Verbindung ist 1 MBit/s (brutto) für 40 m Leitungslänge spezifiziert. Bei 1 km wird die Echtzeitfähigkeit bei 50 kBit/s etwas behäbiger, was aber für den gesicherten Nutzgehalt immer noch gut 3mal schneller als RS232 bei 9600 bit/s ist, im Gegensatz zu RS232 aber inklusive FehlerErkennung und -Korrektur (und 1 km ohne galvanische Trennung ist für das unsymmetrische RS232 eher unsicher).

So manche CAN-Interface-Hardware ist übrigens immun gegen einfachen Kurzschluß oder Unterbrechung. Das PKW-Chassis bzw. die Spannungsversorgung werden zum Ersatzleiter.

Typisch gibt es max. 2032 Identifier. Bis 30 (bzw. 200 bei mehr HardwareAufwand) CAN-Knoten sind möglich, jeder kann verschiedene Sensoren, Aktoren usw. bedienen. Das orientiert sich an CiA (CAN in Automation e.V.). Eine Variation des RS485 auf 9-pol Sub-D wird als physikalische Grundlage genutzt.

❑

fer gefunden wird, kommt TRUE auf den TOS. Zur späteren flexibleren Handhabung kopiert ?RXPACK das Datenpaket dann nicht auf den Stack, sondern in das User-Array CANDAT. Der Identifier dieses Datenpakets wird in der User-Variablen

RXID

übergeben; zur einfacheren Handhabung hat sich die Reihenfolge bewährt: RXID LowByte = erstes CAN-ID-Byte, RXID HighByte = 2. CAN-ID-Byte.

Normalerweise sind aus einem FIFO gelesene Daten aus der Sicht des lesenden Programms im FIFO gelöscht.

Zur praktischen Restaurierung bereits "ver"werteter Datenpakete werden 2 User-Variable eingeführt:

HEADMARK

(=zuletzt vergebene Eingangsnummer) und

RXMARK

(=Eingangsnummer des zuletzt mit ?RXPACK bzw. ?REMOTE aus dem FIFO geholten Datenpaketes). Mit

BACKTAIL (– T|F)

kann der Puffer um 1 Datenpaket "zurückgespult" werden. FALSE wird geantwortet, wenn nicht innerhalb 22 Bytes die gesuchte Identifier-Markierung gefunden wird. Bei Sucherfolg werden die rückverfolgten Daten wie bei ?RXPACK abgespeichert.

Die spezifischen Stärken von Forth kommen allerdings erst mit der Möglichkeit einer interaktiven Fernprogrammierung des CAN-Netzwerks voll zum Tragen. Nach der Erprobung verschiedener Varianten hat sich folgende Lösung als recht praktikabel erwiesen: Gedanklich wird der Datenaustausch des Forth-Kerns mit der Applikation von der Konsolenfunkion getrennt. Jedem Forth-Kern wird ein Konsolen-Empfänger-Identifier zugeteilt, der in der User- Variablen

CCRXID

gespeichert wird und der nur einmal im Netzwerk vergeben werden darf. Für Konsolen-Sendungen wird analog zu TXID eine eigene User-Variable definiert:

CCTXID

Beim momentanen Entwicklungsstand haben alle Konsol-Nachrichten eine Datenlänge von 1 Byte. Das ist insofern von Vorteil, als die Worte für Sendung und Empfang analog denen der seriellen Schnittstelle programmiert werden können und ein im Hintergrund laufendes XON/XOFF Handshake eingeführt werden kann. Es ist allerdings auch von Nachteil wegen der stark reduzierten Übertragungsrate (sie sinkt auf etwa 1/4 des theoretisch erreichbaren Werts). Der Empfangs-Interrupt-Handler sortiert alle eingehenden Konsol-Datenpakete von vornherein aus, schreibt sie nicht in oben beschriebenen Puffer, sondern in ein eigenes Konsol-FIFO. Hier werden- wie von der seriellen Schnittstelle her bekannt- nur die Datenbytes aufgereiht, die Identifier sind nach Zuordnung zur Konsole ja informatorisch wertlos geworden. Parallel zum eForth Wort STDIO, das die Vektoren von KEY und EMIT auf die serielle Schnittstelle richtet wurde das Wort

CANIO (–)

eingeführt, das KEY und EMIT auf den CAN-Bus leitet. Auf dieser

Feldbusse von A bis Z — akg

A		A-Bus
B		Bitbus
C	*	CAN
D		DIN-MeßBus (eichfähig)
E	*	'EnergyLine' (Netz 115/230V)
G	*	GPIB (HPIB, IEEE-468.1/.2+SCPI)
H		Hart-Protokoll
I	*	IC
K	*	MAKbus + MAKnet (Dr. Weiss, Schriesheim)
L		'Lichtleiter-(billig)-Bus'
M	*	Meter-Bus
N		P-Net
O	*	ONF - OpenNetworkForth
P		ProfiBus
R		RBUS (Rekers)
S		SERCOS
V		VAN
W	*	Wiesemann & Theis: CombiBus
X		Interbus-S (Phoenix // DRIVECOM)
Y		phyNET (PHYTEC)
Z		('zentral'-) Berghoff-SDIO

(*) == diese Busse empfiehlt der Autor besonders zu beachten. (siehe folgenden Text)

Jedem Bus seinen 'eigenen' Buchstaben zuzuordnen, konnte nicht ideal gelingen, ein Rest von Mnemotechnik ist jedoch gesichert.

Der Begriff Feldbus wurde hier bewußt weit gesteckt. Alle Busse, die ein paar Meter schaffen, egal ob linear, Stern, Ring oder Baum, sind oben aufgeführt, auch wenn sie nicht die zum Feldbus gehörenden OSI-Layer voll umfassen. Erwähnt werden muß vielleicht noch SCSI (insbesondere als 'RS-485'-symmetrisches SCSI-2), aber auch da fehlt mir eine Buchstaben-Idee.

Der HP-IL (InterfaceLoop) hat es bis heute nicht überlebt, dieser Aktentaschen-Token-Ring war insbesondere bei HP-41 und HP-78 Aktivisten verbreitet.

CAN
In der VD: ein CAN-GateWay in Forth von W. Schemmert.
dazu: Erläuterungen zum CANbus und CiA (!) von (akg). Insgesamt ein recht universeller Bus mit Echtzeitfähigkeiten und recht breiter IndustrieUnterstützung.

'EnergyLine'
DatenÜbertragung über die Netzleitung. Auch zum Schalten, Steuern, sowie Ablesen der Zähler. Ein Gebiet in dem verteilte (Forth-)Systeme sinnvoll genutzt werden können.

GPIB
die schnelle (parallele) Standard Schnittstelle für MeßtechnikSysteme. Forth als meßtechnische Software ASYST oder als interne Software der Meßgeräte.
kostengünstige und einfache Verbindung nicht nur zwischen ICs. Verwandt zu CAN und COMBIbus. Die Zeitschrift 'Elektor' hat sich diesem Bus mehrfach gewidmet.
In der VD: im TDS2020 ist dieser Bus implementiert per Forth.

MAKbus MAKnet
Ein (schnelles) Netz unter Forth im Anwendungsbereich SPS und industriellen Regel-Prozessen inklusive Fuzzy. Vor Jahren bereits richtungsweisend.

Meter-Bus
Speisung der Sensoren, Aktoren über den Bus selbst, für Haustechnik und verwandte Bereiche.

ONF
Auf den ForthTagen:
von Heinz Schnitter 1989 in Aachen vorgestellt (per ARCnet, zuerst unter GEM),
von Udo Schütz 1993 in Nürberg: Einbindung unter comFORTH für windows in MS-Windows 3.1

CombiBus
In der VD: Buchbesprechung dieses LowCost LowPowe Busses (akg)

Und nicht zu vergessen: der UNIVERSAL-Feldbus Processor 'I-X' (Stack-orientiert, bit-optimiert, in Forth programmierbar) von delta-t, Hamburg.

❑

Grundlage sind die Worte
`?CCRX ( – c T|F )` und
`CCTX! ( c – )`
als Analogie zu `?KEY` und `TX!` ohne weiteres verständlich. (Als Adressierungsparameter dienen implizit `CCRXID` und `CCTXID`.)

Soweit so gut. Wie aber kommt man an den entfernten Knoten heran um ihn vom bequemen Operator-Sessel aus zu bearbeiten7 Dazu wird neben Applikations-Datenpaketen und Konsol-Datenpaketen eine dritte Klasse von Busnachrichten eingeführt, als "Spezial-Konsolen-Nachrichten" bezeichnet. Das sind Datenpakete von 8 Bytes, mit `CCRXID` des angesprochenen Knotens als Identifier. Diese Pakete werden ebenfalls vom Interrupt-Handler vor der Pufferung herausgefischt und in einem Primitiv-Parser (in Assembler) ausgewertet. Derzeit sind dem Interrupt-Handler zwei derartige Datenpakete bekannt: Zunächst ein Konstrukt der Form "GCIO xyz", wobei "xyz" die ASCII- Darstellung einer 3-stelligen Hex-Zahl mit dem Wert von `CCRXID` des anrufenden Knotens ist. Wenn der angesprochene Interrupt-Handler dieses Datenpaket findet, ruft er "hintenrum" `CANIO` auf, d.h. legt unabhängig von der momentanen Aktivität des Knotens die Konsole auf den CAN-Bus und trägt den Wert von "xyz" in sein `CCTXID` ein. Dabei wurde zur Vereinfachung bislang auf die in [2] diskutierten Maßnahmen zur Sicherung der Exklusivität von Konsolaktivitäten verzichtet. Als Gegenstück hierzu wurde das eForth-Wort
`GETNODE ( w – )`
dem CAN-Vokabular hinzugefügt. Es sendet genau dieses Datenpaket mit xyz hergeleitet aus dem eigenen Wert von `CCRXID`. w ist der Konsol-Identifier des adressierten Knoten. Nach einigen Fernprogrammier-Versuchen erwies es sich als unumgänglich, das Konstrukt "COLD xyz" einzuführen, das interruptgesteuert den Kaltstart des angesprochenen Knotens durchführt. (Die Angabe "xyz" ist hier eigentlich überflüssig, wurde aber aus Symmetriegründen aufgenommen und könnte ggf. zur Prüfung einer remote-COLD Berechtigung herangezogen werden.) Da der Interrupt-Handler zum Überleben nur einen RAM-Bereich von wenigen Bytes benötigt, übersteht dieses Fern-COLD mit hoher Wahrscheinlichkeit auch mittelprächtige Systemabstürze, von festgelaufenen Programmschleifen ganz zu schweigen. (100% Absturzsicherheit wäre möglich, wenn `CCRXID` und der Vektor des CAN-Interrupts aus der freien Programmierbarkeit heraus ins EPROM verlegt würden.) Das eForth-Gegenstück hierzu ist
`COLDNODE ( w – )`
wobei xyz wieder aus dem eigenen CCRXID hergeleitet wird.

Als letztes Kettenglied fehlt nun noch eine Möglichkeit, die Konsolfunktion des entfernten Knotens mit dem Operator-PC zu verbinden, der ja beim eForth-Modell auch den Massenspeicher und Source-Editor enthält. Dazu muß zunüchst der PC an die serielle Schnittstelle eines beliebigen eForth-Knotens des CAN-Netzes angeschlossen werden. Dann wird interaktiv das eForth-Wort
`CCLINK ( w – )`
aufgerufen. Ähnlich wie ein Modem im Datenmodus verwandelt sich der am PC hängende Knoten nun in eine transparente bidirektionale Datendurchreiche zwischen serieller Schnittstelle und CAN-Bus. Er hört am Bus auf den Identifier `CCRXID` und verwandelt jeden Character der seriellen Schnittstelle in ein CAN-Datenpaket mit dem Identifier `CCTXID`. Alle anderen Konsolaktivitäten des Knotens sind damit blockiert (Hintergrund-Tasks laufen aber weiter). Durch Eingabe der Escape-Sequenz "+++" wird wie beim Modem die Verbindung aufgebrochen, der entfernte Knoten bekommt vorher seine I/O-Vektoren auf die serielle Schnittstelle zurück gestellt.

Gegenüber den gegenwärtig marktüblichen in C, Pascal oder Assembler geschriebenen CAN-Bibliotheken hat diese Lösung den großen Vorteil, daß keine CAN-Kopplerkarte im PC notwendig ist, folglich auch kein spezieller CAN-Treiber im PC. Alle für die serielle Schnittstelle entwickelten Tools reichen aus für den Zugriff auf das gesamte CAN- Netzwerk.

Obwohl eigentlich nicht Gegenstand dieses Textes, sei der Vollständigkeit halber noch auf ein anderes Vokabular hingewiesen, um das eForth H8/325 mittlerweile ergänzt wurde: Neben EPROM und RAM enthält die von mir benutzte H8/325-Platine ein 28C256 EEPROM, von dem - pageadressiert - der Mikrocontroller zur Schonung seines Adreßraums jeweils nur 256 Bytes sieht. Es besteht die Möglichkeit, durch Aufrufen des eForth-Wortes
`FREEZE ( – )`
den momentanen Systemzustand (Dictionary Erweiterung seit dem letzten `FREEZE`, User-Datenbereich und aktuelle Interrupt- Sprungtabelle) inkrementell in das EEPROM zu brennen und somit relativ sicher abschaltfest zu speichern. In den zugrunde liegenden eForth-Worten wird auf dem Stack die relative EEPROM-Adresse übergeben; die Pageadressierung wird automatisch verwaltet. Sofern die entsprechenden Funktionen aktiviert sind, werden die Datenblökke beim Kaltstart aus dem EEPROM in das RAM zurückkopiert. Außerdem läßt sich im EEPROM veränderlich aber abschaltstabil ein Vektor für eine Kaltstart-AUTO-Applikation unterbringen.

Diese dezentrale Speichermöglichkeit ist ergänzend zum CAN-Bus Wortpaket eine wesentliche Unterstützung der Fernprogrammierung. Darüber hinaus lassen sich mit dem bereits in [1] beschriebenen eForth-Terminalprogramm Sourcen via Bus kompilieren, komplilierte Dictionary-Erweiterungen oder auch beliebige Binärfiles sichern und zum Mikrocontroller zurück laden.

❑

[1] 4.Dimension, Heft 4/1992, S.13 ff
[2] H.Zöller,H.Löwe: Forth in der Automatisierung, Düsseldorf 1990

Wanted:

Wer hat Erfahrung und/oder Geräte zum HP-IL-BUS?

Nachricht bitte an 'akg' (siehe Impressum).

F68K konnte es, F68KANS besser

von Zbigniew Diaczyszyn
Pommernstr.20 91126 Schwabach email: Z.Diaczyszynsontap.franken.de

Entwicklungsbericht zum Update des 68000-er Forthsystems von Jörg Plewe

Das Konzept

Zur Erinnerung: F68K war ein subroutine-threaded Forthsystem mit optionaler Makroerweiterung, das im Prinzip auf allen Rechnern mit dem 68000-er Prozessor lauffähig ist. Es bestand aus zwei Teilen: einem in Assembler geschriebenen betriebssystemunabhängigen Kern und einem Ladeprogramm, das den Kontakt zum jeweiligen Betriebssystem herstellte. Solche Ladeprogramme gab es für Atari, Amiga und Sinclair.

An diesem Konzept ist nichts geändert worden. F68K sollte aber durch die Anpassung an den ANS-Standard professioneller werden, und so war F68KANS geboren. Diese Umstellung ist nicht trivial, und sie ist immer noch im Gange, aber das Dokument dpANS6 scheint doch wohl das letzte gewesen zu sein.

Dadurch, daß sich der Interpreter von F68KANS ANS-konform verhält, ist es im Gegensatz zu seinem Vorgänger möglich, Quellcode aus sequentiellen Dateien einzulesen, da ein ANS-konformer Interpreter mit allen möglichen Eingabequellen zurechtkommen soll. Das Ladeprogramm für den ATARI, auf das ich im folgenden Bezug nehmen will, erlaubt auch die serielle Schnittstelle als Ein- und Ausgabequelle.

Stichworte

F68KANS
680x0-Forthsystem
C-Schnittstelle
Fensterumgebung
Entwicklungsbericht

Der neue GEM-Lader

Nun muß aber doch noch der Atari-Lader eingehender erwähnt werden, da hier von Jörg Plewe viel Entwicklungsarbeit geleistet worden ist. Der einfache Lader in F68K sorgte mit 16KB für die Übergabe rudimentärer Ein- und Ausgabefunktionen des Atari-TOS an den Assemblerkern. Mittlerweile umfaßt der Lader 190KB und bietet denkbar viel Komfort für den Forth-Anwender: Da ist zum einen die C-Schnittstelle , die es erlaubt, das reichhaltige Angebot getesteter Routinen aus C-Bibliotheken in den Lader einzubinden (z.B. durch das "Linken" von Fließkommaroutinen) und an den Forth-Kern weiterzureichen. Von dieser Fähigkeit des Laders nun wird der Forth-Anwender aber nichts merken, wenn er vor seinem Monitor sitzt; von der Einbindung des Forthsystems in das GEM des Atari aber wird er sehr wohl überrascht sein. Der Atari-Lader sorgt nämlich dafür, daß F68KANS in einem GEM-Fenster läuft [für DOS-User: Graphic Environment Manager, wie WINDOWS, doch nur mit Pseudomultitasking]. Gibt man jetzt WORDS ein, geht nach dem Durchscrollen der Liste kein einziges Wort verloren, denn über die Scroll-Leiste kann man sich wieder an den Anfang der Wortliste klicken. Eine History-Funktion, die sich die letzten eingegebenen Zeilen merkt, ist überflüssig geworden, da es mög lich ist, sich mit der Maus an jede beliebige Fensterposition zu klicken, beliebige Textausschnitte zu markieren und zu interpretieren. Man kann den so markierten Text aber auch ins Clipboard schieben, um ihn anderen Applikationen zur Verfügung zu stellen, einer Textverar beitung etwa, oder man kann ihn über einen Puffer in eines der anderen drei F68KANS-Textfenster transportieren, die wie das Forthfenster selbst auch über eine SUCHEN-ERSETZEN-FUNKTION verfügen. Weil F68KANS durch seinen Lader völlig in das GEM eingebunden ist, bereitet es keine Probleme, Accessories aufzurufen (Hintergrundprogramme, die aus der Menüleiste von jedem beliebigen Programm aus aufrufbar sind). In meinem Fall habe ich z.B. so neben F68KANS gleichzeitig einen Taschenrechner, eine Datenbank, und ein Modemprogramm zur Verfügung, mit denen über das Clipboard Informationen ausgetauscht werden können. Das neue multitaskingfähige Tos des Atari gestattet zusätzlich den Informationsaustausch zwischen den einzelnen Prozessen mittels "pipes", und auch diese akzeptiert der F68KANS-Interpreter.

Resümee

Ich glaube, es ist deutlich geworden, daß F68KANS nicht nur ein Update eines Public-Domain-Programmes ist, sondern ein modernes Desktop-Forth, das nicht nur innere Werte hat (subroutine-threaded-Code, Makrooption, C-Schnittstelle), sondern auch von seinem Erschei nungsbild her sehr benutzerfreundlich (Fenster-Graphic-Interface) gestaltet ist, so daß auch der Forth-Einsteiger einen Zugang finden kann. Ähnliches hat nur das 32FORTH von Richard Aumiller auf dem Atari geleistet, (das übrigens genauso komfortabel auf dem GEM der Msdos-Rechner läuft, und ohne daß man einen 486/66 braucht...), das aber noch indirekt gefädelt und ganz auf das Block-Format fixiert war. Man konnte aber immerhin schon 8 Editorfenster öffnen, leider aber nur mit sehr verhaltener Geschwindigkeit. F68KANS ist da sowohl von der Ausführungsgeschwindigkeit als auch vom Scrolling-Tempo entschieden schneller. BigForth, entstanden aus dem volks-Forth, hat sehr viel innere Werte (schnellstes Forth für den Atari ST, Modul-Konzept, Makrooptimierung,

fortsetzung auf Seite

Das legendäre ZF

Eine Vorstellung

von Friederich Prinz
Hombergerstraße 335, 47443 Moers

ZF, ein Forthsystem von *Tom Zimmer*, wird seit einigen Jahren von der Moerser Forthgruppe gepflegt, in vielen Bereichen eingesetzt und auf Anfrage kostenlos an jeden Interessenten weitergegeben. Die Nachfrage ist wegen des vorbildlichen Begleitmaterials speziell für Anfänger recht groß. Es fehlte bisher eine Vorstellung in der VD. Hier ist sie!

Tom Zimmer's ZF ist ein auf das F83 aufbauendes Forth mit sequentiellem Editor, separierten Wortbodies- und headern und sich an den Bedürfnissen des PC's orientierenden Sourcen. Damit sind unter ZF erstellte Applikationen nicht mehr problemlos auf andere Systeme portierbar, wobei sich allerdings die Inkompatibilitäten auf die cirka 10% von F83 differierenden Interna beschränkt. Letztlich sind ZF's Definitionen eine 'Untermenge' des jüngeren F-PC, wodurch ZF's Applikationen kompatibel sowohl zu diesem System, als auch zu dem mit dem F-PC mitgelieferten TCOM sind. Zumindest bei der späteren Übersetzung von ZF-Quellen durch TCOM sind aber noch in dem Maße 'Nacharbeiten' fällig, wie diese auch bei 'puren' F-PC Quellen notwendig sind.

ZF, das von der Moerser Forthgruppe auf zwei 3,5" Disketten versendet wird (auf Anfrage selbstverständlich auch in anderen Formaten), ist angenehm klein. Der Forthkern, der in dem File SKERNEL.COM zu Verfügung steht, ist gerade 21 KByte klein. Das vollständige ZF-System, inclusive des sequentiellen Editors, belegt als ZF.EXE lediglich 70 KByte auf dem Massenspeicher. Zum eigentlichen System gehören alle Quelltexte der Forthumgebung, so daß der Nutzer sich 'sein' ZF in einer Metakompilierung beliebig an die eigenen Bedürfnisse anpassen kann. Hier fällt besonders angenehm auf, daß ZF.EXE mit allen Quellfiles noch immer auf einer 360 KByte Diskette des PC/XT Platz finden würde ! Diese Feststellung ist nicht eben unbedeutend, wenn man weiß, daß von der Verfügbarkeit der Quelltexte die korrekte Arbeit der Utility-Definition VIEW abhängt. Zusätzlich zu den Quellfiles des eigentlichen Systems stehen noch cirka 100 KByte an Beispielen und 'Erweiterungen' bereit, die bei Bedarf zum ZF hinzukompiliert werden können.

> Letzlich sind die ZF's Definitionen einer 'Untermenge' des jüngeren F-PC

Stichworte

ZF
F-PC
TCOM

ZF trennt Wortbodies- und header und lagert beide Definitionsteile in unterschiedlichen Segmenten. Zunächst wird beim Aufruf von ZF.EXE durch das System ein 64 KByte großes Segment für den Kern, den 'Extended Code', weitere Definitionen, für den Stack, den TIB und den Returnstack angelegt.In diesem Segment wächst der Returnstack vom Segmentende dem TIB entgegen. Die Startadresse des TIB beträgt als Offset auf das Segment FD24h, woraus folgt, daß für den TIB und für den Returnstack knappe 730 Byte zu Verfügung stehen. Da ZF mit 16-Bit Pointern arbeitet, bleiben somit für Rückkehradressen auf dem Returnstack maximal 365 'Nest-Einträge' übrig. Tatsächlich ist die maximale Anzahl von Nestings in ZF auf 350 begrenzt, so daß eine Kollision des Returnstacks mit dem oberhalb von TIB beginnenden Datenstack definitiv ausgeschlossen ist.

Der Datenstack wächst wieder 'absteigend' dem Start des freien Code-Bereiches entgegen, d.h. der Adresse HERE. Neben diesem ersten Segment legt ZF.EXE ein 32 KByte großes Segment für die Wortheader an. An der Startadresse dieses Segmentes wird jedoch zunächst die Hashtabelle für schnelle Dictionary-Suchläufe abgelegt und gepflegt, so daß für die eigentlichen Wortheader etwas weniger Platz übrig bleibt, was allerdings in der Praxis kaum von Bedeutung sein wird. Ein drittes, wieder 64 KByte großes Segment, legt ZF.EXE für den Editor an, bzw. für die zu bearbeitenden Quelltexte. In diesem Segment werden neben den eigentlichen Texten mehrere Puffer bereit gehalten (Bildschirmpuffer, Puffer zum 'Entlöschen' von Zeilen usw.), so daß auch hier nicht die vollen 64 KByte zu Verfügung stehen. Die maximale Zeichenanzahl für in sich geschlossene Quelltexte liegt bei cirka 60.000 . Aus der Addition der einzelnen Segmente ergibt sich, daß ZF insgesamt mit 160 KByte freiem Arbeitsspeicher auskommt! Wenn man berücksichtigt, daß von diesen 160 KByte bereits 64 KB für den Editor reserviert sind, wird deutlich, daß ZF eine der wenigen Programmierumgebungen ist, die mit den Resourcen des Programmierers nicht so umgehen, als würden die Entwickler der Umgebung die Kosten für Arbeits- und Massenspeicher selbst tragen.

> ZF.EXE findet mit allen Quellfiles noch immer auf einer 360 kByte Diskette des PC/XT Platz!

Das legendäre ZF
von Friederich Prinz

Beim ersten Aufruf von ZF zeigt das System in seiner Statuszeile im Kopf an, daß dem Programmierer noch cirka 30 KByte für Definitionen zur Verfügung stehen. Das mag auf den ersten Blick wenig erscheinen. Tatsächlich sind dies aber 30 KByte aus dem Code-Segment, die sich die vom Programmierer erstellten Applikationen lediglich mit dem Datenstack teilen müssen. Für die überwiegende Mehrzahl aller 'forthigen' Programme reicht diese Menge mehr als aus. Entwickler von Applikationen mit größerem Speicherhunger werden in der Regel ohnehin von DOS weiteren Speicher allokieren. Ein Aufruf von `WORDS *.*` läßt 1205 Worte über den Bildschirm huschen, von denen rund 90% dem erfahrenen Forther zumindest aus F83 bekannt sind. Die restlichen 10% der Definitionen sind Worte, die sich auf DOS-Interna beziehen (z.B. Allokierung von Speicher), oder Worte, die das System nutzt und die zumindest nicht problemlos durch den Programmierer verwendet werden können. Wie auch immer, für den Quereinsteiger oder Anfänger ist diese Wortmenge zu groß, so daß die Moerser Forthgruppe dem ZF neben dem forthigen `WORDS` noch ein `WORTE` verpaßt hat. Der Aufruf von `WORTE`, ohne jeglichen Parameter, listet 25 'Kategorien' auf, in denen die gebräuchlichsten Worte zur Arithmetik, zum Compiler oder zum Assembler definiert sind. Der Aufruf von `WORTE`, gefolgt von dem Namen einer dieser Kategorien, listet dann die entsprechenden Definitionen auf. Hierbei muß gesagt werden, daß die Inhalte der Kategorien relativ willkürlich sind und den Erfahrungen der Moerser Gruppe entsprechen, die, entsprechend den RISC Überlegungen, einfach die Worte zusammengefasst hat, mit denen sie in der Praxis arbeitet.

Der Editor des ZF, der mit `SED` aufgerufen wird, meldet sich zunächst mit einem Fenster zur Dateiauswahl. Hier kann der Nutzer bequem mit den Pfeiltasten Dateien anwählen, löschen oder in der Editor rufen, Pfade und Laufwerke angeben und sich 'kreuz und quer' über seinen Massenspeicher hinweg bewegen. `SED` ist ein sequentieller Editor mit 'fließenden', zusammenhängenden Texten. Das Blockkonzept ist in ZF passé. Zwei Fenster mit Hilfstexten, die durch die F1-Taste erreicht werden, bieten dem Programmierer alles, was er zu seiner Arbeit benötigt, selbstverständlich in deutscher Sprache. Die dem alten WordStar Editor entlehnten Kommandos des reinen ASCII Editors sind vielen FORTHern geläufig und in der Anzahl auf das Wesentliche beschränkt. Selbstverständlich fehlen hier solche Optionen wie das Auslagern von Textteilen und das blockweise Kopieren von Textteilen nicht. Und selbstverständlich ist es möglich, den Quelltext zu verlassen, ohne eventuelle Änderungen abzuspeichern.

> **ZF ist eine der wenigen Programmierumgebungen, die mit den Resourcen des Programmieres nicht so umgehen, als würden die Entwickler der Umgebung die Kosten für Arbeits- und Massenspeicher selbst tragen.**

Mit der Tastenkombination ALT-P erreicht der ZF-Nutzer ein einfaches Druckermenü, welches es ihm ermöglicht seitenweise nur Auszüge aus seinem aktuellen Text zu drucken, Randeinstellungen vorzunehmen und die Anzahl der Kopien zu bestimmen. Alles in allem bietet `SED` exakt den Leistungsumfang, den man vom Editor einer Programmierumgebung erwarten darf - nicht mehr und nicht weniger.

Nachdem der Forther einen Quelltext bearbeitet hat, übergibt das Wort `DOIT` diesen Text an den Compiler, der eine Referenz auf eventuelle Fehlerstellen an den äußeren Interpreter übergibt, so daß nach dem Auftreten eines solchen Fehlers sofort an die entsprechende Stelle im Quelltext 'gesprungen' werden kann.

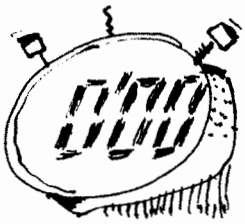
ZF arbeitet intern mit indirekt gefädeltem Code und 16-Bit Pointern. Die Fädelung in den Parameterlisten von Colon-Definitionen erfolgt über die Einträge der CFAs der enthaltenen Definitionen, so daß ein relativ flinkes Umschalten zur jeweils nächsten Definition durch `NEXT`, bzw. `NEST/UNNEST` gewährleistet ist. Die tatsächliche Geschwindigkeit mit der solche Fädelungen letztlich abgearbeitet werden, ist vor allem von den intern arbeitenden Definitionen `NEXT`, `NEST` und `UNNEST` abhängig. Hier sind zum Teil kommerzielle Systeme, allen voran LMI's PC/FORTH, dem ZF leicht überlegen. Allerdings sollte auch deutlich sein, daß neben den internen Definitionen heute solche Größen wie der CPU-Typ und dessen interne Organisation (z.B. Cache, Prefetch-Queue) eine ebenfalls wesentliche Rolle bei der Gesamtperformance eines Systems spielen. Benchmarks, gleich welcher Art, können deshalb kaum relevante Aussagen über die Performance einer Entwicklungsumgebung machen.

Summa sumarum ist ZF ein Forth-System, das allen konventionellen Ansprüchen gerecht wird. Es ist gerade 'umfangreich' genug, um nicht ständig zur Neuerfindung der verschiedenen Räder zu zwingen, aber noch nicht 'groß' genug, um damit Quereinsteiger und Anfänger zu erschlagen und abzuschrecken. Klein, aber leistungsfähig, mit genau dem Komfort, den sich der Profi wünscht und den der Anfänger benötigt, mit ausreichend guten Werkzeugen zur Fehlersuche wie `VIEW`, `SEE`, `DEBUG`, und `DUMP` - stellt ZF eine forthige Entwicklungsumgebung dar, die immer häufiger von der Moerser Forthgruppe angefordert wird. Nicht nur für den Anfänger sind neben dem ZF selbst auch die inzwischen mehr als 2 MB deutschen (!) Kurstexte und Aufsätze zu den unterschiedlichsten Themen interessant, die ebenfalls auf einer Diskette mitgeliefert und in Moers intensiv gepflegt werden.

❑

Si-Nuß

von Jörg Staben
Hagelkreuzstraße 23, 40721 Hilden, Tel. 0 21 03 - 5 56 09



Wie lange braucht man wohl, um ein gegebenes Problem (hier: Ausgabe einer Sinuskurve auf dem Bildschirm) im System seiner Wahl (hier: F-PC) zu lösen? Jörg Staben hat für uns Protokoll geführt.

Kaum hatte ich Gelegenheit, mich mit meinem Modem in den nationalen und internationalen Netzen herumzutreiben, schon stolperte ich über solch' harsche Aussagen wie: "Die F-PC-Grafik von Dr.Smiley läuft nicht." Statt einer solchen Pauschalaussage wünsche ich mir lieber in unserer Mailbox eine Ecke F-PC-Bugs, in der ich dann eine detaillierte Beschreibung des Fehlverhaltens finde. Mit einer solchen Beschreibung kann man dann die Fehlerkorrektur sinnvoll in Angriff nehmen.

Das F-PC und seine (VGA-)Grafik haben unter den Entwicklungssystemen eh' eine Sonderstellung, da hier mindestens zwei Leute (*Tom Zimmer & Mark Smiley*) unabhängig voneinander arbeiten; dies zeigen gerade die neuesten Grafikarbeiten von Tom Zimmer wie z.B. das Malprogramm DRAW.SEQ.

Das Konzept 0'00

Eine Sinus-Kurve auf den VGA-Schirm zu bringen, braucht es nur wenig - denkt man. Man nimmt Funktionen,

- die auf Grafik umschalten (das alte Leiden der PC's unter DOS),
- die einen Punkt zeichnen,
- die die Werte der Sinusfunktion liefern.

Setzt man den Grafik-Treiber von Dr.Smiley ein, sind die ersten beiden Punkte mit `vga640` und `words dot` schnell gelöst; schwieriger wird es bei der Sinusfunktion. Das F-PC kennt von Haus aus kein `sin` oder `cos`. Woher nehmen? Wie immer ist der "grüne" Zech, "Forth83", die erste Wahl; nur werden dem Leser hier ab S.147 sofort die "äqui-distanten Stützstellen" um die Ohren geschlagen und eine Sequenz wie `here ' : @ , ] Does>` läßt schon auf den ersten Blick Schwierigkeiten fürchten. Zudem weckt die Aussicht, eine Tabelle mit 100 Werten von Hand anlegen zu dürfen, wenig Vorfreude.

Stichworte

F-PC
VGA-Grafik
Probleme beim Aufbau einer Modul-Bibliothek
Sinus/Cosinus-Funktion

Die Suche 0'10

An dieser Stelle wäre es für den hoffnungsfrohen Forthler schön, mit seinem Modem in die Forth-Mailbox zu gehen und dort aus F-PC\LIB\MATH schnell eine Datei SINCOS.SEQ downloaden zu können. Aber da die Mailbox noch nicht soweit ist(?), war in dieser Situation die Suche nach Forth-Quellen in der guten alten Diskettenbox angesagt.

Nach kurzer Zeit - 45 Min. - hatte sich dank der diversen F-PC-Utilities wie `LOOK` und `LIST` einiges angesammelt. Leider hatte `LOOK` nicht ein Programm gefunden, sondern direkt vier Programme! Welches soll man denn nun nehmen, und warum gibt es so viele verschiedene Programme für den SINUS??

Dr.Ting 0'55

Egal - erst mal sehen, was die Lösung SINCOS.SEQ von Dr.C.H.Ting aus dem Jahr 1986 bringt: Die Ausga-

Sinus- und Cosinusfunktionen

```
\\ 16bit integer sin and cos; highlevel and assembler code; tested version

   J.Staben
   Hagelkreuzstr.23
   4010 HILDEN
   Germany

   $rev: $
   $date: $
{


Create SinusTable     \ sine 0 - 90  multiplied by 10,000   integral degrees
 0 , 175 , 349 , 523 , 698 , 872 , 1045 , 1219 , 1392 , 1564 , 1736 , 1908 ,
2079 , 2250 , 2419 , 2588 , 2756 , 2924 , 3090 , 3256 , 3420 , 3584 , 3746 ,
3907 , 4067 , 4226 , 4384 , 4540 , 4695 , 4848 , 5000 , 5150 , 5299 , 5446 ,
5592 , 5736 , 5878 , 6018 , 6157 , 6293 , 6428 , 6561 , 6691 , 6820 , 6947 ,
7071 , 7193 , 7314 , 7431 , 7547 , 7660 , 7771 , 7880 , 7986 , 8090 , 8192 ,
8290 , 8387 , 8480 , 8572 , 8660 , 8746 , 8829 , 8910 , 8988 , 9063 , 9135 ,
9205 , 9272 , 9336 , 9397 , 9455 , 9511 , 9563 , 9613 , 9659 , 9703 , 9744 ,
9781 , 9816 , 9848 , 9877 , 9903 , 9925 , 9945 , 9962 , 9976 , 9986 , 9994 ,
9998 , 10000 ,


\- cells        ' 2* Alias cells
false \ true
#IF
}
********************  jrg  03.09.92 09:24  ****************************
                SinusTabelle mit
                High-Level-Zugriff
*****************************************************************************
{
: isin          ( n1-degrees ---, n2-sine*10K )
                360 mod
                dup
```

befunktion SHOW-SINE hinterläßt - entgegen dem Stackkommentar - einen Wert auf dem Stack. Die ersten Schritte, 45 SIN . und 45 COS . , bringen nicht das gewünschte Ergebnis von 7071. Damit ist - nach weiteren 30 Min. - die erste Lösung aus dem Rennen.

B.Gunton 125

Von der nächsten Lösung, die sich durch ganz viel Assemblercode auszeichnet, ist mir der Autor nicht sicher bekannt; bestimmt war es *Mr. B.Gunton* oder einer der anderen, die schon sehr früh mit Grafik in Forth beschäftigt haben. Leider läßt sich auch die aufregend aussehende Datei SINARC.SEQ nicht kompilieren; bestimmt weil sich der Gebrauch der "local labels" in der Zwischenzeit geändert hat. (Hat jemand in der FG Ahnung, wie das mit den Local Labels im F-PC so ist?)

Allen, die jetzt über die Inkonsistenz des F-PC jammern, sei gesagt, daß es den TurboPASCAL-Jüngern auch nicht besser geht. Eine TP3.0-Datei läuft nicht unter 4.0, eine TP4.0-Datei nicht unter 5.0 und was das BorlandPASCAL (=TP7.0) anrichten mag, mögen die TurboPASCAL-Jünger selbst ausprobieren. Und am TurboPASCAL arbeitet nur Borland...

I.Mathyl 150

In der Grafikbibliothek von *I.Mathyl*, Hamburg, fand ich dann eine Lösung SIN&COS.SEQ, die viel Ähnlichkeit mit der von *R.Zech* hatte und sich sofort kompilieren ließ.

Nicht nur das, ich konnte auch noch zwischen Forth und Assemblercode wählen. Das wünscht man sich eigentlich für jedes Programm: Erst die Lösung in Hochsprache, dann eventuell die Ergänzung in Maschinensprache.

In geringfügig anderer Form (Werte-Tabelle und Zugriff in Assembler) ist diese Sinusfunktion auch im Lieferumfang des F-PC enthalten; Sie finden sie im Verzeichnis TCOM in der Datei TSHAPES.SEQ.

Nun macht eine Basisfunktion in

Fortsetzung: Sinus- und Cosinusfunktionen

```
                180 >   IF      180 -  -1 ( flag )
                        ELSE           1 ( flag )
                        THEN           >r      \ temporary store
                dup
                90 >    IF      180 swap -
                        THEN
                cells sinustable + @            \ access to table position
                r> ?negate
                ;


#ELSE
)
********************  jrg  03.09.92 09:30  ****************************
                SinusTabelle
                mit LowLevel-Zugriff
*****************************************************************************
(
Code isin       ( n1-degrees ---, n2-sine*10K )
                pop bx
                xor cx, cx
                mov dx, # $168
                BEGIN   cmp bx, cx
             <  WHILE   add bx, dx
                REPEAT
                BEGIN   cmp bx, dx
            >=  WHILE   sub bx, dx
                REPEAT
                mov dx, # $B4
                cmp dx, bx
              < IF      inc cx
                        sub bx, dx
                THEN
                mov ax, # $5A
                cmp ax, bx
              < IF      sub dx, bx
                        mov bx, dx
                THEN
                shl bx, # 1
                add bx, # sinustable
                mov ax, 0 [bx]
                cmp cx, # 0
           0<>  IF      neg ax
                THEN
                push ax
next end-code

#THEN

: icos  negate 90 + isin ;

' isin Alias sin
' icos Alias cos
```

❑

vga-sinus-kurve

```
\\ anew dot.sinus
```

Zeichnet eine sich bewegende SINUS-Kurve auf den Grafik-Bildschirm. Dadurch, dass die Existenz des Vokabulars GRAFIK geprüft wird, lässt sich das Programm sowohl für den Grafik-Treiber von Dr. M.Smiley als auch für den Grafik-Treiber von Dipl.Ing. I.Mathyl laden.
Verbesserungsmöglichkeiten:
Mit den Cursortasten die Amplitude u. Frequenz, mit der SPACE-Taste die Farben ändern.

```
(
\- task         : task ;
forget task

fpath+ e:\4th\fpc\wrk\sin&cos
fload sin&cos.seq
```

➪

einem Programm nicht viel Sinn, wenn man nicht sicherstellt, daß sie auch richtig funktioniert.

Test

Diese Sinustabelle habe ich mit einigen - hoffentlich typischen - Werten und einem CASIO-Taschenrechner überprüft. Gibt es denn keine anderen Testmöglichkeiten als einen Taschenrechner zu Hilfe zu nehmen?

Aber dennoch: Nun, über zwei Stunden später, steht eine minimal getestete, funktionierende Sinusfunktion zur Verfügung. Damit dürften auf dem Weg zur Sinuskurve in VGA-Grafik-Qualität keine weiteren Probleme auftauchen. Aber eines ist aber hier schon deutlich geworden: In diesem Jahrzehnt, in den 90er Jahren, ist nicht mehr der große Wurf, der geniale Rundumschlag angesagt, sondern mühsames Aufräumen. Die fehlerfreie, solide, einfache Lösung zählt; denn nur darauf lassen sich weitere Arbeiten aufbauen.

Jetzt haben wir eine so schöne Sinus/Cosinus-Funktion; wie bewahren wir sie am besten auf? In einer Datenbank? Oder mit einer Diskettenverwaltung oder wie?

Aber diese Frage ist vielleicht schon mit dem Aufkommen der Indexer geklärt worden; dies sind Spezialprogramme, die Dateien nicht nur unter ihrem Dateinamen irgendwo ablegen, sondern den Dateien noch eine Kurzbeschreibung, den Autorennamen mitgeben. Vor allem legt ein Indexer einen Verweis für jedes Wort im Text an, das so als Stichwort dienen kann.

Der Indexer von WordPerfect 6.0 hört auf den Namen QuickFinder und indiziert ein 2KB großes Dokument wie diesen Text in Sekundenbruchteilen. Nach der Indizierung kann nach jedem Wort des Textes, einem Wortschema oder einer Kombination mit UND, ODER, NICHT gesucht werden. Sie sehen, auch Leute, die Textbearbeitung machen, haben das Problem, ihre Sachen wiederzufinden.

Das Programm zum schwebenden Sinus ist ja schon ganz nett; schöner wäre eine Benutzerschnittstelle, damit dieser interaktiv sowohl Frequenz als

Fortsetzung des Listings vga-sinus-kurve

```
defined grafik nip

#IF     grafik  also
        : init  ( -- )
                hiresmode
                auto-init
                graf-mode
                ;
        red color !
        : plot.dot      ( x y -- )
                rot drop
                plot
                ;
        : end   ( -- )
                text-mode
                ;
#ELSE   : init  ( -- )
                vga640
                ;
        : plot.dot      ( x y -- )
                cdot
                ;
        : end   ( -- )
                text
                ;
#THEN

: scale         ( n -- scaled[n] )  \ Anpassung an 480/2 Pixels
                dup 0< >r
                abs
                240 um*         \ Kandidat für Amplitude
                10000 um/mod nip
                r> IF negate THEN
                480 swap -
                240 -
                ;

: .sinus        ( color x-coord angle -- )
                isin scale
                plot.dot
                ;

: plot.sinus ( -- )
    init
    cls
        time-reset
        640 0
        DO
                ltred i i .sinus
        LOOP
    .elapsed
    ." || Stack: " .s
    key drop
    end
    ;

: Waving ( -- )
  init
  0
  BEGIN
        640 0 DO
                black i 2 pick
                  i ( Frequenz ) + 1- 0max .sinus
                ltred i 2 pick
                  i (     "    ) + .sinus
        LOOP
  1+  360 mod
  key? UNTIL
  drop
  end
  ;

previous forth
: wave waving bye ;
\s
```

Buchbesprechung

Arndt Klingelnberg

Der Tischer

Michael Tischer,
PC intern 3.0, Systemprogrammierung,
1. Auflage, 1992,
Data Becker, Düsseldorf,
720 kB, Hardcover 1404 Seiten, > 2 kg,
ISBN 3-89011-591-8, DM99.-

Als deutschsprachiges Buch füllte PC intern 1.0 damals ein Vakuum und wurde kurz prägnant 'Der Tischer'. Mittlerweile sind wichtige Dinge ergänzt worden, die auch für oder gerade Anfängern die 3.0 empfehlenswert erscheinen lassen. Wir schreiben DOS 5 bis 6, vieles sollte nicht mehr so gemacht werden, wie im DOS 3.x Mittelalter. Große Partitionen, VGA, Mouse, EMS, XMS sind Standard, und die allerwichtigsten 'UNdocumented' Funktionen sollte ein jeder bei der Hand haben. Das Buch hat sich mittlerweile auch davon gelöst, daß es sich zu sehr an der UrBibel von Ray Duncan inklusive Fehlern orientierte. Mit 250 Seiten mehr ist es fast gleich dick und schwer, es ist kleiner gesetzt aber eher besser lesbar.

Die Assembler Beispiele sind für FORTH, soweit das System normale (MASM) PREfix-Notation verdauen kann, gut nutzbar. Die Beispiele ab Diskette können nach ein wenig Editieren schnell zum laufen gebracht werden. Außer C , Pascal und Assembler gibt es hier auch BASIC Beispiele, das mag für manche wichtig sein.

Für ein solch gewichtiges Nachschlage-Werk ist die Qualität des Inhalts- und Schlagwortverzeichnisses von sehr hoher Bedeutung. Noch so vielseitige Information ist unnütz, ja konterkarriert sich, wenn man nicht (schnell genug) Zugang findet.

Trotz 10 Seiten Index und 7 Seiten Inhaltsverzeichnis handelt sich das Buch deutliche MinusPunkte ein. Nun hat ein deutsches Buch es sehr schwer, mit den vielartigen und zudem oft abenteuerlichen Übersetzung der ursprünglich Englischen Begriffe klarzukommen. Entweder man führt die üblichen Englischen Begriffe (auch) auf, oder versucht die Sorgfalt früherer DDR Fachbücher in diesem Punkt zu erreichen. Zudem muß ein Eintrag auch auf (alle) wichtige Stellen verweisen, und wenn eine sehr wichtige, wenn auch kleine Bemerkung irgendwo fällt, muß auch dorthin verwiesen werden: nicht verschollen in einem Berg von Papier und (im Augenblick gerade) unwichtigen Informationen. Dieses Manko kostet Zeit bzw. läßt dann doch zu anderen Büchern greifen.

All die IndexErgänzungen, die ich mir zu 2.0 geschaffen hatte, sind nun wieder hinfällig und die Suche beginnt von Neuem. Trotzdem ein Tipp: alles, was beim Überfliegen wichtig erscheint, selbst im Index ergänzen und zwar dort, wo man suchen würde.

Soweit ein deutsches Buch gewünscht ist, ein empfehlenswertes Buch zum starten, da es alle relevanten Bereiche ersteinmal abdeckt. *(siehe auch Buchbesprechung Seite 33)*

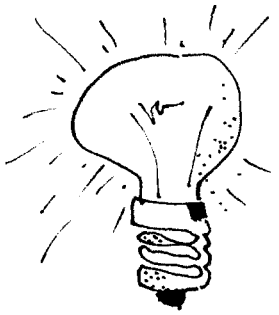
Wanted for C64

Wer hat innerhalb der letzten Jahre für diesen Käfer unter den Computern noch etwas in Forth entwickelt? Wer hat vor allem unter Geos-64 schon mal Schritte in Forth getan? Für alle Hilfen diesbezüglich wäre ich sehr dankbar. Der Hintergrund ist folgender: Im Geos-User-Club habe ich eine Reihe von Leuten getroffen, die an Informationen über die Sprache interessiert sind. Eventuell kann ich in diesem Geos-Club eine kleine Forth-Sektion aufbauen. Außerdem entwickle ich zur Zeit ein Standard-Forth (nach FIG/F83), das unter Geos einige Features des Fort(h)schrittes einfangen soll. Jeder, der noch irgendwelche alten Quelltexte, z.B. ein altes FIG-Assembler-Listing rumfliegen hat, könnte mir eine Kopie der entsprechenden Disk senden...

MfG... Klaus Heinisch
Schachtstr. 34, 30655 Stadthagen,
Tel. 05721-82633 (Btx ..0001)

Inserentenverzeichnis

UCS ?



von Dipl. Phys. Jörg Plewe
Haarzopfer Str. 32, 45472 Mülheim an der Ruhr, 0208/497068

Dreibuchstabige Kürzel sind 'in' und wirken immer sehr professionell und geheimnisvoll. Deshalb will ich zunächst auch gar nicht verraten, um was es geht und hoffe, daß mir diese Taktik wenigstens für den ersten Abschnitt die Leserschaft sichert.

Auch will ich dieses Mal nicht von eigenen Ideen und Taten berichten, sondern mich etwas auf den Lorbeeren anderer tummeln. Kurzgesagt geht es um einen Artikel, der in der 'Forth Dimensions', dem Fachmagazin des großen Bruders der Forth-Gesellschaft, der FIG, im September '91 erschienen ist. Da nur wenige Mitglieder der FG auch gleichzeitig Mitglieder der FIG sind und ich diesen Artikel für besonders interessant halte, will ich versuchen, die wesentlichen Gedanken hier wiederzugeben.

Dabei will ich zunächst die Intention des Autors *Kevin Haddock* aus Chico, Kalifornien, darstellen. Danach werde ich die Funktion mit eigenen Worten beschreiben und dabei päpstlicher vorgehen als Kevin Haddock selbst. Zuletzt werde ich ein paar Bemerkungen zur UCS-Implementation auf verfügbaren Forthsystemen machen.

Wer hat ein Problem mit Forth?

Kevin Haddock ist ein Forthprofi. Deshalb bemerkt er Dinge, die dem 0-8-15-Programmierer erst dann auffallen, wenn er mit der Nase darauf gestoßen wird.

Forth ist eine interaktive Sprache. Man kann jederzeit aus der Kommandozeile auf den gesamten Sprachumfang zugreifen. Also los:

```
base @ DECIMAL 16 =
   IF
     .( hex )
   ELSE
     .( ??? )
   THEN
```

Huch, was ist das?

```
IF compile only!
```

Na ja, fast der gesamte Sprachumfang. Kontrollstrukturen sind eben etwas besonderes. Dafür erzeugt Forth aber besonders effektiven Code.

```
3 array x
: funny-range ( -- flag )
0 x @ 15 0 x @ 17 and
1 x @ 27 1 x @ 39 and and
2 x @ 71 2 x @ 82 and and;
```

Zur Verdeutlichung der entsprechende C-Code (peinlich aber wahr):

```
int x[3];
...
if ((x[0]15 && x[0] ) &&
    (x[1]27 && x[1]') &&
    (x[2]71 && x[2]R))
...
```

Sei nun x[0]=0. Damit ist die Bedingung immer FALSE, da bei UNDierten Ausdrücken ein einziges FALSE (x[0]15) ausreicht, um den gesamten Ausdruck zu FALSE auszuwerten. Diese Tatsache aber wird vom Forthwort `FUNNY-RANGE` schlicht ignoriert. Im Gegensatz zum C-Pendant werden alle Vergleiche durchgeführt, obwohl das Ergebnis schon nach dem ersten feststeht. Nur soviel zum Thema 'besonders effektiver Code'.

Sie werden es gemerkt haben: Kevin Haddock hadert mit den Forth-Kontrollstrukturen. Die zig verschiedenen Ansätze z.B. beim `CASE`, die dann auch noch von System zu System verschieden sind, haben seinen Unmut dann auf die Spitze getrieben.

Daß Ärger der Vater der Kreativität ist, erkennt man an der Lösung, die in besagter *'Forth Dimensions'* dann vorgestellt wird. Sie zeigt meines Erachtens alles, wofür Forth steht: Einfachheit, Mächtigkeit und syntaktische Eleganz. Und UCS steht natürlich für *'Universal Control Structure'*.

Universal Control Structure

Kevin Haddock orientiert sich bei der Realisierung seiner UCS etwas an bekannten Mustern aus 'C' oder UNIX-Shellskripten. Bevor ich jetzt allerdings versuche, seinen Ansatz abstrakt hier darzulegen, zeige ich lieber, wie es später einmal aussieht und erläutere dann die Funktionsweise anhand dieses Beispiels.

```
: .char ( char -- )
{ { dup 32 | dup 126 |
vv } drop ascii . } emit
;
```

`.CHAR` gibt ein Zeichen, dessen ASCII-Wert zwischen 32 und 126 liegt, als solches aus. Alle anderen Zeichen werden als Punkt (.) ausgegeben.

Die strukturierenden Elemente in dieser Definition sind die Zeichen `'{}|'` und `vv`. Alles andere ist gewöhnlicher Forthcode. Die geschweiften Klammern markieren Anfang und Ende eines sogenannten Kontrollblocks. Diese Stellen sind mögliche Sprungziele für alle denkbaren Verzweigungen. Die Verzweigungen werden in `.CHAR` durch `'|'` und `vv` dargestellt.

`'&'`, `v`, `vvv`, `vvvv`, `vvvvv`, `^`, `^^`, `^^^`, `^^^^` und `^^^^^` sind weitere Möglichkeiten für die Darstellung von Verzweigungen, sind aber schlecht alle zusammen in einem einzigen Beispiel unterzubringen.

Wie geht das nun? Eigentlich ist es aus dem Beispiel schon fast zu erkennen. `'|'` spricht sich `IFNOT`. Findet `'|'` ein TRUE-Flag auf dem Stack vor, so wird der aktuelle Kontrollblock verlassen und das Programm unmittelbar hinter dem aktuellen Kontrollblock (hinter der nächsten

Stichworte

Kontrollstrukturen
F68K
Forth Dimensions

'}') fortgesetzt. Ist das Flag FALSE, so passiert gar nichts. Dies kann am Beispiel leicht erläutert werden.

Angenommen, das übergebene Zeichen (char) sei 50. Der Vergleich

```
dup 32 <
```

liefert FALSE. Das Programm läuft dann geradeaus weiter und der nächste Vergleich (126?) wird ausgeführt. Wäre das Zeichen 17 gewesen, so hätte der Vergleich TRUE ergeben. '|' hätte dann hinter die nächste '}' verzweigt, wo dann durch

```
drop ascii .
```

die 17 durch den ASCII-Code des Punktes ersetzt worden wäre. Das folgende `EMIT` hätte diesen Punkt dann dargestellt. Das Zeichen '&' (sprich IFSO) reagiert so ähnlich wie '|', nur da die Bedeutung des Flags entgegengesetzt ist. '|' und '&' stellen also BEDINGTE Verzweigungen jeweils zum Ende des aktuellen Kontrollblocks dar. Was passiert nun weiter mit einem druckbaren Zeichen, z.B. unserer 50?

Nachdem es die bedingte Verzweigung

```
dup  32 < |
```

überwunden hat, wird auch

```
dup 126 > |
```

kein Problem mehr sein.

Das Programm findet nun das Wort 'vv'. Die kleinen v's sollen Pfeile nach unten symbolisieren und sprechen sich 'SINK'. Zwei v (SINK 2) bedeuten dann nicht nur den Austieg aus dem aktuellen Kontrollblock, sondern auch aus dem nächst höheren, einschließenden Kontrollblock. In unserem Beispiel entspricht dies schlicht dem Sprung zur zweiten '}'.

Ein 'v' entspricht also einem UNBEDINGTEN Sprung ans Ende eines Kontrollblocks, wobei die Anzahl der v's angibt, wieviel Kontrollblockebenen dabei zu überwinden sind.

Passend dazu hat *Kevin Haddock* noch ein oder mehrere '^' (sprich: 'FLOAT') eingeführt, die entsprechend unbedingte Sprünge zum Anfang eines Kontrollblocks '{' symbolisieren. Mit ihrer Hilfe können Schleifen dargestellt werden:

```
: .asc ( addr count -- )
{ over c@ .char
  \ Zeichen ausgeben
    1- swap 1+ swap
  \ addr++, count--
    ?dup &
  \ !=0 IFSO fertig
    ^
  \ zurück zum Anfang
}
```

Um auch den letzten Zweifler von der sprichwörtlichen Universalität von UCS zu überzeugen, sei ein weiteres Beispiel angeführt, das die Anwendung von UCS auf einen CASE-Fall zeigt (Abb. 1).

Ich überlasse es dem Leser, sich die Funktion selbst zu Gemüte zu führen.

Ist etwas aufgefallen? Gut. Sicher haben Sie bemerkt, daß für die Ausgabe der Strings das interpretierende '. (' an Stelle des an solchen Stellen üblichen '." ' verwendet wurde. Danach haben Sie sich vielleicht gefragt, ob es kein Druckfehler ist, daß diese Codesequenz nicht mit ': Name' eingeleitet und mit ';' abgeschlossen wird?

Sie sind auf der richtigen Spur: das alles wird gar nicht kompiliert, wie in den vorangegangenen Beispielen, sondern aus einem Block interpretiert. Neben der Universalität ist dies der zweite Clou der UCS: sie sind interpretierbar! Das bedeutet, daß eine umfangreiche aber keineswegs zeitkritische Definition wie im Beispiel bis auf ein freundliches

```
<u> load
```

keinen Speicherplatz verbraucht!!

Beispiel einer CASE-Anweisung in UCS

```
{
        { cr .( Enter a letter: ) key $7f and
                { dup 27 = & .( bye! ) vvv }
                { dup ascii a = & .( alpha ) vv }
                { dup ascii e = & .( edward ) vv }
                { dup ascii i = & .( ida ) vv }
                { dup ascii o = & .( ocean ) vv }
                { dup ascii u = & .( union ) vv }
                cr bell dup emit .( is not a vowel! )
        } drop ^.               \ zurück zum Anfang
} drop
```

Abb. 1

Dies ist wohl mehr als ein netter Nebeneffekt, auf den *Kevin Haddock* nicht ohne Grund recht stolz zu sein scheint. Als mögliche Anwendungen nennt er Kompilationsskripte, nichtzeitkritische Routinen, speicherkritische Programme oder Debugging ohne Kompilieren. *Kevin Haddock* selbst hat auf diese Weise ein objektorientiertes System mit dynamischem Binden (late binding) erstellt, auf das er aber nicht weiter eingeht (leider).

Noch ein kleiners Schmankerl am Rande: *Kevin Haddock* fühlt sich durch die beiden geschweiften Klammern an Gesichter erinnert. Zwei solcher Gesichter zusammen zeigen ihm die doppelseitige Maske des Theaters, die Komödie und Tragödie symbolisieren. Entsprechend führt er für die geschweiften Klammern '{' und '}' die Sprechweisen 'COMEDY' und 'TRAGEDY' ein. Ich finde, allein diese phantasievollen Bezeichnungen verleihen der ganzen Geschichte einen gewissen Reiz. Ich jedenfalls würde gerne mehr davon sehen.

Implementation

Jetzt geht's an's Eingemachte. Nach der Beschreibung des UCS ist man zunächst erstaunt, wenn man das Listing abtippt. Nach dem Warmtippen (ca. 3 Screens) ist man schon fertig! Hält man sich an die Regeln für Quelltextformatierung, die zu befolgen wir ja alle gelobt haben, wird es etwas mehr, aber immer noch nicht viel. Nachdem man vorhin noch etwas Angst hatte, so leicht ins tiefe 'C' abzurutschen, weiß man jetzt wieder sicher: es ist Forth!

Trotzdem, trotzdem, so ganz einfach ist die ganze Geschichte nicht.

UCS in kompilierender und interpretierender Form funktionieren ganz verschieden, so daß man sie auch getrennt beschreiben kann. Nachher schlagen beide Herzen in einem einzigen Wort, was durch die etwas in Verruf geratenen state-smart-Worte erreicht wird.

Interpretieren

Das Einfachere zuerst. Zur Erinnerung: durchzuführen sind bedingte

und unbedingte Sprünge vorwärts oder rückwärts. Sprünge für den Interpreter sind nun leicht dadurch zu verwirklichen, indem man den Interpreterzeiger >IN, der den Offset in den Quelltext angibt, ab dem WORD sein Glück versucht, an die passende Stelle setzt.

Glücklicherweise sind alle möglichen Sprungziele durch '{' oder '}' dargestellt. Die Worte '|' und '&' müssen nun einfach im Quelltext nach der nächsten '}' suchen und >IN auf die Stelle dahinter zeigen lassen. 'vvv' z.B. sucht nach der dritten '}' vorwärts, '^^' sucht die zweite '{' rückwärts.

Nach dem Justieren von >IN darf der Interpreter dann fröhlich weiterlaufen. COMEDY und TRAGEDY sind in diesem Fall Dummyworte, da sie einfach nur vorhanden sein müssen, um die Sprungziele zu markieren.

Dieses Verfahren ist zwar sehr einfach, bringt aber auch einige Einschränkungen mit sich, die sich aber nach meinem Dafürhalten ertragen lassen. Zum einen muß der gesamte Quelltext am Stück, daß heißt nicht über mehrere Blöcke oder mehrere Zeilen bei Eingabe von der Kommandozeile, vorhanden sein. Für 'normale' Forthsysteme bedeutet das eine maximale Quelltextlänge des höchsten Kontrollblocks von 1kB, bei dem von mir verwendeten F68K 2kB. Weiterhin dürfen keine weiteren '{' oder '}' beliebig im Quelltext auftauchen, da die UCS sich an diesen Zeichen orientieren. Es wird eben einfach bis zu einer geschweiften Klammer vorwärts oder rückwärts gescannt. Syntaktische oder semantische Besonderheiten finden dabei keine Berücksichtigung. Ein unverhofft auftauchendes Wort mit dem Namen {FOO kann da für einige Verwirrung sorgen.

Kompilieren

Das Verhalten der UCS-Worte während der Kompilation verlangt uns Hobbyprogrammierern einen zweiten Blick ab, wenn wir ihre Funktionsweise verstehen wollen.

Im Grunde existieren in den meisten Forthsystemen nur zwei Primitive, mit deren Hilfe sich eigentlich jede Verzweigungsstruktur erstellen läßt. Es sind dies 'BRANCH' für eine unbedingte Verzweigung und '?BRANCH' für eine bedingte Verzweigung. Beide erwarten ein In-Line-Argument, da die Zieladresse der Verzweigung angibt.

Zur Erinnerung und als Beispiel für die gebräuchliche Verwendung dieser Primitiven mögen die Definitionen von 'IF' und 'THEN' dienen (Abb. 2).

'>MARK' und '>RESOLVE' könnten in einfachen Forthsystemen auf einfache Weise definiert sein (Abb. 3).

Nach diesem kurzen Exkurs in das Basiswissen 'Kontrollstrukturen' zurück zum UCS. Auch hier werden lediglich die Primitiven 'BRANCH' und '?BRANCH' zum Einsatz gebracht. Dies sichert UCS auch eine gewisse Portabilität, doch dazu später mehr.

Zunächst kann man grob einteilen: die bedingten Verzweigungen IFSO und IFNOT verwenden ausschließlich die Primitive '?BRANCH', die unbedingten Verzweigungen SINK und FLOAT entsprechend 'BRANCH'.

So weit so gut. Zusätzlich unterscheiden sich die beiden Typen durch ein ganz wesentliches Kriterium: alle IFSO und IFNOT eines Kontrollblocks enthalten einen bedingten Sprung zu ein und derselben Stelle im Programm, nämlich dem Ende des Kontrollblocks. Bei den SINK- und FLOAT-Operatoren, die immer an der gleichen Stelle unmittelbar vor TRAGEDY stehen, verhält es sich genau umgekehrt. Hier kann zu allen Stellen gesprungen werden, die durch COMEDY und TRAGEDY gekennzeichnet sind.

Einmal hat man es also mit einem Sprung von verschiedenen Stellen zu einem einzigen Ziel, anderfalls mit einem Sprung von *einer* Stelle zu einer Vielzahl möglicher Ziele zu tun. Diesem kleinen aber feinen Unterschied, der bei genauerer Betrachtung der Grund für die universelle Einsetzbarkeit ist, wird auch bei der Implementation (natürlich) Rechnung getragen.

Initiiert wird der gesamte Mechanismus durch COMEDY. Hier wird die Grundinformation für beide Verzweigungstypen erzeugt, indem zwei Werte auf dem Stack abgelegt werden: der aktuelle DP (HERE, siehe MARK) und eine 0.

Der erste Wert dient allen Rückwärtssprüngen, die durch FLOATs erzeugt werden. Da dieser Wert die Kompilationsadresse von COMEDY darstellt (COMEDY selbst erzeugt ähnlich wie 'BEGIN' keinen Code, sondern 'arbeitet' ausschließlich zur Kompilationszeit), wird bei ^, ^^, ^^^, ... an eine solche Adresse zurückverzweigt. Da jedes COMEDY seine Adresse auf dem Stack hinterlegt, muß bei der Kompilation von FLOATs lediglich mit einem bestimmten Offset, der sich aus der Anzahl der FLOATs berechnet, in den Stack zugegriffen werden (Bemerkung dazu s.u.), um sich die Adresse des Sprungziels zu verschaffen.

Dies wird u.a. im Wort '~' (siehe Listing) erledigt. '~' ist das Wort, das den notwendigen Code für alle FLOAT und SINK-Operationen generiert. Wie SINK-Operationen arbeiten, erkläre ich später.

Üblicher BRANCH Gebrauch

```
: IF
postpone ?branch \ Primitive kompilieren[compile]
?branch          \ so hie das früher mal
mark             \ Platz für In-Line-Argument
                 \ reservieren und seine Position
                 \ auf dem Datenstack ablegen
; immediate
: THEN
resolve          \ an der Adresse, die von 'MARK'
                 \ hinterlassen wurde, wird das
                 \ Sprungziel (z.B. HERE) einget.
; immediate
```

Abb. 2

Etwas schwieriger liegt die Situation bei den IFSOs und IFNOTs, die während der Kompilation zwar schon wissen, daß sie einmal springen möchten, aber noch gar nicht wissen WOHIN, denn das potentielle Sprungziel TRAGEDY erscheint im Quelltext erst später. Dies ist so ähnlich wie beim 'IF', das auch noch nicht weiß, wo das Sprungziel 'THEN' letztendlich einmal stehen wird. Deshalb hinterlegt es auch einfach nur seine Adresse auf dem Datenstack (zur Kompilationszeit), damit 'THEN' diese offene Sprungreferenz dann auflösen kann.

In dieser Art lösen auch die IFSOs und IFNOTs ihr Problem, allerdings mit der kleinen Komplikation, daß MEHRERE IFSOs oder IFNOTs zur

UCS finde ich schlichtweg pfiffig

gleichen TRAGEDY springen möchten. Um die Sprungreferenzen aller IFSOs und IFNOTs später auflösen zu können, hinterlegt COMEDY die bereits erwähnte 0 auf dem Datenstack. Sie dient quasi als Anker für eine gelinkte Liste von Adressen, an denen sich offene Sprungreferenzen befinden. Jedes IFSO und jedes IFNOT nimmt nun dieses Stackelement und tauscht es gegen die Adresse aus, an der es später sein Sprungziel eingetragen haben möchte. Anstelle der Sprungadresse wird das besagte Stackelement eingetragen. Beim ersten IFSO oder IFNOT ist dies die 0. Später dann die Referenzadresse des vorangegangenen IFxx.

Dadurch entsteht eine Liste der offenen Referenzen, die durch die 0, die von COMEDY abgelegt wurde, abgeschlossen wird. Diese Liste wird dann durch TRAGEDY abgearbeitet. Dabei kann es alle Referenzen auflösen.

Klingt etwas schwierig, ein Blick ins Listing schafft aber schnell Klarheit.

Bleiben die SINKs. Da auch diese Vorwärtsreferenzen darstellen, nutzen sie ebenfalls die Listen und hängen sich in diese ein. Nur gibt es hier die Komplikation, daß der Sprung über mehrere Kontrollblockebenen erfolgen kann. Es gilt also, sich in die RICHTIGE Liste einzutragen, da jedes COMEDY eine solche Liste anlegt, deren Zeiger alle auf dem Stack liegen. Hier benutzt *Kevin Haddock* einen Trick, der nicht so ganz sauber ist. Um die richtige Liste zu benutzen, adressiert er schreibend den Stack direkt (`SP@`), ohne Benutzung von Stackoperatoren. Dies ist zwar einfacher und schneller, es bleibt aber dennoch ein kleiner Beigeschmack. Auf der anderen Seite wäre aber der Einsatz von ebenfalls etwas unschönen Stackoperatoren `PICK` und `ROLL` notwendig geworden. Man hat also die Wahl, auf welche Weise man es unschön haben möchte.

Natürlich habe ich UCS abgetippt, so daß das Listing in maschinenlesbarer Form für Jedermann vorliegt. Da Forth bekanntlich unglaublich portabel ist, können Sie sicher sein, daß es auf Ihrem System nicht funktioniert. Meine Version läuft unter F68K. Dabei waren gegenüber dem Original nur geringe Anpassungen vorzunehmen, was hoffen läßt, daß sich dies für andere Systeme ähnlich verhält. Etwas benachteiligt sind Systeme, die keine Blocks benutzen, da ein Hin- und Herlaufen im Quelltext für die interpretative Ausführung nicht so ohne weiteres möglich ist. Hier sind weitreichende Anpassungen erforderlich. Das Forth-Modell von F-PC verlangt zudem ein Nachsehen bei den Kontrollstrukturen, da F-PC seinen Code in verschiedenen Speicherbereichen erstellt.

Schlußbetrachtung

UCS finde ich schlichtweg pfiffig. Ich habe lange nicht mehr so viel Spaß beim Ausprobieren gehabt. Wenn man sich etwas an die Logik gewöhnt hat, sind tatsächlich schnell auch komplexe Abläufe zu erstellen. Kleine Wermutstropfen wie der direkte Stackzugriff stören in der Regel nicht und sind im Notfall umgehbar.

Vielleicht kann man noch etwas über die Syntax nachdenken. Ich finde, '|' sollte man vielleicht nicht nur 'IFNOT' sprechen, sondern auch so schreiben, da dieses schmale Zeichen im Quelltext schnell untergeht. Auch für `COMEDY` und `TRAGEDY` würde ich mir andere Schreibweisen wünschen. Wie wäre es mit `COMEDY` und `TRAGEDY`? Und wie wäre '`>>TRAGEDY`' anstelle von '`vv`' oder '`<<<COMEDY`' für '`^^^`'?

Wie das dann aussehen könnte, zeigt Abb. 4.

❑

>mark, >resolve

```
: >mark ( -- adr )
  here    \ Position merken
  0 ,     \ Platz für In-Line-Sprungziel reserv.
;
: >resolve ( adr -- )
  here    \ dies ist das Sprungziel
  swap !  \ bei der von 'MARK' hinterlassenen
          \ Adresse eintragen
;
```

Abb. 3

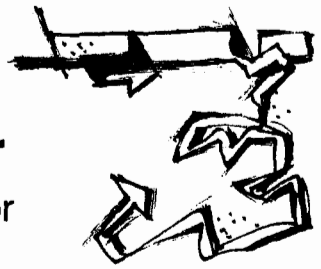
Anwendungsbeispiele

```
: .char ( char -- )
  COMEDY
    COMEDY
      dup  32 IFNOT
      dup 126 IFNOT >TRAGEDY
    TRAGEDY
    drop ascii .
  TRAGEDY
  emit ;

: .asc ( addr count -- )
  COMEDY
    over c@ .char   \ Zeichen ausgeben
    1- swap 1+ swap \ addr++, count--
      ?dup
      IFSO Y
  TRAGEDY ;
```

Abb. 4

Vorwärts -

und dann kreuz und quer



von Friederich Prinz und Michael Major
c/o F. Prinz, Homberger Str. 335, 47443 Moers

Wenn man Benutzeroberflächen programmiert, stößt man auf das Problem der Verwaltung verschiedener Funktionsebenen. In diesem Artikel werden zwei unterschiedliche Lösungen eines speziellen Aufruftyps beschrieben.

Bei der Programmierung einer Applikation unter "ZF" stellte sich mir ein Problem, das in ähnlicher Form 'immer wieder einmal' aufgetaucht war und dem ich bisher stets spezifische, der jeweiligen Applikation angepaßte Lösungen zugeführt hatte. Der 'Weg' spezifischer Lösungen beinhaltet aber die Notwendigkeit, jedes Mal 'das Rad neu erfinden' zu müssen. Das erschien mir diesmal als zu unbefriedigend, so daß ich nach einer generellen Lösung suchte.

Zwei dieser generellen Lösungen, oder besser: potentiell mögliche Wege, möchte ich hier vorstellen, wobei ich vorsorglich bereits an dieser Stelle darauf hinweisen muß, daß zumindest der erste Vorschlag nicht bedenkenlos auf jedes beliebige, andere System übertragbar sein wird.

Zunächst möchte ich die grundsätzliche, für den folgenden Aufsatz vereinfachte Problematik beschreiben:

In einer Applikation (Programm) soll an einer ganz bestimmten Stelle in eine Gruppe von Subfunktionen eingesprungen werden. Diese Subfunktionen dienen der Auswahl bestimmter 'Servicefunktionen', die nach einem Druck auf noch zu definierende Tasten aufgerufen werden. Dabei benutzen die einzelnen Subfunktionen zu einem Teil gleiche Tasten für unterschiedliche Aktionen. Einige Tasten werden nur von einem Teil der Subfunktionen oder sogar nur von einer Subfunktion genutzt. Nur eine einzige Taste hat in jeder Subfunktion die gleiche Aktion zur Folge.

Während der Arbeit der 'Funktionsgruppe' kann jede Subfunktion jede andere Subfunktion aufrufen.

Der 'Abbruch' einer beliebigen Subfunktion soll das Arbeitsende der gesamten Funktionsgruppe bedeuten.

Zur Verdeutlichung (Bild 1): Eins, Zwei und Drei bilden eine 'Funktionsgruppe', die von 'Applikation' aufgerufen wird. Die Funktionsgruppe wird aus einer beliebigen Stelle heraus verlassen, woraufhin die Arbeit von 'Applikation' fortgesetzt wird. Während die Funktionsgruppe arbeitet, kann `Eins` die Subfunktionen `Zwei` und `Drei` aufrufen. `Zwei` kann `Eins` und `Drei` aufrufen und `Drei` kann auf die Subfunktionen `Eins` und `Zwei` zugreifen. Grundsätzlich sollen hier 'beliebig' viele Subfunktionen definierbar sein. Zum besseren Verständnis will ich es aber bei diesen drei Funktionen belassen.

Aus der o.a. Skizze wird sofort deutlich, daß hier Vorwärtsrefrenzen erzeugt werden müssen. Wenn die drei Subfunktionen Eins Zwei und Drei in eben dieser Reihenfolge definiert werden, dann ist der `'Drei'` sowohl `Eins`, als auch `Zwei` bekannt. Aber der Compiler wird schon bei der Definition von `'Eins'` seine Arbeit unterbrechen, wenn in `Eins` die Subfunktionen `Zwei` und `Drei` aufgerufen werden, die zu diesem Zeitpunkt noch gar nicht definiert sind. Diese Vorwärtsreferenzen lassen sich aber, wie in FORTH üblich, relativ einfach via `DEFER` realisieren, wie in dem an späterer Stelle folgenden Beispiel zu sehen sein wird.

Ein anderes Problem ergibt sich aus der Steuerung 'zwischen' den einzelnen Subfunktionen. Hier sollen innerhalb einer Subfunktion Abfragen der Tastatur den Aufruf einer jeweils anderen Subfunktion bewirken. Da in der vorliegenden Aufgabe nur drei Subfunktionen insgesamt behandelt werden müssen, lassen sich die drei Funktionen, wie in der o.a. Skizze angedeutet, als 'Ring' organisieren. Diese Ringorganisation macht es möglich, sich sehr einfach mit der Tabulatortaste, bzw. mit der Kombination Shift-TAB, 'links' oder 'rechts herum' in diesem Ring zu bewegen. Wären mehr als drei Subfunktionen in der Funktionsgruppe enthalten, und gälte die Maxime, daß jede Funktion jede andere Funktion aufrufen können muß, dann würde das den 'Tastaturtreiber' jeder einzelnen Subfunktion vergrößern, und zwar linear zur Anzahl der Subfunktionen. Dies ist hier aber nicht der Fall, so daß man -zunächst- das Beispiel (Bild 2) durchaus akzeptieren kann:

In der Zeile (1) wird eine beliebige Taste abgefragt. Wenn diese Taste eine Kombination aus Shift & TAB war, dann wird der zuvor duplizierte Tastenschlüssel vom Stack entfernt und die Subfunktion (c = Drei) aufgerufen. War die gedrückte Taste nicht Shift-TAB, wird in der Zeile (2) ge-

Stichworte

ZF
Vorwärtsreferenzen
Assembler
Funktionsaufruf
Menüsteuerung

Eine Funktionsgruppe wird aufgerufen

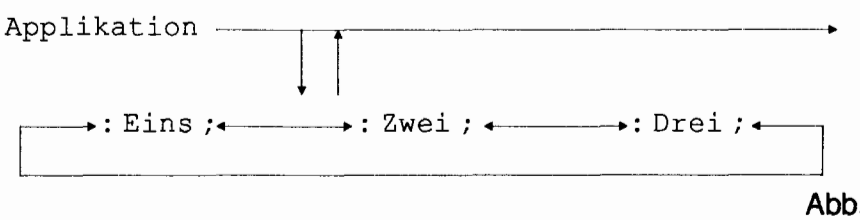


Abb. 1

prüft, ob auf dem Stack der Tastenschlüssel für TAB liegt. Ist dies der Fall, ruft die 'Eins' die Subfunktion (b = Zwei) auf, nachdem der Tastenschlüssel vom Stack entfernt wurde. In der Zeile (3), die hier nur angedeutet ist, können beliebige, weitere Auswertungen der in (1) gedrückten Taste folgen, wenn diese weder TAB noch Shift-TAB war. Letztlich wird in der Zeile (4) definiert, daß die Subfunktion Eins als beendet gilt, wenn die Taste ESC gedrückt wurde.

Gerade die Definition in (4) muß aber noch nicht bedeuten, daß damit die Arbeit aller Subfunktionen, bzw. der gesamten Funktionsgruppe beendet ist ! Tatsächlich ist dies in der obigen Definition von 'Eins' auch nicht der Fall.

Hier kommt der Aufgabenstellung das Wort NEST 'in die Quere'. NEST sorgt dafür, daß nach der Abarbeitung von Verzweigungen stets 'an der richtigen Stelle' weiter gearbeitet wird. Dazu legt NEST die Adresse in der PFA auf den Returnstack, die nach Abarbeitung der Verzweigung als nächstes abgearbeitet werden sollte (siehe Forthgruppe Moers: FORTH Intern). Das heißt im Beispiel von Eins, daß, würde in der Zeile (1) nach 'Zwei' verzweigt, die Adresse der PFA in 'Eins' auf dem Returnstack liegen würde, an der die CFA des DUP aus der Zeile (2) liegt. So kompliziert das auch klingt - die Abarbeitung dieser 'Listen' ist im Prinzip recht einfach. Dazu wird beispielhaft in Bild 3 die komplette 'Liste' der Definition 'Eins' aufgezeigt.

Wenn man sich die Parameterfeld-Liste ansieht, wird die oben gemachte Aussage deutlicher. Noch einmal: wird in 'Eins' die Prüfung der gedrückten Taste auf Shift & Tab als WAHR erkannt, dann verzweigt der innere Interpreter ab der Speicherstelle 31221 nach 31225 und 31227, wo die Adresse der CFA der Funktion "C" eingetragen ist. Über das Wort ?BRANCH wird aber die Rückkehradresse aus der Verzweigung, hier die Adresse 31229, auf dem Returnstack abgelegt, weil nach Beendigung der Verzweigungsdefinition exakt dort weiter gearbeitet werden soll !

Nun stelle man sich vor, die Funktion 'Eins' würde gestartet, von dort würde nach 'Drei' verzweigt werden, und von 'Drei' führte der Weg weiter nach 'Zwei'. Jeder der drei Aufrufe ließe einen Wert auf dem Returnstack zurück ! Beliebig oft ließe sich das nicht durchführen, weil definitiv die Resourcen des Returnstacks 'irgendwann' (systemabhängig !) erschöpft wären. Aber auch die Rückkehr via ESC funktioniert dann nur bedingt ! Tatsächlich brächte das erste ESC, ausgelöst in 'Zwei', den Interpreter wieder in die Funktion 'Drei' zurück. Das nächste ESC, diesmal in 'Drei' ausgelöst', würde nach 'Eins' zurück führen, und erst das dritte ESC könnte die Arbeit der gesamten Funktionsgruppe beenden. Das eingangs verlangte Verhalten ist aber anders definiert !

Wenn man sich allerdings deutlich macht, daß dieses 'Fehlverhal-

Tastaturabfrage

```
DEFER a
: Eins ( -- )
    BEGIN
       10 10 AT ." Definition Eins (A)"    \ Funktionskontrolle
 ( 1 )  KEY DUP 143 = IF DROP c THEN       \ Reaktion auf Shift-TAB
 ( 2 )       DUP   9 = IF DROP b THEN      \                   TAB
 ( 3 )   ( DUP - tue irgendwas )           \ weitere Auswertung von KEY
 ( 4 ) 27 = UNTIL ;                        \ Reaktion auf ESC
       ' Eins IS a                         \ Zuweisung der PFA-Liste von "Eins"
                                           \ an den Header von 'a'.
```

Abb. 2

'Liste' der Definition EINS

Achtung: Die tatsächlichen Adressen, bzw. die tatsächliche CFA als 'Startadresse' wird in anderen Systemen in der Regel anders lauten.

```
' Eins        31177          CFA von 'Eins' - (Offset auf das CodeSegment)
              31179          (LIT)          - Start der PFA von 'Eins'
              31181          10
              31183          (LIT)
              31187          AT
              31189          (.")
              31191          19             - COUNT Byte für den Text
              31192
              31193
                ..           ...            - Zeichen des Textes
              31209                         "Definition Eins (A)"
              31210
              31211          KEY
              31213          DUP
              31215          (LIT)
              31217          143
              31219          =
              31221          ?BRANCH        - Verzweigung bei Shift-TAB
              31223          31229          - wenn NEIN, weiter ab 31229
              31225          DROP           - wenn JA  ....
              31227          C              - Aufruf der Funktion 'Drei'
              31229          DUP
              31231          (LIT)
              31233          9
              31235          =
              31237          ?BRANCH        - Verzweigung bei TAB
              31239          31245          - wenn NEIN, weiter ab 31245
              31241          DROP
              31243          B              - Aufruf der Funktion 'Zwei'
              31245          (LIT)
              31247          27
              31249          =
              31251          ?BRANCH        - Verzweigung bei ESC
              31253          31179          - wenn NEIN, weiter ab 31179
              31255          UNNEST         - wenn JA  , Ende der Arbeit
              31257
```

Abb. 3

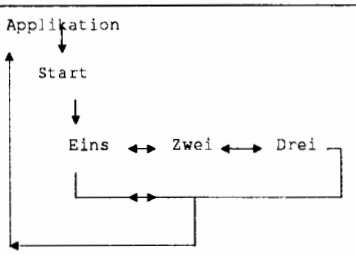
ten', das im Übrigen für ein 'normales' Arbeiten des inneren Interpreters unabdingbar ist, einzig durch den Wert auf dem Returnstack bestimmt wird, dann drängt sich der Gedanke auf, daß durch eine entsprechende Manipulation dieses Wertes ein anderes Verhalten erzwingbar sein müßte. Und genau so ist es ! Das Wort UNNEST am Ende einer 'Parameterliste', und damit am Ende jeder Definition, sorgt lediglich dafür, daß der Inhalt des TOR (Top Of Returnstack) in den IP (InterpreterPointer) geladen wird. Wenn der TOR nun aber keine Rückkehradresse der vorher aktiv gewesenen Subfunktionen enthält, dann kann gar nicht in diese zurück verzweigt werden ! Dies macht sich das Beispiel (Bild 4) zunutze.

Applikation mit Tücken

```
       DEFER a    ' NOOP IS a          \ WortHeader für Vorwärtsreferenzen
       DEFER b    ' NOOP IS b
       DEFER c    ' NOOP IS c
\ ------------------------------------------------------------------------
: Eins ( -- )
    R> DROP                           \ Rückkehradr. vom RET-Stack löschen
    BEGIN
      10 10 AT ." Definition Eins (A)"  \ Funktionskontrolle
      KEY DUP 143 = IF DROP c THEN      \ Reaktion auf Shift-TAB
          DUP   9 = IF DROP b THEN      \                    TAB
      ( DUP - tue irgendwas )           \ weitere Auswertung von KEY
     27 = UNTIL ;                       \ Reaktion auf ESC
     ' Eins IS a                        \ Zuweisung der PFA-Liste von "Eins"
                                        \ an den Header von 'a'.

: Zwei ( -- )
    R> DROP                           \ siehe "Eins"
    BEGIN
      10 10 AT ." Definition Zwei (B)"
      KEY DUP 143 = IF DROP a THEN
          DUP   9 = IF DROP c THEN      \ potentielle Rückkehradresse -1-
     27 = UNTIL ;                       \                             -2-
     ' Zwei IS b

: Drei ( -- )
    R> DROP                           \ siehe "Eins"
    BEGIN
      10 10 AT ." Definition Drei (C)"
      KEY DUP 143 = IF DROP b THEN      \ auf die "potentiellen" Rückkehradr.
          DUP   9 = IF DROP a THEN      \ würde UNNEST den InterpreterPointer
     27 = UNTIL ;                       \ setzen, wenn der nicht vorher
     ' Drei IS c                        \ entfernt worden wäre.

: Start ( -- )                        \ HilfsInit, siehe unten
     Eins  ;

: Applikation                         \ 'Applikation' startet die
     Start ;                          \ Funktionsgruppe...
```

Abb. 4

Der Fluß der Aktionen




```
\ Start arbeitet als 'Hilfsstarter',
\ dessen IP Informationen
\ durch den ersten Aufruf von Eins
\ vom RET-Stack entfernt werden

\ Die 'Funktionsgruppe' arbeitet
\ zunächst 'in sich geschlossen'.
\ Jede Teilprozedur kann jede
\ andere Teilprozedur aufrufen.
\ auf ESC reagieren alle TeilProzed.
\ gleich, durch direkten Rücksprung
\ nach 'Applikation', 'hinter' den
\ Aufruf von Start.
```

Abb. 5

Die Applikation darf die Funktionsgruppe nicht selbst und direkt starten. Der erste Aufruf einer der Subfunktionen in der Gruppe, hier 'Eins', nimmt bereits die 'Rückkehradresse' der nach dem Aufruf folgenden Definition vom TOR. Das ist im Falle von 'Start' das Semikolon ! Wird die Funktionsgruppe aber, wie in 'Applikation', über eine Hilfsroutine initialisiert, dann setzt der innere Interpreter seine Arbeit 'hinter' dem Aufruf von 'Start' in Applikation fort!

Skizze 5 macht den 'Fluß' der Aktionen deutlich.

Dieser Lösungsweg ist sicher nicht im Sinne Derjenigen, die sich zu recht, um syntaktisch korrektes Programmieren bemühen. Vermutlich ist dieser Weg auch für den einen oder anderen FORTHer nicht so einfach nachvollziehbar. Die gerade aufgezeigte Lösung, so interessant sie in technischer Hinsicht auch sein mag, entspricht weder den unter FORTH üblichen Gepflogenheiten, noch ist sie problemlos auf andere Systeme portierbar. Deshalb soll ab dieser Stelle ein zweiter Lösungsvorschlag folgen, den *Michael Major* (Forthgruppe Moers) erarbeitet hat. Zunächst stelle ich die fertige Lösung vor, die danach abschließend kommentiert werden soll:

Michael's Lösung scheint auf den ersten Blick komplexer zu sein, was aber ausschließlich an dem etwas umfangreicheren Quelltext liegt. Tatsächlich ist der 'Majorsche Weg' wesentlich einfacher nachvollziehbar. Er verläßt an keiner Stelle die unter FORTH üblichen Konventionen und verletzt nirgendwo syntaktische Regeln. Sieht man einmal von der nicht gerade üblichen, aber sehr geschickten Lösung der Vorwärtsreferenzen ab, dann verbleibt 'pures Forth vom Feinsten'. Modular aufgebaut werden die Prüfungen der Tastatur in eine eigene Definition gelegt, wo sie problemlos jederzeit an unterschiedlichste Bedürfnisse anpassbar sind. Michael's Version erfüllt die gestellten Aufgaben technisch vollkommen korrekt. Allerdings hat er die Steuerung der Abläufe zwischen den einzelnen Subfunktionen nicht diesen selbst überlassen, sondern 'nach außen' in die Routine 'Wahl' verlegt. Das führt zunächst intern zu etwas komplexeren

Applikation, Lehrbuchmäßig

```
  CREATE AKTION  6 ALLOT                \ Feld zur Aufnahme von drei CFA's
         AKTION  6 ERASE
  CREATE ZEIGER  2 ALLOT                \ Zeiger auf die CFA's in "AKTION"
         ZEIGER  2 ERASE
  CREATE ENDE    2 ALLOT                \ Flag für das ProgrammEnde und das
         ENDE    2 ERASE                \ Ende der 'Programmgruppe'.
         ( ENDE = 0 - Prozedur läuft / ENDE = 1 - Prozedur Ende )
         ( ENDE = 2 - Prozedurgruppe Ende                      )
\ ------------------------------------------------------------------------
: GRENZEN? ( -- )                     \ Abfrage ob die Grenzen für
    ZEIGER @ 0 < IF 4 ZEIGER ! THEN   \ Shift - TAB = Linksrichtung
    ZEIGER @ 4 > IF 0 ZEIGER ! THEN   \ TAB         = Rechtsrichtung
;                                     \ erreicht wurden
\ ------------------------------------------------------------------------
: DEF_HAUPT ( key -- key )            \ Gleiche Funktion in allen Prozeduren
    DUP 143  = IF -2 ZEIGER +! 1 ENDE ! \ Shift-TAB = Bewegung 'nach links'
             THEN                     \ in "AKTION"
    DUP   9  = IF  2 ZEIGER +! 1 ENDE ! \       TAB = Bewegung 'nach rechts'
             THEN                     \ in "AKTION"
    GRENZEN?                          \ Korrektur bei Verlassen der
                                      \ Feldgrenzen
    DUP  27  = IF 2 ENDE ! THEN       \ ESC gedrückt dann " 2 "
;                                     \ Flag Ende
\ ------------------------------------------------------------------------
: Eins ( -- )
  10 10 AT ." Definition EINS (A)"    \ Funktionskontrolle
  0 ENDE !                            \ Die Prozedur läuft...
    BEGIN
       KEY                            \ Taste abfragen
       DEF_HAUPT                      \ Reaktion auf TAB, Shift-Tab, ESC
       ENDE @ 0 <> IF DROP EXIT       \ Auswertung von ENDE
                   ELSE DROP          \ Weiterverarbeitung von KEY, wie
                   THEN               \ in der Applikation verlangt...
    AGAIN
;
: Zwei ( -- )
  10 10 AT ." Definition ZWEI (B)"
  0 ENDE !
    BEGIN
       KEY
       DEF_HAUPT                      \ siehe "EINS"
       ENDE @ 0 <> IF DROP EXIT
                   ELSE DROP
                   THEN
    AGAIN
;
: Drei ( -- )
  10 10 AT ." Definition DREI (C)"
  0 ENDE !
    BEGIN
       KEY
       DEF_HAUPT                      \ siehe "EINS"
       ENDE @ 0 <> IF DROP EXIT
                   ELSE DROP
                   THEN
    AGAIN
;
\ ------------------------------------------------------------------------
  ' eins AKTION     !                 \ CFA's von EINS, ZWEI, DREI
  ' zwei AKTION 2+  !                 \ in "AKTION" einsetzen
  ' drei AKTION 4 + !
\ ------------------------------------------------------------------------
: WAHL ( -- )
  0 ZEIGER !                          \ Offset '0' auf "AKTION" = EINS
  BEGIN
     ENDE @ 2 = IF >NORM 0 23 AT      \ Reaktion auf ENDE Programmgruppe
                 EXIT
              THEN
     AKTION ZEIGER @ + PERFORM        \ 'Prozedur' entsprechend [Zeiger]
  AGAIN                               \ aufrufen.
;
: Applikation ( -- )                  \ Wahl wird von 'Applikation'
    ( .... )
     Wahl                             \ aufgerufen.
    ( .... )
    ;
```

Abb. 6

Abläufen, die in Bild 5 verdeutlicht werden sollen.

Die eigentliche Steuerung wird von der Definition DEF_HAUPT übernommen, worin sowohl das Flag ENDE besetzt, als auch festgelegt wird, welche TeilProzedur als jeweils nächste Funktion aufgerufen werden soll. Diese 'äußere' Steuerung wird durch die Definition WAHL exekutiert, die entweder die gesamte 'Funktionsgruppe' aufgibt, oder entsprechend den Angaben die von DEF_HAUPT in [ZEIGER] zurückgelassen wurden, die jeweils nächste Teilprozedur anstößt.

Nun ist diese etwas größere, innere Komplexität der Abläufe durchaus kein Nachteil in dem Sinne, daß daraus irgendeine unbeherrschbare oder undefinierte Situation entstehen könnte - im Gegenteil: wie bereits weiter oben gesagt: "Das ist FORTH wie aus dem Lehrbuch".

Der interessierte Leser mag sich jetzt fragen, welcher Lösung er in Zukunft bei gleicher oder ähnlicher Problematik in seinen Applikation den Vorzug geben sollte. Diese Frage kann ich nur für mich selbst beantworten.

Mein Lösungsvorschlag arbeitet intern etwas schneller, was aber in der von mir angestrebten Applikation irrelevant ist, weil sowohl die Aufrufe der Subfunktionen untereinander, als auch die in den Subfunktionen zu definierenden Serviceroutinen eine Schnittstelle zum Anwender bilden werden. Wo immer der Rechner darauf warten muß, daß der Mensch endlich wieder einmal einen Finger auf eine der Tasten fallen läßt, sind Routinen nicht mehr zeitrelevant. Mein Lösungsvorschlag belastet den 'CodeSpace' im ZF geringer. 318 Byte gegenüber 434 Byte sind m.E. schon ein gewichtigeres Argument. 116 Bytes bereits bei einer relativ kleinen Subfunktion (bzw. Funktionsgruppe) gespart zu haben, sind mir persönlich Argument genug.

Trotzdem möchte ich aber grundsätzlich dazu auffordern, im Zweifelsfall dem Vorschlag von Michael Major zu folgen. Zum Einen führen korrekte Definitionen (in aller Regel) auch zu korrekt arbeitenden Applikationen, und zum Anderen sollten Ma-

nipulationen auf dem Returnstack stets nur 'das letzte Mittel' sein. Adressmanipulationen auf dem Returnstack sollten, zumindest in dem gezeigten Rahmen eigentlich völlig unterbleiben, es sei denn "man macht solche Dinge für sich selbst". Das zu solchen Manipulationen notwendige Wissen um die Interna des jeweilgen FORTH-Systems darf nicht grundsätzlich vorausgesetzt werden. Und letztlich führen solche systemabhängigen Manipulationen zu unportierbarem Code, was in den Augen vieler FORTHer deutlich gegen meine Version sprechen wird.

Uns, das heißt Michael Major und mir, hat die Arbeit an unseren jeweiligen Versionen großen Spaß gemacht. Hier konnten wir uns einmal mehr 'austoben', experimentieren und ein wenig mehr darüber lernen, was in 'unserem' FORTH ("ZF") wirklich passiert. Wir hoffen, daß die Leser dieses Aufsatzes den Spaß nachvollziehen können und vielleicht zu ähnlichen Aufgabenstellungen einige Anregungen erhalten.

❑

Der Fluß der Major'schen Lösung

```
WAHL - Auswertung des Flags ENDE, Prüfen, ob 'GesamtEnde' gegeben ist,
   Aufruf einer 'TeilProzedur' entsprechend dem Inhalt von [ZEIGER]
   PROZEDUR (Eins | Zwei | Drei) - Abfrage einer Taste, Übergabe der Taste
                                   an DEF_HAUPT

DEF_HAUPT - Beurteilung, ob einer der anderen TeilProzeduren aufgerufen
            werden soll, oder ob das Flag 'GesamtEnde' gesetzt werden
            soll. Setzen des Flags, bzw. des Inhaltes von [ZEIGER],
            Rückgabe der Taste an 'PROZEDUR'

PROZEDUR nach DEF_HAUPT      - Überprüfen, ob die TeilProzedur als
                               beendet gilt, wenn ja, dann Rückkehr
                               zu "WAHL", sonst Auswertung der Taste
                               und Wiederholung der TeilProzedur.

  - Auswertung des Flags 'GesamtEnde', wenn ja, dann sofortiger Ausstieg
    sonst "PROZEDUR" entsprechend dem Inhalt von [ZEIGER] aufrufen.
```

Abb. 7

Buchbesprechung

akg

'CombiBus' Handbuch 2. Aufl.

Das Buch enthält die 'open-system' Dokumentation eines LowCost LowPower Busses. Verwandt mit CAN und I²C wird ein recht einfach zu beherrschender Bus vorgestellt (2-polig, Cinch-'Phono'-Stekker). Viele Sensoren können direkt über den BUS ge-POWER-t werden, ja es können sogar zeitweise analoge Signale übertagen werden. Typischerweise arbeitet der Bus mit 1200 bit/s und mit 'Klarschrift' (ASCII), also auch low speed, aber das reicht für sehr viele Applikationen aus bzw. vereinfacht diese.

Gleichzeitig dient das Buch als ausführlicher Katalog der von Wiesemann und Theis vertriebenen CombiBus Module und Datenlogger mit allgemeinen Hinweise zur Bus-Implementation. Eine Anbindung an einen PC mit RS232 kostet neben intelligenter Software: 8 D, 4 R, 3 C, 3 Schalt-FET, die Anbindung an einen Controller: 1 SchottkyD, 3 R, 1 NPN. Der Bus benötigt keinen Kontroller, ein LichtSensor kann so auch direkt ein LCD oder einen Centronics-Drucker ansteuern.

Ideal für KühlraumÜberwachung, MaschinenLaufzeiten, KundenFrequenzen aber auch MerhfachHausklingeln, HeizungsSteuerung.

Ein wahrlich intelligenter Gag ist der ErschütterungsSensor im batteriebetriebenen Datenlogger (bis 1/2 Jahr Betriebsdauer): es kann erfasst werden, ob der direkt mit Temperatur- und FeuchteSensor ausgestatte Logger in einen anderen Raum verbracht wird!

Neben der drucktechnisch schwachen Darstellung der vielen Schaltpläne, sollte das Kapitel rund um die Adressierung/Arbitrierung/Priorisierung überarbeitet werden, erstens gibt es da Fehler im Buch, zweitens muß bei diesem kritischen Thema mit Mistverständnissen des Lesers gerechnet werden.

Der CombiBus entspricht der ForthMaxime 'KISS': Keep It Simple Stupid. Für Forthler (Hobbyisten und Profis) empfehlenswert, ideal auch zum generellen Einstieg in FeldbusTechniken.

(ASM-Quellcode für 8051 auf Disk DM100.–)

Wiesemann & Theis, Wuppertal,
144 Seiten, DM42.–

Taaaagungsband in Sicht!

Endlich ist es soweit! In Kürze wird der Tagungsband zur Forth-Tagung 1993 in Nürnberg ausgeliefert. Die Gelegenheit, den Tagungsband zu bestellen, wenn die Tagung nicht besucht werden konnte. Die Themen (die Autoren):

- FORTH-RISC-Prozessor FRP 1600 (N. Schuhmann)
- Datenstrukturen unter FORTH (Dr.-Ing. E. Woitzel)
- Objektorientierte Programmierung in der Automatisierungstechnik (M. Köller)
- FORTH auf dem Transputer (M. Hendrix, Niederlande)
- F-PC - Stand der Dinge (J. Staben)
- F-PC - Topics for Discussion (T. Zimmer, USA)
- ONF - Open Network FORTH (H. Schnitter)
- FORTH-Programmierung unter Windows (U. Schütz)
- ANS-GNU-FORTH (B. Paysan)
- : F68KANS (--) F68K -FIFI ANS ; (J. Plewe)
- FORTH for distributed Real-Time Control/CAN (Meuris, Niederlande)
- Kompilieren ins EPROM mit dem RAMBLER (K.-P. Schleisiek)
- FORTH am laufenden Band (A. Klingelnberg)
- Die FORTH-Mailbox (J.Wilke)
- Einsatz von FORTH in Protocol-Testern (H. Neuendorff)

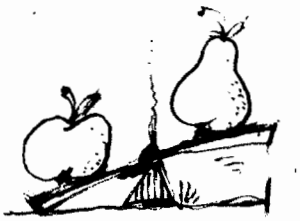
Dirk Brühl Brühl EE GmbH,
Hegelstraße 10,
90409 Nürnberg
Tel.: 0911/359088 Fax: /359044

Fortsetzung von Seite 17

auch Amplitude des Sinus ändern kann. Mit der Leer-Taste durch alle Farben zu rollen (true color=65000 Farben), wäre auch ganz toll. Außerdem ist der Sinus - egal ob *Mathyl* oder *Smiley* - immer gleich schnell. Malen sie doch den schnellsten Sinus mit der schönsten Bedieneroberfläche und Ihnen geht als Siegespreis der Fraktalgenerator JULIAM von *Dr. M. Smiley* zu. Ich zähl' auf Sie.

❑

BASIC oder Forth?



Kurs für fortgeschrittene Einsteiger

von Zbigniew Diaczyszyn und Hans-Georg Forster
Zbigniew Diaczyszyn, Pommernstr. 20, 91126 Schwabach
email: Z.Diaczyszyn@SONTAP.zer.de oder H.G.Forster@SONTAP.zer.de

An einigen Standardproblemen des Zugriffs auf Dateien sollen beispielhaft die Eigenarten der Programmiersprache Forth gezeigt werden. Zur Verdeutlichung wird parallel dazu das jeweilige Problem in BASIC programmiert.

Warum ausgerechnet BASIC als Vergleichssprache?

Um Forth-Einsteigern die Hürden zu eigenen Programmen etwas niedriger zu legen, sollen in lockerer Folge kleine Programme zu Standard-Aufgaben geschrieben und kommentiert werden, um die Eigenarten der Forth-Programmiertechniken etwas verständlicher und damit verdaulicher zu machen. Es bietet sich dabei an, das jeweilige Programm auch in BASIC zu schreiben, da es als direkten Konkurrenten zu Forth eigentlich nur die Interpretersprache BASIC gibt, die genauso komfortabel wie Forth auch auf kleineren Rechnern das unmittelbare Austesten kleiner Programmeinheiten erlaubt. Zudem dürften BASIC-Kenntnisse auch bei Einsteigern vorausgesetzt werden. Vor- und Nachteile jeder Programmiersprache sollen an jedem Beispiel diskutiert werden.

Und welches System?

Als BASIC-Implementation wurde QBASIC unter DOS gewählt, das repräsentativ erscheint für ein modernes BASIC mit Prozeduren, lokalen Variablen und strukturierten Verzweigungsstrukturen. Die Forth-Seite wird vertreten von F-PC für DOS-Computer von *Tom Zimmer* und F68K von *Jörg Plewe* für die 68000-er Linie wie z.B. den Atari ST.

Die Übertragung der Routine in Bild 1, bzw. des Wortes SHOW ins F68K-System ist so ohne weiteres nicht möglich, da F68K ein blockorientiertes File-Interface (=) besitzt. Die Nachfolgeversion F68KANS wird dagegen einen ANS-konformen Dateizugriff ermöglichen.

F68K verdeutlicht eine Hürde für Forth-Einsteiger: Viele Forth-Systeme (vgl. das volksForth der Forth-Gesellschaft) stützen sich ausschließlich auf das forthtypische Blocksystem. Dadurch muß der Einsteiger nicht nur bei der Bedienung des Editors umdenken, sondern er wird auch Mühe darauf verwenden müssen, das Wort R/W als alleinige Schnittstelle zum Massenspeicher erfolgreich einzusetzen. Da eine Datensicherung, die direkt einen Block von Daten (in der

Stichworte

Forth
BASIC
Dateizugriff

BEISPIEL 1: Daten aus einer sequentiellen Datei einlesen

Basic-Programm

```
DECLARE SUB show (name$)

CLS
show "temp.seq"

SUB show (name$)
STATIC daten$
    OPEN name$ FOR INPUT AS #1      ' Öffne eine sequentielle Datei
    DO                              ' Einleseschleife
        LINE INPUT #1, daten$       ' überlies die Trennkommata
        IF EOF(1) ,THEN EXIT DO     ' End Of File?
        PRINT daten$                ' CRLF-Bytes sind schon
        herausgefiltert
    LOOP                            ' Datei ist eingelesen
    CLOSE #1                        ' Schliee sie
END SUB
```

Forth-Programm (F-PC)

```
: show ( | <filename> --- )
    open
    BEGIN
        lineread            \ übergibt String im Forth-Format
        dup c@              \ hole das Countbyte
        0 <>                \ Ist noch nicht Null?
    WHILE                   \ Dann
        cr count 2- type    \ gib ihn aus, ohne CRLF am Ende
    REPEAT  drop            \ die Datei ist gelesen
    close ;                 \ schliee sie

cls
show temp.seq
```

❑

Regel 1 KB) auf den Massenspeicher schreibt, ohne das Disk-Operation-System (DOS) eines forthfremden Betriebsystems zu benützen, auch von groem *Vorteil* sein kann, wird einem Einsteiger aber zunächst nicht einleuchten. Doch spätestens dann, wenn ihm einmal ein paar Bytes in der FAT (File Allocation Table) gelöscht worden sind und sich seine Dateien ins Diskettennirwana verflüchtigt haben, wird er es zu schätzen wissen, wenn er mit `34 LIST` *immer* den Block 34 finden wird...

Nun, heutige Betriebssysteme sind so sicher, da viele moderne Forth-Systeme wie z.B. F-PC sogar vorrangig den Datenzugriff über ein Filesystem unterstützen.

Die Prozedur SHOW

Eine wichtige Eigenart der Sprache Forth wird bereits am Aufruf des Wortes SHOW deutlich: Forth ist so transparent, daß man die inneren Schräubchen nicht nur sehen kann, sondern, wenn man will, auch für seine Zwecke ausnützen kann. Der äußere Interpreter z.B., der normalerweise unbemerkt mit Hilfe des Wortes `WORD` die einzelnen Worte aus dem Eingabestrom herausfiltert, kann zu eigenen Zwecken eingesetzt werden.

`SHOWTEMP.SEQ...`
sei der Eingabestrom

Definiere ich `SHOW` etwa so:

```
: SHOW
( -- adr-of-counted-str)
BL WORD ;
```

dann kann ich mich mit
`SHOW TEMP.SEQ COUNT TYPE`
auf dem Terminal davon überzeugen, daß ich den Forth-Interpreter für meine Zwecke eingesetzt habe. *Tom Zimmers* `OPEN` macht genau das.

Dieses Einsetzen eines bereits vorhandenen und hochoptimierten Werkzeugs wie `WORD` ist in BASIC nicht möglich. Der BASIC-Interpreter gibt seinen inneren Aufbau nicht preis. Er tut, was man ihm beigebracht hat, und sonst nichts. Auch versteht er die Wörter, mit denen man ihn füttert, nur dann, wenn sie genau so erscheinen, wie er denkt, daß sie erscheinen müssen.

Will ich einem Unterprogramm in BASIC also einen String übergeben, *muß* ich die dafür vorgesehene Methode einhalten, in dem Fall: Deklaration einer Variablen `NAME$` im Subprogramm, die dann den Stringparameter aufnimmt, was Laufzeit und Speicherplatz kostet.

Das Positionieren des Dateizeigers

Den unterschiedlichen Zugang zum Problem des Dateizugriffs kann man auch daran verdeutlichen, wie jede Sprache den Dateizeiger positioniert.

BASIC sieht Dateien unter praktischen Gesichtspunkten und bietet von seinem Standard her nur Lesefunktionen (`LOC(kanal)`) des Dateizeigers bis auf den `PUT`-Befehl, der den Dateizeiger so setzt, da ein relativer Zugriff auf einen Datensatz in einer random-access-Datei ermöglicht wird. Das moderne QBASIC bietet zwar die Möglichkeit, mit `SEEK #kanal,position` den Dateizeiger in einer sequentiellen Datei beliebig zu manipulieren, in random-access-Dateien dagegen greift `SEEK` in QBASIC nach wie vor nur auf den einzelnen Datensatz zu.

Zwar kann Qbasic eine Datei auch im Binärmodus öffnen, in dem dann der Befehl `PUT` den Dateizeiger beliebig setzen kann, doch dann findet man sich auf der Ebene der `PEEK`s und `POKE`s wieder, wenn man an die Dateidaten herankommen will, und das ist in BASIC kein Spaß.

Für Forth ist dagegen eine Datei zunächst nichts als eine Ansammlung von Daten, deren Strukur nicht schon bei den Öffnungsbefehlen bekannt sein muß (vgl. in BASIC `OPEN FOR INPUT|OUTPUT|RANDOM`). Spezialisierte Zugriffskommandos liegen auch im neuen ANS-Standard nicht vor, so daß *Tom Zimmers* `SEEK` nicht davon abhängig ist, welche Datenstruktur in der konkreten Datei anzutreffen ist.

Natürlich gibt es auch in Forth unterschiedliche Dialekte. Doch es ist prinzipiell immer möglich, das fehlende oder anders formulierte Wort nachzubilden, notfalls in Assembler. In BASIC dagegen ist man auf den Wortschatz angewiesen, den man beim Kauf erhalten hat und den man so anwenden muß, wie es sich der Implementierer des jeweiligen BASIC-Dialekts gedacht hat. Überspitzt formuliert: In BASIC paßt sich der Programmierer der Programmiersprache an, während Forth es dem Programmierer gestatten möchte, die sprachliche Formulierung dem Problem anzupassen.

Fazit:

Um am Boden und in Praxisnähe zu bleiben: Im Alltag tauchen in Dateien kaum andere Datenstrukturen als sequentielle oder index-sequentielle auf, und für diese genügen die Standardbefehle eines BASIC-Interpreters allemal, ja es ist gerade für den Einsteiger eine große Hilfe, wenn ihm ein Befehl fast alle Arbeit abnimmt, so daß er in kurzer Zeit auch ein Ergebnis auf dem Monitor sieht. Von der großen Freiheit in Forth hat er dagegen gar nichts, wenn ihm als einzige Möglichkeit, auf eine Datei zuzugreifen, das blockorientiert Wort R/W angeboten wird wie etwa im volks-Forth für den Atari ST, das in dieser Hinsicht gar nicht volksnah ist (war?).

So nebenbei bemerkt: Um die Datei FPCWORDS.TXT mit `SHOW` auszugeben, braucht Qbasic 18,8 Sekunden. F-PC schafft das in 6,48 Sekunden!

So. Das war's fürs erste. Geplant sind noch Beiträge zu folgenden Themen:

- Array-Zugriffe (Vgl. `DIM array$(anzahl)` etc)
- Index-sequentielle Dateien
- Assemblereinbindung
- Fehlerbehandlung
- Debugging
- Aufruf anderer Programme (Shell-Funktionen)

Rückmeldungen sind natürlich höchst erwünscht.

❑

Benchmark für 8-Bit Microcontroller

von Raphael Deliano
Steinbergerstr. 37, 82110 Germering, Tel: 089-8418317

Ein simpler Benchmark für 8-Bit-CPUs und die Ergebnisse des Tests von mehreren Systemen.

Warum ein Benchmark ?

Er gibt dem Anwender eine konkrete Grundlage zur Beurteilung eines Systems. Dabei wird keine Implementierung komplett beurteilt, sondern nur einige wesentliche, vergleichbare Merkmale werden herausgestellt; so z.B beim folgenden Benchmark Geschwindigkeit und Speicherverbrauch. Zwischen beiden besteht offensichtlich ein Zusammenhang: schnelle Implementierungen verbrauchen meist mehr Speicher. Die in Bild 1 dargestellten Ergebnisse wurden im Lauf der Zeit in der Mailbox unter Mithilfe der Benutzer der Mailbox zusammengestellt.

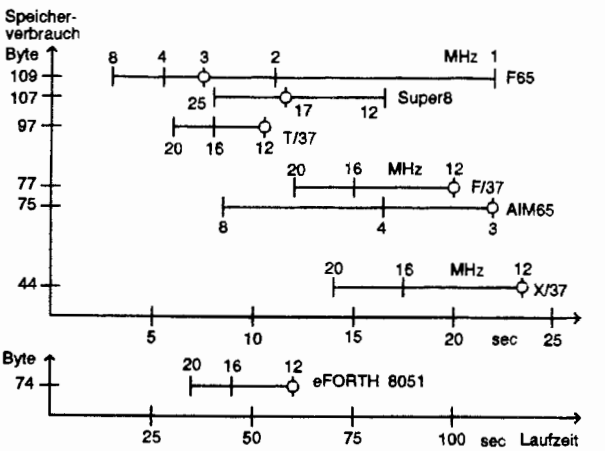
BENCH

Das hier verwendete Programm (siehe Listing) erfüllt folgende Kriterien:

- Es verwendet Befehle, die in praktisch jedem System vorhanden sind. Und sollte damit überall ohne große Modifikationen laufen.
- Es ist sinnlos. Und gibt damit niemandem Gelegenheit, es zu "verbessern". Das "Sieb des Erathostenes" ist heute kaum noch als Benchmark verwendbar, weil es zuviele Varianten gibt.
- Es ist kurz und damit leicht einzutippen.

Normierung

Die Geschwindigkeit hängt natürlich nicht nur davon ab, wie gut die Implementierung ist, sondern auch von der CPU und der Taktfrequenz. Kraftmeierei mit hohen Taktfrequenzen ist offensichtich unbegründet. Sie deutet nur auf erhöhten Stromverbrauch und EMV-Abstrahlung hin. Wesentlich für die Rechenleistung hingegen ist, wieviele Opcodes die CPU tatsächlich pro Sekunde abarbeitet; die MIPS also. Bei einem 2 MHz-6502 sind es z.B. 0,503 MIPS. Und hier hängt die CPU wiederum von der Zugriffszeit des Programmspeichers ab. Deshalb wurde die Taktfrequenz des Controllers an der Markierung o auf die Zugriffzeit eines EPROMs, das als externer Programmspeicher dient, normiert. Angenommen Sie haben ein 150nsec-EPROM. Mit welcher Taktfrequenz dürfen Sie den Controller betreiben, damit die Zugriffszeit gerade noch reicht ? Das ist die Taktfrequenz an der Markierung.



System:	getested von:	CPU:	Technik:	Zeit:	Speicher:
AIM65 FORTH	Rafael Deliano	6502	indirect-threaded	66sec 1MHz	75Bytes
F65	Rafael Deliano	6502	JSR-threaded	22sec 1MHz	109Bytes
Super8 FORTH	Claus Kühnel	Super8	?	10sec 19,6MHz	107 Bytes
eFORTH 8051	Klaus Seegebarth	80C535	?	64sec 11,06MHz	74 Bytes
F/37	Johannes Teich	8051	direct-threaded	19,7sec 12MHz	77 Bytes
T/37	Johannes Teich	8051	JSR-threaded	10,5sec 12MHz	97 Bytes
X/37	Johannes Teich	8051	token-threaded	23,5sec 12MHz	44 Bytes

Bitte um Mithilfe

Wenn Sie einen Controller haben, der hier nicht aufgeführt ist, sollten Sie ihn testen. Dokumentieren Sie:

- Speicherverbrauch in Bytes
- Laufzeit in Sekunden
- eventuell nötige Modifikationen, um das Programm lauffähig zu machen.
- genauen CPU-Typ und Taktfrequenz der Hardware.
- anhand des Datenblatts der CPU verfügbare und angekündigte Taktfrequenzen und die zugehörigen Zugriffszeiten für externen Programmspeicher. Setzten Sie das Ergebnis in die Mailbox oder nehmen Sie Kontakt zum Autor auf, damit der es tut.

❑

Stichworte

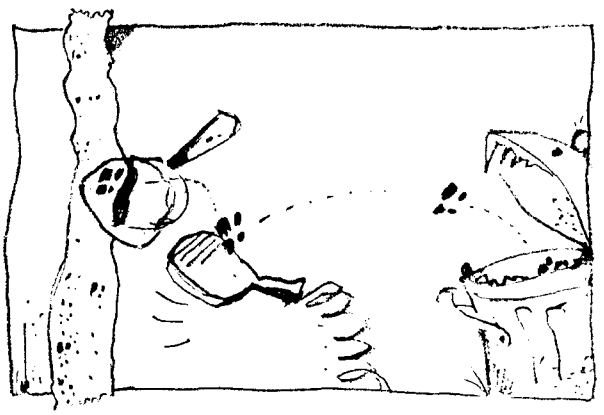
Benchmark
Microcontroller
Geschwindigkeitsnormierung

Listing zu Benchmark (Rafael Deliano)

```
HEX
5 CONSTANT FIVE
2 VARIABLE BVAR
: BENCH
100 0 DO 1 BEGIN DUP SWAP DUP ROT DROP 1 AND
IF FIVE + ELSE 1- THEN BVAR ! BVAR @ DUP 0100 AND UNTIL
DROP LOOP ;

\ Das Listing ist für F65 gedacht.
\ Die 2 vor VARIABLE bedeuted, daß die Variable 2 Bytes enthält.
```

❑

Vom Suchen, Finden und Wegwerfen



von Claus Vogt
Bülowstr. 67, 10783 Berlin, Tel: 030-2168938

Das Suchen und Ersetzen von Zeichenketten gehört zum Standardfunktionsumfang auch des dümmsten Editors. Trotzdem hat sich der Autor entschlossen, die Funktion als alleinstehendes Programm in F-PC3.56 zu implementieren. Das ist manchmal ganz praktisch und nebenbei lernt man ein wenig über DOS-Pipes.

...eigentlich sollte ja nur (!) eine Adreßdatenbank auf die neuen Postleitzahlen umgestellt werden. Schon war die Datenbank in ASCII exportiert und in das (recht preiswerte) Konvertierungsprogramm geladen. Doch dieses verlangt leider Kommas als Trennzeichen statt Tabs; außerdem will es alle Parameter in doppelten Hochtüddelchen eingeschlossen sehen. Das läßt sich mit der Funktion 'Suche und Ersetze' des Editors sicher schnell erledigen. Doch leider ist das ASCII-File für den Editor viel zu groß. Welches Textverarbeitungssystem konnte nochmal beliebig große Files verarbeiten? Das Wordstar von 1983 ist nicht im Diskettenkasten - hat sich wohl jemand ausgeliehen und nicht zurückgebracht. Na, dann starten wir halt Windows und Word für Windows. Doch siehe: Ausgerechnet die gewünschte Ersetzefunktion führt bereits nach 100 Zeilen auf eine Meldung 'Nicht genug Speicher, um die Funktion auszuführen. Bitte Datei speichern' - und das bei 16 Mb RAM. Nun ja da muß ein eigenes Replace-Programm her.

Stichworte
F-PC
Replace

Listing zu Suchen, Finden und Wegwerfen (Claus Vogt)

```
\\ REPL.SEQ : A Search & Replace Filter                clv1993

       REPL [[str1] str2] <infile >outfile

Takes input from stdin. Puts output to stdout.
Replaces all occurances of str1 by str2.
str1 and str2 may contain Ascii-codes surrounded by \
In inFile a Ascii-0-char or Ascii-26-char is treated as End of File!
On return to DOS the errorcode refers to:
0=ok 1=parameter 2=inFile 3=outFile 255=forth-error(including ^Break)

samples:
REPL                        \ copies stdin to stdout
REPL abc de                 \ replaces all occurances of abc by de
REPL \09\  TAB\09\          \ inserts TAB in front oi all tabs
REPL \$0D\\$0A\  RET\13\\^J\ \ inserts RET in front of all CRLF


Claus Vogt (Berlin), Mitglied der Forth-Gesellschaft e.V. 1993

clv 07/22/93 11:50:29.65 : 2nd version. hex supported. Trailing \. No char
apended
clv 07/21/93 13:23:54.19 : 1st version




{ \ compiler setup
Only forth also definitions   decimal
}

{ \ Variables
Create str-search  256 allot  str-search off    \ string to be searched for
Create str-repl    256 allot  str-repl off      \ replace by this
Variable str-ind   str-ind off                 \ so much chars found till yet
}

{ \\ Routines for keyboard i/o for test purposes
: readc ( -- c ) key ;
: writec ( c -- ) emit ;
: writestr ( a l -- ) type ;
}

folgende Handles sind unter DOS bereits standardmäßig definiert und geöffnet:
   0 - STDIN  Standardeingabe (i.a. Tastatur, aber mit <-Zeichen umleitbar)
   1 - STDOUT Standardausgabe (i.a. Bildschirm, aber >-Zeichen umleitbar)
   2 - STDERR Standardfehler (Bildschirm)
   3 - STDAUX Hilfsgerät (COM1:)
   4 - STDPRN Drucker (LPT1:)
(vgl. 'PC-Referenz für Programmierer. MS-Press. dtsch:Systhema Verlag 1992'
S.220. Übrigens endlich ein Buch, wo alle Infos über PC-Internas drinsteht!)

{ \ stdi/o
Code dosret ( errCode -- )   POP AX  MOV AH, # $4c  INT $21  END-CODE

: msgstr ( a n -- ) \ message auf STDERR
       2 -rot swap ( hndl cnt addr -- cnt' err? ) $4000 hdos4 2drop ;
Create $ret
       2 c, $0d c, $0a c,
: msgs ( aN cN .. a1 c1 n -- )
       " REPL " msgstr  0 ?DO  msgstr LOOP $ret count msgstr ;
: replEnd ( errCode -- ) RESTORE_VECTORS BYEFUNC dosret ;

: readstr ( a n -- ) \ lesen von STDIN
       0 -rot swap $3f00 hdos4 nip
       IF  " read error" 1 msgs  2 replEnd THEN ;
: readc ( -- c ) 0 sp@ 1 readstr ;

: writestr ( a n -- ) \ schreiben auf STDOUT
```

Fortsetzung des Listings zu Suchen, Finden und Wegwerfen (Claus Vogt)

```
      1 -rot swap $4000 hdos4 nip
      IF  " write error" 1 msgs  3 replEnd THEN ;
: writec ( c -- ) sp@ 1 writestr drop ;
}



{ \ parameter crack routines

: str\! ( str -- )      \ converts  \###\ in str into Ascii char
      dup>r   count
      BEGIN   dup                   \ length?
      WHILE   over c@ Ascii \ =
          IF                            \ -- a len ;r str
              2dup 1 /string Ascii      \ scan 2swap \ -- a' 1' a 1
              bl 4 pick c!              \ Replace trailing '\' with bl
              dup 3 pick -  2 pick c!   \ make count-byte
              over %number              \ %number ignores 1st char
              0= or over $ff00 and or
              IF                        \ no number
                  >r  2dup 1 /string  " bad parameter "  2 msgs  r>
                  1 replEnd
              THEN
              2 pick c!                 \ save char -- a' 1' a 1
              2 pick swap - r@ c+!      \ update count -- a' 1' a
              2 pick 1+  over 1+  3 pick 1- 0max cmove      \ delete
              rot drop swap             \ -- a+1 1'
          THEN
          1 /string
      REPEAT  2drop r>drop ;

: dos-par
      $80 count
      str-search 256 bl fill
      str-repl 256 bl fill
      bl skip  2dup bl scan tuck 2>r - str-search place 2r>
      bl skip  2dup bl scan tuck 2>r - str-repl place  2r>
      str-repl count  "  " str-search count  3 msgs    \ Start message
      str-search str\!
      str-repl str\!
      2drop ;
}

{ \ search & replace
: allmatches? ( -- flag )  str-ind @  str-search c@ = ;

: matches? ( c -- flag )
      allmatches?
      IF      drop false                    \ search string exhausted
      ELSE    str-search 1+ str-ind @ + c@ =
      THEN ;

: flush-found
      str-ind @ IF str-search 1+  str-ind @ writestr  str-ind off THEN ;

: repl
      BEGIN   readc
          dup 0= IF drop flush-found EXIT THEN        \ no more available
          dup matches?
          IF      1 str-ind +!                \ one more matches
              allmatches? IF                  \ all match
              str-repl count writestr         \ replace found string
              str-ind off
          THEN
          ELSE    flush-found  dup writec
          THEN
          dup 26 = IF drop flush-found EXIT THEN      \ EOF detected
          drop
      AGAIN   ;

}
{ \ F-PC-turnkey application
: main  dos-par repl 0 replEnd ;
: repl-?error ( a n f -- ) IF  1 msgs  255 replEnd THEN ;

' repl-?error Is ?error
' main Is initstuff
turnkey repl          \ also returns to DOS
}
```

Fortsetzung von Seite 12

OOP-Modul), doch sind nur die Editoren in GEM eingebunden. Es scheint auch von *Bernd Paysan* nicht mehr gepflegt zu werden, denn auf meine vor langer Zeit gestellten Nachfragen habe ich jedenfalls keine Antwort erhalten. Schon in seiner F68K-Zeit hat Jörg Plewe dagegen geduldig und prompt meine Fragen beantwortet. Daß auf dem Atari solche 32-Bit-Forthsysteme zur Verfügung standen, hatte ich keinen Grund, nach der PC-Seite zu schielen, da dort das F83 von Laxen und Perry Stand der Dinge war (16 Bit!!!), bis ich in der c't vom F-PC gelesen habe. Es ist zwar sehr schnell und gemäß seinem Autor ein Ozean, der von den Quellen guten Forthcodes gespeist wird, doch es bleibt ein 16-Bit-System, das sich an die Launen von Mrs. Dos halten muß. Da ich mein Geld nicht mit der Programmierung von Dos-Computern verdienen mu, kann ich es mir es leisten, F68KANS für das modernere und effizientere Forthsystem zu halten, da es auf den 680x0-Prozessor von Motorola setzt. (Wer schreibt ein Ladeprogramm für die Macintosh-Rechner???). Nach der Lektüre des Artikels von *M.Major* und *Fr. Prinz* über FORTH/2 wäre F68KANS wohl nur mit diesem Forth in OS/2 zu vergleichen, sowohl was die komfortable Fensterumgebung als auch die lineare 32-Bit-Adressierung des Speichers betrifft. Die Fülle an Programmierideen im Ozean F-PC ist natürlich ohne Konkurrenz, womit wir bei der Frage wären, was F68KANS außer seinem Lader sonst noch anzubieten hätte. In bezug auf die Tools ist noch alles am Werden. Momentan sind ein Decompiler und die Bibliothek der GEM-Routinen verfügbar. Geplant sind ein 68000-Assembler im Motorola-Format und ein Disassembler. Wer Lust hat, ein Programmierwerkzeug oder Beispielprogramme hinzuzufügen, möge sich beim Autor *Jörg Plewe* melden. Wie gesagt, alle Applikationen, die in F68KANS geschrieben sind, können auf allen Rechnern mit einem 680x0-Prozessor laufen, sobald ein Lader erstellt ist, der dem Kern die Systemroutinen zur Ein- und Ausgabe übergibt (-: `KEY`, `EMIT`, `FOPEN` etc). Der in C geschriebene Lader von F68K umfaßt dabei wie gesagt nur 16KB. Was F68KANS kosten soll? Darüber kann ich nichts Genaues sagen, der Autor hat mir aber versichert, daß es ein Produkt aus dem Low-Cost-Bereich sein soll. Er freut sich bestimmt über eine Kontaktaufnahme. Seine Adresse lautet:

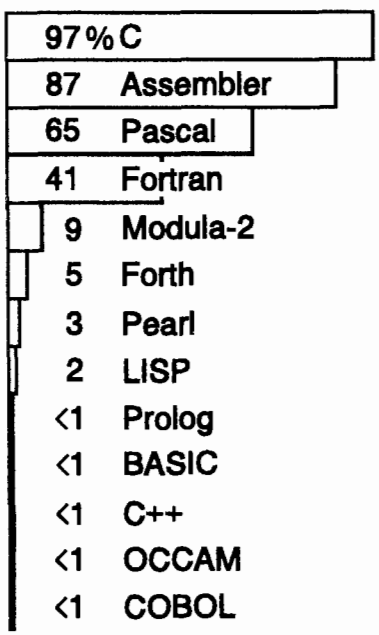
Jörg Plewe, Haarzopfer Str. 32
45472 Mülheim an der Ruhr
Tel.: 0208/497068
email:
joerg.plewempi-dortmund-mpg.de

Erbsenzähler

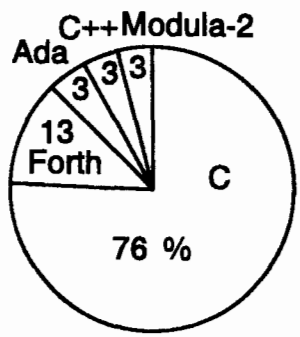
Raphael Deliano

Ab und zu sind Statistiken möglich, anhand denen man abschätzen kann, in welchem Umfang Forth in der Industrie auf embedded Controllern verwendet wird. In anderen Bereichen ist ohnehin keine nennenswerte Verbreitung von Forth zu erwarten. Statistik 1 zeigt das Ergebnis einer Befragung der Zeitschrift VMEbus vom August 89 zum Thema "Systemintegration im Auftrag des Kunden". Deutsche IngBüros und Firmen gaben an, welche Sprachen sie dafür benutzen. Bei den Systemen handelt es sich um 16-bit-Industrierechner, bei denen die HLL typisch auf ein Betriebsystem aufsetzt. Statistik 2 basiert auf dem von Motorola 1991 herausgegebenen Katalog von Compilern, die von verschiedenen Herstellern für Motorola-Controller angeboten werden. Das Spektrum reicht dabei vom 68HC05 bis zum 68300. Die Anbieter sind durchweg US-Firmen. Generell zeigen derartige Statistiken, daß es eine 80% Standardsprache gibt, auf die sich die Anbieter und zwangsläufig die Anwender konzentrieren. Wobei hier ein laufender Generationswechsel stattfindet. Bei Microcomputern z.B. von BASIC (Ende 70er) zu Pascal (Mitte 80er) zu C (Ende 80er). Daneben gibt es eine Unzahl von Sprachen, die über 1 - 2% nicht hinauskommen. Forth liegt dazwischen mit ca. 5% und scheint diese Verbreitung, jedenfalls bei kleinen Systemen, relativ stabil beizubehalten.

❑



Statistik 1: FORTH auf VME-Bussystemen



Stat.2: FORTH-Compiler für Motorola-Controller

Fundsachen

(Gesucht und gefunden in den letzten beiden Ausgaben der FD, USA)

- *"Forth hasn't taken over the world - not for any technical reason, but for a marketing reason." (E. Rather)*
- *"My number-one priority is merging the capabilities and strengths of Forth and UN*X." (J.Schneider)*
- *"Now is the time to use Forth as a labor-saving, service-generating engine that fits into open system models." (M.Elola)*
- *"I do not feel comfortable with files.(...) It is often fogotten that blocks represent a window into virtual memory..." (G.Carlton)*
- *"I (...) discovert that there was a class of problems that are trivial (or nearly trivial) in C that are difficult in Forth. (Of course, the opposte is more usually true.)" (J.Schneider)*
- *"To make porting, maintaining, and modifying easier, I tried to keep definitions dialect-invariant."(J.V. Noble)*
- *"After twenty years of running a leading Forth business, Ms. Rather has heard all the arguments against Forth and has an impresive arsenal of evidence and case histories with which to shoot down each of those objections." (M. Ouverson)*

Buchbesprechung

Arndt Klingelnberg

Dr. Dos und Ms. Dos

Arne Schäpers,
DOS 5 für Programmierer:
'Die endgültige Referenz',
1. Auflage, 1991,
Addison Wesley, Bonn,
360 kB, Hardcover 1123 Seiten, 1980 g,
ISBN 3-89319-350-2, DM99.–

Dieses Buch handelt von dem Pärchen Dr. Dos und Ms. Dos und ihrer Ahnengalerie. Ein Buch, daß nicht die Augen verschließt vor individuellen Problemen der Versionsvielfalt. In mancher Hinsicht ein Gegenpart zu Schulmann's UNdocumented DOS. Hier werden die Bugs und Quirks der vermeintlich gut dokumentierten DOS Funktionen erörtert, undokumentiertes fällt dabei aber keineswegs unter den Tisch(er). Wer nicht nur durchschnittlich programmieren will, benötigt auch die besonderen Informationen.

Arne Schäpers hat sich ausgiebig mit Turbo Pascal / Borland befaßt, das können auch seine Beispiele (Pascal, Assembler, C) nicht verleugnen.

Mir zumindest sagt sein (er-)frisch(-end) aufgemischter Schreibstil sehr zu. Gut angepasst an die wirklichen Erfordernisse wärend der Systemprogrammierung nutzt Arne Schäpers gegenüber Ray Duncan's MsDos UrBibel eine anders geartete Verdichtung. Arne Schäpers stellt eine Thematik auch gebündelt dar, inklusive Querverweisen, Warnungen, Nebeninformationen. Es verliert sich nichts wie üblich über das gesamte Buch.

Damit wäre es eines der wenigen deutschen ComputerBücher, bei dem eine Übersetzung ins Englische denkbar wäre.

Sehr empfehlenswert für den, der detailiert, schnell und umfassend über DOS Funktionen und Fehler informiert werden möchte. Vorsicht, verleitet zu interessanten Ideen.

Forth in der *elrad*

akg Da viele Forth-Experimente Schnittstellen zur realen Welt bedürfen, soll auf die in *elrad* 93/10 beschriebenen DA-Wandler (8*12 bit) hingewiesen werden. Zum Betrieb wird ein Parallelport verwendet. In elrad bearbeitet ein Pascal Programm den seriellen Datenstrom am Parallelport; wie leicht geht sowas in Forth.

Nicht nur Briefmarken-Sammler sollten sich den Artikel des Autors *Dr. Claus Kühnel* (ein Forther) in elrad 93/10 heraus-PICken. Unter anderem beschreibt Herr Kühnel das Parallax-Basic und die Kommunikation zum PC und über I^2C. Das Stack-orientiere Forth tut sich auf einem PIC sehr schwer, aber als Peripherie zu einem kleinen Forth-Controller ... und das alles wieder als Peripherie zu 'imaginary'-time Windows-PCs oder einer UNIX-Maschine ... .

In 93/11 werden technisch-wissenschaftliche Taschenrechner gestestet, inklusive der Diskussion stack/UPN.

'Die Uhr'- kein TSR, ein TWC!

Leserbrief von Arndt Klingelnberg,
Straßburger Str. 12, 52477 Alsdorf

'Die Uhr' von Friedrich Prinz ist kein TSR, sie erweist sich als ein TWC-Programm (Terminate and Wait for Crash, oder auch: Transient, Will Crash). Es wird eine falsche DOS-Service Routine benutzt, um Speicher zu allocieren, genauer: es gibt gar keine richtige. Für den Bau von TSRs müssen völlig andere Verfahren angewendet werden. Und zwar sollte das Mutter-Programm eine besondere (meist individuell abgestimmte) Struktur aufweisen, in jedom Fall muß es aber auf eine 'unübliche' Weise beendet werden. Mit ein paar 'dirty tricks' kann Friedrichs Lösung allerdings ohne sehr viel Änderung hingebogen werden (siehe unten).

Das Problem besteht darin, daß nach dem Verlassen des Mutter-Programms (Forth) der Speicher (hier: die Uhr Interrupt Routine) wieder automatisch zum Überschreiben freigegeben wird. Hatte sich der Code in eine kleine Lücke geklemmt, so kann man länger warten, bis es crasht. Beispielsweise wird ein F-PC aber -- bei erneutem Laden -- auch kleine Bereiche allocieren, und: up up in the air ... . Hat man aber dann doch den allocierten Bereich -- nicht ganz legal -- geschützt, so steht er üblicherweise mitten im schönsten freien großen Memory Block und blockiert viele Programme (hier ist nun F-PC speziell gutmütig und allociert Speicher drumherum, meistens kann man dann immer noch recht große Dateien edititeren, es fällt kaum auf).

Meiner Meinung nach kann 'Die Uhr' keineswegs als FORTH-TSR bezeichnet werden. Forth wird lediglich als Hilfmittel genutzt, um reinen Assembler-Maschinen- Code als TSR zu nutzen (Maschinen-Code mit relativen internen Sprüngen). Unter einem Forth-TSR verstehe ich, daß ich da natürlich auch Colon-Definitionen nutzen kann, mir also nicht so viel umständliche low-level Arbeit machen muß.

Der Uhren-Artikel war für mich der Anreiz, das Thema TSR generell wieder aufzugreifen. Erneute Diskussionen mit Tom Zimmer führten dazu, daß Tom nun TSR-Beispiele in TCOM integriert hat. TCOM hat heute eine andere interne Struktur (frühere Versionen waren TSR-feindlich, die Stacks lagen falsch). Ein in Forth geschriebenes TSR benutze ich fortwärend. Es bietet die endgültge Lösung des Ctrl- und Caps-Problems: KBDR.com , dieser 'KeyBoardDRiver' kommt von Tom Almy (ForthCom == 4C.com).

Eine verbesserte und kürzere Version der Uhr, Crash-frei geschützt und ab MSdos 5.0 sogar auch echt nntzbar (im UpperMemoryBlock platzsparend allociert) und zudem sowohl von F-PC ladbar, als auch von ZF, spiele ich auf die MehlSchachtel. Weitere TSR-Info gerne bei mir.

Noch was zur 'Uhr'...

Leserbrief von Michael Schröder,
Magdeburger Str. 26 39387 Oschersleben
Tel.: 03949/4086

Ich möchte noch etwas zu dem Beitrag "Die Uhr" in den letzten VD hinzufügen. Das Konzept der TSR-Programmierung aus dem Forth Interpreter heraus, das Friederich Prinz dort aufzeigt, ist ein sehr ordentliches Stück Forth und eine faszinierende Idee...

Leider kann nach meinen eigenen Erfahrungen mit TSR-Programmierung - die auch auf diversen Systemabstürzen und langen Debug-Sitzungen beruhen - dieses Konzept aber nicht sicher funktionieren, obwohl es das den Werken der Systemprogrammierung zufolge tun müßte. Zumindest in meinem Nachschlagewerk, Tischers "PC Intern", wird ausgeführt, daß Speicher, der mit der DOS-Funktion 48h allokiert wurde, vom Programm explizit freigegeben werden muß, ansonsten aber auch nach Programmende den DOS-Speicher blockiert. Auf der Richtigkeit dieser Aussage baut aber die Idee der Auslagerung des TSR-Codes in ein separates Segment auf.

Meine Erfahrungen mit Ms.Dos waren aber ganz anders. Auch der mit dieser Funktion allokierte Speicher (Forth-Wort 'alloc') wird nach dem Ende eines Programms wieder freigegeben. Überprüfen Sie dieses mit der folgenden Sequenz:

1) Ermitteln Sie auf DOS-Ebene die Größe des verfügbaren Speichers; neuere DOS-Versionen haben dazu das Kommando MEM und in den älteren geht es mit CHKDSK.

2) starten Sie Forth und allokieren Sie Speicher mittels 'alloc'; beispielsweise allokiert '1000 alloc' einen Bereich von 16kB Größe.

3) verlassen Sie Forth, ohne vorher 'dealloc' auszuführen

4) überprüfen Sie erneut den im DOS verfügbaren Speicher und vergleichen Sie mit dem ersten Wert

Laut den Büchern müßte beim zweiten Mal 16kB weniger Speicher zur Verfügung stehen; bei mir ergaben sich jedoch trotz meines ungläubigen Staunens jedesmal zwei gleiche Werte. Ich habe das zuerst unter DR-DOS 6.0 festgestellt, kam aber auch unter einem alten MS-DOS 3.3 zum selben Ergebnis. Es scheint ein besonderer Service (besonders zweifelhaft) von DOS zu sein, mit dem unsauber programmierte Programme unterstützt werden sollen.

Für die Uhr ergibt sich nun als Konsequenz: Nach der Rückkehr ins DOS wird sie ohne weiteres noch funktionieren, da Ihr Speicher zwar freigegeben, aber noch nicht überschrieben ist. Sie wird genau solange funktionieren, bis ein großes Programm den Code der Uhr mit seinen Daten überschreibt. Danach bekommen Sie wieder einmal Gelegenheit den Reset-Knopf an Ihrem Computer auszuprobieren.

Müssen wir uns nun von der Idee 'TSR-Start aus der laufenden Interpreteroberfläche' trennen oder nicht? - Vielleicht nicht unbedingt, aber ein paar 'Klimmzüge' wären schon noch nötig, um eine praktisch einsatzfähige, sichere Lösung daraus zu machen. Ein Lösungsansatz dafür könnte vielleicht so aussehen:

1) das Wort 'bye' muß soweit ausgeführt werden, daß nur noch der Funktionsaufruf für das Programmende fehlt (alle Speicheraufräumungsarbeiten müssen fertig sein und die von Forth verbogenen Interrupts müssen wieder hergestellt sein). Dann müssen statt der DOS-Funktion 00h "Programmende" die folgenden Schritte ausgeführt werden

2) Der Code der Uhr muß beim Beenden des Forth-Interpreters an den Anfang des Codesegmentes von Forth kopiert werden (Segment ?cs: Offset 100h um den PSP unangetastet zu lassen).

3) der verbogene Interrupt muß noch einmal zurechtgerückt werden, so daß er auf den neuen Uhreneinsprung zeigt.

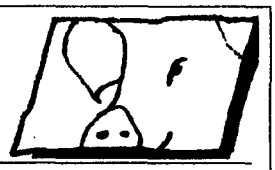
4) zuletzt wird die DOS-Funktion 31h aufgerufen, die das Programm als TSR beendet. Dieser Funktion wird die noch benötigte Speichergröße wieder in 'Paragraphen' zu 16 Byte im Register DX übergeben. (Hier kommen allerdings noch 16 Paragraphen für den PSP zu der ursprünglichen Speicherplatzberechnung hinzu.)

Die dazu benötigte Routine sollte meiner Meinung nach unbedingt nach dem Uhrencode compiliert werden; am besten in Assembler ohne Verwendung von Worten aus dem Forthkern, da ein Teil dieser Worte durch den Uhrencode überschrieben würde.

Da ich mit FPC arbeite und den internen Aufbau von ZF nicht kenne, will ich nicht versuchen, so eine Routine zu schreiben. Unter F-PC würde es sich auch eher anbieten, das Konzept von Friedrich Prinz zu nutzen, um den Ablauf eines TSR-Programmes im Interpreter zu testen. Nach einem erfolgreichen Test, kommt man dann mit TCOM leichter ans Ziel.

Das R-Schwein

von Arndt Klingelnberg
Straßburger Straße 12, 52477 Alsdorf, Tel. 02404 / 61648



Dieses ist nun mal ein Schwein auch für Anfänger und zudem von hohem Nährwert, zumindest eigentlich. Denn eigentlich denkt keiner an diese Schweinerei und dabei wird es oft richtig falsch, was eigentlich meist übersehen wird !!! Daher auch etwas zum Nachdenken. Umsteiger von BASIC greifen sich verwundert an den Kopf und nennen es eine Schweinerei, Forth83geschrittene sprechen dann liebevoll von maschinennah, was stimmt und oft hilfreich ist. Ein eingefleischtes fig-Forth verneint das und hält gar nichts von solcher Schweinerei, keiner braucht es, und meine untenstehende Schweinerei macht dort auch zuviel Mist. Schwein oder nicht, ANS läßt oft die Wahl, auf den ersten Blick aufwendig, aber eigentlich einzig richtig. Insider schmunzeln nun eigentlich schon wissend.

```
ONLY FORTH ALSO DEFINITIONS   decimal   autoTitle
\ ?NEGATE  scheint nicht unbedingt üblich zu sein, daher:
\       : ?NEGATE       ( n1 n2 -- n3 )  \  If n2 is negative, negate n1.
\               0 IF   NEGATE  THEN ;
: SchweinR      ( ??? -- ??? )  \ Forth83 !!!
       2dup XOR 0 R abs R abs R@ /MOD swap 2* R = - R ?negate ;
```

☐ Wie sollte die Funktion heißen (mein bisheriger Name ist lang/ungenau)
☐ Was macht sie (aber bitte *EXAKT* beschreiben, genau darauf kommt es an)?
☐ Wie geht es kürzer und/oder schneller ?!?
Ich hege den Optimismus, High Level (zumindest ZF , F-PC) gut ausgereizt zu haben, per Code oder auch abwechselnd HighLevel und `inline` Code liefe das ganze natürlich deutlich schneller einfacher.
☐ Wie ist sie in ANS oder fig zu schreiben ?
Für ANS reicht *EINE* Lösung nicht, es sei denn, es wird conditional compiliert.

(Anm. d. Redaktion: Wie von zuverlässiger Quelle (J. Staben) verlautet, hat Michael Schröder aus Radebeul den Haskell-Kurs ins Deutsche übersetzt! Den Namen merken wir uns...)

Hilfsbereit

Leserbrief von R. Freitag
Edelweissweg 9 50769 Köln

Ich habe in meinem Leserbrief "Boshafte Betrachtung" bereits anklingen lassen, warum sich unter den Umständen meines Einsteigens in Forth der Kreis der Forth-Gemeinschaft nicht vergrößert. Aus meinen Erfahrungen möchte ich einige Vorschläge abgeben.

Es gibt eine sehr verständliche Einführung in Forth von *F.Prinz*, die als Einsteigerhilfe sehr zu empfehlen ist. Diese sollte aber durch ein zweites Werk an Tiefe weitergeführt werden. Außerdem rege ich an, diese Einsteigerseminare zu überarbeiten und als Buch herauszugeben. Koordiniert werden sollte das alles von der Forth Gesellschaft eV. Wer das Buch haben will und nicht Mitglied ist, soll zu den Herstellkosten einen angemessenen Aufschlag zahlen. (...)

Anspruchsvoll

Leserbrief von Dipl.-Ing. A. Kochenburger
Georg-Römer-Str. 3, 67273 Herxheim

Ich habe vor kurzem die VD probeweise abonniert mit der stillen Hoffnung, darin regelmäßig über Internationale Forthaktivitäten und den Entwicklungsstand der Sprache informiert zu werden - neben cleveren Programmierlösungen. Ich gehe davon aus, daß die Forth-Gesellschaft auch Mitglied bei wenigstens einer amerikanischen Gruppierung ist und auch deren regelmäßige Publikationen bezieht (was ist mit unseren Frères von jenseits des Rheins à la JEDI etc.?). Es wäre sicher ein leichtes, Kopien von interessanten Artikeln im Originaltext im Fotosatz abzulichten und abzudrucken; ein Forth'ler kann sowieso Englisch lesen. Ich kann mir nicht vorstellen, daß sich innerhalb eines Vierteljahres nicht irgendwo ein interessanter neuer Artikel findet. Sollte sich meine o. g. stille Hoffnung nicht erfüllen, werde ich die VD ganz sicher wieder abbestellen.

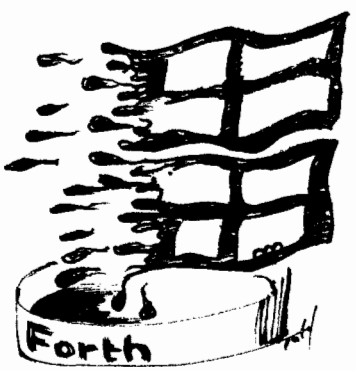
Forth WINner

rk

Gleich zwei deutsche Firmen bieten Forth-Systeme für Windows-Freunde an. Wir haben die wichtigsten Fakten für Sie aus den Produktinformationen abgeschrieben. Wenn Sie genaueres wissen wollen, setzen Sie sich mit den Firmen in Verbindung. Und wenn Sie die Produkte getestet haben, schreiben Sie einen kleinen Bericht für die VD! Abgemacht?

comFORTH für Windows...

...gibt es bei der Firma *FORTecH Software GmbH, Rostock* als Beta-Test-Version.ComFORTH für Windows ist ein segmentiertes 16-Bit-Forth-System, das zum Standard Forth-83 kompatibel ist. Das Basissystem unterstützt den Aufruf sämtlicher API-Funktionen von Windows Version 3.1 und enthält alle Vereinbarungen von Konstanten und Strukturen zum API in Form von Quelldateien. Für die Definition applikationsspezifischer Ressourcen wie beispielsweise Dialogboxen wird die Benutzung systemfremder Werkzeuge vorausgesetzt, wie sie z.B. im optional mitgelieferten Microsoft-SDX für Windows enthalten sind. Die Definition von Callback-Funktionen und Message-Handlern wird ebenso wie die Nutzung von externen DLL-Funktionen auf High-Level-Niveau angeboten. Die Implementation von Turnkey-Applikationen ist problemlos möglich. Ein Demo der Betaversion ist im Filebereich der Münchener Mailbox zu finden. Gegen eine gestaffelte Schutzgebühr, die beim Kauf angerechnet wird, erhält man von FORTecH neben der Beta-Version laufend aktualisierte Dokumentationen und Fehlerberichte. Interessenten wenden sich bitte an FORTecH Software GmbH, Joachim-Jungius-Straße 9, 18059 Rostock, Tel.: 0381-4659472, Fax: 0381-4654971



LMI WINFORTH...

...der Firma *FS Forth-Systeme GmbH, Breisach* bietet ebenfalls eine komfortable Entwicklungsumgebung, um mit Forth Standalone Windows Programme zu entwickeln. Es besteht voller Zugriff auf die gesamte Windows-API und es können zusätzlich andere Libraries (z.B. C-Libraries) benutzt und integriert werden. Alle dokumentierte Funktionen lassen sich direkt als Forth-Worte aufrufen. Es genügt bereits eine Zeile Forth-Code, um eine Messagebox am Bildschirm darzustellen. Viele mitgelieferte Demoprogramme und Online Hilfe erleichtern den Einstieg und den Umstieg vom MS-DOS, 80386 und OS/2 Paket von LMI UR/FORTH. Zur Systems '93 wird erstmals LMI WINFORTH NT, WIN 32 S, eine Entwicklung von Forth für Windows NT, vorgestellt. Dadurch ist auch die Entwicklung von Forth Programmen unter Windows mit voller Integration des 32 Bit Betriebsystems möglich. Ausführliche Informationen erhalten Sie bei FS Forth-Systeme GmbH, Postfach 1103, 79200 Breisach

Forth-Gruppen regional

Berlin
Claus Vogt
Tel. 0+30 - 2 16 89 38 p
Treffen: nach Absprache

Rhein-Ruhr
Jörg Plewe
Tel. 0+208 - 49 70 68 p
Treffen: jeden 1. Samstag im Monat im S-Bahnhof Derendorf Münsterstr. 199, Düsseldorf

Moers
Friederich Prinz
Tel. 0+2841 - 5 83 98 p
Treffen: jeden Samstag 14:00 Arbeitslosenzentrum, Donaustr. 1 Moers

Aachen
Klaus Schleisiek,
Tel. 0+241 - 87 34 62 gaf
Treffen: jeden 1. Montag im Monat als Gruppe des Computer-Club der RWTH, Seminargebäude, Raum 214, Nähe Wüllnerstraße Aachen

Darmstadt
Andreas Soeder
Tel. 0+6257 - 27 44

Mannheim
Thomas Prinz
Tel. 0+6271 - 28 30 p
Ewald Rieger
Tel. 0+6239 - 86 32 p
Treffen: jeden 1. Mittwoch im Monat, Vereinslokal Segelverein Mannheim e.V., Flugplatz Mannheim-Neuostheim

µP - Controller Verleih

Rafael Deliano
Steinbergstr. 37,
82110 Germering
Tel.:0+89 - 8 41 83 17

Gruppengründungen, Kontakte

Regional

Stuttgart
Wolf-Helge Neumann
Tel. 0+711- 8 87 26 38 p

Fachbezogen

8051 ... (Forth statt Basic,e-FORTH)
Thomas Prinz
Tel. 0+6271 - 28 30 p

Forth-Hilfe für Ratsuchende:

Forth allgemein

Jörg Plewe
Tel. 0+208 - 49 70 68 p
plewe@mpi-dortmund.mpg.de

Karl Schroer
Tel. 0+2845 - 2 89 51 p

Jörg Staben
Tel. 0+2103 - 24 06 09 p

Spezielle Fachgebiete

Anfänger und Wiedereinsteiger
Gerd Limbach
Tel. 0+2051 - 25 51 12 p
Mo. + Di. 20:00 - 22:00

32FORTH (Atari)
Rainer Aumiller
Tel. 0+89 - 6 70 83 55 gp

FORTHchips (FRP1600, RTX, Novix ...)
Klaus Schleisiek-Kern
Tel. 0+40 - 2 20 25 67 p

F-PC & tCOM, ASYST (Meßtechnik), embedded controller (H8/5xx//TDS2020, 8051 ... eFORTH...)
Arndt Klingelnberg
Tel. 0+2404 - 6 16 48 agp

Gleitkomma-Arithmetik
Andreas Döring
Tel. 0+721- 59 39 35 p

HS/Forth (Harvard Softworks)
Wigand Gawenda
Tel. 0+30- 44 69 41 p

KI (Künstliche Intelligenz), OOF (Object Oriented Forth)
Ulrich Hoffmann
Tel. 0+431 - 80 12 14 p

Unterricht mit FORTH
Rolf Kretzschmar
Tel./Fax 0+2401 - 8 88 91 ap

UUCP (FORTH ... per eMAIL)
Andreas Jennen
Tel. 0+30 - 3 96 52 27 ap

volksFORTH/ultraFORTH/RTX-FG-Forth/Super8Forth
Klaus Kohl,
Tel. 0+8233 - 3 05 24 p
Fax. 0+8233 - 99 71 f

Forth-Vertrieb

volksFORTH (PC, ST, C64) / F68K (68000) / ...
Johannes Teich
Ettaler-Mandl-Weg 19
82418 Murnau
Tel. 0+8841 - 29 43
jgt@bbs.forth-ev.de

Forum: Forth-Mailbox

Forth-Mailbox
Jens Wilke (SysOp)
Tel. 0+89 - 8 71 43 52 p
Mailbox 0+89 - 8 71 45 48,
300-2400 baud (8N1)

Hinweise

Zu den Telefonnummern

f == FAX
a == Anrufbeantworter, hier können Sie Ihren Ansprechpartner eventuell vorinformieren, erwarten Sie bitte keinen (kostspieligen) Rückruf
g == geschäftlich, zu erreichen innerhalb typischer Arbeitszeiten
p == privat, zu erreichen außerhalb typischer Arbeitszeiten

Die Adressen des Forth e.V. (Forth Büro) und der Redaktion / Anzeigenverwaltung finden Sie im Impressum.

Fundsachen (rk)

1989年8月10日 星期四 第三版

衛星電視雙節目接收系統

大型設備用激光雙波長精密測量系統

面向FORTH語言的多思新一代計算機

計算機中英、英中輔助翻譯系統

計算機輔助漢語教學系統

Annonce

Freitag, 13.3.1992, 11.Woche 17

(0241) [illegible] Edgard
■ Suche 2 Sommerreifen, 145SR12 auf Ford Fiesta Felgen (*Vielen lieben Dank für die Kaffeekassenspende...Karin*) Tel. (02406)
...ungssatz (02424)
Stoßstan... (0241)
■ 4 Digitalreifen für Anhänger gesucht. Tel. (0031) 4542
■ Suche Alufelgen für VW Polo, 6J13 mit

FORTH-SYSTEME GMBH

Postfach 1103,
79200 Breisach

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

Telefon Schweiz:
(055) 53 65 55

UR/FORTH

- FORTH-83 Standard
- Für MS-DOS, OS/2, 80386
- Direkt gefädelte Code Implentationen 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

WinFORTH

- UR/FORTH kompatibel
- Windows Funktionen werden voll unterstützt
- Erweiterte Debugging-Hilfsmittel
- Online Windows Hilfe
- Coprozessor Unterstützung möglich
- Software-Gleitkomma-Paket
- Viele Beispielprogramme
- Upgrades von UR/FORTH Systemen auf WinFORTH sind preisgünstig zu erhalten

SRS II

- Serieller ROM Emulator
- Unterstützung folgender Bausteine:
  27256, 27512, 271000, 27010, 27020, 27040
- Minimale Zugriffszeit 100 ns
- Maximale Baudrate 115.200 bits/s
- Highspeed Interface als Option
- Gleichzeitiger Zugriff von Host und Zielprozessor
- Zusätzliche serielle Schnittstelle über den ROM-Sockel
- Intel-Hex, Motorola-S oder ASCII/binär Formate werden unterstützt
- Der SRS II ist nur 157 x 94 x 36 mm groß
- SRS63 kompatibel

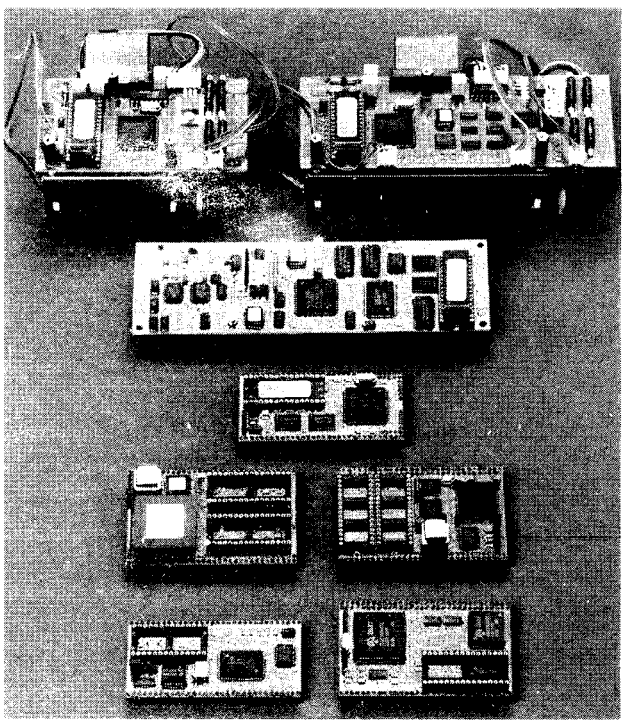
FORTH-83 Metacompiler

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
- Z80/HD64180
- 8080/8085
- 68000
- Z8
- 1802
- 6809
- 8096/97
- 78310
- 8031/32/535
- 6303
- 6502
- V25
- 68HC11
- RTX 2000
- 80C166

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

ModuNORM



CPU-Steck-Module im Scheckkartenformat:

- 8 Bit z.B. 6303
- 16 Bit z.B. V25
- Highspeed RTX-2000/1
- 80C166
- Softwareunterstützung durch SwissFORTH
- Thermodrucker und Controller
- LCD Grafik-Controller

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