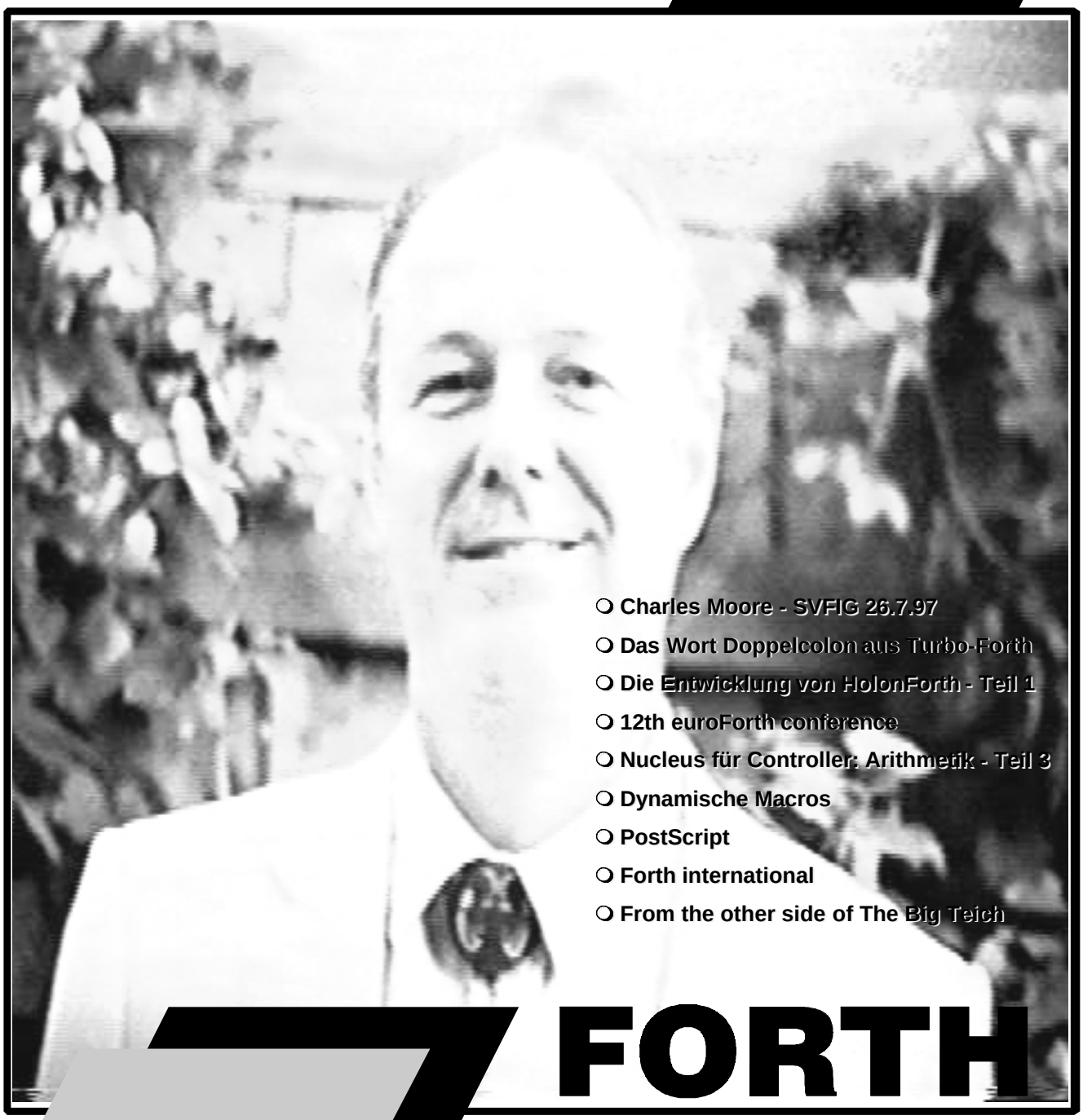
VIERTE DIMENSION

3+4/1997

13. Jahrgang 1997, 3.+4. Quartal, DM



- Charles Moore - SVFIG 26.7.97
- Das Wort Doppelcolon aus Turbo-Forth
- Die Entwicklung von HolonForth - Teil 1
- 12th euroForth conference
- Nucleus für Controller: Arithmetik - Teil 3
- Dynamische Macros
- PostScript
- Forth international
- From the other side of The Big Teich

FORTH MAGAZIN

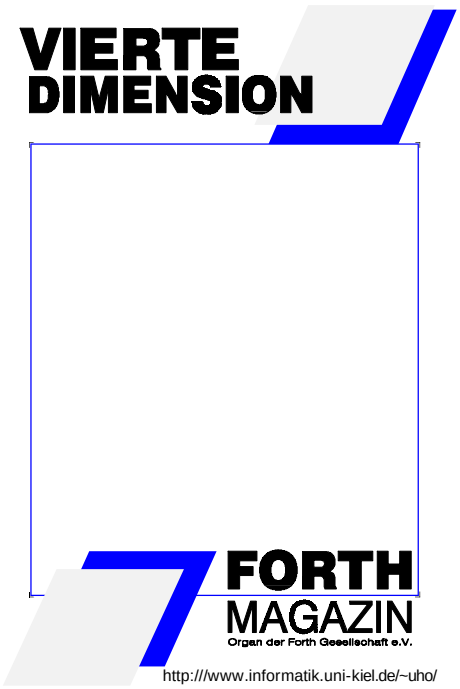
Organ der Forth Gesellschaft e.V.

http://www.informatik.uni-kiel.de/~uho/VD/

Neue Redaktionsadresse seit 1.10.1997:

Forth Magazin Vierte Dimension
c/o Friederich Prinz
Homberger Str. 335
D-47443 Moers

Tel.: 02841-58398 (Q)
F.Prinz@MHB.gun.de



http:///www.informatik.uni-kiel.de/~uho/

Dienstleistungen und Produkte von Forthlern und/oder für Forthler

Ingenieurbüro Dipl.-Ing. Wolfgang Allinger

Tel. (+Fax.) 0+212-66811
Brander Weg 6
D-42699 Solingen

Entwicklung von µC, HW+SW, Embedded Controller, Echtzeitsysteme 1 bis 60 Computer, Forth+Assembler PC / 8031 / 80C166 / RTX2000 / Z80 ... für extreme Einsatzbedingungen in Walzwerken, KKW, Medizin, Verkehr / >20 Jahre Erfahrung.

FORTecH Software GmbH

Tel.: 0+381 -405 94 71 (Fax: -4059.471)
Joachim-Jungius-Str. 9
D-18059 Rostock

PC-basierende Forth-Entwicklungswerkzeuge, System comFORTH für DOS und Windows, Cross- und DownCompiler für diverse Microcontroller, Controllerboards mit 80C196, 80C537 und H8, Softwareentwicklung für Microcontroller und PC's, auch unter Windows (und fremdsprachig)

Ing.Büro Klaus Kohl

Tel.: 0+8233-30 524 (Fax: --9971)
Postfach 11 73
D-86406 Mering

FORTH-Software (volksFORTH, KKFORTH und viele PD-Versionen). FORTH-Hardware (z.B. Super8) und -Literaturservice. Professionelle Entwicklung für Steuerungs- und Meßtechnik.

Dipl.-Ing. Arndt Klingelnberg

Tel.: ++32 +87 -63 09 89 (Fax: -630988)
akg@aachen.forth-ev.de
Waldring 23 , B-4730 Hauset, Belgien

Computergestützte Meßtechnik und Qualitätskontrolle, Fuzzy, Datalogger, Elektroakustik (HiFi), MusiCassette High-SpeedDuplicating, Tonband, (engl.) Dokumentationen u. Bed.-anl.

Möchten auch Sie oder Ihre Firma hier aufgeführt werden? Bitte wenden Sie sich an die Forth-Gesellschaft (s. Impressum).

Dienstleistungen und Produkte von Forthlern und/oder für Forthler

IMPRESSUM

Name der Zeitschrift

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

Herausgeberin

FORTH-Gesellschaft e.V.
Postfach 1110
85701 Unterschleißheim
Tel./Fax: 089/3173784
secretary@admin.FORTH-eV.de

Redaktion & Layout

Claus Vogt

Katzbachstr. 23; D-10965 Berlin
Tel.: 030/786 84 60 (Fax nach Bedarf)
vd@FORTH-ev.de

Anzeigenverwaltung:

Ulrike Schnitter c/o Forth-Gesellschaft

Internationale Kontakte:

Fred Behringer

Planegger Str. 24; D-81241 München
behringe@statistik.tu-muenchen.de

Zeichnungen:

Rolf Kretzschmar;Hans-Georg Schmid

Redaktionsschluß

Erste Januar/April/Augustwoche

Erscheinungsweise

Dreimal im Jahr.

Preis

Einzelpreis: DM 10,-

Direktorial...

Was sich schon in der Mitgliederversammlung ‘97 abgezeichnet hat, ist nun zu einer wenig erfreulichen Wirklichkeit geworden - unserer Vereinszeitschrift steht, neben anderen Veränderungen, ein Wechsel im Editoriat bevor.

Zwar kommt diese Entwicklung nicht eben überraschend, aber trotzdem sollen im folgenden die Hintergründe dieses Wechsels noch einmal kurz angesprochen werden.

Der Grund besteht in der Finanzlage des Vereins, der im wesentlichen durch die Mitgliedsbeiträge finanziert wird. In den letzten Jahren hat sich die Mitgliederzahl der FG kontinuierlich verringert. Zwar konnten die Kosten für den Druck der VD vom bisherigen Editor, Claus Vogt, reduziert werden, aber die gleichzeitig gestiegenen Forderungen für die Edition verhinderten eine Entlastung der Vereinskasse. Dies führte zu einem finanziellen Defizit des Vereins im Jahr 1996.

Um keine übereilten Entschlüsse zu provozieren, hat die Jahresversammlung ‘97 beschlossen, dem Direktorium ein weiteres Defizit für das Jahr 1997 zur Finanzierung der VD zu genehmigen, wobei aber bereits vorgegeben wurde, im 2. Halbjahr ‘97 ‘nur’ ein Doppelheft 3+4/97 erscheinen zu lassen. Das zu erwartende Defizit in der Vereinskasse wird uns voraussichtlich Dank einer großzügigen Spende für die VD im Anschluß an die Jahresversammlung erspart bleiben!

Gleichzeitig wurde das Direktorium damit beauftragt, bis zur Jahresversammlung ‘98 eine Lösung zu finden, die langfristig stabil ist, also keine weiteren Defizite im Vereinshaushalt verursacht, weil dem Verein anderenfalls die Auflösung durch das Amtsgericht droht.

Angesichts der Tatsache, daß die ‘Druckkosten’ durch den Editor bereits auf ein Minimum reduziert wurden, blieb als erster Schritt, die Kosten für die Edition, bzw. für den Editor zu senken. Diese Kosten waren allerdings für Claus Vogt zu keiner Zeit ein Verhandlungsgegenstand. Das Direktorium war gezwungen, nach Alternativen zu suchen.

Ab sofort wird darum das Editoriat der VD von der Moerser Gruppe übernommen werden, die sich die Arbeit teilen wird, so daß die Belastung für den Einzelnen einen erträglichen Rahmen nicht sprengen wird. Die Moerser Gruppe wird das Editoriat in ehrenamtlicher Arbeit übernehmen. Das heißt, daß mit Ausnahme von einzeln nachzuweisenden Kosten, die direkt der Edition der VD zuzuordnen sind, keine Kosten für das Editoriat anfallen. Die Kosten für den Verein werden drastisch sinken! Gleichzeitig kann ‘der Editor’ Mitglied der Forthgesellschaft bleiben.

Die VD soll sich auch zukünftig so eng wie möglich an das innere und äußere Konzept der VD der letzten drei Jahre anlehnen. Trotzdem wird es auch in diesem Bereich Veränderungen geben. So werden die Leser zum Beispiel auf farbige Ausgaben verzichten müssen, was selbstverständlich ebenfalls dazu beiträgt, die Kosten für die VD zu reduzieren.

Dem Direktorium bleibt an dieser Stelle, dem bisherigen Editor das Bedauern über die beschriebene Entwicklung auszudrücken, sowie ihm für seine dreijährige, anerkannt gute Arbeit zu danken. Gleichzeitig wünschen die Direktoren der FG dem neuen Editoriat viel Glück und Erfolg, vor allem in der anstehenden, schwierigen Zeit des Wechsels.

Das Direktorium

Neue Redaktionsadresse seit 1.10.1997:

Forth Magazin Vierte Dimension
c/o Friederich Prinz
Homberger Str. 335
D-47443 Moers
Tel.: 02841-58398 (Q)

... eines Tages an der Strippe bei der PC-Hotline geschah, was einmal geschehen mußte

Ein Super DAU

(dümmster anzunehmender User)

Hotline: Hotline, Guten Tag.

DAU: Guten Tag, mein Name ist Daumeier. Ich habe da ein Problem mit meinem Computer.

Hotline: Welches denn, Herr Daumeier?

DAU: Auf meiner Tastatur fehlt eine Taste.

Hotline: Welche denn?

DAU: Die Eniki-Taste!

Hotline: Wofür brauchen Sie denn die Taste?

DAU: Das Programm verlangt diese Taste.

Hotline: Was ist denn das für ein Programm?

DAU: Das kenne ich gar nicht, aber es will, daß ich die Eniki-Taste drücke. Ich habe schon die STRG-, die ALT- und die Groß-Mach-Taste ausprobiert, aber da tut sich nichts.

Hotline: Herr Daumeier, was steht denn auf Ihrem Monitor?

DAU: Eine Blumenvase.

Hotline: Nein, ich meine, lesen Sie mir mal vor, was auf Ihrem Monitor steht.

DAU: I be em.

Hotline: Nein, Herr Daumeier, was auf Ihrem Schirm steht, meine ich.

DAU: Moment, der hängt an der Garderobe.

Hotline: Herr Daumeier ... ?!

DAU: So, jetzt habe ich ihn aufgespannt, da steht aber nichts drauf ?!?!

Hotline: Herr Daumeier, schauen Sie mal auf den Bildschirm und lesen Sie mir mal genau vor, was darauf geschrieben steht.

DAU: Ach, so, Sie meinen ... oh, Entschuldigung. Da steht: "Please press any key to continue".

Hotline: Ach Sie meinen die any-key-Taste. Ihr Computer meldet sich in Englisch

DAU: Nein, wenn der was sagt, piepst er nur.

Hotline: ..... Drücken Sie mal die Enter-Taste.

DAU: Jetzt geht's. Das ist also die Eniki-Taste! Das könnten die aber auch draufschreiben!
Vielen Dank und auf Wiederhören!

Hotline: ...????... Bitte, bitte, keine Ursache!

❑

Rubriken

Forth-Gesellschaft

Forth-Systeme

Presseschau

Forth Inside

Bücher

Was noch

Was fehlt

Die Rubriken 'Adressen und Ansprechpartner', Anfänger, ANS-Forth, Firmen, Inserenten, Musik, Online, Produktinfo, Prozessorgeflüster, Termine und die Kolumnen 'ANS Forth' und 'Forth Online'.

= Listing auf Diskette

```
1.585 5.07.97 22:54 nucleus3\lists.txt
nur etwa 1500 bytes in 1 Datei(en)
```

Leserforum

Nur drei VDs pro Jahr?

Hallo Claus,

ich habe heute die VD bekommen. Laß mich ganz kurz zwei Absätze dazu verlieren:

a) Der größte Teil der Zeitung ist wieder prima geworden, die Bilder von der Tagung sind allerdings 'bescheiden'. Auf die Dauer können wir uns so eine Qualität nicht leisten.

b) Laut Protokoll der Tagung haben wir 'nicht'!!!! beschlossen, daß es in Zukunft nur noch drei Hefte pro Jahr geben wird. Das wurde zwar zwischenzeitlich diskutiert, aber dann verworfen. Ich muß Dich daran erinnern, daß wir beschlossen haben, 'in diesem Jahr' drei Hefte herauszubringen, wovon das letzte ein Doppelheft wird.

Ich glaube, Du hast der Forth-Gesellschaft mit dieser Äußerung im Editorial einen Bärendienst erwiesen. Du solltest darüber nachdenken, wie Du dies gegenüber den Lesern klarstellen kannst (z.B. eine Meldung d.c.l.f. und eine entsprechende Korrektur im nächsten Heft.

Thomas Beierlein, tb@tb.forth-ev.de, Juni'97

[Auf der Tagung wurde lange über die Kosten der VD beraten. Im Protokoll findet sich dazu der Satz: "Es sollen im Jahr 1997 nur drei Ausgaben der VD erscheinen, wobei eines davon als Doppelheft ausgeführt wird."/clv]

Zu VD 2/97, S. 14: Mac & Midi

Leider ist im Druck ein Semikolon in die URL gerutscht. Richtig muß es natürlich heißen:
http://netsurf.citylink.de/users/hhorch/macmidi.html

(Ich reite nur deswegen drauf rum, weil da die Listings zu sehen sind, die den Weg ins Heft offensichtlich leider nicht geschafft haben.)

Bei dieser Gelegenheit: Mein Dank geht nochmals an den unermüdlichen Claus, an Rafael für die großzügige Spende, sowie an das Forth-Büro.

Helge Horch, hhorch@STUFF-.wuerzburg.netsurf.de 28.06.97

[Bei dem von Rafael gespendeten Autorengeschenk handelte es sich um das Buch von Susan Lammer: "Programmers at Work", das jetzt nicht mehr gewünscht werden kann. /clv]

Zu VD 2/97, S.28 Warum nicht Forth, sondern C ?

Die VD 2/97 ist ja sogar noch im 2. Quartal gekommen! Zwar müssen wir auf den Sponsor der blauen Titelseitenfarbe verzichten, das tut dem Inhalt aber keinen Abbruch.

Nur eine Bemerkung: Robert Freitags Artikel "C statt Forth" sieht aus, als sei er irgendwie aus der verstaubten Schublade eines Voreditors gekommen. Die Kritik am fehlenden Standard von Forth ist ja inzwischen gegenstandslos (es gibt ein ANS Forth (das sogar ISO-Standard ist!), das gibt's bei C, aber nicht mal bei C++, geschweige denn für Java, Visual Basic, Delphi oder andere Hypes). Die kritisierten Forth-Systeme sind allesamt um die 10 Jahre alt - aber mit 10 Jahre alten C-Compilern kann man auch nicht ANSI-konform programmieren. *[Der Artikel war tatsächlich 'gut abgelagert'. Preisfrage: Welches ist der älteste Beitrag in dieser VD? /clv]*

Übrigends hat mir Jens Wilke eine Windows-SDK-CD überlassen und so ermöglicht, MINOS (das Visual bigFORTH) in relativ kurzer Zeit auf Windows 95 zu portieren (ganz fertig ist der Port noch nicht). Die "normalen" Programmierer, die häufige spontane Abstürze, verkorkste Systemschnittstellen, despotische Allmachtsansprüche des Herstellers und allerhand seltsames Verhalten wie hängende AltGr-Tasten tolerieren, können aufatmen. Wer das nicht toleriert, kann unter Linux programmieren und das Ergebnis nachher auch Windows-Nutzern zur Verfügung stellen (oder umgekehrt).

Probleme gibt's noch beim Vertrieb: Die Firma, die mich versklavt hat, duldet keine Firma neben sich (obwohl sie nur für 40 Stunden eher mäßig zahlen); wenn sich da keine Verhandlungslösung anbahnt, schreibe ich halt noch ein paar Bewerbungsschreiben.

Bernd Paysan, paysan@informatik.tu-muenchen.de, Juni'97

Zu VD 2/97, S.28 Warum nicht Forth, sondern C ?

Hallo Claus!

Erst Mal vielen Dank für das Heft der "Vierten Dimension".

Nach dem ersten Durchblättern gleich bei "Warum nicht Forth, sondern C ?" (von Robert Freitag, VD2/97, S.28) hängengeblieben.

Während eines Treffens mit den Boxgewaltigen der CN hatte ich vor kurzem eine ähnliche Diskussion. *[Die CN ist die Berliner Mailbox Chat Noir 030 3822699, vgl. VD 2/97, S.39 /clv]*

Desweiteren geht eine Ergänzung für die Literaturliste (DDR) in den nächsten Tagen an die angegebene Adresse. *[vgl S.36/clv]*

Tschüß, Gerd (G.Bretschneider@BBrandes.Berlinet.de) Juli'97

Forth-Befehle für Anfänger

Hans-Jürgen Geiss wrote:

"Gibt es, für Anfänger in Forth-Programmierung, ein brauchbares Text-File, das die wichtigsten Forth-Befehle beschreibt? Das Wichtigste ist: Es sollte in deutscher Sprache verfasst sein, damit ich nicht gleich zwei Hürden überwinden muß. ;-)"

Vor ein paar Jahren habe ich anhand einiger Assemblerlistings, die mir vorlagen, eine Definition des alten 8086-FIG-Forth in Forth rekonstruiert.

(Bekanntlich kann man Forth - abgesehen von einem gewissen Kern, der in Assembler implementiert werden muß - in Forth definieren.)

Die Datei enthält ausführliche deutsche Kommentare, sowohl zur Funktionalität als auch zur Implementierung der einzelnen Forth-Worte, enthält also wahrscheinlich die von Dir gewünschten Informationen, allerdings - wie gesagt - nur für das alte FIG-Forth. Sie dient als Vorlage für einen Meta-Compiler, der daraus ein lauffähiges Forth erzeugt (über den Umweg eines MASM-Listings). Auf Wunsch bootet das erzeugte Forth auch, ohne DOMES-DOS, direkt von der Diskette.

Die entsprechenden Dateien habe ich unter meiner Homepage abgelegt. Sie sind unter
http://www.informatik.uni-rostock.de/~ajung/Forth/index.html

erreichbar. Wenn Interesse besteht, werde ich die entsprechenden Quellen und Programme irgendwo unter meiner Homepage ablegen.

MfG,
Andreas Jung
ajung@informatik.uni-rostock.de,
/z-netz/sprachen/forth, April'97

"An unpleasant fact of editing and producing a periodical (or

performing any sort of community service) is that it 'eats' enthousiastic people. It's a lot of work and if you're not careful they burn out after a few years, never to be seen again. One should not make too many jokes about this."
Marcel Hendrix.

Letzte VD 3+4/97 aus Berlin

=== I N E I G E N E R S A C H E ===

Liebe Forthler,

Das Direktorium hat mir mit Datum vom 14.8.97 mitgeteilt, daß sie ab 1998 eine preiswertere Möglichkeit für das Forth Magazin Vierte Dimension wahrnehmen werden. Das Doppelheft 3+4/97 wird also das letzte Heft unter meinem Editoriat sein.

Daher muß ich darauf hinweisen, daß alle mit mir getroffenen Absprachen, die über das Jahr 1997 hinaus gehen, nunmehr mit der neuen Redaktion bzw. mit dem Direktorium zu erneuern sind. Ich hoffe, daß dies für die Aktiven nicht allzu frustrierend wird ...

Auf diesem Weg möchte ich mich bei allen bedanken, die mich und die VD in den letzten drei Jahren unterstützt haben und möchte sie bitten, mit meinen Nachfolgern genauso gut zusammenzuarbeiten wie bisher mit mir.

Claus Vogt, Editor des Forth Magazins Vierte Dimension
clv@clvpoint.forth-ev.de, in /de/comp/lang/forth, Aug'97

Erste Reaktionen ...

"gut zusammenzuarbeiten ..."

Mache ich.

Schade daß diese Sache so endet, ich war mit Dir als Editor ganz zufrieden und damit wohl ja auch nicht allein, wie Du ja in Ludwigshafen gesehen hast. Einige Dinge in dieser Angelegenheit verstehe ich nicht und kann mir, mangels Kenntnis der Hintergründe auch kein Urteil darüber erlauben. Ist aber vermutlich, wie bei jeder Scheidung, eine Sache, bei der beide Seiten ihren Anteil dazu beigetragen haben :-(

Also nochmals danke und alles Gute

Tschüß Wolfgang
All@business.forth-ev.de
(Wolfgang Allinger)

Hi, Claus

Klingt nun ja endgültig. Dir viel Erfolg in allem, was du tust weiterhin, es hat mir gefallen wie du die Sache angepackt hast.

Grüße aus Malente, Michael
michael@malente.forth-ev.de
(Michael Kalus)

Und mir fing es gerade an Spaß zu machen mit Claus was für die VD zu schreiben.

"wie bei jeder Scheidung..."

Manches verstehe ich auch nicht. Aber vielleicht sind die Menschen wirklich so, Olli? :-(Warum muß man einen Editor rauskanten, wenn man doch bloß die preiswertere Variante durchsetzen muß als Direktorium.

Aber die Dinge sind meistens komplizierter, als man vermutet. Also, liebes Direktorium, bitte erkläre mir (uns) das genauer. Es geht um unsere Beiträge, unsere Spenden, unsere Arbeit, unseren Spaß an der Sache.

Bitte wartet nicht bis Weihnachten mit einer Info.

Wenn Claus einen Brief kriegt, warum dann wir nicht eine Erklärung? Und wenn das nicht ganz öffentlich sein soll, gibt es PMs und SnailMail.

"Also nochmals danke und alles Gute"

Und das hört sich wie eine Scheidung vom eV an? Hoffentlich nicht!

Ansonsten war's prima, was Claus gemacht hat. Dank auch von mir.

Tschüssikowski - Ekkehard
eks@velten.forth-ev.de
(Ekkehard Skirl)

Ich bin zwar (noch?) kein Direktor, es hat bei der Wahl nur zum vierten Platz gereicht, aber verstehen kann ich das schon. Unser Forth-Verein ist nicht der ADAC, und noch dazu schrumpfen die Mitgliederzahlen, wenn auch langsam. Die VD ist immer schon der teuerste Einzelposten gewesen, und mit den aktuellen Kosten kann man dieses Jahr knapp 3 VDs zahlen, und nächstes Jahr dann wohl nur noch 2 oder so. Claus hat zwar die Kosten für den Druck reduziert, für den Verein hat sich das aber nicht ausgewirkt. Eine Zeitung in VD-Qualität (und mit der aktuellen Vorgehensweise) kann man aber nicht an einem Samstag Nachmittag setzen und die Kopiervorlage ausdrucken. Jedenfalls nicht, wenn man ein DTP-Programm verwendet und auch noch redaktionelles macht.

Natürlich *kann* man eine 40-Seiten-Zeitschrift mit den richtigen Werkzeugen und den entsprechenden Autorenrichtlinien an einem Nachmittag setzen. Aber wenn ich jetzt sagen würde, für die Bernd Paysan-VD (oder Uli Hoffmann-VD, der würde das auch so machen) müssen die Artikel als LaTeX-File eingereicht werden, steigen mir alle die auf's Dach, die behaupten, LaTeX sei "Texte programmieren"*, das LaTeX-Frontend LyX liefe nicht auf ihrem "Betriebssystem", und Linux wollten sie nicht installieren.

Dann kommen die Artikel halt nur noch von den LaTeXern, und die VD hat statt 40 nur 10 Seiten. Das spart Druckkosten, Editorkosten und Porto, und führt zu einer Austrittswelle, was man aber dadurch kompensieren kann, daß man die VD nach erfolgreichem Setzen gleich nach HTML wandelt (geht dann auch automatisch) und ins Web legt. Oder man den Autoren nahelegt, das Zeugs gleich als HTML-Files abzuliefern, Paßwort für den WWW-upload gibt's auf Anfrage.

Weil dann sowieso nur noch per E-Mail erreichbare Vereinsmitglieder vorhanden sind, wird auch das Forth-Büro wegrationalisiert, weil Werbung machen kann auch ein Junk-Mailer (wer in comp.lang.* postet und noch nicht in der Liste ist, kriegt 'ne Mail), Anmeldung erfolgt über ein CGI-bin-Script, und wer nicht bezahlt, kriegt einmal X11R6.3 zugestellt, beim zweiten Mal sämtliche Linux-Hacker-Kernels...

Naja, irgendwie gefällt mir diese Lösung auch nicht so recht ;-).

*) Ja, man kann in LaTeX auch die ersten 100 Primzahlen berechnen oder einen Phrasengenerator à la "Starting Forth" schreiben, aber eigentlich ist LaTeX eine Markup-Language, d.h. man tippt das ein, was man in WinWord anklickt: Das ist eine Überschrift, und dieses Wort soll fett sein. TeX programmieren muß nur der können, der die VD setzt, wenn er das Erscheinungsbild beibehalten will.

Bernd Paysan
paysan@informatik.tu-muenchen.de (Bernd Paysan)

Mit Erstaunen und Bedauern habe ich feststellen müssen (Inhalt der 95`er VD über "http://www.informatik.uni-kiel.de/~uho/VD/"), daß ich zu früh aus dem FORTH-Verein ausgetreten bin. Da gab es doch tatsächlich noch Artikel, die sich mit FORTH und PC als Meß-, Regel- und Steuerrechner beschäftigt haben. Auf diese habe ich immer gewartet, statt dessen gab es nur Grabenkriege über das beste FORTH. :-((Wenn diese Wende auch durch den Editor mitveranlaßt wurde, Gratulation.

Die Artikel in LaTeX einzureichen wäre sicher eine nette Sache, vielleicht gibt es ja eine Möglichkeit in WORD (oder was auch immer eine Alternative darstellen soll) Formatierungsmacros zu definieren, so daß ein Minimal-LaTeX-file daraus entsteht. Der Feinschliff wäre dann immer noch Sache des Editors, aber die Arbeit könnte so verringert werden. Nun sind die "auf's - Dach - Steiger" gefragt, sich ein wenig Mühe zu machen. (In welcher Form muß die VD denn in den Druck gegeben werden?)

Piet
psc@informatik.uni-kiel.de (Peter Schneider)

In my experience with the Dutch Forth GG ("Het Vijgeblad"), one should prevent *at all costs* that the magazine does not appear regularly (and frequently). We had the problem there were not enough articles, you're very lucky that this seems not be an issue here!

"Het Vijgeblad" was done with LateX. The authors were asked to deliver straight ASCII. This worked very well, but it had nowhere near the looks of VD (evidently, the power is not in the tool, but what you do with it). I've suggested several times that "Het Vijgeblad" go on-line, but 99.9% of the readers are still *several years* away from that point. This is impossible to believe for people with usenet intravenously connected :-)

An unpleasant fact of editing and producing a periodical (or performing any sort of community service) is that it "eats" enthousiastic people. It's a lot of work and if you're not careful they burn out after a few years, never to be seen again. One should not make too many jokes about this.

marcel

mhx@iaehv.IAEhv.nl
(Marcel Hendrix)

"Direktorium, bitte erkläre mir (uns) das genauer..."

Eigentlich kommt diese Wendung nicht ganz überraschend, eine solche Entwicklung hat sich eigentlich schon

auf der Mitgliederversammlung abgezeichnet. Vielleicht mal zu den Hintergründen (Bernd Paysan hat sie ja auch schon beschrieben). Der Grund besteht einzig und allein in der Finanzlage des Vereins, der im wesentlichen nur durch die Mitgliedsbeiträge finanziert wird. In den letzten Jahren hat sich die Mitgliederzahl kontinuierlich verringert, die Kosten für den Druck wurden von Claus auch verringert (was gar nicht so einfach war, da in diesem Auflagenbereich die einmaligen Grundkosten gegenüber den Kosten für das einzelne Heft dominieren), allerdings stiegen im selben Maße die Kosten, die er für seine redaktionelle Tätigkeit verlangte. Die Vereinskasse hat also *keine* Entlastung zu spüren bekommen. Dies führte zu einem Defizit im Jahr 1996.

Um keine übereilten Entschlüsse zu provozieren, hat die Jahresversammlung '97 beschlossen, dem Direktorium ein weiteres Defizit für das Jahr 1997 zu genehmigen, wobei schon einkalkuliert wurde, daß im 2. Halbjahr ein Doppelheft 3+4/97 erscheinen wird. (Das Defizit wird uns voraussichtlich Dank einer großzügigen Spende für die VD im Anschluß an die Jahresversammlung erspart bleiben!)

Gleichzeitig wurde aber das Direktorium damit beauftragt, bis zur Jahresversammlung '98 eine Lösung zu finden, die langfristig stabil ist, also keine Defizite im Vereinshaushalt verursacht. (Sowas kann nur der Bundestag ungestraft, bei uns würde der Verein per Konkurs aufgelöst werden.) Das läßt sich aber nur ereichen, indem die Kosten für den Editor reduziert werden. Der Preis war aber für Claus *leider* kein Verhandlungsgegenstand. Da Claus diese Situation lang genug kannte, halte ich die Formulierung "rauskanten" eigentlich nicht für angemessen.

Spaß hat das allemal *nicht* gemacht. Aber wie Du schon sagst, es geht hier eben um die Beiträge und Spenden der Mitglieder. Ich hoffe nur, daß trotz solcher Sachen auch der Spaß erhalten bleibt. Die VD wird von der Moerser Gruppe übernommen werden, die sich die Arbeit teilen wird, so daß die Belastung für den einzelnen nicht so groß sein wird. Die Kosten für den Verein werden drastisch sinken. In der VD 1/98 und auf der Jahresversammlung '98 werden die Moerser ihr Konzept für die VD vorstellen, die VD 1/98 soll erst mal noch so eng wie möglich am Erscheinungsbild der jetzigen VD bleiben.

Egmont Woitzel (einer der drei FORTH eV Direktoren)
woi@baltic.e-technik.uni-rostock.de

"In welcher Form muß die VD denn in den Druck gegeben werden?"

Die VD wird derzeit aus dem Laserdrucker 'rausgeworfen und das wird dann im Xerox-Verfahren kopiert. Inzwischen bieten manche Copyshops auch an, Postscript-Files direkt mit dem Laserdrucker 'rauszulassen, das gibt ein besseres Ergebnis, und im Prinzip könnte es sogar billiger sein (der Seitenpreis von einem guten Laserdrucker ist jedenfalls unter dem eines guten Kopierers). *[nach VD 1/96 würde ich pro Heft mit mehreren Hundertern allein für Dateitransfers rechnen! /clv]*

Um das Erscheinungsbild der VD in LaTeX zu erhalten, muß man sich einen passenden Headings-Style schreiben, für umflossene Bilder, verschiedenspaltigen Text etc. gibt's auch Stile. Wie Marcel Hendrix sagt, es kommt immer darauf an, was man daraus macht, aber DTP macht TeX auch nicht von selbst.

In Word-Basic (oder VBA für Words) kann man sicher auch ein Speicher-als-LaTeX-Makro schreiben, das die wichtigeren Sachen wie fett oder monospaced festhält. Da sind aber die gefragt, die immer behaupten, LaTeX sei "Texte programmieren". Das kann man mit Word auch, viel besser sogar, denn einen TeX-Virus hat noch keiner geschrieben. Die Alternative "plain ASCII" (oder ISO-Latin1, mit Umlauten) geht dann immer noch.

Trotzdem: Das aufwendige Erscheinungsbild ist den Mitgliedern so wichtig, daß Uli Hoffmann jede Forth-Tagung mit seiner "Spar-VD" durchfällt, obwohl die wirklich professionell gesetzt ist - aber eben Buchsatz, nicht Zeitschriftensatz. Ebenso richtig ist, daß wir Interneter-Junkies uns nicht vorstellen können, daß man ohne WWW-Zugriff, News und E-Mail überhaupt noch leben kann (es ist schon schwer genug, mehr als 2 Wochen ohne auszukommen - dann sind ja all die Artikel, die man unbedingt lesen will, schon expired ;-). Aber es geht, und gerade unter den Forthern sind viele Internetabstinenzler.

Bernd Paysan
paysan@informatik.tu-muenchen.de

Naja, mann kann's auch so machen wie die ACM SIGPLAN Notices und viele Konferenzen, und einfach verlangen, daß die Leute ihre Beiträge camera-ready abgeben. Dann schaut die Zeitschrift zwar etwas uneinheitlich aus, aber man hat sich viel Arbeit erspart. Man kann ja auch

noch Stilrichtlinien herausgeben ("zweispaltig, 2cm Rand, 10pt-Times im Fließtext"), bzw. Style-files zur Verfügung stellen.

anton@mips.complang.tuwien.ac.at (Anton Ertl)

"wie bei jeder Scheidung ..."

Hmm. Könnte es sein, daß die FG einfach nicht mehr das Geld hat um eine VD in der zuletzt erreichten Qualität herauszubringen? Umsonst gibts die nun mal nicht - selbst wenn der Editor umsonst arbeiten würde nicht.

Diese Papierform geht ab 500 Mitglieder gut, behaupte ich mal so aus meiner alten Erfahrung damit, besser wären 1000. Da liegen wir aber inzwischen ziemlich drunter, Tendenz fallend hörte ich :-(

Nun müssen wir uns was Neues einfallen lassen.

michael@malente.forth-ev.de
(Michael Kalus)

Hallo Claus,

An dieser Stelle auch von mir vielen Dank für deine Arbeit als Editor. Bei der derzeitigen Finanzsituation war etwas ähnliches aber zu erwarten.

topeban@rng.forth-ev.de (Thomas Prinz)

"In der VD 1/98 und auf der Jahresversammlung '98 werden die Moerser ihr Konzept für die VD vorstellen, die VD 1/98 soll erst mal noch so eng wie möglich am Erscheinungsbild der jetzigen VD bleiben."

Das klingt ja schon ganz gut. Wenn sich die Moerser für die LaTeX-Lösung erwärmen lassen (auch mehrere Leute, die sich jeden Samstag treffen, wollen nicht dauernd DTPen), wäre ich auch bereit, einen VD-lookalike-Stil (oder was immer man ausheckt) in LaTeX zu programmieren. Aber wie ich die Leute kenne, erwärmen sie sich nur dann dafür, wenn sie sehen, wie es hinterher aussieht ;-).

Bernd Paysan
paysan@informatik.tu-muenchen.de

Seufz - kaum ist man aus dem Urlaub zurück, schon hat man mehr Arbeit am Hals als zuvor :-(

Glückauf,
der Fritz
F.PRINZ@MHB.gun.de (Friedrich Prinz)

c't 8/97: Elrad verkauft

Der Heise-Verlag hat die Elrad - zum Schrecken vieler Hobby- und Profibastler - an den Bruckberger Bruchmann-Verlag verkauft. Der Käufer will mit einer neuen Redaktion beginnen, was einige Irritationen in der Fangemeinde auslöste. Auf Seite 52 der August-c't erklärt uns Thomas Schult das Rätsel der Auflösung: Man will "frisches Blut für die c't- Redaktion".

Nur die Älteren unter uns haben noch "die Erinnerung, daß die Proportionen einmal ganz anders waren - vor achtzehn Jahren hieß c't noch Computing Today und war ein achtseitiger Innenteil von Elrad".

Pessimisten mögen die Elrad - gleichsam als Retourkutsche -zwischen den c't-Kleinanzeigen verschwinden sehen. Natürlich müssen mit Rücksicht auf den Bruckmann-Verlag die neuen Tätigkeitsfelder der Abgelichteten - allen voran Ernst Ahlers - mit 'Hardwaretest' und Multimedia angegeben werden.

Mir persönlich scheint sich hier die längst fällige Rückbesinnung der c't auf ihre hardwarenahen Wurzeln immerhin anzudeuten. Ich freue mich auf eine (notfalls auch etwas dickere) c't, die die Themen der Elrad im Controller- und hardwarenahen Bereich übernimmt. Und das zum Preis einer einzigen Zeitschrift - das wäre eine Sparmaßnahme, von der Verlag und Leser profitieren.

Und natürlich wünsche ich allen zwölf Menschen viel Erfolg und viel Spaß mit ihrem neuen (alten) Arbeitgeber.

Claus Vogt, clv@FORTH-eV.de, Juli'97

Forth-Systeme: hForth v0.9.A for 8086 and Z80

I uploaded hForth v0.9.A for 8086 and Z80, HF86V09A.ZIP and HFZ-80V9A.ZIP, to:
ftp://ftp.taygeta.com/incoming/forth

I expect them soon to be moved to:
ftp://ftp.taygeta.com/pub/Forth/Reviewed/

There were two bugs in v0.9.9. I fixed internal word 'pack"' which returned unaligned address and made some words not function properly. Also I revise SIO.F in 8086 package according to new EXE program structure.

Wonyong Koh, Ph.D.
wykoh@pado.krict.re.kr,
29.06.97, /comp/lang/forth

Forth-Systeme: Aztec für Win95

Aztec Forth ist ein dpAnsForth - T.W.Worthington UK. Es erzeugt native intel code. Windows API Aufrufe werden wie normale Forthworte behandelt und DLL's wie Wortlisten. Laut Autor ist kein spezielles Umschalten erforderlich um Windowsfunktionen aufzurufen. Die Adressen werden in Forth und Windows gleich behandelt, es ist keine Übersetzung von Forth in Windowsadressen nötig. Die Quellen werden in Blocks gehalten und über den beigefügten Windows basierten Editor bearbeitet.

Erstaunlicherweise hat der Autor seinen ersten Versuch in Intel Code genutzt, um ein ANS Forthsystem zu machen! Wer mag kann es sich holen:
http://www.ncl.ac.uk/~n6388131/PROG.HTM

"Hope someone out there likes it.
Thomas Worthington"
michael@malente.forth-ev.de (mka)
nach /comp/lang/forth Aug'97

Forth-Systeme: bigForth goes Windows

MINOS ist ein 'Rapid GUI Designer' für bigFORTH von Bernd Paysan.

Die aktuelle Demoversion soll, wenn es nach dem Willen des Autors geht, schon ab September '97 durch eine verbesserte Version ersetzt sein. MINOS kann frei als ganzes (ca. ein Mbyte) bezogen werden bei:
http://www.informatik.tu-muenchen.de/~paysan/bigforth.zip

MINOS hat zwei Teile: Eine library (ebenfalls MINOS genannt) und einen Editor als Werkzeug, um MINOS zu handhaben, 'Theseus' genannt. MINOS benutzt ein box&glue-Modell um Objekte anzuordnen - pack' alles Benötigte in die Box, dann navigiere damit. Es gibt dazu Cursorbuttons in der Icon-Leiste. Zudem die Editier-Icons Text, Code, Name, Delete und Try. Weitere Icons legen fest, wie neue Objekte angehängt werden: Als erstes in der aktuellen Box oder als letztes, vor oder nach dem aktuellen Objekt. Schließlich kann die aktuelle Form gesichert, geladen oder ausgeführt werden. Ein "designer open" im Dialogfenster führt zur Geburt eines weiteren Theseus.

Der Autor weist darauf hin, daß MINOS noch nicht fertig ist und man daher nicht erwarten sollte, daß alles schon reibungslos geht. Um die Features einfach mal anzusehen ist ein "include testwidgets.str" in einem Dialogfenster vorhanden.

Kontakt zum Autor von MINOS:

http://www.informatik.tu-muenchen.de/~paysan/

oder:

paysan@informatik.tu-muenchen.de

mka nach /comp/lang/forth Juni'97

Forth-Systeme: Wimp Forth strong on the RISK

"Wimp Forth unter RISK OS 3.7 - läuft auf strongARM."

Diese schlichte Nachricht über sein Forth veröffentlichte Charles Esson (charlese@cvs.com.au) in comp/lang/forth.

Ja was ist denn ein 'strongARM'? Henning Hansen konnte diese Frage beantworten: Der 'starkeARM' ist die jüngste Entwicklung des 32-bit-ARM-Prozessors, welcher ursprünglich von Acorn in 1987 entwickelt worden ist. Die Entwicklung wurde in ARM Ltd. weitergeführt, bis nun Digital die Lizenzen für diese Technologie erwarb, um nun den noch schnelleren StrongARM herauszubringen. Nach dem Motto "Keine Plattform ohne Forth" ist also auch hierfür ein System zu haben.

H.Hansen führte dann noch aus, der ARM Processor habe eine sehr saubere RISC Architektur, sei schnell, billig, leicht zu programmieren und verbrauche sehr wenig Strom. Er werde von Acorn in 'advanced computers' eingebaut, sei aber auch für 'embedded systems' gut und kürzlich für Oracle NC verwendet worden. *[Nun, die Experten werden hoffentlich wissen, was damit gemeint ist - der Sezza].*

So ist RiscOS ein GUI Operating System das Acorn 1988 für Computer auf ARM-Basis gemacht hat. Es sei noch immer das schnellste, sicherste und am leichtesten anzuwendende graphische User Interface, das es gäbe, mit Features wie "plug&play, drag&drop, multitasking and antialiasing" seit damals. Alles in 32-Bit, gepackt in 4 MB ROM zur Zeit. RiscOS 3.7 sei die jüngste Version die für den Acorn StrongARM RiscPC Computer gemacht worden ist.

Weitere Informationen:

http://www.acorn.com
- Acorn Group in Cambridge, UK.
- Where the brains are.
http://www.arm.com
- ARM Ltd in Cambridge,
- developer of ARM processors
http://www.acorn.com/acorn/products/strongarm/ - The StrongARM RiscPC
http://www.digital.com/semiconductor/strongar.htm - Digital StrongARM

"Disclaimer: I'm an Acorn enthusiast, and I preach Acorn as much as I preach Forth. I have tried to provide some objective info, though ;-)"
Henning Hansen,
Ebenso objektiv versuchte zu übersetzen: mka, Juni'97

Forth-Systeme: PilotFORTH 0.3.0 for Palm Pilot

PilotFORTH 0.3.0 available! Source files can now be read from the memopad. Several other enhancements included. Have a look! <http://www.interlog.com/~nbridges>

PilotFORTH is an on-board native FORTH compiler for the USR Pilot and Palm Pilot series of PDAs.

Neal Bridges (nbridges@interlog-.com) in /comp/lang/forth, Aug'97

Forth-Systeme: FORTH to PIC-compiler V1.00 out

Announcing the NEW and IMPROVED Forth-to-PIC compiler!

Thanks to the many responses from users there is now a new improved version of F2P, a Forth compiler for the Microchip PIC16Cxx mid-range microcontrollers. This version, 1.0, incorporates new features as well as support for several PIC-16Cxx-family microcontrollers. F2P now generates code for multiple code and ram banks and supports 16C61, -C64, -C620, -C621, -C622, -C71, -C73, -C74, -C84 and the flash -F84.

F2P is a compiler that reads Forth source for the PIC16Cxxfamily of microcontrollers and generates a file ready to be assembled by Microchip's MPASM. You *will* need MPASM or MPLAB to be able to generate executable code. It is however free and may be downloaded from Microchip's web site http://www.microchip-.com. F2P will help in structuring the code so that routines hopefully can more easily be reused, with gains in both smaller executable and more rapid program development.

The compiler generates subroutine threaded code without any normal Forth kernel, instead it incorporates code parts from an external archive when needed. So that, if you do not need a word it is not loaded and does not take any code space.

Details on the implementation are given in the revised and extended F2P-READ.ME file included in the F2PIC10.EXE self-extracting executable.

Try it out!

It should be available via anonymous ftp at

ftp://ftp.taygeta.com/pub/Forth/PIC/F2PIC10.EXE

The latest version with incremental bug fixes can be obtained from

ftp://lagrange.isy.liu.se/pub/F2PIC

As always, I am keen to hear from users. Suggestions for improvements are welcome, as are bug reports and success stories...

Cheers,
Micke
Michael Josefsson (mj@isy.liu.se)
in /comp/lang/forth, Aug'97

Forth outside: Gibt es Forth auf dem Mars?

Diese Frage beschäftigt die Menschheit schon seitdem sie das erste mal von der Existenz von Leben, Kanälen, Sklavenaufständen, Brett-BoardSystemen und preiswerten Schuhkartons auf dem Mars gehört hat. Statt diese Frage noch weiter auszuwälzen, hier die 'keep it simple'-Formulierung aus comp.lang.forth:

Is it?

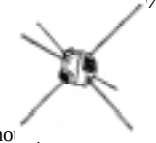
John Garst garst@sunchem.che-m.uga.edu

Und die Antwort der Fachfrau:

Not that I know of, but the NEAR satellite currently studying an asteroid (and also in the news last week) was produced at Johns Hopkins Applied Physics Lab and includes 4 RTX processors programmed in Forth.

Cheers,
Elizabeth
erather@forth.com (Elizabeth D. Rather)
(clv nach /comp/lang/forth, Juni 1997)

Forth ist ...



Forth is art witho

@austin.finnigan.com
ner), comp.lang.forth, Jun'97

Programming is the right words in the right order.

Poetry is the best words in the best order.

wilbaden@netcom.com
(Wil Baden), comp.lang.forth, Jun'97

Und ich frage mich schon seit Jahren, wer eigentlich diesen Satz erfand:

"Forth ist die Fortsetzung der Hardware mit anderen Mitteln!"

(clv)

Der Teufel steckt im PC

"Macs & Forth. Sollte ich besser einen PC kaufen?"

Ja, solltest du. Allem Apple Macintosh'igen (und seinen Klonen) haftet Linkes an und ist von übel. Ich rate dir ausdrücklich: Nimm' einen PC. Aber nimm' weder DOS noch Win95 noch NT noch OS/2, sondern FreeBSD UNIX. Denn diese Plattform hat eine Menge guter FORTH compiler/interpreter - und alle kostenlos.

Eric DeArment in comp.lang.forth
ejd@efn.org
(Übersetzt von mka, Jun'97)

New FORTH rap CD !

Jack Woehr's "Rappin' Forth"!!!!
which features the hits:

- Stairway to ANSI
- What kind of Forth am I?
- BEGIN Strawberry-Fields AGAIN
- Every Brodie loves some Brodie sometime
- JAX the Knife
- My Brodie lies over the ocean
- Comment your code, Tom Dooley
- He ain't heavy, he's my Public-Domain Implementation (formerly: I Did It My Way)
- I ain't got no Brodie

Special Cameo appearences by 'Chuck' and the "Buttons Three", Ray Duncan and the Doughnuts, Elizabeth and the Rather-Not's.

The entire Forth community joins in on the final selection, a treme dously emotional rendition of "We are the World".

(special BLOCK-format-compatible CD player required)

aus comp.lang.forth von Mike Haas (mikeh@starnine.com)

Vom AOL-Umschlag 08.07.96

Die Welt der Smilies ...

:-)	Der lächelnde Smiley
:-(	Der traurige Smiley
;-)	Der augenzwinkernde
:´-(	Der weinende Smiley
8-)	Smiley mit Brille
:-X	Smiley Kuß
(:-*	Der küssende Smiley
{}	Umarmung
:D	Der herzhaft lachende
%-(	Der verwirrte, traurige
(:-$	Der kranke Smiley
%*}	Der sturzbetrunkene Smiley
(:-&	Der böse Smiley
:-Q	Der rauchende Smiley
(-.-)Zz	Der schlafende Smiley
:-[	Der beleidigte Smiley
:o	Der erstaunte Smiley
%-	Smiley mit Lachkrampf
:>	Der teuflisch grinsende
:@	Der fluchende Smiley

Charles Moore

- SVFIG 26.7.97 -

Die folgende Rede hielt der Erfinder von Forth, Chuck Moore, am 26. Juli 1997 vor dem Silicon Valley Chapter (örtliche Gruppe) der Forth Interest Group (US-amerikanisches Pendant zur Forth-Gesellschaft). Wir drucken sie ab, da sie einige interessante Gedanken zur Geschichte und Zukunft von Forth enthält, die wir unseren LeserInnen nicht vorenthalten möchten.

Der englische Text wurde von Jeff Fox nach eigenen Mitschriften rekonstruiert und im Juli in comp.lang.forth gepostet. Joachim Merkel (Berlin) und Fred Behringer (München) übertrugen den Text ins Deutsche, die Redaktion führte die Übersetzungen leicht überarbeitet zusammen.

Stichworte: SVFIG Charles_Moore Geschichte OKAD

Am Sonnabend, den 26. Juli 1997 veranstaltete Chuck Moore [Charles Moore] vor der örtlichen Gruppe Silicon Valley der Forth Interest Group eine Vorführung. Vor seiner Vorführung wurde ihm eine Auszeichnung in Würdigung der Entwicklung des ursprünglichen Forth verliehen. Chuck wurde gebeten, den zeitlichen Ablauf der Dinge aufzuhellen. Seine Antwort: Alles fing 1968 mit dem ersten Forth bei Mohasco [eine Firma, für die C.M. damals arbeitete] an. Dann kam Forth bei NRAO [Amerikanische Radioastronomische Gesellschaft] 1970. Forth Inc. [eine der Firmen C.M.s] wurde 1973 gegründet, und jetzt, da wir bald 1998 schreiben und er 1938 geboren ist, sei zu vermelden, daß er schon fast die Hälfte seines Lebens mit Forth gearbeitet hat.

Ich habe das aufgeschrieben, damit mehr Leute mit den vielen Ideen, die Chuck Moore vortrug, bekannt werden. Das ist keine offizielle Mitschrift, da ich einiges von dem, was er sagte, verpaßte. Meine Anstrengungen, einen Mitschnitt zu bekommen, scheiterten an technischen Problemen. In der eigentlichen Rede waren einige Gedanken etwas besser mit dem Material verknüpft, das mir entging. Ich habe Fragen und Diskussionen mit aufgenommen. Mögen andere, die dort waren, das ergänzen. Ich habe auch eine Kopie ins Internet unter http:/www.dnai.com/~jfox/color4th.html plaziert.

Jeff Fox 27.7.97

Charles Moore:

Da bin ich wieder.

OKAD war ein Fehler. Ich habe dem OKAD Forth hinzugefügt; in der zweckmäßigsten Weise, aber nicht in der besten. Forth läuft unter OKAD, und ich werde dann wohl OKAD unter Forth zum Laufen bringen.

Vor ein paar Jahren habe ich Forth aufgegeben. Es war zu kompliziert. Es war ungefähr die Zeit der Arbeit des Standardisierungs-Komitees und ich wollte etwas Einfaches. Ich schaffte den Quellcode ab und benutzte einen Decompiler, um mir das Maschinensprache-Programm [object code] anzusehen. Es klappte nicht wirklich. Ich wollte auch von der Tastatur wegkommen.

OKAD war Code für den Pentium mit einem 7-Tasten-Interface. Ich halte es immer noch für den besten Weg, Layouts zu erstellen, für alles andere taugt es aber nicht. [OKAD war ein Cad-Programm zur Entwicklung von Chip-Layouts, die C.M. bearbeitete. Der Name ist eine Zusammensetzung von OK, also dem Wort, das Forth - jedenfalls bei Moore - immer ausgibt, und CAD. OKAD setzt rein menügesteuert alle Befehle oder Worte zu einem Programm zusammen, es gibt keine Kommandozeile.] Das Problem mit den Menüs ist das Skript. Es ist nicht leicht, Skripts zu erstellen, die Menüs steuern sollen. Das wurde mir

klar, als ich Chip-Layouts testete. Ich habe eine ganze Menge Tests durchzuführen und es war schwierig, sie automatisch zusammenzustellen oder zu wiederholen.

Jahrelang habe ich gesagt, „Die Landkarte ist nicht das Gebiet". Die Beschreibung eines Programms ist nicht das Programm, die Beschreibung einer Testreihe ist nicht die Testreihe, die Beschreibung eines Chips ist nicht der Chip. Ich habe mich da etwas geirrt. Die Karte ist eine bessere Art der Darstellung.

Ich habe niemals behauptet, immer recht zu haben oder mich nie in Widersprüche zu verwickeln. *(Lachen)* Ich hoffe, ich werde weiterhin lernfähig bleiben.

Davon ausgehend, daß ich nicht mit Maschinencode hantiere, brauche ich mehr als nur 7 Tasten. Ich brauche eine standardmäßige Tastatur mit 83 Tasten. Für meine Quelltexte brauche ich nicht alle 101 Tasten. Nach 30 Jahren halte ich Forth immer noch für das beste Verfahren.

Gibt es etwas Einfacheres?

Vor allem hat es einen Daten-Stack. Forth ist die einzige Sprache, die einen getrennten Daten-Stack hat. Dann hat es Colon-Definitionen. `: XYZ ;` ist eine schöne und einfache Konstruktion. Gibt es etwas Einfacheres? Da steckt alles drin, was man zur Modularisierung braucht. [„It forms the entire metaphoric background for factoring."]

Es gibt da ein Problem, das ich zuerst bei iTV gewahr wurde [Firma, die mit dem i21 ein Ein-Chip-Projekt mit sehr leistungsfähiger Grafik verfolgt]. Es war das erste Mal, daß ich mit einem Forth-Projekt zu tun hatte, bei dem ich nicht selbst mitwirkte. Ich sah, wie andere Programmierer Forth einsetzten. Die bekommen das nicht immer ganz hin.

Neben Stacks und Colon-Definitionen gibt es da noch etwas anderes. Man müßte nochmal ein Buch über das Schreiben guter Programme schreiben.

Perfektionismus

In meinem Fall geht alles auf meinen Vater zurück. Dort, wo ich aufwuchs, kannten wir die Jahreszeit Herbst. Im Herbst hatte ich unter anderem die Aufgabe, das Laub aus der Hecke zu holen. Meine Hände waren klein und ich konnte dahin gelangen, wo er nicht hinkam. Ich sollte alle Blätter einsammeln. Das war unmöglich, da ja der Vorrat an Blättern schier unendlich war. Ich entwikkelte eine Strategie, die man unter dem Namen Perfektionismus kennt. Sie ist nicht sehr beliebt, aber bei einigen Dingen führt sie doch zum Erfolg, so zum Beispiel beim Programmieren.

Das beste Forth, das ich auf die Beine gestellt habe, war cmForth. Ich betrachte es aber als zu komplex. Ich möchte darüber sprechen, wie Forth sein sollte.

(Chuck zeigt eine Overhead-Projektion mit einem Bildschirmauszug der OKAD-Diagnose mit vielen Leitungen.)

Forth als Skriptsprache

Hier ist der Grund, weshalb ich wieder zu Forth zurückkehren wollte. Ich verwende Forth als Skript-Sprache zum Austesten von Chips mit OKAD. Die Möglichkeit, mehrere Signale gleichzeitig anzuzeigen, ist neu und äußerst wirkungsvoll. Hier sehen sie Leitungsführungen, diese Punkte zeigen den Zustand der einzelnen Netze auf dem Chip an und diese Linien sind Signalleitungen. Das hier ist eine Hard-Copy vom Bildschirm in OKAD. Ich habe OKAD um „print screen" erweitert. Ich habe die Farbe des Hintergrunds invertiert und einige andere Veränderungen vorgenommen. Das ist nicht das PrtSc von Windows, es ist mein eigenes. Da steckt ein bißchen Arbeit drin. Ich habe jetzt die Möglichkeit, einen Bericht zusammenstellen, in dem steht, welche Tests durchgeführt wurden und mit welchen Ergebnissen für welchen Chip, vorausgesetzt, ich kann den Test irgendwie beschreiben (Forth).

Was meine ich mit Forth? Hier ist ein Programmausschnitt für den i21 [soll als 'Internet-Chip' kostengünstige Netzanbindung ermöglichen].

(Chuck zeigt eine weitere Overhead-Projektion mit einem anderen Bildschirm-Abzug von OKAD.)

Beachten Sie die großen Buchstaben und daß das ein 256-Byte-Block ist. Ein 1024-Byte-Block wäre zu groß, um auf den Bildschirm zu passen. Ich habe vier i21-Compiler geschrieben. Das hier ist ein Quelltext-Assembler von 1K Länge. Er gefällt mir gut. Es ist der kleinste und es scheint, daß man in 256-Byte-Blökken mehr unterbringen kann als in 1024-Byte-Blöcken.

(Chuck zeigte einen weiteren Bildschirm-Abzug.)

Das da oben ein Kommentar. Der Kommentar endet mit einem einfachen Anführungsstrich [Hochkomma]. Ich verwende meinen eigenen 0-Z-Zeichensatz. Es ist ein 6-bit-Zeichensatz. Er enthält nur Großbuchstaben und zwischen dem Buchstaben „O" und der Null wird nicht unterschieden. In den Menüs war das kein Problem, aber jetzt macht es mir im Quelltext zu schaffen. Das Wort `COUNTER'` ist ein Kommentar. Das Wort `EMPTY` ist wohlbekannt. `11 LOAD` ist der Befehl, mit dem der „Screen 11" geladen wird. Dann sehen Sie `: G0 C0 ON G G G G G G G G G G PRINT ;` Zehn „G"s sind einfacher als eine Schleife.

„`Smudge`" und „`unsmudge`" werden bei mir zur Wortdefinition nicht verwendet. Sie müssen das Wort unter einem neuen Namen nochmal definieren. Definieren Sie ein Wort nochmal, dann ruft es sich selbst rekursiv auf. Das ist die einfachste Art, sowas zu erreichen.

Kommandozeilen-Interpreter überholt

Ein Problem ist, daß Forth als Kommandozeilen-Interpreter betrachtet wird. Diese Auffassung ist überholt. Ich verwende das Leerzeichen, um das Ende anzuzeigen, und die Worte werden sofort ausgeführt. CR zur Kennzeichnung des Zeilenendes gibt es nicht. Kein CR, nur Zwischenräume als Trennzeichen. Zur Zeit verwende ich BS und Delete. Ich möchte nur zwei Sondertasten behalten, BS und eine „Exit"-Taste. Ich werde wohl das BS dem Delete vorziehen.

Hier wird ein Ganzseiten-Editor verwendet. Ich bin nicht sicher, ob ich einen Ganzseiten-Editor brauche. Vielleicht werde ich den alten 'Find-and-Replace'-Editor aus der Forth-Inc.-Zeit verwenden.

Ich verwende nur ganz wenige Befehle auf dem Pentium, um i21-Befehle zu implementieren. Damit wird Portierbarkeit zum i21 erreicht.

Im allgemeinen macht es einem nicht allzuviel aus, ganze Blökke von Zeichen vor sich zu haben, sogar ohne Zeilenumbruch. (Er bezieht sich auf das Aussehen des Textes in dem Block ohne Trennung durch CRs.) Ideal ist das jedoch nicht.

Ich rate den Leuten, ihr eigenes Forth zu schreiben. Standardisierung heißt nicht, daß Sie nichts erfinden können. Ich schlage hier den Einsatz von Farbe vor.

In `: GO` erscheint der durch das Colon (Doppelpunkt) definierte Name in Rot, so daß wir kein Colon brauchen. Der Hauptteil der Colon-Definition ist in anderer Farbe gehalten. Ob compilierend oder interpretierend, kann man aus der Farbe erkennen. An Stelle von `COMPILE` geben Sie einfach die Farbe an. Machen Sie ein Wort über die Farbe zu einem `IMMEDIATE`-Wort.

Farbe als Trennzeichen

Die Farbe wird durch ein Farbsetzzeichen gesetzt, das wie ein Leerzeichen aussieht. Der Kommentar hat eine andere Farbe. Andere Trennzeichen als Farbe und Leerzeichen werden nicht benötigt. Dezimalzahlen können von Hexadezimalzahlen über ihre Farbe unterschieden werden. Und auf dem i21 verwenden wir auch noch eine Hexadezimaldarstellung, die mit AAAAA ge'xor't wird, und die hat eine andere Farbe.

Wie viele Farben sind sinnvoll? Dinge wie `DECIMAL` innerhalb und außerhalb einer Definition wirken auf Anfänger verwirrend.

Zielcompilierung [target compiling] ist nicht das Ziel [the target]. (Es handelt sich um ein interaktives Forth.) Ich kann Colon [Doppelpunkt], Semikolon, Klammern, Zahlenbasis und kompilierende Worte weglassen. Ich habe das noch nicht getan. (Wie Sie sehen, sind Colon und Semikolon noch im Druckauszug.)

Keine gute Schleifenkonstruktion

Leute, die Forth nicht mögen, könnten sich auf folgendes versteifen: In 20 Programmabschnitten habe ich keine bedingten Sprünge oder Schleifen. Gutes Forth minimiert die Anzahl der bedingten Sprünge. Das Minimum ist Null. Beim i21 gibt es keine gute Schleifenkonstruktion. Ich kann `: 5X X X X X X ;` `: 20X 5X 5X 5X 5X ;` sagen. Das ist genauso gut wie eine Schleife. Wenn ein Programmteil den Speicher durchläuft, sollte es lieber eine Aussprungbedingung abfragen als einen Schleifenzähler. Wenn ich bedingte Sprünge brauche, würde ich gern Wil Badens Flußdiagramme verwenden.

OKAD ist in Assembler geschrieben. Es ist einfach nur eine Binärcode-Umsetzung. Forth ist ein Batzen OKAD. OKAD enthält eine Symbol-Tabelle, und die wird von Forth verwendet. Der Compiler compiliert Code für den Pentium und ist unterprogrammgefädelt. `+ AND OR 2* 2/` werden direkt eingebaut, andere Dinge in Form von Unterprogrammaufrufen. Er läuft schnell. Er compiliert schnell. Bei meinen kleinen Programmen ist von Compilieren überhaupt nichts zu merken. Man lädt einen Block und schon sieht man die Definitionen vor sich, noch ehe man die Finger von den Tasten genommen hat. Man braucht gar nicht zu wissen, daß da ein Compiler am Werk ist, nur daß es da einen Stack und Colon-Definitionen gibt.

In fünf bis zehn Minuten sollte man diese Sprache erlernen können.

In 30 Jahren hat sich viel verändert. Einerseits sind da die riesigen Mengen an Arbeitsspeicher. Computer mit kleinem Speicher können Sie gar nicht mehr kaufen. Die Blöcke werden also im RAM gehalten. Ich setze 100 Block-Puffer zu je 256 Byte ein. Wenn ich den Computer einschalte, liest mir DOS das vom Laufwerk ein. Während des Programmlaufs brauche ich keinen Zugriff auf das Laufwerk. Zudem haben wir ja jetzt das Internet. Es gibt keinen Grund, Computer nicht ans Internet anzuschließen. Und wenn der Computer mit dem Internet verbunden ist, braucht er kein Laufwerk. Man liest einfach alles über den Server ein. Server und Tastatur, größer braucht der Computer nicht zu sein.

Kürzlich sah ich einen Artikel über Web-Sprachen. Ich las den Artikel und suchte die Web-Seite auf. Forth war nicht unter den Sprachen. Java ist natürlich nur eine dieser Sprachen unter vielen

und Forth sollte auch dabei sein. Ich war in Tokenized-Forth verliebt. Warum nicht Tokenized-Forth, so wie Java? Dazu wäre aber ein Standardisierungs-Komitee nötig.

Vorbehalte gegen ANSI

Ich hatte Vorbehalte gegen ANSI [American National Institute for Programming Systems Languages] . Ich hatte Angst, das Ganze werde nicht nur zu einer Verbesserung von zweifelhaftem Charakter, sondern zu einem kompletten Reinfall. Meine Ängste wurden wahr und kein Vorteil war zu sehen. Jeder Erneuerungsansatz war im Keim erstickt worden. Untergrund-Forthe sind immer noch nötig. Ich sagte denen, ich dachte, die Standardisierung sollte sich auf die Veröffentlichung von Forth beziehen [„publication standard"]. Was die aber wollten, war eine Standardisierung der Ausführung [„execution standard"].

Quelltext austauschen

Im Netz haben wir keine Zeit, einen Standard für Netz-Forth hochzuziehen. So, das bedeutet also Quelltext. Meiner wird farbabhängige Worte verwenden.

Es sind mehrere Bestandteile, die Forth ausmachen. Das große Geheimnis dabei ist, warum es überhaupt so gut funktioniert. Man könnte als Erklärung anführen, daß es mit logarithmischem statt mit linearem Wachstum zu tun hat. Und eine Modularisierung, die von der Sprache gefördert wird, lädt zu einer Modularisierung des zugrunde liegenden Problems ein.

Was ausgetauscht werden soll, sind Ideen, nicht Programme. Sage mir, wie Du Deinen Heap-Sort angepackt hast, und behalte Dein Programm für Dich. Ich hatte immer etwas gegen die Veröffentlichung von seitenlangen Programmen in der Forth Dimensions, wo meist ein kleiner Teil schon den Kerngedanken enthielt. Es macht es mir nur schwerer zu finden, was ich eigentlich wissen will. Ich werde Ihr Programm sowieso nicht verwenden. Ich schreibe meinen eigenen Code.

Problematisch ist die Überführung eines Teils von OKAD von einer menügesteuerten zu einer kommandozeilen-orientierten Fassung. Ich schätze, etwa die Hälfte von OKAD befaßt sich mit Bildschirmanzeigen. Windows ist zum größten Teil damit beschäftigt, dem Anwender Ein- und Ausgaben sichtbar zu machen. Bei einer Kommandozeilen-Fassung gäbe es das nicht.

Internet

Über das Internet haben Sie ganz einfachen Zugriff auf die Programme anderer Leute. Sie können Programmteile gemeinsam verwenden, wenngleich ich selbst das nicht mache. Programmteile können Sie gemeinsam verwenden, wenn Sie beide einen ähnlichen Quelltext haben. Quelltext besteht aus Zeichen. Meine Zeichen sind keine Ascii-Zeichen. Ich verwende bunte Trennzeichen. Ich könnte auch Ascii verwenden, aber ich mag den 0-Z-Zeichensatz zu 6-Bit. Ob die NSA [eine Art Geheimdienst in den USA] wohl je dahinterkommt, daß sie meinen Text nicht zu 8-Bit-Zeichen decodieren kann? Kein Ascii verwenden, heißt Kompression. Auch eine Art von Verschlüsselung. Ich weiß, daß E-Mail-Protokolle 7-Bit- statt 8-Bit-Zeichen verwenden.

Ich möchte mehrfache Eintrittspunkte für Wort-Definition, damit ich kein Semikolon benötige. Ich möchte kein Semikolon haben müssen, um das Kompilieren zu stoppen. Das läßt sich ganz leicht so einrichten.

Die Idee eines Forth, ob alt oder neu, das alles kann, mag ich nicht.

Für's Netz brauchen Sie eine Anbindung. Ich weiß nicht, ob man wirklich ein vollständiges TCP/IP-Protokoll braucht, nur um etwas herunterzuladen und zu plaudern.

Der Vorteil der Konfiguration des [Net-] Browers ist praktisch gleich Null. Nur bei Netscape hilft das. Ich kann cookies zwar nicht vollständig ausschalten, aber ich kann sie melden lassen und aufhalten, wenn sie angefordert werden. Die Auswahl am Netz ist viel größer als mir bewußt ist. Da gibt es Massen zum Herunterladen und ich tue es nicht. Sie können leicht eigene Software schreiben, wenn Sie die Software anderer Leute einfach nutzen können.

Betriebssystem ist ein Unding

Mit der Software, mit der ich mich herumschlagen muß, bin ich höchst unzufrieden. Windows übersteigt jedes Fassungsvermögen! UNIX ist nicht besser. DOS auch nicht. Es gibt keinen vernünftigen Grund für ein Betriebssystem. Es ist ein Unding. Vielleicht brauchte man es früher einmal.

Netscape ist mir zuwider. Ich bin zum Internet Explorer übergegangen, obwohl mir MicroSoft noch mehr zuwider ist als Netscape. MASM ist mir zuwider. Ich habe herausbekommen, daß MASM meinen reservierten Speicherplatz angreift. MASM sagt zwar, das tue es nicht, das tut es aber. Ich habe Word, WordPad und Edit ausprobiert. Sie sind unbrauchbar.

Ich hätte gerne eine Floppy, ein Megabyte wäre groß genug. Da wäre ein 10Kb-Programm drauf, und der Rest bleibt für Daten. Das kommt beim Booten in den Arbeitsspeicher und ist betriebsbereit, während der Rest in den Speicher geladen wird. Es sollte in 1-2 Sekunden starten. Sie wollen es ja gar nicht auf ihre Festplatte tun und keine Verwandlung [Ihres Systems] in eine Spezialmaschine.

Bei den Betriebssystem-Herstellern hat sich im Laufe der Zeit nichts geändert. Man braucht das nicht mehr, aber sie stehen sich ganz gut damit.

Ich schlage den freien Zugriff aufs Internet vor. Das könnte sich aber als illegal herausstellen. Meine Zeichen sind 6-Bit breit, also sogar noch kompakter als 8-Bit-Token. Ich würde gern nach eigenem Gusto im Internet surfen.

Es gibt Stimmen gegen Standard-Forth. Das Wort `WORD` ist ein Fluch. Man sollte es abschaffen. Das Wort `WORDS` ist genauso schlecht. Auf einen Anfänger wirkt es einfach verwirrend, wenn er eine lange Liste von Worten sieht, die er nicht versteht.

Was ich brauche, ist ein Browser, ein Text-Programm, das zur Erstellung kurzer E-Mail Nachrichten geeignet ist, und OKAD. Ich will keine Spiele und keine Tabellenkalkulation. Falls ich Spiele will, lege ich sie auf einen anderen Rechner.

Spaß haben klappt immer

Mit einem Computer können Sie drei Dinge tun. Sie können versuchen, Geld zu verdienen, was kaum möglich ist. Sie können versuchen, berühmt zu werden, was aber nie vorkommt. Und sie können einfach nur Spaß daran haben. Das klappt immer. Obwohl es für mich zeitweise eher Arbeit als Spaß wurde.

Es ist nicht mein Geschäft, für Forth Reklame zu machen. Ich mache keine Reklame für das, was ich hier tue. Ich berichte einfach nur darüber. Ich weiß, daß Forth eine bessere Lösung ist.

Mir bleiben noch zwanzig Jahre. Die ganze Welt war auf hirnlose Pull-Down-Menüs aus. Also habe ich es auch versucht. Es geht nicht. Das führt uns wieder auf „die Landkarte, die nicht das Gebiet selbst ist" zurück. Ich muß Test-Scripts in OKAD zusammenstellen, also verwende ich Forth.

Wenn mir die Beschäftigung ausgeht, werde ich mich der Spracherkennung zuwenden. Das Problem ist nur, daß ich meinem Computer nichts zumurmeln möchte, wenn jemand in der Nähe ist. Ich brauche eine Tastatur.

Die Auffassung, man brauche zum Programmieren eine besondere Ausbildung, ist falsch. Sowas vertritt nur die Priesterschaft.

Vor einem Jahr habe ich jeden ermutigt, sein eigenes OKAD zu schreiben. Ihr habt das nicht getan, also fühle ich mich auch nicht schuldig. Ich versichere Euch, daß ich jetzt auf dem rechten Weg bin, und empfehle Euch, experimentiert mit Farb-Forth.

Jeff Fox, jfox@dnai.com, http://www.dnai.com/~jfox/

Das Wort Doppelcolon aus Turbo-Forth

von Fred Behringer
Planegger Str. 24; D-81241 München

In der internationalen News-Group comp.lang.forth ist eine Diskussion über Lieblingsworte (pet words) im Gange. Das sind Worte, die in keinem Standardverzeichnis stehen, die aber für viele Anwender nützlich sind oder nützlich sein können.

Stichworte: Turbo-Forth, Lieblingswort, Schwierigkeiten, Lösungsvorschlag

Was mich betrifft, eines meiner Lieblingsworte ist „::“ aus Turbo-Forth. Man kann es beispielsweise aus der Eingabeebene (vom Interpreter) heraus wie folgt verwenden:

```
:: 100 0 DO I . LOOP ;    [RET]
```

Ausgegebenen werden die Zahlen 0 bis 99, simuliert interpretativ, ohne dass im Dictionary Wortrückstände bleiben.

Wirkungsweise von „::“

Das Wort „::“ ist in Turbo-Forth wie folgt definiert:

```
: ::
     HIDE HERE >R
     [ ' : @ ] LITERAL ,
     !CSP ]
     R@ EXECUTE
     R> DP ! ;
```

Innerhalb der Klammern „[„ und „]“ wird der Inhalt der cfa von „:“ eingeholt. Im anschließenden „LITERAL ,“ wird er in das aufzubauende Wort „::“ compiliert. Damit steht bei Aufruf von „::“ die Adresse von NEST zur Verfügung. (Diese Adresse kann man in Turbo-Forth (und in anderen F83-basierten Systemen) nur während der Metacompilation erreichen, später nicht mehr.) Das Wort „::“ leitet also bei Aufruf den NEST-Prozess aus „:“ ein, ohne für „::“ einen „Header“ CREATEt zu haben. Der Wert von HERE wird zwischengespeichert und dient nach Abarbeitung der Konstruktion mit „::“ dazu, DP wieder auf ebendiesen Wert zu setzen. Dadurch wird die auf „::“ folgende Wortkette (nach Abarbeitung per EXECUTE) schließlich wieder aus dem Dictionary entfernt. Abgeschlossen wird das Zwischenspiel über ein „normales“ „;“. In „;“ ist aber REVEAL enthalten. REVEAL würde das Forth-System restlos durcheinanderbringen, wenn es sich nicht auf ein vorausgegangenes HIDE beziehen könnte. Das ist die Aufgabe von HIDE im Wort „::“. Auf was bezieht sich nun aber HIDE ? Auf einen Header nicht. Es wurde ja in „::“ gar keiner erzeugt. Auf das in LAST festgehaltene jüngstdefinierte Wort. Das aber birgt zwei Schwierigkeiten in sich. Beide Schwierigkeiten könnte man durch Neufassung der auf Laxen and Perry in F83 zurückgehenden Implementation von HIDE , REVEAL und FORGET ausräumen. Ich möchte nicht zu viel ändern und schlage stattdessen eine Neufassung von „::“ aus Turbo-Forth vor. Zunächst ein paar Worte zu den Schwierigkeiten.

Zwei mögliche Schwierigkeiten

(1) Wenn alles gut geht, ist die Welt in Ordnung. Was aber, wenn die Konstruktion :: ... ; ein nichtexistierendes Wort enthält? Dann bleibt das jüngstdefinierte Wort ausgeklinkt, da ja auf das einleitende HIDE kein auflösendes REVEAL mehr folgt. Es kann also passieren, dass das Wort „::“ an der Substanz des Forth-Systems nagt.

(2) Hat man andererseits im Hauruck-Verfahren ein fiktives Wort XXX (nur Header, kein Body) erzeugt, um „Luft zu schaffen“, und das dann gleich wieder per FORGET XXX beseitigt, dann läuft alles gut, solange man nicht zwischen FORGET XXX und der Verwendung von „::“ eine Operation durchgeführt hat, die den Platz hinter HERE in Anspruch nimmt und bis zur Linkadresse des (an sich ja schon beseitigten) Wortes XXX reicht. Das hängt damit zusammen, dass FORGET in F83-basierten Systemen den Wert der Variablen LAST nicht anpasst (nicht zurücksetzt).

Meine Lösung

Ich führe ein Hilfswort XXX ein, das keine andere Funktion hat als die, geeignete Linkadressen für HIDE und REVEAL zu liefern. XXX wird sofort nach Abarbeiten der in „::" enthaltenen Worte per FORGET XXX wieder beseitigt. Die ebengenannte Schwierigkeit (1) kann nicht auftreten, da bei Fehlerausstieg aus „::" das Wort XXX , das über HIDE ausgeschaltet wurde, einfach nur ausgeschaltet bleibt, was über FORGET XXX ja sowieso hätte geschehen sollen. Schwierigkeit (2) kann aber auch nicht auftreten, da ja die lfa von XXX zur Zeit des Abarbeitens der auf „::" folgenden Worte effektiv (unüberschrieben) zur Verfügung steht.

Neudefinition von „::"

Es folgt das kommentierte Listing, aus welchem die Funktionsweise des neudefinierten Wortes „::" hervorgeht. Ich verwende „::" in der zur Zeit in Bearbeitung befindlichen Version F-TP 1.10 meines Forth-Systems für Transputer.

```
:  ::  WARNING @              \ Zur Wiederherstellung
   WARNING OFF                \ Falls XXX schon vorkommt
   " : XXX ; " $EXECUTE       \ Dummy-Wort erschaffen
   WARNING !                  \ Wiederhergestellt
   HIDE                       \ XXX ausklinken
   HERE >R                    \ Anfangsadresse
                              \ der NEST-Kette
   [ ' : @ ]                  \ Adresse von NEST
   LITERAL ,                  \ in :: hineincompilieren
   !CSP                       \ Stack absichern und in
   ]                          \ Compile-Zustand gehen
   R@ EXECUTE                 \ NEST-Kette ausführen
   R> DP !                    \ Kette beseitigen
   " FORGET XXX " $EXECUTE ;  \ XXX beseitigen
```

Bemerkenswert ist das Wort $EXECUTE

Marc Petremann sagt im KERNEL seines Turbo-Forth-Systems: $EXECUTE ist ein mächtiges Hilfsmittel zur Ausnutzung der Fähigkeit von Forth, gleichzeitig interpretierend wie auch compilierend zu wirken. Dem möchte ich beipflichten. Es gestattet eine „Laufzeitcompilation". Es gestattet, den Compilationsvorgang oder Teile daraus in die Laufzeitebene zu verlagern. Der bedingten Compilation sind damit alle Möglichkeiten eröffnet. Man kann den Compiliervorgang unterbrechen, das Compilat ausführen und in Abhängigkeit von bestimmten Laufzeitergebnissen den Compiliervorgang fortsetzen oder zumindest ergänzen, ohne dabei die Laufzeitumgebung (das Programm) zu verlassen. Sicher geht das auch anders. Aber mit $EXECUTE geht es wunderbar einfach und durchsichtig.

Das werfen die anderen den Forthlern vor

„Warum wird Forth in der Programmierfachwelt nicht akzeptiert und warum zieht kein Lehrbuch der Programmiersprachen auch nur in Erwägung, Forth zu erwähnen?", fragte Wolf Wejgaard in seinem interessanten Vortrag „The Invisible Language" auf der Forth-Tagung 1997 in Ludwigshafen. Die pure Möglichkeit, die Erschaffung von Worten wie $EXECUTE dem „Anwender" in den Schoß zu legen, scheint mir eine Antwort darauf zu sein, eine unter anderen. Compilier- und Ausführungsvorgänge werden verwischt. Zumindest fördert Forth die Möglichkeit der Verwischung. Vielen suspekt. Jawoll! Zu Recht. Struktur muss sein, um den Durchblick und den Überblick nicht zu verlieren. Aber kann man die nötige Strukturiertheit nicht auch auf höhere Ebene heben? Eine Sprache, die sich im Verlaufe ihrer Verwendung selbst verändern darf? Code, der sich korrigiert? In den natürlichen Sprachen sind da keine Grenzen gesetzt. Sprache und Sprache zur Beschreibung dieser Sprache (Metasprache) werden, nicht immer sofort unterscheidbar, miteinander vermengt, auch in der naturwissenschaftlichen Literatur, ohne dass die Verständlichkeit unbedingt und immer darunter leidet. Mir machen solche Dinge wie „::" und $EXECUTE , aber auch unerwartete Colon-Definitionen in Code-Definitionen wie : NEXT >NEXT #) JMP ; (siehe Beitrag „Grafik 'ohne Ende'" von Friederich Prinz in VD 1/97), Spaß.

Kontinuierliches Booten oder das Münchhausen-Prinzip in Forth

Eine Sprache, die sich selbst zu beschreiben gestattet, ist vergleichbar mit einer Menge, die sich als Element selbst enthält. An sich noch kein Unding. Ein Inhaltsverzeichnis darf sich natürlich selbst mit aufführen und es wäre übertrieben, zwischen „Inhaltsverzeichnis" und „Meta-Inhaltsverzeichnis" zu unterscheiden. Verboten werden müssen aber, oder milder ausgedrückt, nicht zugelassen werden dürfen solche widerspruchsbehafteten Begriffe, solche Unbegriffe, wie die Menge aller Mengen, die sich selbst (als Element) nicht enthalten. Ein Compiler, der sich selbst compiliert, in Forth nichts Überraschendes, ist das nun schon ein Unbegriff, der verboten werden sollte? Ein Compiler, der sich am eigenen Schopf aus dem Sumpf (des Forth-Urwortes - welches ist das?) zieht. Das wäre eine Verkörperung des Prinzips aller Bootvorgänge schlechthin. Erst stufenweise und dann, wie in der Zinseszinsrechnung bei der e-Funktion, zur Grenze übergehend. Ein kontinuierlicher Bootvorgang. In Forth ist vieles machbar, was in anderen Sprachen Kopfschütteln hervorruft. In einem solchen Zusammenhang taucht dann auch sofort die Frage nach einem „minimalen Forth", was auch immer das heißen soll, ein erster Schritt zur Forth-Ursuppe, aus der heraus kontinuierlich gebootet werden soll, auf. Auf meine diesbezügliche Leserbrieffrage in der VD 3/95 wurde im Vortrag „Gforth EC - Minimal Forth auf Minimal Instruction Set Computer" auf der Forth-Tagung in Ludwigshafen von Bernd Paysan und Jens Wilke eine glänzende konstruktive erste Antwort gegeben.

❑

Die Entwicklung von HolonForth - Teil 1

Wie bringt man Ordnung in ein Programm?

von Wolf Wejgaard
Forth Engineering, CH-6045 Meggen <wejgaard@acm.org>
+41 (0)41 377 3774 Fax ..4774

Wenn Sie Forth lieben und dennoch das Gefühl nicht loswerden, dass das Programmieren in Forth noch ein bisschen einfacher sein sollte, dass die vielen Worte geordneter und der Zugriff auf jedes Teil direkter sein sollte, dann habe ich eine gute Nachricht für Sie. Mit einer kleinen aber zentralen Korrektur können wir den Umgang mit Forth deutlich klarer und einfacher gestalten

Stichworte: Forthsysteme Holon Dictionary Direktheit Browser Geschichte

1985 wusste ich endgültig und ohne jeden Zweifel, dass Forth die ideale Programmiersprache ist. Ich hatte fünf Jahre lang mit Forth verschiedenartige Projekte durchgeführt, und Forth auch immer wieder mit den Alternativen verglichen (C, Pascal, Modula-2, Smalltalk, u. a.). Forth blieb die beste Wahl. Doch vollkommen glücklich war ich nicht. Irgendwie war Forth noch nicht richtig organisiert. Ich überlegte, wie das ideale Forth aussehen könnte, und damit begann die Entwicklung von HolonForth (kurz Holon) [1].

Das Resultat sehen Sie in Figur 1. Holon hat eine neue Bedienungsoberfläche. Das „OK" ist ersetzt durch einen Browser, der immer das ganze Programm zeigt. Das Programm ist geordnet in Module, Gruppen und Worte. Ich habe so den vollen Überblick und kann doch auf jedes einzelne Wort direkt zugreifen. Ich kann die Worte direkt im Browser testen und das Resultat im Stackfenster sehen. Ich kann die Worte einzeln ändern und die Korrektur unmittelbar testen. Und mit dem Debugger kann ich den Programmablauf im Einzelschritt durch den Quelltext im Browser verfolgen. Sie werden verstehen, dass ein solches Entwicklungssystem nicht nur Spass macht, sondern auch ein hocheffektives Arbeiten erlaubt. Es ist ein neues Forth-Gefühl, das Sie selbst auch erleben können. [2]

Ich beschreibe hier, wie ich zu Holon gekommen bin. Es begann mit der Frage, wie man am besten mit dem Programmtext umgeht.

Ordnung in das Programm bringen

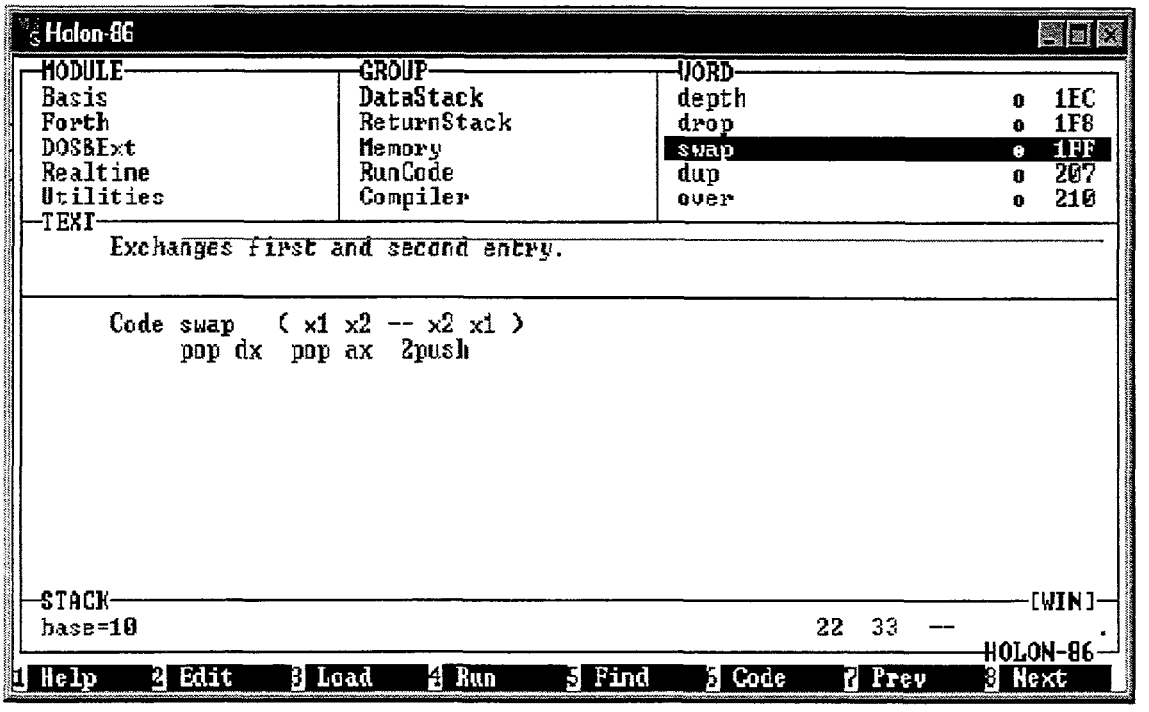
Ein Forth-Programm wird formuliert als Text auf Textdateien oder in Textblöcken. Dateien und Blöcke bilden eine grobe Ordnung, aber sie geben keine Übersicht über die Worte. Wenn ich die Worte ordnen will, muss ich sie in ein Glossar schreiben. Das ist ein mühsamer Zusatzaufwand, für den ich beim Entwickeln keine Zeit fand. Ich wünschte mir, dass das Forth-System automatisch die Worte ordnet wie in einem Glossar.

Wie der Bildschirm aussehen könnte, war mir relativ klar, nachdem ich Smalltalk gesehen hatte. Mein Forth-System zeigte einen leeren Bildschirm mit einem freundlichen(?) „OK". Dahinter war alles verborgen und ich musste die Worte auswendig kennen oder aus der kryptischen WORDS-Liste herauslesen. Oder im Glossar nachschauen, falls vorhanden. Das Smalltalk-System dagegen besass einen Browser, der pausenlos das ganze Programm zeigt. In den verschiedenen Fenstern des Browsers sieht man Listen von Klassen und Methoden und anderen Programmelementen, und ein Editierfenster präsentiert den Text des aktuellen Elementes. Das war's! So ähnlich sollte sich doch auch ein Forth-Programm darstellen lassen.

Smalltalk ist objektorientiert, und ein objektorientiertes System ist bekanntlich von Haus aus geordnet in eine Klassenhierarchie. Hier ist also das Problem gelöst, wie man Ordnung in das Programm bekommt. Leider ist das nicht die Ordnung, die ich suche. Ich schätze Objekte sehr und verwende sie auch in Holon, doch die Klassenhierarchie ist für mich eine sekundäre Ordnung, die nur in speziellen Fällen wirklich sinnvoll ist. Ich halte das Forth-Wort weiterhin für die Grundeinheit von Programmen, und einige Worte werden halt am besten als Objekte definiert.

Es gab bis zur augenblicklichen Blüte der objektorientierten Systeme etliche Versuche, prozedurale Sprachen zu ordnen. Modula-2 hat wohl den tiefsten Eindruck hinterlassen und auch viele Forth-Programmierer angeregt, Module für Forth zu erfinden, mit eleganten Importmechanismen und eventuell sogar Exportbeschränkungen. Ich habe mehrere Erfahrungsberichte gelesen, die alle meine eigenen negativen Resultate bestätigen. Das Verstecken der Worte in Modulen löst das Problem der Ordnung nicht und vereinfacht auch nicht das Programmieren. Nur den Begriff „Modul" brauche ich seither für die oberste Strukturstufe des Programms.

In der Praxis entspricht ein Modul dem Inhalt einer Textdatei. Eine Textdatei enthält zu viele Worte, um übersichtlich zu sein. In einem typischen Projekt hatte ich vielleicht 2000 Worte in ca. 20



Dateien, also ca. 100 Worte pro Datei. Beim näheren Hinsehen merkte ich, dass die Worte einer Datei sich gut gruppieren lassen. Ich fand ca. 10 Gruppen mit je ca. 10 Worten. Damit war die Struktur zerlegt in Grössenordnungen, die ich überblicken konnte.

Also war auch die Bedienungsoberfläche klar: Je ein Listenfenster für Module, Gruppen und Worte, und darunter ein Textfenster zum Editieren des aktuellen Wortes, d. h. bis auf das Stack-Fenster genau die Einteilung von Figur 1. Es war ein schöner Traum: Man konnte durch die Listen blättern und zu jedem Modul die zugehörigen Gruppen, und zur aktuellen Gruppe die zugehörigen Worte sehen, und darunter den Text des gewählten Wortes.

Ein Wort pro Block

Wie bringt man den Text einer Datei in einen Browser? Oder wären Blöcke vielleicht besser geeignet? Ich machte ein Experiment und verwendete im damals aktuellen Projekt Textblöcke mit der klaren Regel: Nur ein Wort pro Block! Am Anfang fiel mir das schwer, all der schöne ungenutzte Platz! Aber meine Festplatte störte das nicht, und die Regel zeigte erstaunliche Wirkungen. Plötzlich wurden die Blöcke wertvoller. Es gab genug Raum, um zu jedem Wort einen genügenden Kommentar unterzubringen. Auch war es jetzt einfach, Worte im Programm herum zu schieben, ich musste ja nur den ganzen Block umordnen. Ich schrieb wie üblich den Namen des Wortes auch in die Indexzeile (Zeile 0) und hatte in der Indexliste eine gute Übersicht über das Modul. Ein Modul war eine Datei mit Blöcken. Die Indexliste machte ich interaktiv, so dass ich hin und her blättern konnte, und baute auch die Möglichkeit ein, Blöcke direkt in der Liste zu verschieben. Das heisst, ich baute mir einen Index-Editor, in dem jede Zeile einen Block darstellt, und von dem aus ich auch den Blockeditor rufen konnte. Das war eine kleine Mühe für die positiven Resultate, die ich mit diesem Projekt erlebte. Beim Drucken gab es auch keine Platzverschwendung, denn die Druckroutine ignorierte die leeren Zeilen im Block und auch die Indexzeile. So sah das Listing aus wie bei einer Textdatei, nur stand bei jedem Wort noch die Blocknummer.

Ich hätte den Indexeditor erweitern und auch die Verwaltung der Module da hineinbringen können, aber nun wollte ich den ganzen Schritt tun und den echten Browser realisieren. Die wichtigste Erkenntnis war für mich, dass es sich lohnt, sich auf einzelne Forth-Worte zu konzentrieren. Aber mein Grundproblem war noch nicht gelöst. Wie verwalte ich den Text, damit ich ihn im Browser darstellen kann? Das Blocksystem war ein guter Anfang, aber nicht die ganze Antwort.

Ist ein Programm wirklich ein linearer Text?

Ich hatte den Eindruck, an ein fundamentales Problem zu stoßen. Irgendwie ist die Art, wie wir ein Programm behandeln und den Text bearbeiten, nicht der Natur des Programms angepasst. Irgendwie werden wir einer Grundeigenschaft von Programmen nicht gerecht. Ich ahnte langsam, was das Problem war.

Ein Forth-Programm besteht aus vielen gleichartigen Elementen, den sogenannten Worten. Wie ordnen wir normalerweise eine Sammlung von gleichartigen Elementen, zum Beispiel eine Adressenliste? Schreiben wir die Adressen in eine Textdatei und bearbeiten wir die Liste mit einem Texteditor? Die meisten Leute ziehen wohl eine Datenbank vor, denn damit haben sie ganz andere Möglichkeiten. Sie können direkt auf einzelne Adressen zugreifen und sie bearbeiten, die Adressen leicht ordnen, umstellen, sortieren und filtern.

Warum eigentlich machen wir das gleiche nicht auch mit einem Forth-Programm? Warum werden Computerprogramme, die doch aus vielen gleichartigen Elementen bestehen, als reiner Text behandelt und nicht als eine Datenstruktur?

Es gab schon damals Ausnahmen, sogar in Forth. Click und Snow [3] hatten ein Forth-System namens Fifth entwickelt, das tatsächlich wie eine Datenbank organisiert war; leider haben die Autoren diesen Aspekt von Fifth nicht betont und so hat es keiner gemerkt. Ein anderes Beispiel ist Smalltalk, das den Programmtext ebenfalls als Teil eines Records speichert, der im Workspace liegt. Beides habe ich aber erst später verstanden. Damals ahnte ich es, heute weiß ich aus der Erfahrung mit HolonForth, dass ein Programm tatsächlich eine Datenbank ist und dass es auch von Anfang an als Datenbank behandelt werden sollte.

Ein Programm ist eine Datenbank

Sie sind immer noch skeptisch und finden, es bestehe ein Unterschied zwischen einer Adresse und einem Forth-Wort? Eine Adresse hat ja verschiedene Teile (Name, Strasse, Ort, u. a.), die in der Datenbank in verschiedenen Feldern erscheinen.. Aber auch ein Programmwort hat verschiedene Teile, die wir in Feldern eingeben könnten. Es gibt mindestens fünf verschiedene Teile in einem Forth-Wort, nämlich der Typ des Wortes (CODE, „:", VARIABLE, u. a.), sein Name, das Stackverhalten (Parameter und Resultate), der Körper (Definition oder Datenwerte), und der Kommentar.

Das war meine Sicht der Dinge vor rund zehn Jahren. Ein Programm zerfällt in Module, Gruppen und in Worte, die als Records einer Datenbank betrachtet werden können. Ich dachte damals daran, den Worttext in einzelne Felder aufzuteilen, also je ein Feld für den Worttyp, den Namen, den Stackkommentar, die Definition und den Wortkommentar einzuführen. Aber das war dann doch zu weit entfernt von der herkömmlichen Forthwelt. Deshalb habe ich mich mit den zwei Feldern für Definition und Kommentar begnügt, die Sie in Figur 1 sehen. Auch das war schon ein Fortschritt, denn da der Kommentar getrennt verwaltet wird, muss ich ihn dem Interpreter gar nicht zeigen und kann daher auf Kommentarzeichen wie „\" am Zeilenanfang verzichten.

Aber wie sollte ich dieses System realisieren? Datenbank und Fensteroberfläche waren ein bisschen viel auf einmal für ein Nebenprojekt. Ich suchte ein Forth-System, das mir die Arbeit mit fertigen Bibliotheksmodulen erleichtern würde, fand HS-FORTH und begann endlich, an Holon zu arbeiten. 1989 existierte Holon als ein funktionsfähiges System, und seither habe ich alle Projekte in Holon entwickelt und daneben an Holon selbst weiter gebaut. Das ging in kleinen Schritten so weiter, bis es mir gelang, Holon von HS-FORTH abzunabeln. Endlich konnte ich die Vorteile von Holon auch für Holon selbst nutzen und das System bekam einige hübsche neue Eigenschaften, wie den mit dem Editor integrierten Debugger.

Dictionary

Holon ist ein Forth-System, das auf einer Datenbank beruht. Text und Code werden in der Datenbank verwaltet. Ich habe Holon 1989 auf der EuroFORML Konferenz vorgestellt und dann, beim Schreiben meines Artikels, merkte ich endlich, dass Forth schon immer eine Datenbank war! Ohne diese Datenbank, dem Dictionary, haben wir kein Forth vor uns, es existiert einfach nicht. Die Neuerung von Holon bestand darin, den Quelltext mit in das Dictionary aufzunehmen und ihn geordnet zu präsentieren.

Laut ANS Forth besteht das Dictionary aus Namen, Code und Daten. Der Programmtext wird nicht erwähnt. ANS Forth lässt die Frage offen, wie das Dictionary organisiert wird. Namen, Code und Daten können ineinander geschachtelt sein. In einigen Forth-Systemen ist die Wortliste getrennt von Code und Daten und enthält für jeden Worteintrag einen Zeiger auf den Code. Außerdem enthalten die Worteinträge zumeist auch einen Zeiger auf den Text des Wortes. Das erlaubt den Zugriff auf die Definition mit dem Befehl SEE (oder LOCATE, VIEW).

Das Dictionary wird im konventionellen Forth vom Compiler gebildet und ist daher erst verfügbar, *wenn das Programm geladen ist*. Und wenn das Programm geladen werden kann, ist es schon recht weit entwickelt. Die sehr nützlichen Textzeiger im Dictionary können wir praktisch erst in der Testphase benutzen und leider nicht schon während des Programmentwurfs.

Warum muss der Compiler den Eintrag in das Dictionary bilden? Das kann der Editor doch genau so gut machen, denn der Editor weiß bereits alles über das Wort außer der Codeadresse, welche der Compiler später einfügen kann.

Wenn der Editor das Dictionary bildet, wie in HolonForth, machen wir eine fundamentale Korrektur am System. Bisher sind Text und Editor außerhalb des Forth-Systems. Wir können einen beliebigen Editor nehmen, um ein Forth-Programm zu bearbeiten. Das Programm wird erst beim Laden durch den Compiler effektiv an Bord geholt. In HolonForth sind dagegen Editor und Text in das Forth-System integriert. Bei der Definition eines neuen Wortes wird direkt ein Eintrag im Dictionary gebildet mit einem Zeiger zum Textraum. Der Textraum besteht gleichberechtigt mit dem Coderaum.

Im Forth-System Fifth, das, wie erwähnt, wie eine Datenbank organisiert ist, liegt der Text neben dem Code im Dictionary, also voll im Programmraum! (Da denkt man sofort an Platzprobleme, aber Click und Snow haben das gut gelöst. Fifth ist ein 32 Bit System mit Speicherverwaltung, und der Text wird komprimiert abgelegt.) Holon macht genau das Gegenteil. Code und Text werden je in eigenen Räumen getrennt von der Wortliste verwaltet.

Die Komplexität des Forth-Systems ändert wenig. Zu jedem System gehört sowieso ein Editor. In Holon sind einfach die Funktionen neu verteilt. Der Editor wird erweitert um die Browserroutinen und der Compiler wird entlastet vom Generieren von Datenbankeinträgen. Beides hat sehr positive Wirkungen. Das Programm ist leichter zu bearbeiten und es wird schneller geladen.

Implementierung

HolonForth organisiert das Programm mit einer zentralen Datenstruktur (Wortliste), die dem Forth Dictionary entspricht. Der Unterschied besteht darin, dass die Holon-Struktur zusätzliche Felder enthält und dass sie vom Editor erstellt wird. Jeder Eintrag in der Datenstruktur belegt 40 Bytes. Es gibt Einträge für Worte, Gruppen und Module (Figur 2). 20 Bytes sind für den Namen reserviert, der also bis zu 19 Zeichen lang sein darf plus Längenbyte. Durch das feste Namenfeld braucht der Eintrag nicht verschoben zu werden, wenn sich der Name ändert. Der Rest des Eintrags besteht aus 10 Feldern von je 16 Bit.

Worteintrag:	Vorgänger	Nachfolger	Alpha	Text	Code	Meta (4)	Reserve	Name
Gruppeneintrag:	Vorgänger	Nachfolger	Letzte Gruppe	Erste Gruppe	Aktive Gruppe	Text	Reserve	Name
Moduleintrag:	Vorgänger	Nachfolger	Letztes Wort	Erstes Wort	Aktives Wort	Text	Reserve	Name

Figur 2: Aufbau der Datenstruktur von Holon. Die Struktur entspricht dem Forth Dictionary. Da das Dictionary vom Editor gebildet wird, kann es auch zum Ordnen des Textes verwendet werden.

Im Worteintrag werden zwei Felder für die Programmordnung benutzt. Das eine zeigt auf den Vorgänger und das andere auf den Nachfolger des Wortes. Vier Felder sind für den Metacompiler da und ein Feld ist Reserve.

Das Alphafeld wird für eine alphabetische Verkettung der Worte verwendet. Auch das ist eine Möglichkeit, die sich anbietet, wenn man eine Datenbank unter der Haube hat. Sobald ein Wort eingetragen wird, hängt es Holon in die alphabetische Liste. Das ist ein kleiner Mehraufwand, der sich aber reichlich bezahlt macht. Die Suchroutinen werden einfacher und mit einer kleinen Hilfstabelle kann man die Suche auch noch auf die Worte mit dem gegebenen Anfangsbuchstaben beschränken.

Die Worte sind global sichtbar und haben einen für das ganze Programm eindeutigen Namen.

Die Einträge für Gruppen und Module ähneln dem Worteintrag, 20 Bytes für den Namen und der Rest für die Ordnung und einen Text (neuerdings können auch zu Modulen und Gruppen Texte eingegeben werden). Ein Gruppeneintrag hat Zeiger auf die Vorgängergruppe und den Nachfolger, sowie drei Felder für die Worte der Gruppe: für das erste, das letzte und das aktuelle Wort. Analog ist der Moduleintrag aufgebaut. Der erste Eintrag in der Struktur bei Offset 0 enthält den Modulanker mit Feldern für das erste, das letzte und das aktuelle Modul. Die Einträge werden durch ihre sequentielle Nummer in der Struktur identifiziert. Wenn Sie sich die Struktur genauer anschauen wollen, drücken Sie in einem Holon-System die Tasten Alt+S. [2]

Die Texte werden in einer gemeinsamen Textdatei gesammelt. Die Einträge beginnen auf 16 Byte Grenzen, so dass der 16 Bit Textzeiger für 1 MB Text ausreicht.

Projekte und Dateien

Ein Holon-Projekt ist mit zwei Dateien definiert: Struktur (.STR) und Text (.TEX). Ich muss nicht mehr mit 20 Dateien jonglieren. Als zusätzliche Sicherheit lasse ich Holon bei jedem Start eine Kopie der Projektdateien bilden mit den Kennungen .SBK und .TBK. Falls irgend etwas schief geht oder falls ich die neuen Änderungen rückgängig machen will, greife ich auf diese Dateien zurück. Die Struktur liegt während des Arbeitens resident im Speicher. Bei Änderungen am Programm werden die Dateien spätestens nach einer Minute nachgeführt.

Außerdem führt Holon eine Codedatei, die den aktuellen Codestand festhält. Und in der Arbeitsversion von Holon wird eine Logdatei geführt, die fortlaufend jede Programmänderung notiert.

Das Entwicklungssystem und die Dateien tragen den Namen des Projektes (für jedes Projekt wird eine eigene Kopie des Entwicklungssystems verwendet). Das Entwicklungssystem öffnet beim Start selbsttätig die Projektdateien und stellt den aktuellen Zustand der Entwicklung wieder her, so dass die Arbeit am Projekt unmittelbar weitergeführt werden kann.

Wie Sie sehen, sind in Holon mehr Funktionen realisiert, als im normalen Forth-System. Wer in Forth Projekte entwickelt, muss sich wie jeder andere Softwareentwickler mit den Aspekten des Software Engineering befassen und es ist schön, wenn ihn sein Entwicklungssystem auch hierbei unterstützt. Holon hat mir gezeigt, dass Forth auch auf diesem Gebiet noch einiges zu bieten hat!

Import & Export

Ein Problem blieb noch zu lösen: Wie kann ich Programmteile in neuen Projekten wieder verwenden, wenn alles in einer Datenbank liegt, die ich nur mit Holon selbst bearbeiten kann? Die Antwort ist ein Export/Import Mechanismus für Module. Module können exportiert und in ein neues Projekt importiert werden. Holon kann auch bestehende Forth-Programme aus Textdateien importieren.

Zusammenfassung

Im konventionellen Forth ist der Programmtext außerhalb des Systems. Das Programm wird beim erst beim Compilieren in das Forth-System geholt und dabei in ein Dictionary verwandelt. Das Dictionary ist eine Datenbank mit Namen, Code und Daten. HolonForth nimmt den Programmtext von Anfang an in das System. Das Dictionary wird vom Editor gebildet und enthält nun auch den Text. Der Compiler fügt später Code und Daten hinzu. Diese kleine Verschiebung der Funktionen gibt Forth neues Leben. Eine klare Programmordnung, stets volle Übersicht, direkte Wortzugriffe, Hypertext u. a. sind leicht zu verwirklichen.

Außerdem erlaubt die Systemänderung auch eine neue Behandlung des Programmcodes, und das ist das Thema des zweiten Teils dieses Artikels.

Referenzen

[1] Fritz Prinz, HOLON - Das ganz andere FORTH, VD 2/96, Seite 33-35 und VD 3/96, Seite 7

[2] Die Holon-Version HOLON4DE steht Ihnen zur freien Verwendung zurVerfügung. Sie können darin alle beschriebenen Funktionen selbst erleben. Sie finden HOLON4DE in http://www.holonforth.com . Das System ist voll funktionsfähig und eignet sich wegen der klaren Ordnung der Worte und der vollen Interaktivität u. a. sehr gut als Lernsystem zur Einführung in Forth.

[3] Cliff Click und Paul Snow, Fifth: A Forth based Programming Environment, Proc. Rochester, 1986, Seite 89-92

❑

12th euroForth conference

4.-6. Oktober'96, Hotel Rus, St. Petersburg, Russland

von Markus Dahm, dahm@rwth-aachen.de

Die ausländischen Teilnehmer waren im Hotel Rus untergebracht. Doppelzimmer mit Bad in einem äußerlich renovierten Gebäude mit etwas martialisch anmutender 'security' in der Eingangshalle. Ihre Notwendigkeit wurde zum Glück nicht offensichtlich. Trotz Schauergeschichten von anderen Besuchern St. Petersburgs wurde weder einem Teilnehmer dieser Konferenz etwas gestohlen, noch passierte sonst etwas Unangenehmes. Lediglich die Öffnungszeiten der Hotel'bars' waren gewöhnungsbedürftig. Sie schlossen spätestens um 23.00 Uhr; aber das kann auf der nächsten Konferenz in Oxford ebenfalls passieren.

Petersburg im Oktober zeigte sich von einer sehr sonnigen, herbstlichen Seite, für die leider viel zu wenig Zeit blieb. Die wenigen Stunden in der alten Zarenstadt waren jedoch überaus eindrucksvoll. Sowohl durch die Pracht und Größe der Paläste und Kirchen als auch durch den Kontrast zum heutigen, alltäglichen Zustand der Stadt.

Die Vorträge fanden von Freitag Nachmittag bis Sonntag Mittag in den Konferenzsälen des Hotels statt. Viele Beiträge kamen diesmal von Mitgliedern des St. Petersburger Instituts für Informatik und Automation, des Ausrichters der Konferenz und von anderen Institutionen aus St. Petersburg, Moskau und Biysk. Aus der State University Moskow und dem Institute for System Research of the Russian Academy of Sciences wurde von S.A.Sidorov und M.N.Shumakov über ein Forth sehr ähnliches System namens DSSP berichtet, das sie, nach ihren Worten, nicht *er*funden, sondern *ge*funden haben. Die theoretischen Grundlagen und die praktischen Anwendungen von DSSP wurden in zwei Vorträgen beschrieben (Kontakt: shomakov@systud.msk.su).

Ein Real Time Multitasking System aufbauend auf einer Virtuellen Forth Maschine und Virtuellen Controllern wurde von V.A. Vigul et al. aus Biysk vorgestellt, das dort in Applikationen und in der Ausbildung im Kurs „Designing Processes" eingesetzt wird.

Die Beiträge aus St. Petersburg waren sehr theoretisch orientiert. M.Y.Kolodin beschrieb Meta-Features von Forth, u.a. die Implementierung von 'literate programming' mit Hilfe eines aktiven Forth-Textes. Michael Gassanenko beschrieb in zwei Vorträgen Theorie und Einsatz von Backtracking in Forth, insbesondere bei Kontrollstrukturen. Er stellt z.B. in seinem BakForth Schleifen durch Backtracking Operatoren dar: AMONG <iterator> EACH <body> ITERATE <more> (Kontakt: mlg@iias.spb.su).

Ebensfalls theoretisch orientiert war Bill Stoddards Beitrag, der ein Schichtenmodell von Forth vorstellte. Es teilt die Programmierumgebung ein in eine Daten-, eine Prozess- und eine Verhaltens-Perspektive, die er als State-Machine mit den formalen Mitteln der Spezifikationssprache Z beschreibt (Kontakt: Bill@tees.ac.uk).

Den Übergang von der Theorie zur Praxis bildete Christian Pirkers Bericht über die Fortschritte in der Compilation von Native Code, die er zusammen mit Anton Ertl in Wien im Projekt RAFTS erzielt hat. Sie basieren auf der effizienten Nutzung der Register des verwendeten MIPS-Prozessors, sowie auf einer optimierten Umsortierung der Befehle (Kontakt: {anton,pirky}@mips.complang.tu-wien.at).

Objektorientiertes Forth (OOF) war auch diesmal wieder vertreten: A.L.Medyntsev aus St. Petersburg stellte ein OOF vor, das für Datenbank-Entwicklung eingesetzt wird. Es ist modular aufgebaut, in C und Pascal implementiert und unterstützt runtime typechecking, Windows-DLLs und Windows-Libraries (Kontakt: mal@novavox.spb.su). Von Markus Dahm wurde die Portierung seines natürlichsprachlich zu programmierenden OOF, das ab dieser Version MOON heißt, auf diverse Plattformen beschrieben, sowie die Schwierigkeiten, die sich bei der Umstellung von Linux auf Windows und MacOS ergeben. Außerdem wurde ein Konzept für generische Klassen, z.B. List und Tree vorgestellt (Kontakt: dahm@rwth-aachen.de). Ein Tutorial in MOON für Netscape Navigator ist erhältlich unter www.lfm.rwth-aachen.de/research/moon. Ebenfalls ein Windows-Forth war das Thema von Peter Knaggs, der die Freeware Tools Tcl und Tk an ein 32bit Windows Forth angebunden hat und so die besten Seiten beider Ansätze vereinigt hat. Darauf baut ein von ihm entwickeltes Visual Forth auf, das die Entwicklung von Forth Programmen ähnlich der bekannten Visual-IDEs von Microsoft oder Borland möglich macht (Kontakt: pjk@tees.ac.uk).

Europäische Dimensionen wurden von Jonatan Lee und Stephen Pelc erreicht, als sie beschrieben, wie sie im Rahmen des ESPRIT-Rahmenprogramms der EU im Projekt SENDIT einen portablen Zwischencode (PIC) für die Kommunikation in verteilten industriellen Steuerungen entwickelten. Ihr PIC baut auf einer Stackmaschine mit zwei Stacks auf, die Forth sehr ähnlich sieht und in einem internationalen Projektrahmen funktioniert: spanische Hardware, französisches BIOS, britischer Interpreter, amerikanische Libraries und eine belgische Applikation. Der kommerzielle Erfolg von Forth sei also möglich, wenn man den Namen Forth nicht nennt.

Die kleineren, gemeineren Schwierigkeiten bei der Eingabe von Zahlen in verschiedenen Systemen wurden von Michael Millendorf gezeigt, sowie eine Lösung mittels Präfixen (d#, h#, o# und b#).

Die größeren Schwierigkeiten echter Internationalisierung wurden von Nick Nelson dargestellt, der, um eine erfolgreiche Anwendung auch in Japan verkaufen zu können, alle Text e auf Japanisch ausgeben mußte. In seiner farbigen Schilderung wurde klar, daß es möglich ist, daß Microsoft allerdings nicht viel Unterstützung bietet. Mit Hilfe eines Übersetzers und viel Geduld gelang es trotzdem (Kontakt: 100655.2002@compuserve.com).

Geradezu philosophisch war der Vortrag von Wolf Wejgaard, der sich fragte, warum Forth eine unsichtbare Sprache sei. Seine Begründung gründet sich auf einem verblüffenden Vergleich mit Ansätzen zur Verkehrslenkung. Hierzulande wird an Kreuzungen der Verkehr mittels Ampeln gelenkt. Das Steuerungssystem hat alle Autorität, die Autofahrer gehorchen (meistens). Im Gegensatz dazu regeln die Autofahrer den Verkehrsfluß bei einem Kreisverkehr selber. Der Verkehr fließt aktiv, besser und schneller und das ohne teuren äußeren Eingriff. Er vergleicht diesen Ansatz mit Forth; und Hersteller von Ampelanlagen sind naturgemäß nicht an Kreisverkehren interessiert. Für dieses einprägsame Bild bekam Wolf Wejgaard übrigens den diesjährigen Preis für den besten Vortrag.

Ich möchte mich sehr herzlich bei Irina P. Podnozova, Sergei N. Baranov und Marina Kern bedanken für die Organisation, die Gastfreundschaft und die herzliche Betreuung.

❑

Nucleus für Controller: Arithmetik

Teil 3

von Rafael Deliano
Steinbergstr. 37; D-82110 Germering

Im zweiten Arithmetikkapitel werden Routinen für Multiplikation und Division vorgestellt. Da für sie keine direkte Unterstützung durch die Hardware von 8 Bit Controllern vorhanden ist, sind die Programme relativ kompliziert und langsam. Der Programmierer sollte deshalb auf Controllern, wo immer möglich, Shifts verwenden.

Stichworte: Controller Arithmetik Assembler

Schiebebefehle

Verschiebung um ein Bit entspricht Multiplikation oder Divison mit der Konstanten 2. Bild 1 zeigt die Assemblerbefehle die man auf einer 16 Bit CPU findet. Dabei werden die „logischen" Befehle LSR, und LSL, für vorzeichenlose Zahlen verwendet. Die „arithmetischen" Befehle ASR, und ASL, funktionieren mit Zweierkomplementzahlen. Dabei fällt auf, daß sich ASL, und LSL, identisch verhalten. Die CPUs haben deshalb nur einen von beiden implementiert.

Auf einer 8 Bit CPU geht das Bit das bei LSL, und LSR, herausgeschoben wird nicht verloren, sondern wird im Carryflag gespeichert. Zusammen mit den Rotationsbefehlen ROL, und ROR, die Bits aus dem Carry entnehmen und dort abspeichern sind deshalb leicht 16 Bit Befehle zusammenzustellen (Bild 2). Etwas umständlicher ist ASR, auf CPUs wie dem 6502 die den Befehl nicht implementiert haben. Man muß dabei erst eine Kopie des obersten Bits ins Carry laden, bevor man die beiden Schiebebefehle ausführen kann.

```
HB    LDA.  \ Get High Byte in Accu
A.    LSL.  \ Shift Accu: MSB in Carry
HB    ROR.  \ Rotate High Byte in Memory
LB    ROR.  \ Rotate Low Byte in Memory
```

Die arithmetischen Befehle ASL, und ASR, haben in FORTH meist den Namen 2* und 2/. Für LSR, und LSL, werden verschiedene Namen verwendet. Am bekanntesten ist

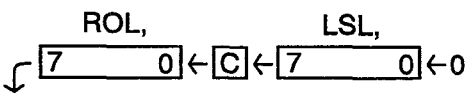


Bild 2: 16-Bit-LSL, aus 8-Bit-Befehlen

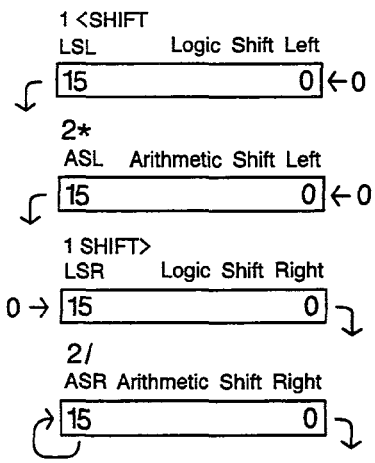


Bild 1:Shifts

<SHIFT und SHIFT> aus FORTH-83. Für ANS-Forth wurden diese Befehle in LSHIFT und RSHIFT umbenannt. Ferner werden in FORTH-83 und ANS nicht 1 Bit Operationen verwendet, sondern die Befehle werden über einen Parameter gesteuert, der auf dem Stack übergeben wird:

```
4 <SHIFT  \ ( UN1 -  ) 4 mal LSL. ausführen.
```

Das wäre sinnvoll, wenn die darunterliegende CPU einen Barrelshifter hätte. Das ist ein gewaltiger Multiplexer der von einem Datenwort gesteuert innerhalb eines Taktzyklus alle Bits beliebig weit verschiebt. Die meisten CPUs können allerdings nur bitweise sequentiell schieben. Es ist deshalb bei einer 8 Bit CPU

effizienter, Befehle wie 8<SHIFT 4<SHIFT 1<SHIFT zu definieren. 8<SHIFT wird dabei durch Umspeichern von Bytes und nicht durch Schiebeoperationen realisiert.

Multiplikation und Division von Konstanten

Die Multiplikation mit den üblichen „binären" Konstanten kann man direkt durch Schiebebefehle realisieren. Das funktioniert für Zahlen mit und ohne Vorzeichen. Beispiele:

```
: 2*  1<SHIFT  ;
: 4*  1<SHIFT  1<SHIFT  ;
: 8*  1<SHIFT  1<SHIFT  1<SHIFT ;
```

Zwischenwerte können durch zusätzliche Additionen erzeugt werden:

```
: 3*  DUP  1<SHIFT  +  ;
: 5*  DUP  1<SHIFT  1<SHIFT  +  ;
: 6*  DUP  1<SHIFT  1<SHIFT OVER  + + ;
```

Derartige Befehle können den Zugriff auf Arrays mit krummer Größe wirksam beschleunigen. Offensichtlich ist das aber nur effizient für Werte die nahe an den „natürlichen" Konstanten liegen.

In der Trickkiste der MISC-Implementierer Paysan&Wilke findet sich ferner eine Methode wie man LSL, auf einer CPU realisiert, die diesen Opcode nicht hat:

```
: 2*  X  =  X + X  \ : 2* DUP + ;
```

Ähnlich wie die Multiplikation von Konstanten kann Division durch Rechtsshifts und Subtraktion ausgeführt werden. Die Nützlichkeit von 2/ ist durch das Rundungsverhalten bei negativen Zahlen allerdings eingeschränkt:

```
-5  2/  =   -3    \ soll:     -2.5
-5  2/  2/  =  -2  \ soll:     -1.25
```

Damit unterscheiden sich 2/ und 2 /. Runden gegen Null bei negativen Zahlen kann man durch NEGATE 1SHIFT> NEGATE erzeugen.

Multiplikation

Z = X * Y

Die Funktion ist am einfachsten zu realisieren, indem man den Wert X Y-mal aufsummiert. Mit Subtraktion könnte man dementsprechend auch dividieren. Das Verfahren ist jedoch extrem langsam und deshalb ungebräuchlich. Üblich ist es, beide Operationen auf eine Kombination von Schiebebefehlen und Additionen bzw. Subtraktionen zurückführen. Als einführendes Beispiel ist

```
                    Y=B9=
                    10111001
                  00000000
X=DA=           +11011010   1
                  00000000  0
                 00000000   0
               +11011010    1
              +11011010     1
             +11011010      1
             00000000       0
           +11011010        1
Z=9DBA = 1001110110001010
```

Bild 3:
Beispiel für Multiplikation

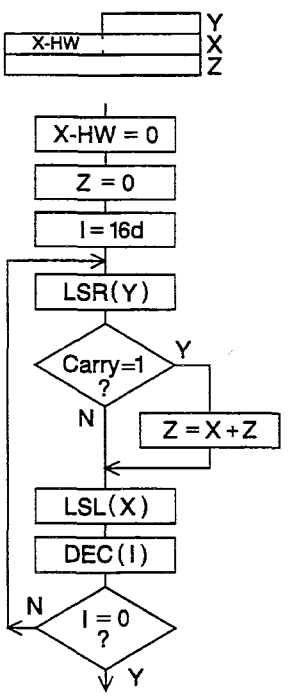


Bild 4:
Flußdiagramm Multiplikation

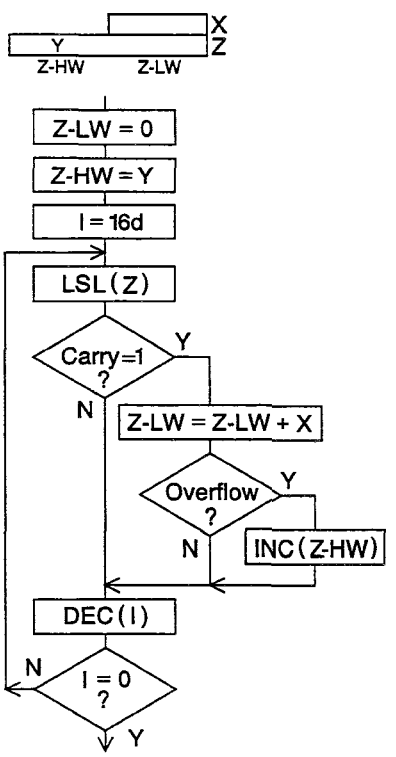


Bild 5:
Flußdiagramm optimierte Multiplikation

die Multiplikation zweier vorzeichenloser 8 Bit Zahlen mit einem 16 Bit breiten Ergebnis in Bild 3 dargestellt. X wird dabei Schritt für Schritt nach links geschoben. Ob es in die Summe addiert wird oder nicht, hängt vom Inhalt des jeweiligen Bits in Y ab. In Bild 4 findet sich das Flußdiagramm für eine 16x16 Bit Multiplikation mit 32 Bit Ergebnis nach diesem einfachen Verfahren. Dabei wird eine 32 Bit Addition benötigt. In Listing 1 ist als praktisches Beispiel 6502-Code dargestellt. Das Flußdiagramm dazu befindet sich in Bild 5. Statt X wird hier Z verschoben. Die 32 Bit Addition kann dann durch eine 16 Bit Addition und einen 16 Bit Inkrement ersetzt werden. Um die Zahl der belegten Register zu senken, wird Y in der oberen Hälfte des 32 Bit Z-Wortes abgelegt, das sonst unbenutzt ist. Y wird hier schrittweise durch die LSL-Shifts eliminiert. Die Reihenfolge der Additionen ist damit gegenüber der ersten Version, wo Y mit LSR ausgelesen wurde, umgekehrt. Die Reihenfolge der Additionen ist jedoch unerheblich, solange der Versatz von X zu Z stimmt.

Eine Multiplikation mit 32 Bit Ergebnis ist überlaufsicher:

FFFF * FFFF = FFFE0001

Der Namen für diese Operation war U* in fig-FORTH und ist UM* in FORTH-83 und ANS. Meist ist es der grundlegende Befehl der in Assembler ausgeführt wird und von dem alle anderen Multiplikationen abgeleitet werden.

Division

Z = X / Y

Bei der Division entsteht neben dem Quotient Z ein Rest R, der meist nicht Null ist. Während Z, Y und R normale Wortlänge haben, wird der Divisor X normalerweise in doppelter Wortlänge ausgeführt.

Die Durchführung der Divison erfolgt durch Shifts und Subtraktion. Als einführendes Beispiel der Rechengang für zwei vorzeichenlose Zahlen mit jeweils 8 Bit Wortbreite in Bild 6. Y wird von rechts hereingeschoben. Von dem oberen Teil von X, der gleichauf oder rechts vom Bit 0 von Y steht, wird Y subtrahiert. Wenn Y größer als X ist, d.h. bei Borrow, werden nur Nullen produziert. Nur wenn kein Borrow eintritt, kommt eine Eins ins Ergebnis. Nach einer festen Zahl von Durchläufen, die der Wortbreite von Y entspricht, ist die Division beendet und es bleibt eventuell ein unteilbarer Rest übrig.

In Bild 7 findet sich das Flußdiagramm für einfache Division. X wird in der unteren Hälfte des 32 Bit Registers gespeichert und nach links geschoben. Die Subtraktion von Y findet mit der oberen Hälfte des Registers X statt. Das Ergebnis wird in M zwischengespeichert und nur dann in die obere Hälfte von M rückgespeichert, wenn kein Borrow stattgefunden hat. In Z wird durch Shifts entweder eine Eins oder eine Null geschoben.

In Listing 2 ist 6502-Code dargestellt. Er entspricht dem Flußdiagramm in Bild 8. Die erste Verbesserung ist, daß das Register Z eliminiert wurde. Die Z-Bits werden in die untere Hälfte des X-Registers nachgeschoben. Durch die Polarität beim 6502 (Borrow = 0) ist das hier direkt möglich. Bei anderen CPUs müßte man Z abschließend mit einem NOT invertieren. Die andere Verbesserung ist die Erweiterung der oberen Hälfte des X-Registers auf 24 Bit durch das Register EXT. Die Zwischensumme kann nämlich 17 Bit breit werden. Oft findet man 6502-Code in fig-FORTH und manchen FORTH-83 der mit 16 Bit Subtraktion arbeitet. Er ist nur für ein 15 Bit Ergebnis geeignet. Also nur für Rechnung mit vorzeichenbehafteten Zahlen zu verwenden.

Der Name für diesen Befehl war U/ in fig-FORTH und ist UM/ MOD in FORTH-83 und ANS. Meist setzen alle anderen Divisionsvarianten auf diesen Befehl auf. Division durch Null ist per Definition ein Fehler. Was die Routine machen soll ist implementierungsabhängig. Man kann den Fall erstens ignorieren. Zweitens kann man das Ergebnis auf FFFF setzen, weil die Division durch etwas sehr Kleines ja näherungsweise etwas sehr Großes erzeugt. Drittens kann man mit Fehlermeldung abbrechen. Viertens kann man irgendwo ein Bit als Fehlerflag setzen.

Überlauf

Da das Ergebnis auf 16 Bit beschränkt ist, ist die Divison nicht zwangsläufig überlaufsicher:

```
FFFFFFFF /  0001   =     FFFFFFFF
FFFFFFFF /  FFFF   =     10001
```

Um einen Überlauf des Quotienten zu verhindern, müssen Divisor und Dividend in ihrem Wertebereich beschränkt bleiben. Je kleiner der Dividend, desto kleiner der Divisor:

```
FFFE0001 / FFFF    = FFFF  R: 0000
FFFD0002 / FFFE    = FFFF  R: 0000
FFFC0003 / FFFD    = FFFF  R: 0000
```

Wenn man den Rest mitverwerten kann, schiebt sich die Obergrenze um FFFF nach oben:

```
FFFF0000 / FFFF    = FFFF  R: FFFF
```

* und / mit Vorzeichen

Die Berechnung erfolgt indirekt durch Rückgriff auf die vorzeichenlosen Grundbefehle. Man sichert erst die Vorzeichen der beiden Operanden im Speicher oder auf dem Returnstack. Dann wandelt man negative Zahlen in Beträge um. Anschließend kann die vorzeichenlose Multiplikation oder Divison ausgeführt werden. Abschließend versieht man die Ergebnisse wieder mit Vorzeichen. Dazu verwendet man die Vorzeichen der beiden Eingangswerte. Bei Division mit Vorzeichen kann ein weiterer Fehlerzustand auftreten, wenn das Ergebnis nicht in den Bereich von +32767 ... -32768 paßt. Multiplikation und Division von Zahlen mit Vorzeichen ist umständlich, weshalb man vorzugsweise die vorzeichenlosen Grundbefehle verwendet.

Divisionsrest

Bei der Division von zwei Integerzahlen bleibt meist ein unteilbarer Rest übrig, der oft Modulo genannt wird. Beispiel:

7 / 3 = 2 Rest = 1

Operation die diesen Rest zugänglich machen, tragen in FORTH meist den Zusatz MOD. Für die Definition von Division mit Rest wird diese Gleichung verwendet:

Dividend / Divisor = Quotient ; Rest
(Quotient * Divisor) + Rest = Dividend

Die Rückgewinnung des Dividenden funktioniert immer, aber Quotient und Rest sind nicht mehr eindeutig, sobald man Zahlen mit Vorzeichen verwendet. In Bild 9 sind zwei mögliche Lösungen für ein Beispiel dargestellt.

Symmetrical Division

Hier übernimmt der Rest das Vorzeichen des Dividenden. Der Quotient wird gegen Null gerundet. Ist damit um Null herum symmetrisch, woraus sich der Name ergibt. Symmetrische Division wird in FORTH-79 verwendet. Beispiele:

Dividend	Divisor	Rest	Quotient
10	7	3	1
-10	7	-3	-1
10	-7	3	-1
-10	-7	-3	1

Implementierung: die Division wird vorzeichenlos durchgeführt, nachdem man irgendwo die Vorzeichen abgespeichert hat. Damit liegen nun Quotient und Rest positiv vor. Der Rest erhält

Listing 1 : Multiplikation für 6502

```
\ LB = LowByte
\ HB = HighByte
\ HW = Highword
\ LW = LowWord
00      #. LDA. \ Clear Z-LW
Z-LW-LB    STA.
Z-LW-HB    STA.
Y-LB       LDA. \ Move Y to Z-HW
Z-HW-LB    STA.
Y-HB       LDA.
Z-HW-HB    STA.
10      #. LDY. \ I = 16d2 $:
Z-LW-LB    ASL. \ 32 Bit LSL. of Z
Z-LW-HB    ROL.
Z-HW-LB    ROL.
Z-HW-HB    ROL. \ Shifts Bit 15 of
                \ Y to Carry
1 $        BCC. \ Skip Add if
                \ Carry = 0
           CLC.
X-LB       LDA. \ Add X to Z-LW
Z-LW-LB    ADC.
Z-LW-LB    STA.
X-HB       LDA.
Z-LW-HB    ADC.
Z-LW-HB    STA.
1 $        BCC. \ If Add over-
                \ flows: Inc Z-HW
Z-HW-LB    INC.
1 $        BNE.
Z-HW-HB    INC.
1 $:       DEY.
2 $        BNE.
```

Listing 2: Division für 6502

```
      10 #. LDA.   \ I = 16d
N           STA.
X-LW-LB     ASL.   \ Shift Bit 15 of
                   \ X-LW in Carry
X-LW-HB     ROL.
2 $: 00 #.  LDA.   \ EXT = 00
EXT         STA.
X-HW-LB     ROL.   \ Shift X-HW&EXT .
                   \ Input Carry
X-HW-HB     ROL.
EXT         ROL.
            SEC.
X-HW-LB     LDA.   \ M = X-HW&EXT - Y
Y-LB        SBC.
M-LB        STA.
X-HW-HB     LDA.
Y-HB        SBC.
M-HB        STA.
EXT         LDA.
00       #. SBC.
1 $         BCC.   \ branch on borrow
M-LB        LDA.   \ X-HW = M
X-HW-LB     STA.
M-HB        LDA.
X-HW-HB     STA.
1 $:                \ Shift in  Bit of
                    \ Result
X-LW-LB     ROL.   \ Shift out Bit 15
                    \ of X-LW to Carry
X-LW-HB     ROL.
         N DEC.
2 $         BNE.
\ X-HW = Rest
\ X-LW = Quotient
```

```
        X=D4      Y=17
      11010100 : 10111
           1
-10111                0
          11
 -10111               0
         110
  -10111              0
        1100
  -10111              0
       11010
   -10111             1
      000111
   -10111             0
        01110
   -10111             0
         11100
    -10111            1
R = 05 =  00101

Z = 09 =        00001009
```

Bild 6:
Beispiel für Division

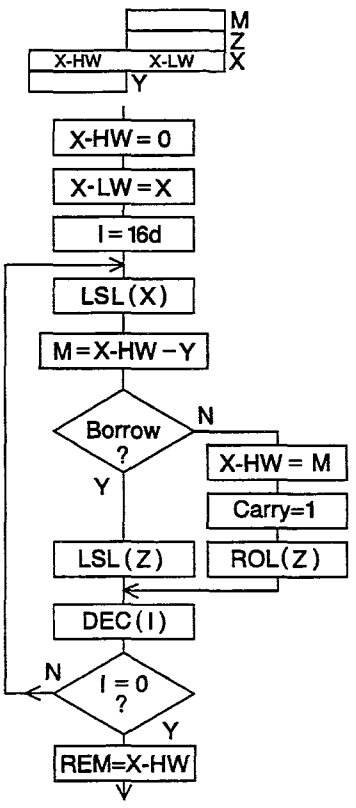


Bild 7:
Flußdiagramm einfache Division

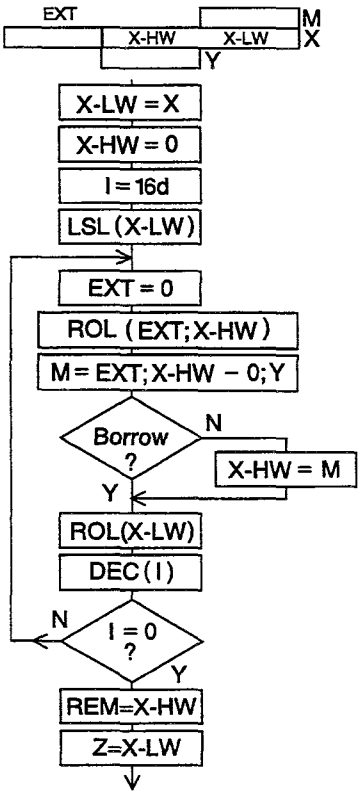


Bild 8:
Flußdiagramm optimierte Division

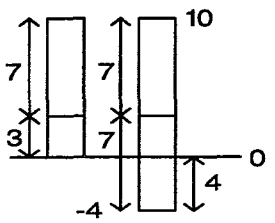


Bild 9: Divisionsrest (10/7=1 Rest 3 oder 10/7=2 Rest -4)

per Definition das Vorzeichen des Dividenden. Der Quotient wird negativ, wenn Dividend und Divisor unterschiedliches Vorzeichen hatten.

Floored Division

Der Rest übernimmt das Vorzeichen des Divisors und der Quotient wird Richtung auf den „arithmetic floor" gerundet. Der „arithmetic floor" ist der größte Integer der kleiner oder gleich einer Zahl ist: +0,6 wird zu 0 , -0,4 wird zu -1. Ergo wird gegen minus Unendlich gerundet. Floored Division wird in FORTH-83 verwendet. Beispiele:

Dividend	Divisor	Rest	Quotient
10	7	3	1
-10	7	4	-2
10	-7	-4	-2
-10	-7	-3	1

Implementierung: nachdem die Division vorzeichenlos durchgeführt wurde, liegen Quotient und Rest positiv vor. Der Rest erhält per Definition das Vorzeichen des Divisors. Der Quotient ist negativ, wenn Dividend und Divisor unterschiedliches Vorzeichen hatten. In diesen Fällen, muß man die Beträge von Rest und Quotient ändern. Der Betrag des Quotienten wird inkrementiert. Den Betrag des Rests ergibt:

neuer Rest = Divisor - alter Rest

Rest in ANS, C & Hardware

In ANS gibt es beide Versionen:

```
FM/MOD \ ( d1 n1 - n2 n3 )  n2 = Rest
      \  floored divison
SM/REM \ ( d1 n1 - n2 n3 ) n2 = Rest
\ symmetrical divison
```

Daß einmal REM für „Remainder" statt MOD verwendet wurde, ist reine Willkür ohne tiefere Bedeutung. Beim altbekannten Befehl /MOD ist es in ANS-Forth implementierungsabhängig, ob er aus FM/MOD oder SM/REM gebildet wurde.

Auch in C ist bei negativen Zahlen das Verhalten des dortigen Modulooperators % implementierungsabhängig. Man kann dort nichtmal auf Grundbefehle mit definiertem Verhalten zugreifen. Soweit CPUs überhaupt vorzeichenbehaftete Divisonsbefehle im Befehlssatz haben, verwenden sie Symmetrical Divison (68000, 80x86). Damit ist die Wahrscheinlichkeit recht hoch, daß der Implementierer, wo er die Wahl hat, diese Variante bevorzugt. ❑

Eine kleine Geschichte ...

Computer: „Hotel International, Guten Tag! Womit kann ich Ihnen dienen?"

Anrufer: „Ja, em, ich wollte wissen, ob Sie ein Doppelzimmer mit Bad haben?"

Computer: „Ja, wir haben 120 Doppelzimmer mit Bad."

Anrufer: „Was, alle frei??"

Computer: „Nach freien Zimmern haben Sie nicht gefragt!"

Anrufer: „Also, wieviel haben Sie frei?"

Computer: „Heute haben wir nichts frei."

Anrufer: „Ich konme ja erst übermorgen, nur eine Nacht!"

Computer:"Ach so! Übermorgen haben wir noch 5 frei. Sie möchten eins dieser herrlichen, preiswerten Zimmer, nicht wahr?! Bitte buchstabieren Sie Ihren Namen."

Anrufer: „Em. ü. El. El. E. Err. „

Computer: „Danke, Herr Emü el Eller. Ich bestätige Ihre Buchung: Ein ruhiges Doppelzimmer mit Bad, übermorgen für eine Nacht, zum Preis von 450 DM. Wann bitte kommen Sie an?"

Anrufer: „Das weiß ich nicht! ...wo ist denn mein Flugschein? ..."

Computer: „Das weiß nur die Flugauskunft, ich verbinde!"

Computer: „Flugauskunft, Guten Tag !"

Anrufer: „.,., ach ja, hier! Flug LH 4523, wann kommt der?"

Computer: „Der Flug EL AL 4523 ist nicht in der Datei."

Anrufer: „Nicht EL AL ! EL Ha, El HAAAA, HAAArghhhhh ! „

Conputer: „Zentralrechner Bundesgesundheitsamt Sie haben behandlungsbedürftige Erkältungssymptome. Zur weiteren Diagnose sagen Sie bitte 'aah'!"

Anrufer: „aa... bin ich denn bescheuert?! Ich wollte ein Zimmer buchen und keinen Arzt! O hätte ich bloß noch einen guten alten Apple!!!"

Computer: (schweigt)

Anrufer: „Hallo, sind Sie noch da ?"

Computer: „Zentrallager. Entschuldigen Sie die Verzögerung! Ich habe einen Apple, Baujahr 1976, im 'Used Comp Shop', Castrop-Rauxel, zum Preis von 1500 DM. Möchten Sie ihn haben?"

Anrufer: „Nein! Vergiß es!!"

Computer: „Ihr Benutzercode erlaubt keine Löschung von Zentrallagerdaten."

Anrufer: „Ich will ein Zimmer, du Hirsch, damischer!"

Computer: „String Recognition error at Substra$ ('damisch $$$$ Aswio52S: Syntax Check at 200DE3 Rec61321: Error Recovery in Proqress, Please wait"

Anrufer: „Herrje! Verschwinde! Ich will zurück zum anderen Computer!"

Computer: „Zentralrechner Bundesgesundheitsamt. Zur Beurteilung, ob Sie bescheuert sind, einige Fragen zu Ihrem Geisteszustand: Gab es früher Probleme?"

Anrufer: „Du verdammter ?$L? Du kannst mich ?$§ $"

Computer: „Achtung, Achtung, hier spricht der Polizeirechner! Ihnen wird eine Straftat nach § 1523 StPO, Beleidigung zentraler Intelligenzsysteme, zur Last gelegt. Ihre Personalmagnetkarte wird eingezogen. Ich weise Sie auf Ihr Aussageverweigerungsrecht hin. Sie können außerdem einen Computer des Rechtswanwaltsverbandes konsultieren."

Anrufer: (bricht zusammen).

Jürgen Beetz

Dynamische Macros

von Friederich Prinz
Homberger Str. 335; 47443 Moers; F.PRINZ@MHB.gun.de

Seit ich mit dem Holon arbeite und lerne, werde ich immer wieder davon überrascht, was das Holon tatsächlich alles kann. Von diesem "nicht ganz gewöhnlichen" Forth vermuten einige Forther, daß es bei der Arbeit als Cross Compiler sozusagen "naturgemäß" weniger flexibel arbeiten müsse, als sie es von traditionell interagierenden Forthen her kennen. Weit gefehlt. Holon läßt mir bis tief hinunter in den Kern des Systems all die Freiheiten, die ich zuvor z.B. am "Moerser" ZF kennen und lieben gelernt habe.

Stichworte: Holon Macro FGDRIVER

Die Arbeit an der Portierung der FGDRIVER Routinen ist eigentlich "kinderleicht", nur eben sehr zeitaufwendig. Um die reine Tipparbeit an den Assemblercodes abzukürzen und die damit verbundenen Fehler zu minimieren, greife ich natürlich möglichst häufig zu Makros. Nachdem Wolf Wejgaard mir dankenswerter Weise gezeigt hat, wie sich via Integers Argumente an Makros übergeben lassen, läßt sich Vieles einfacher und müheloser realisieren. Sogar "dynamische" Makros sind nun möglich, die mir Denk-, Tipp-, und Testarbeiten ersparen.

Das Problem

FGDRIVER erwartet bei einer Reihe von Parameterübergaben "Word-vectors". Das sind im einfachsten Fall 32-Bit Adressen, die auf eine 16-Bit Adresse zeigen, aus der ein Wert entnommen oder in die ein Wert eingeschrieben werden soll.

```
CREATE Zelle 2 ALLOT

CREATE Vector ?CS: , ] Zelle [
```

Eine einfache, forthige Definition im ZF wäre das gerade gezeigte Beispiel. Die eigentliche "Arbeitszelle" ist mit dem Wort "Zelle" definiert. Zelle kann aber von FGDRIVER nicht "short" adressiert werden, weil der Grafiktreiber - als TSR der Applikation vorgeladen - nur segmentübergreifend auf den Speicher zugreifen kann. Deshalb wird die Adresse von "Zelle" in einem Vektor übergeben. Das erste Wort im Vektor beschreibt das Datensegment, in welchem sich die "Zelle" befindet. Das zweite Wort ist die Adresse der Zelle innerhalb dieses Segmentes.

So weit, so einfach. Bei solchen einfachen Konstrukten böte sich natürlich auch die folgende Lösung an:

```
CREATE Vector ?CS: ] Zelle [
                  2 ALLOT
```

Hier stehen alle drei zusammengehörige Bytes direkt "nebeneinander". Leider sind diese einfachen Konstrukte aber eher die Ausnahme. Tatsächlich verlangt FGDRIVER häufiger nach "Records", in denen 2, 3 oder sogar vier WordVektoren zusammengefaßt werden. Um es noch einmal deutlich zu machen: Die WordVectoren bestehen aus Segment- und Offsetangabe einer Speicherzelle. Daß die Speicherzelle direkt "hintenan" steht, ist dabei nicht vorgesehen. Wenn vier Vektoren in einem Record zusammengefaßt werden sollen, dann besteht dieser Record aus vier 32-Bit Adressen und sonst nichts.

Der klassische Ansatz

Ich habe mir angewöhnt, für solche Fälle immer ein Mehrzweckarray "zur Hand zu haben". In der Portierung zum FGDRIVER heißt dieses Feld FGVParam. In seinen 48 Byte werden durch die Arbeit der verschiedenen Funktionen die "merkwürdigsten" Strukturen abgebildet - unter anderem auch die zuvor angesprochenen Records aus Vectoren, samt den zugehörigen "Datenzellen".

Eine Funktion erwartet einen Zeiger auf einen Record mit zwei Vektoren. Dann wird zunächst als "Zeiger" auf den Record die Adresse von FGVParam übergeben, sowie das Datenregister als Quellsegment für FGVParam. Den Record selbst beschreibe ich mit folgender Sequenz:

```
MOV 0 [BX]. DS
MOV 2 [BX]. FGVParam 8 +
MOV 4 [BX]. DS
MOV 6 [BX]. FGVParam 10 +
```

Werden statt dessen drei Vektoren im Record verlangt, muß ich diese Sequenz schreiben:

```
MOV  0 [BX]. DS
MOV  2 [BX]. FGVParam 12 +
MOV  4 [BX]. DS
MOV  6 [BX]. FGVParam 14 +
MOV  8 [BX]. DS
MOV 10 [BX]. FGVParam 16 +
```

Mit anderen Worten: in FGVParam stehen zunächst die geforderten Vektoren, die als Offseteinträge Adressen enthalten, die innerhalb von FGVParam liegen. Die eigentlichen Datenzellen schließen direkt an den letzten Vektor an. Es liegt eben immer noch alles irgendwie im Speicher...und wem die "verbale" Erklärung nicht einleuchtet, dem darf ich an dieser Stelle empfehlen, zu einem bewährten Hausmittel zu greifen, und sich FGVParam und die Belegung seiner Zellen auf einem Blatt Papier zu skizzieren...

Wie die Assemblersequenz für einen oder vier oder beliebig viele Vektoren aussieht, läßt sich aus den beiden Beispielen leicht ableiten. Daß "der Codierer" wenig Interesse daran haben kann, diese Sequenzen jedesmal aufs Neue zu tippen, ist sicher ebenso leicht einsehbar. Deshalb ist hier das Makro gefragt, bzw. für jeden Fall ein separates Makro zu erstellen.

Es geht auch anders...

Nun weiß ich, wie viele Einzelfälle für den FGDRIVER anzusetzen sind. Immerhin habe ich die für das ZF alle tippen müssen. Wenn ich das nicht wüßte, würde es mich sehr ärgern, immer wieder einmal ein Makro "nachschieben" zu müssen. Noch ärgerli-

cher wäre es, ein Makro für nur einen einzigen Fall des Vorkommens geschrieben zu haben. Es wäre einfach angenehm, wenn es ein einziges Makro für alle möglichen Fälle gäbe, bzw. für jede mögliche Anzahl verlangter WordVectors. Und das ist mit HOLON problemlos realisierbar!

Zunächst habe ich mir vorgestellt, das Makro könne, gesteuert durch ein "von außen" vorgegebenes Argument, den Record in einer LOOP zusammenbasteln. In jeder Einzelschleife (zu jedem Index) sollte das Makro das Datensegment eintragen, die Adresse der Datenzelle in FGVParam berechnen und "neben" das Datensegment schreiben. Dazu habe ich mir die Abhängigkeiten der Adressierungen, bzw. Berechnungen als Offsets auf FGVParam auf einem Blatt notiert:

```
   MOV BX. # FGVParam

Arg. Ind.
 1    0   MOV  0 [BX]. DS   MOV  2 [BX]. # FGVParam  4 +

 2    0   MOV  0 [BX]. DS   MOV  2 [BX]. # FGVParam  8 +
      1   MOV  4 [BX]. DS   MOV  6 [BX]. # FGVParam 10 +

 3    0   MOV  0 [BX]. DS   MOV  2 [BX]. # FGVParam 12 +
      1   MOV  4 [BX]. DS   MOV  6 [BX]. # FGVParam 14 +
      2   MOV  8 [BX]. DS   MOV 10 [BX]. # FGVParam 16 +
```

FGVParam wird dabei über das Register BX indiziert. Arg. ist ein Integer, das die Anzahl der Schleifenläufe vorgeben soll. Ind. ist der Schleifenindex einer DO..LOOP.

Es ist leicht zu sehen, daß die Indizierung vor dem ersten [BX] jeder Zeile mit "I 4 *" beschrieben werden kann. Ebenso leicht ist die Indizierung für das zweite [BX] jeder Zeile zu erkennen als "I 4 * 2 +". Und für die Adressenbildung aus FGVParam plus Offset läßt sich definieren "Arg. 4 * I 2 * + +".

Und genau das wollte ich in einem Macro realisiert haben. Dann sollte es möglich sein, dem Macro die Anzahl der Vektoren mitzuteilen, die es in FGVParam anlegen und indizieren soll.

```
MACRO: InitWordVectors
 IS ARG1
 MOV BX. # FGVParam
 ARG1 0 DO MOV I 4 * [BX]. DS
           MOV I 4 * 2 + [BX]. # FGVParam Arg1 4 * I 2 *
+ +
   1 +LOOP ;
```

Ein Wort, das dieses Makro braucht, sieht z.B. so aus:

```
CODE fgSVGAVer ( - )
  3 InitWordVectors
  ...
NEXT
```

Dem Makro ist es einerlei, ob es einen Vektor, oder drei oder beliebig (?) viele davon in FGVParam hineinbasteln soll. Es macht einfach das, was ich mir von ihm erhofft habe.

Kann man jetzt noch Unflexibilität in Holons Cross Compiling vermuten?

□

Erste Lektion in angewandter Mathematik für Ingenieure

Jedem angehenden Ingenieur wird schon zu Beginn beigebracht, z.B. die Summe von zwei Größen nicht etwa in der Form

$$1+1=2 \tag{1}$$

darzustellen. Diese Form ist banal und zeugt von schlechtem Stil. Schon Anfangssemester wissen nämlich, daß

$$1=\ln e \tag{2}$$

und weiterhin, daß

$$1=\sin^2 q+\cos^2 q\,. \tag{3}$$

Außerdem ist für den kundigen Leser offensichtlich, daß

$$2=\sum_{n=0}^{\infty}\frac{1}{2^n}. \tag{4}$$

Daher kann die Gleichung (1) viel viel wissenschaftlicher ausgedruckt werden in der Form

$$\ln e+\left(\sin^2 q+\cos^2 q\right)=\sum_{n=0}^{\infty}\frac{1}{2^n} \tag{5}$$

Es ist sofort einzusehen,daß

$$1=\cosh p\sqrt{1-\tanh^2 p} \tag{6}$$

und da

$$e=\lim_{\delta\longrightarrow\infty}\left(1+\frac{1}{\delta}\right)^{\delta} \tag{7}$$

kann Gleichung (5) zu folgender Form weiter vereinfacht werden:

$$\ln\left[\lim_{\delta\longrightarrow\infty}\left(1+\frac{1}{\delta}\right)^{\delta}\right]+\left(\sin^2 q+\cos^2 q\right)=\sum_{n=0}^{\infty}\frac{\cosh p\sqrt{1-\tanh^2 p}}{2^n} \tag{8}$$

Wenn wir berücksichtigen, daß

$$0!=1 \tag{9}$$

und wir uns erinnern, daß die Inverse der transponierten Matrix die Transponierte der Inversen ist, können wir unter der Restriktion eines eindimensionalen Raumes eine weitere Vereinfachung durch die Einführung des Vektors X erzielen, wobei

$$(X')^{-1}-\left(X^{-1}\right)'=0 \tag{10}$$

Verbinden wir Gleichung (9) mit Gleichung (10), so ergibt sich

$$\left|(X')^{-1}-\left(X^{-1}\right)'\right|!=1 \tag{11}$$

Eingesetzt in Gleichung (8) reduziert sich unser Ausdruck zu der Form

$$\ln\left[\lim_{\delta\longrightarrow\infty}\left|(X')^{-1}-\left(X^{-1}\right)'\right|!+\frac{1}{\delta}\right)^{\delta}\Bigg]+\left(\sin^2 q+\cos^2 q\right)=\sum_{n=0}^{\infty}\frac{\cosh p\sqrt{1-\tanh^2 p}}{2^n} \tag{12}$$

Spätestens jetzt ist offensichtlich, daß Gleichung (12) viel klarer und leichter zu verstehen ist als Gleichung (1). Es gibt noch eine Reihe anderer Verfahren, um die Gleichung (1) auf andere Weise zu vereinfachen. Diese werden jedoch erst behandelt, wenn der angehende Ingenieur die hier verwandten einfachen Prinzipien verstanden hat.

Aushang an der Fachhochschule Ulm
(Quelle:John J. Siegfried / University of Wisconsin "A Fist Lesson In Econometrics". Ins Deutsche übertragen und bearbeitet von Prof. Volkmar Liebig, Ulm)

PostScript

von Rafael Deliano
Steinbergstr. 37; D-82110 Germering

FORTH war nie die einzige stackorientierte Programmiersprache. Es war nur immer die verbreitetste und bekannteste. Zeitweise schien es darin von PostScript abgelöst zu werden. Eigentlich als Seitenbeschreibungssprache konzipiert, ist PostScript leistungsfähig genug, um es mit allgemeinen Programmiersprachen aufzunehmen. Für die auf Tischcomputern wesentlichen Grafikanwendungen hat es mehr zu bieten als FORTH.

Stichworte: PostScript Geschichte Programmiersprachen

Geschichte

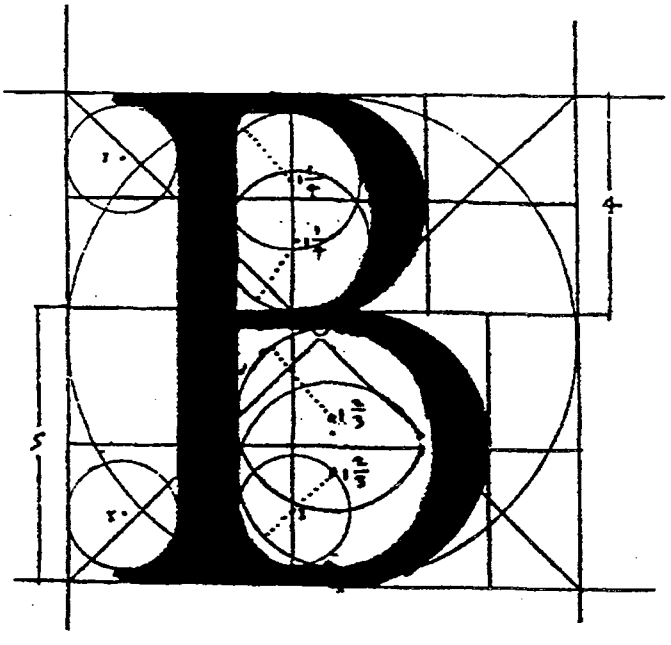
John Warnock hatte sich bereits während seines Studiums an der University of Utah mit Computergrafik beschäftigt. Der Ursprung von PostScript reicht bis 1976 zurück, als er an einer interpretierenden Sprache mit dem Namen „Design System" mitentwickelte. Federführend bei diesem Projekt scheint John Gaffney gewesen zu sein. Warnock: „Although the Design System language and its successors bear a superficial resemblance to the FORTH programming language, their conception and development were entirely independent of FORTH.". Hauptanwendung war die Modellierung eines dreidimensionalen Modells des Hafens von New York, wie man ihn von der Brücke eines Schiffes aus sehen würde. 1978 ging Warnock zu Xerox PARC, wo er sich weiterhin mit Grafik und nun auch Typografie befaßte. Er hätte dort gerne mit LISP programmiert, weil er interaktive Interpreter mag. In seiner Abteilung durfte er aber nicht: „it would have been political suicide to use LISP". Politisch korrekt wäre BCPL gewesen, ein Vorläufer von C. Es war ungefährlicher zusammen mit Martin Newell eine neue Sprache zu erfinden: „JaM" (= John & Martin). Daraus entstand das Xerox Druckerprotokoll „Interpress".

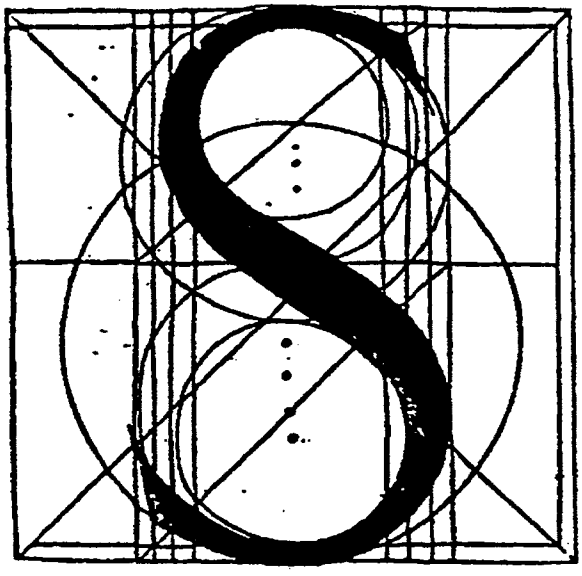
Adobe Inc.

1982 gründeten Dr. John Warnock mit Dr. Chuck Geshke Adobe Systems Inc. Man hatte von PARC das Knowhow übernommen, aber kommerzieller Erfolg kommt nicht von Technologie, sondern von ihrer Anwendung. Adobe wußte, daß verschiedene Firmen an Laserdruckern arbeiteten und ein einheitliches Treiberprotokoll für diese neue Hardware fehlte. Die alternative Anwendung wäre ein Displaytreiber für Workstations gewesen. Das erschien aber als wesentlich schwerer am Markt durchzusetzen. PostScript wurde damit zunächst als portable Seitenbeschreibungssprache für Laserdrucker konzipiert und 1983 auf den Markt gebracht. Aber selbst wenn man die Software als OEM-Produkt an die Hersteller von Laserdruckern lizensierte, war damit der kommerzielle Erfolg noch nicht sicher. Der billigste Laserdrucker von Xerox kostete $15000 und die Stückzahlen waren demenstrechend gering.

Macintosh

Jedoch hatte nicht nur Adobe bei Xerox geklaut. Auch Apple hatte für den gerade entwickelten Macintosh dort über den Zaun geschaut. Der Mac war grafikfähig und billig. Zusammen mit Apples preiswertem Drucker LaserWriter und Adobes Software wurde „Desktop Publishing" möglich und zum Begriff. Widerstände gab es ursprünglich genug. Alle, die bei Apple konventionell dachten, hielten die Idee einen $2000 Computer mit einem $7000 Drucker zu verkaufen für verrückt. Besonders wenn der Drucker mit seinem 12 MHz 68000 wesentlich mehr Rechenleistung als der Mac eingebaut hatte. DTP wurde jedoch zur Killerapplikation für den Mac. Und der Mac ermöglichte Adobe den Durchbruch in den Massenmarkt. Die 5% Lizenz pro Drucker und die Fonts brachten Adobe 1990 $160 Mio ein. Was die Melkkühe Apple & Co natürlich erboste. PostScript war als Drucker-





treiber zum de facto Standard geworden. Um zumindest die Fonts von Adobe zu verdrängen, entwickelten Apple und Microsoft zusammen das TrueType Fontformat für ihre Betriebsysteme.

Display PostScript

Nach dem Erfolg bei den Druckern war der nächste logische Schritt eine Version für Bildschirme: Adobes Display PostScript. Während man bei Druckern hohe Auflösung (300 - 3000 dpi) benötigt, braucht ein Treiber für Bildschirm hohe Geschwindigkeit, aber nur niedrige Auflösung (75 - 100 dpi). Die technischen Probleme bekam man in den Griff. Der Bruch mit Apple bewirkte jedoch, daß man nicht in die Systemsoftware des Mac kam. Damit fehlte die breite Anwendung. Als Trostpreis blieben Unix-Schachteln. Die Verwendung in Steve Jobs NeXT-Computer 1988 brachte nur für kurze Zeit Publicity. Auch DEC und IBM unterstützen Display PostScript. Aber eben nur nominell. Alternativ entstand bei Sun der auf PostScript basierende Bildschirmtreiber NeWS. Auch dieser erfolglos.

R I P

Es gelang PostScript somit nicht, den Sprung aus der Anwendungsnische Drucker in den Bereich allgemeine Systemsoftware zu machen. Die Sprache ist zwar im Bereich DTP immer noch ein verbreiteter Standard und im Lauf der Jahre von Adobe auf derzeit Level 3 erweitert worden. Obwohl es Clones gibt, ist die Sprache aber zu eng mit Adobe verbunden, um ein offener Standard zu sein. Der Name selbst ist ohnehin geschütztes Warenzeichen von Adobe. Letztlich also ein toter Seitenzweig der Entwicklung und damit schon Geschichte.

PostScript und FORTH

Warnock kannte FORTH tatsächlich nicht. Die Namensgebung spricht sehr dagegen: + heißt hier „add" und SWAP heißt „exch". Er hat sich aber wohl einen ähnlichen Anforderungskatalog wie Moore gestellt. Warnock: „first of all, the language property you want for printing is a straightforward syntax. PostScript is like FORTH because the syntax is so simple to parse." Die Sprache sollte vor allem auch einfach erweiterbar sein. In manchem entsprechen sich FORTH und PostScript. Man kann vorhandene Namen mit neuen Funktionen belegen. PostScript wird als Source verteilt, nicht als Binärfile, weil das die Portabilität verbessert.

Stellenweise ist PostScript noch FORTHartiger als FORTH: es hat gleich vier Stacks. Die Dictionaries z.B. werden immer über einen eigenen Stack verwaltet. Auch die Postfix-Schreibweise ist bis zum Extrem weitergeführt worden. Z.B. bei if...else...then:

```
a b gt {x} {y} ifelse      % PostScript
a b >  IF X ELSE Y THEN    \ FORTH
```

PostScript ist ähnlich wie gängige Programmiersprachen reichhaltiger ausgestattet als die meisten FORTHs. Fließkommazahlen, Arrays und Strings sind hier vorhanden. Das, was das KnowHow von Adobe und den Wert von PostScript ausmacht, ist jedoch nicht die Struktur der Programmiersprache, sondern die Qualität der Grafikroutinen, auf die man durch die Sprache Zugriff hat. Die Ursprünge dieses Teils von PostScript reichen bis zu dem deutschen Ikarus-Programm von Dr Peter Karow von 1974 zurück. Dieses Programmpaket führte die Wandlung zwischen Bitmaps und Kurven durch. sowie die Interpolation bei der Größenskalierug innerhalb von Typenfamilien. Letzteres und die Graustufendarstellung in Bitmaps erfordern Fingerspitzengefühl. PostSript-Clones erreichen hier meist nicht die Qualität des Orginals.

Intermediate Code

PostScript-Source wird normalerweise automatisch von anderen Programmen erzeugt. Die Sprache vermeidet die Verwendung von Sonderzeichen als Befehle und verwendet statt dessen relativ lange Schlüsselworte. Interaktives Eintippen wird deshalb keine Freude sein. Praktisch wird die automatisch erzeugte Source vom Programmierer aber nur gelesen, wenn er auf Fehlersuche ist. Deshalb kann die Verwendung langer Schlüsselworte durchaus sinnvoll sein, weil sie die Lesbarkeit fördert.

Zweifellos ermöglicht eine Programmiersprache als Grafiktreiber maximale Flexibilität. Die Effizienz einer umfangreichen Sprache ist jedoch zweifelhaft. Compiler, auch die, die PostScript erzeugen, verwenden normalerweise nicht die angebotenen komplexen Befehle, sondern erzeugen lange Ketten von primitiven Befehlen. Dieser Feststellung liegt der Entwicklung von RISC-Prozessoren zugrunde. Man kann sich bei PostScript nicht des Eindrucks erwehren, daß es sich hier um ein CISC-Monster handelt. Und das zumindest ist unFORTHartig.

Literatur: Die offiziellen Handbücher von Adobe werden vom Verlag Addison-Wesley herausgegeben. Das „Referenzmanual" ist rot, das „Tutorial and Cookbook" blau und „Language Program Design" grün. Weshalb sie auch als Red Book, Green Book und Blue Book bezeichnet werden.

Zitate aus diesen oder aus: Interview mit Warnock in: Lammers „Programmers at Work"

❑

Forth International

von Fred Behringer

Planegger Str. 24; D-81241 München

behringe@statistik.tu-muenchen.de

Gehaltvolles

zusammengestellt und übertragen
von Fred Behringer

Het Vijgeblaadje 5

der hcc FORTH gebruikersgroep, Niederlande, Juni 1997

2 Verslag van de TO-discussie
Willem Ouwerkerk

TO scheint das Wort zu sein, das unsere Nachbarn zur Zeit bewegt. Willem Ouwerkerk berichtet von regen Diskussionen und vier Implementierungsvorschlägen am 12. April 1997 (auf dem regelmäßigen Treffen der holländischen Forth-Gruppe). Diskussionsteilnehmer und Implementatoren waren Marcel Hendrix, Ben Koehorst, Albert Nijhof, Coos Haak und Willem Ouwerkerk. TO wurde offensichtlich von Paul Bartholdi erstmals vorgeschlagen und soll den Transfer von Variable zu Variable lesbarer gestalten. Beispiel: KALTER @ KAFFEE @ + JAVA ! wird zu KALTER KAFFEE + TO JAVA .

4 Finite State Machines
Paul Wiegmans

Der Autor erklärt den Begriff des endlichen Zustandsautomaten anhand eines kleinen Forth-Programms, das Zahleneingaben der Form x.y (x = 0,...,9; y = 0,...,9) akzeptiert (Beispiel: 5.6), Falscheingaben (Beispiel: .56) aber ignoriert. Der Rezensent: Sowas habe ich bisher immer einfach nur gemacht, ohne zu wissen, dass es sich dabei um die Verwirklichung eines endlichen Zustandsautomaten handelt. Interessant, die Dinge mit "systematischen Augen" sehen zu lernen.

5 De Forth bibliotheek
Roelf Toxopeus

Die Bibliothek der Forth-Benutzergruppe wird von der Bibliothek der Fakultät für Naturwissenschaften und Astronomie an der Universität von Amsterdam verwaltet. Bücher ausleihen geschieht über die jeweilige Stadtbibliothek des Ausleihers und kostet 10 Gulden pro Buch. Darin sind die Versandkosten enthalten. Forth-Mitglieder können sich die Bücher in Amsterdam kostenlos ausleihen.

Email-Kontaktadresse der Forth-Benutzergruppe: W.A.A. Kuppen<kuppen@phys.uva.nl> .

Die Postanschriften der sieben Vorstandsmitglieder liegen auch vor.

❑

Forth Dimensions der Forth Interest Group, USA

Januar/Februar 1997

8 Garbage Collection in Forth
Jim Schneider

Der ANS-Forth-Speicherzuordnungs-Wortsatz ist ein guter Anfang in Richtung auf eine Standardmethode zur Speicherbereinigung ("garbage collection") hin, aber eben nur ein Anfang - und der Autor stieß auf einige Unzulänglichkeiten. Er sagt: "Obwohl ich die irritierenden syntaktischen Mätzchen von LISP nicht bewundere, gefällt mir doch, daß LISP die Speicherbelegungen ständig im Auge behält und Speicherteile, die nicht mehr benötigt werden, sofort freigibt. Ich denke, eine Speicherbereinigungsroutine in Forth wäre von unschätzbarem Wert. Die

Ich möchte in dieser Kolumne in unregelmäßiger Folge über Forth-Aktivitäten in anderen Ländern berichten. Ich bin fleißig am Sammeln, würde mich aber natürlich auch über Zusendung von Material von anderer Seite freuen.

Verständlich sind für mich die Sprachen Englisch, Holländisch, Italienisch und Französisch.

Fred Behringer

Programmvorschläge im vorliegenden Artikel sind ein erster Schritt in dieser Richtung."

11 Back to Forth: An Object-Oriented Forth
Anatole L. Medyntsev

Dieser Artikel beschreibt ein experimentelles objektorientiertes Forth-System für MS-Windows. Die vertretenen Ideen beruhen auf Erfahrungen des Autors mit FoxPro 2.5, Visual C^{++}, MS Access, Visual Basic und Delphi bei der Entwicklung von Datenbanken für den Finanzbereich. Für die Entwicklung von Datenbank-Anwendungen eignet sich seiner Meinung nach am besten eine Kombination aus einer interpretativen objektorientierten Sprache und dem üblichen C oder Pascal. Im Gegensatz zu anderen Möglichkeiten ist Forth keine in sich abgeschlossene Sprache. Es stellt eine einfachere und flexiblere offene Technik dar.

15 Zen Floating Point
C.H. Ting

Weniger ist mehr, besonders bei der freien FPU des 486ers. Deren Gleitkomma-Register sind in Stackform arrangiert. Aber ihr Befehlssatz ist unnötig kompliziert. Er gestattet den Zugriff wie bei einem normalen Registersatz. Ein klein bißchen weniger ist ein klein bißchen mehr: Wenn wir die Registeradressierungsbefehle weglassen, können wir ein sehr einfaches und elegantes Gleitkomma-Paket bauen.

In eigener Sache (der Rezensent): Ich übersetze immer die Inhaltskurzangaben, die in den Inhaltsverzeichnissen der FDs zu finden sind. Die FD selbst beziehe ich (noch) nicht. Man kann nicht alles haben. Die letzten drei Kopien (der Inhaltsverzeichnisse) hat mir Henry Vinerts dankenswerterweise geschickt. Ich wäre am obenstehenden Artikel von Professor Ting interessiert. (C.H. Ting ist mir als interessanter Autor bekannt, der Forth mit Mathematik zusammenbringt.) Kann mir jemand eine Kopie des Artikels schicken? - Vorher Ankündigung auf E-mail, um Doppelreaktion zu vermeiden.

19 A Stack-Based Dataflow Operating System
Barry Kauler

"Visual Programming" kann höchst produktiv sein und auch von Leuten verwendet werden, die wenig über die herkömmlichen, textorientierten Sprachen wissen. Aber das hat seinen Preis. Die "visual-dataflow"-Paradigmen haben es dem Autor angetan, Betriebssyteme für Embedded Control auch. Er überlegte sich ein "Dataflow"-Betriebssystem, das auf Embedded-Control-Anwendungen zielt.

22 Can POSIX Threads Be Used as a Standard Forth Multi-tasker?
Dr. Everett F. ("Skip") Carter, Jr.

Threads sind Prozesse, die zu mehreren nebeneinander im selben Speicherbereich laufen - wie virtuelle Forth-Multimaschinen im selben Dictionary, die auf dieselben Daten zugreifen. Die POSIX-Thread-API besteht aus einem Satz von Funktionsaufrufen, die weitestgehend mit denen in einem traditionellen cooperativen Forth-Multitasker übereinstimmen. Diese Übereinstimmung könnte zum Aufbau eines ANS-Forth-Multitaskers, der den internationalen POSIX-Bestimmungen genügt, ausgenutzt werden.

Forth Dimensions der Forth Interest Group, USA

März/April 1997

6 Fuzzy Forth - ANS-compliant Extensions
Rick VanNorman

Unscharfe Logik (fuzzy logic) bietet immer wieder Stoff für Experimente und Diskussionen. Die aufgeworfenen Fragen lenken keinesfalls von den Vorteilen ab, und viele Erzeugnisse konfrontieren den Programmierer mit Fuzzy-Logik-Interfaces. Selten liegen diesen Paketen jedoch Quelltexte bei, anhand derer der Programmierer die Programme analysieren und verstehen lernen könnte. Der vorliegende Artikel liefert einen Überblick über Fuzzy Logic, einen Satz von ANS-Forth-konformen Forth-Quelltext-Codeerweiterungen zur Implementierung einer Fuzzy-Logik-Schlußfolgerungsmaschine und ein einfaches Beipiel eines Fuzzy-Forth-Reglerprogramms.

Bemerkung (des Rezensenten) über den Autor: Ich habe mir mal das OS/2-Forth von Rick VanNorman (rick@thunder-ink.com) von taygeta abgezogen und angesehen: Ich muß sagen, alle Achtung! Das IST es. Auch dort kommt viel "fuzzy" vor. An sich ist das ein Forth für den Protected Mode, das auch unter DOS-DPMI (man denke an Win 3.11) läuft. Im Paket sind die DPANS6-Spezifikationen im ASCII-Format - welche Ausnahme! - enthalten (655 KB); auch die DPMI-Spezifikationen (171 KB) und eine Erweiterung des von Laxen and Perry stammenden 8086-Assemblers auf die 486-Befehle. Nicht fehlen tut der seit F83 übliche Metacompiler, und man trifft auch sonst viel Bekanntes, eben auf 32 Bit erweitert, an. Dieses System reiht sich ein in die Linie F83 --- Turbo-Forth & F-PC --- VanNorman.

14 Object-Oriented Programming in ANS Forth
Andrew McKewan

Programmieren heißt auf unseren heutigen bestausgerüsteten Anlagen zu einem großen Teil, sich mit Komplexität herumzuschlagen. Auf dem Wege, die Komplexität in den Griff zu bekommen, führten die Überlegungen des Autors diesen dazu, die (recht allgemeinen) Class/Object/Message-Ideen von Yerk zu übernehmen. Zunächst übertrug er diese auf Win32Forth, ein Public-Domain-Forth-System für Windows NT und Windows 95. Aber sein Ziel ging weit darüber hinaus. Er wollte jedwedes ANS-Forth-System mit der Möglichkeit ausstatten, die objekt-orientierte Syntax und den objekt-orientierten Programmierstil zu verwenden. Erfolg ist dabei gleichbedeutend mit der Möglichkeit, die Entwicklung einer Bibliothek von ANS-Forth-Objekten in Angriff zu nehmen.

Zusatz: Methoden, zu einem objekt-orientierten Forth zu gelangen: Viele haben sich darin versucht und viele hatten darin Erfolg - zu einem gewissen Grad. Ein Forth-Programmierer kann die Objekt-Orientie-

rung dazu verwenden, an einem normalen Forth Ergänzungen anzubringen, sie als weiteres Werkzeug in einer Werkzeugkiste zu betrachten. Er kann aber auch einfach den Kerngedanken übernehmen und alles von Grund auf neu gestalten.

30 Yet Another Modest Proposal
Richard Astle

Was das objekt-orientierte Programmieren in Forth betrifft, gibt es bis jetzt noch keine gemeinsame Auffassung. Andere Sprachen klaffen in bezug auf das Herangehen an die Objektorientierung ebenso weit auseinander. Statt aber auf einen Standard oder ein neues 'wordset' zu warten, können wir ebenso gut auch das Wesentliche von den Objekt-Techniken abgucken. Ein verwandtes Gebiet, über das wenig geprochen wird und das von einer Standardisierung weit entfernt ist, ist die Modularisierung: Quelltext-Programmteile, die Code, Schnittstellen, lokale und globale Definitionen enthalten. Sie fördert die Übersicht, entwirrt die Ansammlungen von Headers und erlaubt eine Wiederverwendung von Namen. Kurzum: Der Autor hat der Vocabulary/Wordset-Welt den Rücken gekehrt.

32 MPE's Forth Coding Style

Diese Forth-Vorlage behandelt das Erscheinungsbild von Forth-Quelltext in Text-Dateien. Sie geht auf Code und Kommentare ein, auf die Datei selbst, auf die Gründe für eine Standardisierung und die Gründe für bestimmte Entscheidungen und Empfehlungen in den Standards. Dieses Dokument spiegelt den bei MPE zur Zeit gepflegten Programmierstil wider.

36 A Gentle Introduction to Digital Filters
Skip Carter

Digitale Filter sind ein Hilfsmittel, mit denen man digitale Signale so beeinflussen kann, daß sie einer Reihe von Vorgaben genügen. Je nach Problem kann ein Filter beispielsweise unerwünschtes Rauschen reduzieren, einen Signalanteil herausschälen oder unterdrücken oder bestimmte Signalbestandteile verstärken. Was Digitalfilter so unzugänglich macht, ist der Umstand, daß ein guter Entwurf doch etwas Mathematik erfordert - normalerweise in der komplexen Zahlenebene. Was abschreckt, ist die Leichtigkeit, mit der manche Autoren über die doch recht schwierigen mathematischen Fragen hinweggehen. Der vorliegende Artikel soll helfen, den Zugang zur nötigen Mathematik zu finden - ganz langsam, Schritt für

Gehaltvolles

zusammengestellt und übertragen
von Fred Behringer

Forth Dimensions der Forth Interest Group, USA

Mai/Juni 1997

8 A Forth Memoir
John Nangreaves

Es gibt mindestens so viele Forth-Geschichten, wie es Forth-Anwender gibt, und jede für sich wirft ein eigenes Licht auf Forth und erklärt, wie sich Forth in die Werkzeugkiste des aktiven Programmierers einpaßt. Im vorliegenden Fall hat der Autor eineinhalb Jahre lang in Assembler (sogar in hex) programmiert, bevor er dahinter kam, daß es da doch noch einen besseren Weg geben müsse. So fängt die Geschichte an ...

10 A Simple Implementation of the Kermit Protocol in Pygmy Forth
Frank Sergeant

Im vorausgegangenen Artikel spendet John Nangreaves mehr als nur ein freundliches Kopfnicken dem Gemeinschaftsgeist der Forth-Anhänger und besonders derjenigen Mitglieder, die nützliche Programme beisteuern und nicht müde werden, diese ständig weiterzuentwickeln. Einer von diesen ist Frank Sergeant, dessen Pygmy Forth sich unter denjenigen Freunde verschafft hat, die Forth gern rank und schlank sehen. Im vorliegenden Artikel implementiert er Kermit in Pygmy Forth.

12 Transportable Control Structures
Randy Leberknight

ANS-Forth formalisiert einen Aspekt der Erweiterbarkeit von Forth: Die Erzeugung neuer Kontroll-Struktur-Worte, ohne auch nur ein einziges neues Wort in Assembler zu schreiben. IF und WHILE, beispielsweise, führen zur Laufzeit beide einen bedingten Vorwärtssprung aus. Diese gemeinsame Aktivität kann zusammengefaßt und beiden zur Verfügung gestellt werden. Ein solches Vorgehen kann bei Entwicklungsarbeiten Zeit sparen helfen.

20 Working Comments (long)?
Julian V. Noble

Programmteile können schon vor dem Compilieren ausgetestet werden, um ihren Einfluß auf den Daten- und Returnstack herauszubekommen, ohne einen Systemzusammenbruch oder versteckte Fehler zu riskieren. Es wird ein vorläufiges Programm besprochen und es werden mögliche Verbesserungen diskutiert. Ein gutes Beispiel kreativer Arbeit, die ihren Ursprung in einer beiläufigen Bemerkung in comp.lang.forth hatte.

24 Standardizing OOF Extensions
Anton Ertl

Andrew McKewan vertrat im letzten Heft die Meinung, man müsse sich erst auf ein bestimmtes Modell einigen, bevor man darangehen könne, eine objekt-orientierte Bibliothek aufzubauen. Der Autor des vorliegenden Beitrags vertritt die gegenteilige Ansicht: Man schreibe eine gute objekt-orientierte Bibliothek, auf die jedermann einspringt, und schon wird das Objekt-Modell, auf welchem die Bibliothek beruht, zum Standard.

From the other side of The Big Teich

Forth USA SVFIG May&June

Dear friends, it is time to let you know again that Forth still lives in Silicon Valley. Unfortunately, I missed the May SVFIG meeting and have been unable to get a report from someone else. If the planned program went through as announced in the bulletin, then there were only two speakers: the indefatigable Dr. Ting was to give a tutorial on how Forth vocabularies might be "beefed up" to support the current trends of object-oriented programming; and Dr. Skip Carter was to report on what he learned in a conference that he attended, dealing with robotics and remote systems. I missed the meeting because my wife and I took a little vacation in Arizona, in connection with a badminton tournament that I participated in.

Since I lost in the quarterfinals, we had some extra time; the day before the SVFIG meeting we visited Kitt Peak Observatory, where, I understand, around 1973, the number of Forth programmers in the world doubled (from two to four!).

Chuck Moore says in his article in 1980 Byte magazine that in 1970 there was only one Forth programmer and in 1971 the second one--Elizabeth Rather--came along. Kitt Peak is quite a place. It still boasts the largest number of working telescopes in one place--25 of them in all. If Forth is still there in some old machine, that I was unable to find out. We also visited Biosphere 2, the Desert Museum, the Pima Airplane Museum and a few other places of interest around Tucson, but not in search for Forth...

The June 28th SVFIG meeting at Cogswell College drew the usual number--20 to 25 attendees, even though some of the gurus, I was told, had gone to the Rochester conference. Dr. Ting was there again, "walking on one leg," as the title of his talk indicated. He is writing his own Forth to support Open Firmware and is using the compiler only. The "other leg," the interpreter, appears not to be necessary. Carriage return executes what has been compiled. The reason Dr. Ting has tried this, he said, is "because 'normal' computer-science people do not understand interpreters." Having worked on this for a while, Dr. Ting has come to the opinion, however, that it really is not too good an idea to throw away one leg, not in Forth, anyway.

The second speaker, Jet Thomas, had come all the way across the U.S.A. from Maryland, and he amused everyone with a lively presentation of his work on writing a program that would teach a "virtual chicken to dance." It was so interesting that the next week I went to the bookstore and bought the book that inspired Jet to do the program. It is called "Don't Shoot the Dog," by Karen Pryor, an authority on training of dolphins, dogs, cats, children, spouses, anybody (Forth programmers?). Down-to-earth talk about behavioral psychology. As Jet mentioned, he picked the chicken as his subject, because of a quotation in the book that says something to the effect that "no one should be allowed to have children until they have taught a chicken to dance." The program, of course, is in Forth. It offers the virtual chicken appropriate 'reinforcements' for various kinds of behavior, until the steps leading to the "chicken dance" are pretty well learned. So far the output is only in ASCII descriptions of what the chicken does, but animation and sounds are possible, and may be forthcoming. At least I made that suggestion to Jet.

An unannounced, but interesting extra presentation this time came from Kevin Bennett, who probably holds the record of the most commuting miles to our meetings. He comes from about 650 miles away in Northern Oregon, where he works in a factory that makes paint-brush handles; thousands of them are shipped to all kinds of places every week. Purdy-Wooster handles, made by McRae and Sons, from local Oregonian timbers, I presume. Kevin has written some Forth to collect data from a Dallas DS 1620 temperature sensor and to control the drying times and temperatures in the kilns where large loads of timbers are processed. I must add that Kevin is one of our younger members, and chances are that he will move Forth into more applications that we'll hear about. Last, but definitely not least, I wish to mention that the meeting was honored by the presence of Bill Ragsdale, who is one of the founders of FIG. You will find his name in the first editions of Forth Dimensions, back in 1978. I wish that I had known him when he wrote the Forth assembler for the 6502, then I might have learned Forth from the ground up on my first computer, the Apple IIC. I noticed that Bill has a knack of explaining the inner makings of Forth, so that even people like me can understand.

... about 650 miles away in Northern Oregon

Claus, I really did not wish to fill up this much space. You will have to "clip" what doesn't fit your magazine *[Yes, of course I do read this, but if you want to be clipped, it's on* you *to write s.th. clippable!! /clv].* I guess I am trying to make up for having missed the May meeting, or the future ones that I might miss because of Federball. (I'll get back to you on the "Forthy Feather" letter later. Ich weiss wirklich auch nicht, was der Codename bedeutet!)

Tschuess!

-- *Henry*

Forth USA SVFIG Juli 97

Hello, friends,

Before the hot summer days make me too lazy, I better send you what I have managed to put together about last Saturday's Silicon Valley FIG's meeting:

It may have been the weather, the noon-time barbecue, or the fact that Charles Moore was coming to give a little talk, but more than likely it was the latter that caused a larger-than-usual group (30 to 35 people) to show up for the meeting.

In the morning session Skip Inskeep gave a good report of how he developed an Internet interface for iTV by tethering Forth from PC through the serial port and also with I-square-C through the keyboard to the I21 microprocessor, developing his own PIG'E'Forth (piggy-back e-Forth) in the process. Again, too much for yours truly, the eternal novice, to understandingly describe.

Then John Rible, Skip Carter, and Glen Haydon (I understand, the only Rochester Conference attendees from SVFIG), reported on what had happened at the Rochester Conference. Conference was "subdued", they all said, fewer than the usual number of attendees. A number of papers. Larry Forsley gave a seminar on the year-2000 problem. The highlight was the presence of 4 "forth-speaking" representatives of Varian's ion-implementation device group, which is located in Gloucester, Massachusetts. Varian has been making these bus-size (passenger-bus-size, as I gather) machines that are used in doping wild chemicals into wafers that eventually turn into chips. Every chip maker uses these $10,000,000 machines and there are some 400 of them, "regrettably" for Varian, all running on Forth. About a year ago Varian was advertising for Forth programmers to come and convert the systems to C, but, as I gather, Forth programmers are self-respecting people, and Varian was not successful in carrying out such unthinkable degeneration. So they are still looking for Forth programmers (there may be more work in Gloucester for them than there is for Maytag repairmen). Now Gloucester may well appeal to the hardy North-european fishermen types who eat ludifisk or vacation on Helgoland...

Forth, Inc., was there with their Chip-Forth operating system. Rob Chapman showed his TIMBRE. Skip Carter introduced the thought that Forth, albeit technically superior to Java and TCL, cannot compete due to marketing handicaps and due to lack of library material. Therefore, the building of Forth string libraries is essential. Rich Wagner has volunteered to spearhead the effort. If you are interested in helping, please, contact Rich at RICH@TAYGETA.COM. Peter Knaggs' name was mentioned in connection with text styles, but I did not catch the details.

OK. After the barbecue, Chuck Moore was at the podium for almost 2 hours, and it could have been longer, but we must vacate the classroom at 4 p.m.

First, we gave Chuck a Kitt Peak Observatory pin to commemorate the birth of the Forth compiler there, about 25 years ago, and then Chuck told us that he is coming back to Forth after about a 5-year hiatus, when he was doing other things, mostly designing new chips, using his own design tool--OKAD. I think that he was, and still is disenchanted with the ANSI Forth standard. It may have been about 5 years ago that I heard Chuck say that if people are trying to make a standard of something, then it is time to move on and do other things. Today it looks to him that the standard is a disaster. "All of the fears of the Standard have come to pass, and none of the advantages have gained support in the community. It should have been a publication standard, not an execution or interpretation standard."

Now Chuck says that OKAD was a mistake. Seven keys for input are not enough. Five years ago he was disillusioned because Forth was getting to be too complicated, but now he is going to put Forth back into OKAD. He also retracts his fairly recent statement that "The map is not the territory (like the description of a program is not a program)." "The map in some instances is a better representation than the territory."

"Please, write your own Forth," he encourages. "A standard does not mean that we cannot do otherwise." He is going to write another Forth for OKAD, as I understand. Seven keys may not be enough, but fewer than 84 may well do. Parentheses will not be needed, comments can be enclosed in the same apostrophes. Colons and semicolons may not be needed, if we can use color for delimiting compiled and interpreted words. Conditionals can be avoided. Why loop, if the number of repetitions is low? But the "print-screen" button will be important. OKAD will most likely be used for chip diagnostics; a readily obtainable printed record will be invaluable.

Many things have changed since Forth started. Memory is so plentiful now that all of the working Forth can be placed in RAM, disk access is not necessary. The Internet is available to everybody, one can download what is needed, exchange code with others as desired. Java is not the only Web language, there are many, and Forth can be one of them. "I argue against any Forth, modern or classic, that can do everything," Chuck says, but do anything you wish, by loading what you need." "Write your own browser. Customize, suppress the ads and other junk." His system will have to have a browser, a word-processor, and OKAD and fit on one disk within 1MB, with, say 10KB of start-up code. Of course, everyone should be able to order his own system and not be subjected to frustrations with the systems that they are forced to deal with.

Finally, of the three things that one can do with his computer:

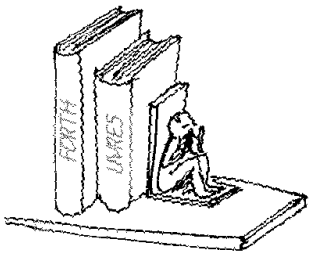
1) Make money,
2) Get famous,
3) Have fun,

Chuck says that the first 2 may be quite disappointing; he would rather pursue the third. And soon Forth will have filled half of his life, and he intends to go on having fun with it.

- - - - -

(As usual, if there are any errors in the above, they are entirely my own. If no one makes any comments, I'll assume that there aren't any, or I'll assume that no one reads my column. *[perhaps a few more paragraphs? /clv]*..)

Forth-Bücher aus Frankreich



MP7
17, allèe de la Noiseraie
93160 NOISY LE GRAND

Frankreich

Unter der obengenannten Adresse, der Firma von Marc Petremann, dem Erschaffer von Turbo-Forth, sind die unten angeführten Bücher erhältlich. Interessant erscheint mir der Titel über Forth für wissenschaftliches Arbeiten. Das Buch ist in englischer Sprache verfasst. Auch das OOP-Buch, ebenfalls englisch, gefällt mir gut. Ich habe es mir aus unserer TU-Bibliothek ausgeliehen und beabsichtige, die Vorschläge des Autors in mein Transputer-Forth-System F-TP 1.00 einzubauen. Die anderen beiden Bücher sind französisch. (Man kann Französisch auch dadurch lernen, dass man Bücher über Forth liest. Hätte man mir das auf der Schule gesagt, wäre ich mit größerem Eifer dabeigewesen.)

Julian V.Noble: Scientific Forth

309 Seiten, englisch, mit Diskette 559,-- FF

Der Autor ist Physikprofessor an der Universität von Virginia

Dick Pountain: Object-Oriented Forth

119 Seiten, englisch 151,-- FF

OOP hat ja jetzt durch Java wieder einen neuen Anstoß bekommen. Forth hat alle Anlagen zum OOP. Hier erfährt man Konkretes.

Leo Brodie: Debutez en Forth

305 Seiten, französisch 199,-- FF

Der bekannte Brodie in französischem Gewand

McCabe: Le Forth

289 Seiten, französisch 151,-- FF

Für Anfänger und Fortgeschrittene

beh

Archiv der Forthliteratur

2. Teil

von Gerd Bretschneider
neue Mail-Adresse:
G.Bretschneider@BBrandes.in-brb.de

Hallo Leute!

Es folgt eine Ergänzungsliste zum Archiv der Forthliteratur (Ost).

Ich hoffe, daß einiges für Euch neu ist.

Sichworte: Archiv Literatur Bücher

- VEB Mikroelektronik “Wilhelm Pieck” Mühlhausen (Autor: Klaus Katzmann): Kleincomputer KC 85 - Beschreibung zu M 026 - FORTH

Inhalt: Beschreibung Handhabung des ROM-Moduls, FIG-Forth für Kleincomputer KC 85/2/3/4 einschließlich Programmierkurs (Liegt vor)

- Claus Kuehnel: Messen und Steuern mit Forth auf dem Kleincomputer (Reihe “Amateurbibliothek”, Brandenburgisches Verlagshaus, 1990)

Inhalt: Vorstellung eines in Forth programmierten Meßsystems für den KC 85/2/3/4 auf Basis des Forthmoduls und der zum System gehörenden ADU- bzw. DAU-Module, einschließlich Quelltexte. (Liegt vor)

- B.Schubert: FORTH für den MRB “Z 1013” (Zeitschrift “Funkamateur”, Militärverlag der DDR, Heft 12/88 S. 586)

Inhalt: Kurzvorstellung eines F-83-Systems für den Mikrorechnerbausatz Z 1013. (Liegt vor)

- Gerd-Ulrich Vack: Programmieren mit Forth (VEB Verlag Technik Berlin, 1990)

Inhalt: Das Standardwerk zu Forth. Es werden Forthsysteme für i8086 und z80 detailliert im Aufbau beschrieben. Umfangreiche Quelltexte zum Systemaufbau. (Liegt vor)

- Claus Kuehnel: Eignung verschiedener Hochsprachen für die Bearbeitung von Meßdaten (Zeitschrift “Der Elektroniker”, 3/1988, S. 97, Verlag????)

Inhalt???? (fehlt)

- Dr. Helmut Hoyer: Forth-83-System für den JU+TE Selbstbaucomputer (Zeitschrift “Jugend und Technik”, 1/1990, S. 70, Verlag:????)

Inhalt: Kurzvorstellung eines Forth-83-(Minimal-) Systems für einen Eigenbaurechner mit CPU U 8830 (Z8-Derivat).

(Liegt vor; Im gleichen Heft beginnt auch ein darauf aufbauender Programmierlehrgang, liegt mir aber leider nicht vor)

- Claus Kuehnel: Erste Erfahrungen mit Forth (Zeitschrift “Radio, Fernsehn, Elektronik”, 9/1986, S. 553, VEB Verlag Technik Berlin)

Inhalt: Vorstellung der Programmiersprache Forth (Liegt vor)

- Th. Leopold: Diskussion: Erste Erfahrungen mit Forth (Zeitschrift “Radio, Fernsehn, Elektronik”, 5/1987, S. 297, VEB Verlag Technik Berlin)

Inhalt: Erfahrungsbericht zu Forth-Systemen (Liegt vor)

- Claus Kuehnel: Definition von Anwenderworten in Forth (Zeitschrift “Radio, Fernsehn, Elektronik”, 9/1988, S. 583, VEB Verlag Technik Berlin)

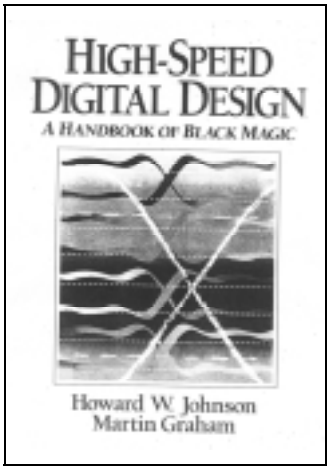
Inhalt: Beschreibung der internen Arbeitsweise eines Fig-Forth-Systems (Liegt vor)

❑

Bücher

Johnson/Graham

"High-Speed Digital Design
A Handbook of Black Magic"
Prentice Hall 1993
ISBN 0-13-395724-1
englisch 450 Seiten
76 DM



Der reißerische Untertitel und der Umstand, daß der Autor eine Kolumne in der amerikanischen Kennzifferzeitschrift "Electronic Design" schreibt, dürfte eine weite Verbreitung bewirken. Erfreulicherweise hat sie das Buch verdient. Es ist praxisorientiert, gut illustriert und enthält viele Details die man sonst nirgends finden wird. Wer braucht so ein Buch ? Leider inzwischen fast jeder. Tischcomputer, embedded Controller und DSPs erreichen heute Busgeschwindigkeiten, bei denen Leiterplatten und Steckverbinder genauso wichtig werden wie ICs. Wissen schützt hier vor teueren Überraschungen.

jrd (Rafael Deliano) April'97

Bücher

Markus Kopp: Die virtuelle Java-Maschine

Java für Profis
In der Zeitschrift
LINUX 5/1997, S. 19-27

Jeder spricht von Java. In der Universitätsbuchhandlung Lachner in München gibt es etwa zwei Meter an Java-Literatur. Wesentlicher Bestandteil des Java-Komplexes ist die virtuelle Java-Maschine (Java virtual machine oder JVM), das Interface zwischen maschinenunabhängigem Bytecode (aus dem Internet in Form von "Klassen-Dateien" beziehbar) und der eigentlichen Maschinensprache der jeweiligen Plattform. Der Forthler ist weniger an Java selbst, als vielmehr an der Bytecode-Maschinerie interessiert. Forth lebt seit eh und je von "Stackorientierung" und "virtueller Maschine" ("FVM" (?)). Über den Java-Bytecode gibt es bei Lachner momentan gar nichts. Umso interessanter ist der oben zitierte Artikel. Im ersten Teil der beabsichtigten dreiteiligen Serie geht es um die vom Java-Compiler erzeugte maschinenunabhängige Java-Klassendatei. Im zweiten Teil soll die Arbeitsweise der JVM beschrieben werden. Im dritten Teil werden dann Übersetzungen von der JVM in Hochsprachen wie C++ besprochen. (Der Autor möchte den C++-Code dann weiterübersetzen und den Binärcode des Compilats (als schnelleres Mittel) anstelle des vom Java-Bytecode-Interpreters gelieferten (notwendigerweise langsameren) Codes verwenden.)

beh

Bücher

Jon Meyer & Troy Downing: Java Virtual Machine

O'Reilly, Köln 1997
426 Seiten (englisch)
mit 3,5"-Diskette mit
Bytecode-Assembler
ISBN 1-56592-194-1
DM 66,--

Das Buch hat 426 Seiten. Etwa die Hälfte davon listet die Opcodes des Bytecodes der virtuellen Java-Maschine auf und erklärt sie. Die Erklärungen sind gut und ausführlich. Für viele Opcodes werden Programmausschnitte gegeben, aus denen deutlich wird, wie der betreffende Opcode bei der Übersetzung eines bestimmten Java-Konstrukts verwendet werden kann. Im übrigen werden folgende Kapitel besprochen: Vor- und Nachteile der Java-Maschine, schneller Überblick, Komponenten der Java-Maschine, Klassen, Sicherheitsfragen, Bemerkungen zur Implementation, Datenbehandlung, Arithmetik, Verzweigungen und Unterprogramme, Ausnahmebehandlung, Threads, Klassendateien und dann eben die Hälfte des Buches Bytecode. Im Anhang werden auf jeweils 5-10 Seiten aufgezählt: (1) die nach Funktionsgruppen geordneten Opcodes, (2) die numerisch geordneten Opcodes, (3) eine Bedienungsanleitung für den Bytecode-Assembler Jasmin, (4) die JAS-Bibliothek, Bestandteil von und Ergänzung zu Jasmin. Das Buch enthält eine 3,5"-HD-Diskette, auf der in komprimierter Form im wesentlichen jene Software enthalten ist, die man sich auch direkt aus dem Internet beziehen kann unter der Adresse: http://mrl.nyu.edu/meyer/jasmin . Es handelt sich um Jasmin, den Bytecode-Assembler-Disassembler der beiden Autoren.

Fazit: Das Buch ist ausführlicher, instruktiver, besser als das kürzlich von mir rezensierte Buch von Dalheimer (VD 1/97, S.38). Es hat höchstens einen Nachteil, nämlich den, dass es in englischer Sprache geschrieben ist. (Das kann aber bisweilen auch ein Vorteil sein.)

beh

Bücher

H.-R. Wernli:
"Die CCD Astrokamera für den Amateur"
Birkhäuser, 1995
240 Seiten, Deutsch
ISBN 3-7643-5218-3
50,00 DM

Eigentlich für Hobbyastronomen gedacht, die mit gekühlten CCDs Bilder vom Mond und Sternen machen wollen. Sollte aber als Einführung in Bildverarbeitung für einen größeren Leserkreis nützlich sein. Behandelt praxisnah Themen wie Linsen, Bildverarbeitung und Grafikformate ohne großes Vorwissen vorauszusetzen. Leider sind bei dem Bemühen um Anfänger auch verzichtbare Kapitel (" Was ist eine Diskette ?") reingerutscht. Hervorzuheben ist die gute Ausstattung mit Fotos und Zeichnungen, allerdings mußte auf Farbfotos verzichtet werden um den günstigen Preis zu ermöglichen. Gegen die Eindeutschung englischer Fachbegriffe kann man hier wenig Einwände erheben, da sie durchwegs recht gelungen vorgenommen wurde.

jrd

Bücher

D. Patterson, J. Hennessy
"Computer Organization & Design: the Hardware / Software Interface"
Morgan Kaufmann, 1994
ca 700 Seiten, Englisch
ISBN 1-55860-281-X
ca. 73,00 DM

Auf Patterson geht der Begriff "RISC" zurück. Er hat zusammen mit seinen Studenten den Berkeley RISC gebaut, aus dem SPARC und MIPS hervorgingen. Er sitzt heute im Aufsichtsrat von MIPS, weshalb in der Darstellung diese CPU als Beispiel dient. Als Lehrbuch für seine Studenten gedacht, bemüht man sich um eine einfache Darstellung. Die interessanten Kapitel dürften echte Anfänger allerdings überfordern, da die Autoren tief in den Innereien von Caches, Pipelines, superscalarer Architektur und parallelen Prozessoren wühlen. Die Einschätzung anderer CPUs als MIPS ist natürlich auch sehr subjektiv ausgefallen. Trotzdem liest man ein Buch gern, das locker geschrieben ist und in dem sich die Entwickler zu Wort melden.

jrd

Die weniger bekannten Programmiersprachen (714. Folge)

Nachdem wir uns in den bisherigen Folgen mit der einen wirklich wichtigen Programmiersprache und ihren Derivaten beschäftigt haben, wagen wir heute den Blick über den Tellerrand.

...

THE LESSER-KNOWN PROGRAMMING LANGUAGES
LITHP

This otherwise unremarkable language is distinguished by the absence of an "S" in its character set; users must substitute "TH". LITHP is said to be useful in prothethting liththth.

THE LESSER-KNOWN PROGRAMMING LANGUAGES
SLOBOL

SLOBOL is best known for the speed, or lack of it, of its compiler. Although many compilers allow you to take a coffee break while they compile, SLOBOL compilers allow you to travel to Bolivia to pick the coffee. Forty-three programmers are known to have died of boredom sitting at their terminals while waiting for a SLOBOL program to compile. Weary SLOBOL programmers often turn to a related (but infinitely faster) language, COCAINE.

THE LESSER-KNOWN PROGRAMMING LANGUAGES
SARTRE

Named after the late existential philosopher, SARTRE is an unstructured language. Statements in SARTRE have no purpose; they just are. Thus SARTRE programs are left to define their own functions. SARTRE programmers tend to be boring and depressed, and are no fun at parties.

THE LESSER-KNOWN PROGRAMMING LANGUAGES
C

This language was named for the grade received by its creator when he submitted it as a class project in a graduate programming class. C- is best described as a "low-level" programming language. In fact, the language generally requires more C- statements than machine-code statements to execute a given task. In this respect, it is very similar to COBOL.

THE LESSER-KNOWN PROGRAMMING LANGUAGES
FIFTH

FIFTH is a precision mathematical language in which the data types refer to quantity. The data types range from CC, OUNCE, SHOT, and JIGGER to FIFTH (hence the name of the language), LITER, MAGNUM and BLOTTO. Commands refer to ingredients such as CHABLIS, CHARDONNAY, CABERNET, GIN, VERMOUTH, VODKA, SCOTCH, and WHATEVERSAROUND. The many versions of the FIFTH language reflect the sophistication and financial status of its users. Commands in the ELITE dialect include VSOP and LAFITE, while commands in the GUTTER dialect include HOOTCH and RIPPLE. The latter is a favorite of frustrated FORTH programmers who end up using this language.

in der KBBS gefunden, gepostet von Juergen Schardt

When someone says "I want a programming language in which I need only say what I wish done," give him a lollipop.

rfy /js
(wird nicht mehr fortgesetzt)

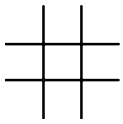
Als Charles wissen wollte, wie Computer Spiele spielen.

Erzählt für Chris, der wissen wollte, wie so eine E-Mail um die Welt kommt

Als mein Sohn zehn Jahre alt war - er heißt Charles - spielte er eine Menge Computerspiele. Dann eines Tages fragte er mich, wie macht das ein Computer, Spiele spielen?

Da erzählte ich ihm einige einfache Flußdiagramme. Und zeigte ihm dann, wie solche Flußdiagramme in Forth "geschrieben" werden. Und mit mir in der Nähe, um seine Fragen zu beantworten, programmierten wir das Spiel Tic-Tac-Toe für den Computer.
Manche nennen das Spiel auch XXO.

Sie kennen das Spiel wahrscheinlich auch alle. Es sieht so aus:



Zwei waagerechte Linien gekreuzt von zwei Senkrechten.

Zwei Spieler ziehen abwechselnd. Der eine setzt X und der andere O in ein Feld. Wer zuerst drei Zeichen nebeneinander bringen kann, hat gewonnen. Dabei ist es egal, ob sie in einer Reihe, Spalte oder Diagonalen stehen.

Ein guter Spieler verliert nie!

Zwei oder drei Tage lang redeten wir über das Spiel, bevor wir irgend etwas auf dem Computer machten. Er fand die Logik heraus und ich zeigte ihm, wie diese in Forth geschrieben werden konnte.

Einen Zufallsgenerator gab ich dazu. Den benutzte er für die Spielzüge des Computers.

Schließlich machten wir dem Computer noch ein Gedächtnis, sodaß er niemals einen Zug wiederholte, bei dem er einmal verloren hatte.

Nach einiger Zeit verlor der Computer nicht mehr.

Es macht keinen Spaß andauernd von einem Computer geschlagen zu werden, aber mein Sohn war sehr stolz darauf, dem Computer nach gut 100 Spielen beigebracht zu haben, ein guter Spieler zu sein. Denn ER hatte es ihm beigebracht.

Nun Chris, vielleicht magst du das auch mit deinem Vater versuchen? Und dann könnt ihr darüber im Forth Magazin berichten.

Wil

(Ü: mka, sept'96)

Landschaft
30 x 40 cm, 1996
Öl auf Leinwand

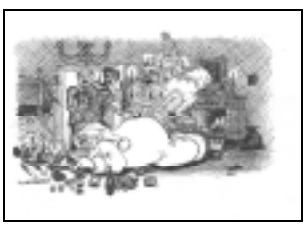
Bilder von Hans-Georg Schmid

- Alle Originale noch erhältlich -

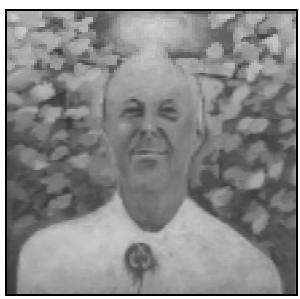
Aufträge und gedeckte Schecks werden gerne entgegengenommen.

Haben Sie auch Interesse, richtige Bilder von H.-G. Schmid zu sehen? Teilen Sie uns Ihre Adresse mit. Wir weisen Sie auf aktuelle Ausstellungen hin.

H.-G. Schmid Tel.: ++49 30 781 16 30 Mail über Claus Vogt:
Ebersstr. 10; D-10827 Berlin clv@FORTH-eV.de



Weihnachtsmann
13 x 18, VD 4/95
Kuli/Buntstift auf Papier



Charles Moore
17 x 17 cm, VD3+4/97
Acryl auf Pappe

Profi oder Hobby?
11 x 12 cm, VD3/97
Acryl auf Pappe