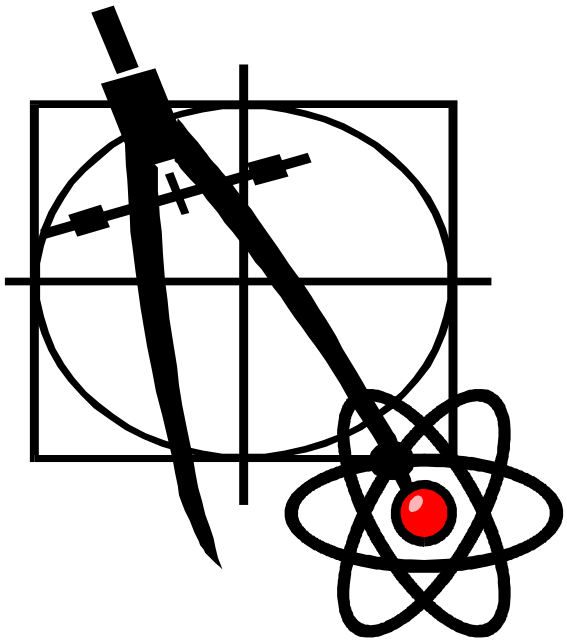
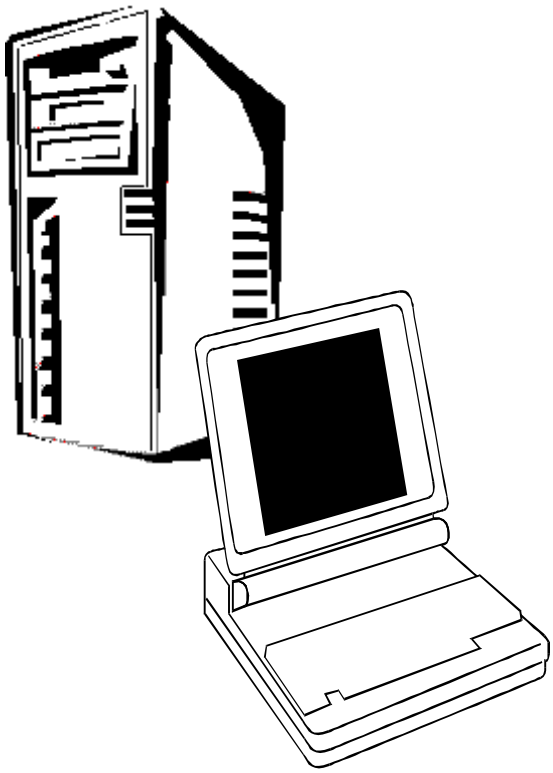
VIERTE DIMENSION

**für Wissenschaft und Technik, für kommerzielle EDV,
für MSR-Technik, für den interessierten Hobbyisten.**





In dieser Ausgabe:

14. Jahrgang - **Ausg. 4/'98** - 10,- DM

Organ der FORTH-Gesellschaft e.V.

Dienstleistungen und Produkte fördernder Mitglieder des Vereins

Editorial

IMPRESSUM

Name der Zeitschrift

Vierte Dimension
Organ der Forth-Gesellschaft e.V.

Herausgeberin

Forth-Gesellschaft e.V.
Postfach 161204
D-18025 Rostock
Tel.: 0381-4007872
E-Mail:
SECRETARY@ADMIN.FORTH-EV.DE
Bankverbindung: Postbank Hamburg
BLZ 200 100 20
Kto 563 211 208

Redaktion & Layout

Friederich Prinz
Hombergerstraße 335
47443 Moers
Tel.: 02841-58 3 98
E-Mail: F.PRINZ@MHB.GUN.DE
FRIEDERICH.PRINZ@T-ONLINE.DE

Anzeigenverwaltung

Büro der Herausgeberin

Redaktionsschluß 1997

März, Juni, September, Dezember
jeweils in der letzten Woche

Erscheinungsweise

1 Ausgabe / Quartal

Einzelpreis
DM 10,- zzgl. Porto u. Verp.

Liebe Leser,

Moers die vierte.

Als ich vor vielen Jahren mein erstes Forth erstand, tat ich es vor allem, weil ich gehört hatte, man könne damit Assembler-Code fabrizieren - also letztendlich alles programmtechnisch Machbare lösen. Nun ja, der Irrtum war - Forth kann es, und Jemand, der weiß wie es geht, kann es. Aber ein Anfänger, so wie ich es damals war, kann erst mal gar nichts (daran hat sich bis heute wenig geändert)! Geholfen haben mir viele relativ gut dokumentierte Quelltexte, ein - zwei schlaue Bücher: Brodie, später Zech und der Hinweis in dem README zur PD-Software (drei Disketten VolksForth für den ATARI-ST) auf eine Mitgliedschaft der Forthgesellschaft mit dem Bezug der VD.

Die erste VD, die ich erhielt, erschreckte mich zunächst: ein Autor wehrte sich gegen die "Entstellung" seines Artikels durch den Redakteur (unabgesprochene Kürzungen), dieser wieder wehrte sich gegen die Anschuldigungen. Mein Eindruck: Vereinsmeierei! Der zweite Blick versöhnte: es gab Tips, kleine Codestücke, die ich sogar auf meinem Rechner :-))> umsetzen konnte: ein dinput, mit französischem Kommentar (Marc Petreman) und ein UNDO von Bernd Pennemann. Da wußte ich, ich bin VD-Leser!

Lange Zeit blieb die VD mein einziger Kontakt zur Forth-Welt. Wesentlich später, als die allgemeine und die berufliche Entwicklung mich dem PC :-(> in die Arme trieben, stieß ich zur Forth-Gruppe Moers. Das Wissen um ihre Existenz nebst Adresse usw. war in der VD zu finden.

Vor knapp einem Jahr bastelte die Forth-Gruppe Moers an "ihrer" ersten billigst-VD. F. Prinz schrieb sinngemäß: Mitglieder: Schreibt! Und viele Mitglieder haben geschrieben! Mit jeder Ausgabe wiederholte sich das Wechselspiel von Spannung (kommt genug) und Freude (es ist genug und es ist gut!). Diese vierte VD aus Moers bietet ein Spektrum von Anregungen, Denk- und Gesprächsstoff und von Kontaktadressen im In- und Ausland (Vijgenblaadje, FIG-UK, Forth-Dimensions).

Nutzen Sie bitte die Möglichkeiten, die die VD Ihnen bietet - erweitern Sie diese Möglichkeiten durch eigene Beiträge!

Für das “Editoriat”
Martin Bitter

PS: Gibt es noch Vierzeiler ?

Quelltext Service

Die Quelltexte in der VD müssen Sie nicht abtippen. Sie können diese Texte auch direkt bei uns anfordern und sich zum Beispiel per E-Mail schicken lassen. Schreiben Sie dazu einfach eine E-Mail an die Redaktionsadresse.

fep

Leserbriefe

Leserbriefe zum Thema ‚Projekt : Ein gutes Forth(Buch)'

ABS: **HELLMUT.KOERBER**@BFS.ADMIN.CH

Hallo Fritz
Kurz vorweg: Danke für Dein Mail. Ich weiss noch zuwenig von Eurem Ansatz. Daher kann ich mir noch schlecht vorstellen, was die Schwierigkeiten sind.
Ich finde Euer Projekt eine sehr gute Idee. Mir scheint besonders wichtig, gute Ansätze zusammenzutragen. Auf jeden Fall sollte Wolf Wejgaards Ansatz von Holon mit einfliessen, in dem Programme zu komplexen, editierbaren Datenstrukturen werden.
Aus Eurer Ecke - den klassischen Forth-Nutzern und Entwicklern - werden viele gute Vorschläge kommen aus der systemnahen Sicht. Ich könnte auf der Ebene Logik - Philosphie - Objektorientierung einiges beisteuern.
Ich habe 1993 auf der Systems und 1994 auf der Forth-Tagung meinen Ansatz von StepForth vorgestellt. Vielleicht erinnert Ihr Euch. Derzeit schlummert er allerdings in der Schublade. Ein Grund, dass die Entwicklung stecken blieb war, dass ich zuviel am zugrunde liegenden Forth basteln musste, um die Funktionalitäten aufzubauen, die für das sehr anspruchsvolle Konzept von StepForth erforderlich waren. Ich denke, es sind nur einige wenige Modifikationen des klassischen Forth-Ansatzes nötig, um in ganz neue Dimensionen transparenter und konsistenter Software-Entwicklung vorstoßen zu können: Das Wichtigste sind beliebig geschachtelte Vocabularies, um Module, Typen, Contexte, Objekte, Klassen und Vererbung elegant definieren zu können.
Meine Vision ist ein sehr gut strukturierter Forth-Kern mit Standard-Implementationen in C, Java und native Code für wichtige Prozessoren. Dieser Kern kann dann als Systemsprache oder als benutzernahe Skriptsprache dienen - und zum Beispiel als Basis für einen neuen Ansatz von StepForth (Das damalige Konzept lege ich bei).
Wenn es Euch interessiert, kann ich diese Aspekte einbringen. Ich bin allerdings derzeit fest beim Schweizer Bundesamt für Statistik in einer heißen Projektphase und habe nicht viel Zeit. Bisher habe ich noch keinen Kontakt zu Wolf Wejgaard, der ja auch hier irgendwo in der Schweiz lebt.

Laßt mal wissen, was bei Euch läuft, wieweit Ihr seid und ob meine Vorstellungen in Euer Konzept passen. Wie kann ich zum Gelingen beitragen?

So weit für heute.
Ciao *Hellmut*

ABS: **Soeren Tiedemann** <tiedema@mail.uni-freiburg.de>

Lieber Friederich!

Ein FORTH von und fuer die FG !
Deine Anregungen und Vorschlaege in der VD finde ich toll, ich bin fest der Ueberzeugung, dass bei dem geballten Know-How der FG die eine oder andere Innovation einfliessen wird, die jetzt noch gar nicht abzusehen ist. Wenn das Projekt startet, waere ich Dir dankbar, wenn Du meine bescheidenen Kraefte einplanst !

Da ich die Vorgeschichte nur zum Teil mitbekommen habe, moechte ich Dir folgende Fragen stellen:

- Es werden wohl zunaechst Anregungen ueber ein Modell gesammelt?
- Auf welchen Maschinen soll diese Modell konkret implementiert werden?

Wir FORTHER lieben ja den Schoepfungsakt eines neuen FORTH-Systems ueber alles, leider gehen, wie Du ja schon geschrieben hast, die Meinungen und Vorstellungen weit auseinander und jeder Kompromiss kann eigentlich nur ein schlechter sein. Deshalb hier einige meiner Gedanken zu diesem spannenden Projekt:

ANS-Kompatibilitaet vorausgesetzt, schlage ich vor, den ANS-Wortsatz (aber nicht nur ihn) staerker zu faktorieren. So bekaeme jeder FORTHER die Moeglichkeit sein eigenes MINIMAL- oder FAT-System auf die Beine zu stellen. Ein Installations-TOOL hierzu waere komfortabel.

Das Thema der kommenden FORML-Konferenz unterstreicht die Wichtigkeit der Schnittstelle von FORTH zur Aussenwelt. Unter der Aussenwelt verstehe ich hier insbesondere die Verbindung und Kombination mit anderen Programmiersprachen . Ich habe mir Frank Seargent's Pygmy angeschaut und seine Anregungen bezueglich der Kombination mit C(++) erstaunt verfolgt. Wenn wir hier noch einen Schritt weitergehen wuerden, nicht nur das FORTH IMAGE von C(++) zu run-time in den Speicher laden und ueber Pointertabellen beidseitig den Einsprung in die jeweils andere Welt mittels speziellen Entry- und Exit- Woertern vornaehmen, sondern wenn das FORTH-System in der Lage waere, Woerter im .OBJ-File Format zu exportieren, die ein C(++)-Linker verarbeiten kann, dann waere die Schnittstelle perfekt. Hier in Freiburg werde ich den Gedanken eigentlich schon nicht mehr los und arbeite bereits daran, zunaechst einmal eine Abschaetzung des Nutzens, des Aufwandes und der Komplexitaet einer solchen Schnittstelle zu beurteilen. Die Problematik besteht darin, dass ein Ansatz dieser Art zutiefst systemimplementationsabhaengig ist. Meine ersten Stand- und Gehversuche in dieser Richtung bestanden darin, FORTH's CODE-Definitionen als .OBJ-Files zu exportieren. Bei High-Level Definitionen kommen wir aber im Prinzip nicht umhin, zwischen indirekt, direkt oder subroutinenverknuepftem FORTH zu unterscheiden. In jedem Fall aber werden die FIXUP Records im .OBJ-File fuer die address-relocation ziemlich umfangreich ...

Ein beliebtes Thema von mir, deshalb moechte ich es hier erwaehnen: Die PARALLELVERARBEITUNG. Meiner Einschaetzung nach liegt die Zukunft in der Parallelverarbeitung. Damit wuerde eine Softwarerevolution bevorstehen. Ich halte es fuer dringend notwendig, dass wir uns auch hierzu bei einem neuem FORTH Gedanken machen. Die Faehigkeit, teure Rechnerkapazitaeten zu kombinieren, besser zu nutzen und damit kostenguenstiger und schneller zu arbei-

Inhalt

In der nächsten Ausgabe finden Sie voraussichtlich:

Objektorientiertes Programmieren in comFORTH 4, Teil - 3 -, die Fortsetzung des Aufsatzes von Egmont Woitzel, plus Sourcen

Evolution – die Geschichte des Forth, von Elizabeth Rather und Anderen (sofern wir die Erlaubnis zum Abdruck bekommen).

Aufsätze, die uns angekündigt, aber – leider – noch nicht vorgelegt wurden...

...und natürlich Ihre Hinweise aus dem Netz, Ihre Leserbriefe und was immer Sie uns schicken.

ten, wird wesentlich die Merkmale und den Einfluss kuenftiger Systeme praegen. Ebenso die Faehigkeit in einem heterogenen Netzwerk mit heterogenen Betriebssystemen Aufgaben zu verteilen und Ergebnisse wieder zusammenzufuehren.
Dies ist eine Domaene fuer FORTH !

In der Hoffnung, dass genuegend Interessenten bereit sind, sich fuer diese Projekt "aufzuopfern", wuensche ich Dir eine gute Zeit und viel Spass beim Motorradfahren,

Soeren

Diese beiden Leserbriefe zeigen – stellvertretend für weitere E-Mails mit gleichem Anliegen – deutlich, daß das Interesse an einem ‚neuem' - oder an einem ‚gutem' - Forth recht groß ist. Ich habe alle Interessenten an diesem Projekt über die Verabredungen, die wir auf der Jahrestagung getroffen haben, detailliert informiert und an Uli Hoffmann verwiesen – in der Hoffnung, daß die Startschwierigkeiten mit MINOS mittlerweile behoben sind und die Arbeit Fortschritte macht.

fep

ABS: clv@clvpoint.forth-ev.de (**Claus Vogt**)

Hallo Fritz,

hier ein Leserbrief zu Fred Behringer: Real-Mode-32-Bit-Erweiterung für Turbo-Forth in VD 2/98, S.10

Fred Behringer stellt eine Möglichkeit vor, mit Turbo-Forth unter DOS im Real-Mode 4 Gigabyte RAM zu adressieren. Er benutzt dabei den von Harald Albrecht in der c't 1990 vorgestellten Trick. Dabei werden die Segment-Cache-Register so verändert, daß mittels der 32-Bit-Indexregister (ab 80386) auch im Real-Mode auf 4 Gb zugegriffen werden kann.

Für F-PC-Nutzer wird es vielleicht interessant sein, daß ich eine Anpassung des Albrechtschen Codes besorgt habe. An dieser Stelle nochmals vielen Dank an den Heise-Verlag für die Genehmigung zur Weiterverbreitung.
Ich nutze die F-PC-Anpassung seit 1993 unter F-PC 3.54, 3.56 und 3.60, um Meßdaten im Extended Memory abzulegen, und habe bisher keine Probleme beobachtet, abgesehen von der von Fred bereits beschriebenen Inkompatibilität mit dem Speicherverwaltungsprogramm EMM386 von MS-DOS.

Außer dem Albrechtschen Code namens FASTMOVE habe ich für F-PC eine Schnittstelle zu HIMEM.SYS namens XMS entwickelt sowie etwa 20 sinnvolle High-Level-Operationen (@G C@G GFILL CMOVEG GDUMP ...) als G@386 zusammengefaßt. Die entsprechenden Dateien finden sich auf der VD-Weihnachts-CD 1995 in der Directory \CLV\F\CLV\ oder auf der KBBS im Archiv F356CLV1.ZIP.

Claus Vogt

Hallo Friederich,
sag ich mal einfach so, denn ich bin zurück vom Fischen in Schweden :-)
Mit in Urlaub ging die VD und der Tagungsband FG98 Neukirchen-Vluyn, weshalb ich mal alles richtig lesen konnte. Als interessierter Hobbyist hat mich diese OO Geschichte beschäftigt und einen Moment lang dacht ich tatsächlich, nun hätt' ich verstanden, was das eigentlich ist.
Aber dann hat mich doch die Angelei mit meinen Jungs mehr gefordert und der Urlaub konnte richtig losgehen ;-)

Grüße aus Malente, *Michael.*

Lieber Michael (Kalus),
Es freut uns hier in Moers natürlich, daß Dich sowohl die interessante „OO Geschichte" als auch der Tagungsband zumindest soweit fesseln konnten, daß nur die Angelei mit Deinen Jungs Dich davon losreißen konnte. Noch mehr freut es uns, daß es auch für Dich noch Wichtigeres gibt als Forth ;-)

fep

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

Face it. FORTH will never be popular.

So sprach Simon am 4.4.98 auf comp/lang/forth. Zu deutsch: "Macht euch nichts vor. FORTH wird nie weitverbreitet sein." Und er begründete es wie folgt. Ich übersetze:

"C wurde geschaffen um große Gruppen kaum kompetenter Programmierer in die Lage zu versetzen, große Quasi-Echtzeit Datenbanken heraus zu bringen. Und dafür ist es erstklassig. Die fähigen Individuen dagegen schreiben Treiber. Die weniger Erleuchteten machen 'production code' - Auftragsarbeiten. FORTH ist geschaffen, um dem fähigen Individuum die volle Kontrolle über sein Computersystem zu geben. Macht euch nichts vor. Es gibt nicht genug fähige Individuen, um Forth weit zu verbreiten."

mka

ABS: <behringe@sunstatistik1.mathematik.tu-muenchen.de>

Mein Transputer-Forth-Paket F-TP 1.00 findet sich jetzt unter
ftp://ftp.leo.org/pub/comp/os/dos/programming/forth/transputer/

Es handelt sich um ein nahezu ANS-vollständiges 32-Bit-Forth für den T800 auf einem INMOS-Board B004 (oder einem dazu kompatiblen Board wie dem TEK 4/8) in Verbindung mit einem IBM-kompatiblen PC. Es funktioniert auch mit einem T400. Der Server auf der Hostseite, der Cross-Assembler und der Metacompiler sind in Turbo-Forth (16 Bit) programmiert.

Das Paket ist Freeware und umfaßt 900 KB in ZIP-Format. Darin ist ein vorcompiliertes Beispiel eines Mutisystems (mehrere Forths in einem) enthalten. Das eigentliche Paket ist wesentlich kleiner.

Fred Behringer

ABS: **G.Bretschneider**@TMB.in-berlin.de

Hallo Leute!
Derzeit prime ich an einer Grafikerweiterung für F-PC, die

die 256-Farb-Modi nach VESA 1.0 unterstützt. Dabei sollte es nach Möglichkeit auch Mausunterstützung geben. Herkömmliche Maustreiber kennen aber bestenfalls 600x800 Pixel in 16 Farben. Hat sich hier schon mal jemand damit beschäftigt und eventuell eine Lösung in Forth/Assembler geschrieben?
Gibt es überhaupt DOS-Maustreiber, die (mindestens) Bildauflösungen nach VESA 1.0 unterstützen und für Anwendungen transparent sind?

Tschüß, *GERD*
Tel. 030-6734583

Ein kurzer ‚Mitschnitt' aus dem Netz – sicher auch für die Nicht-DFÜler interessant...

ABS: kilgus@deuschle.de (**Marcel Kilgus**)

- Antwort auf die vorhergehende E-Mail -

> Hat sich hier schon mal jemand damit beschäftigt und
> eventuell eine Lösung in Forth/Assembler geschrieben?

Selber zeichnen ist das Kompatibelste. Nein, ich hab keine Sourcen rumliegen, aber so schwer ist das nun wirklich nicht.

> Gibt es überhaupt DOS-Maustreiber, die (mindestens)
> Bildauflösungen nach VESA 1.0 unterstützen und für
> Anwendungen transparent sind?

Wenn überhaupt dann packt es der Logitech. Kannst ja mal bei
ftp://ftp.logitech.com/pub/TechSupport/Mouse/ vorbeischauen.

Marcel

...und noch ein Netz-Mittschnitt...

Hi.

Was macht eigentlich die Software Krise der 60iger Jahre?

In einem Artikel über die Geschichte der Sprache SIMULA 67 stieß ich kürzlich wieder auf diesen Begriff der 'Softwarekrise'. Damals wurde der Zustand beim Programmieren kurz so charakterisiert:

- behind schedule
- over budget
- below specifications

Die Theorie ging dann dahin Rettung bei einer 'general purpose language' zu suchen. ("We can not just produce special purpose languages the hole time...") Man begann die universelle maschinenunabhängige Sprache zu entwickeln.
Was ist daraus eigentlich geworden?
Grüße aus Malente,

Michael Kalus

Antwort von **Friederich Prinz**

Nachricht vom 11.08.98 weitergeleitet
Ursprung : /DE/COMP/LANG/FORTH
Ersteller: michael@malente.forth-ev.de

...
> Was ist daraus eigentlich geworden?

FORTH ? ;-)

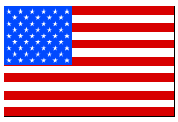
fep

Antwort von **Bernd Paysan**

Michael Kalus wrote:

...
> Was ist daraus eigentlich geworden?

Man hat Ada entwickelt, und sich daran gewöhnt, daß Projekte behind schedule, over budget und below specification sind.
IMHO hätte die echte Lösung sein sollen, daß man eine Sprache nimmt, mit der man maschinenunabhängig special purpose languages entwickeln kann. Also Forth.



Bernd Paysan

Neues aus der FIG Silicon Valley...

Hallo, Friederich,
ich möchte Dir das Kompliment zurückgeben und Dir sagen, daß ich denke, Du bist der "fleißigste Arbeiter" für die VD. Fred Behringer verdient sicher eine Medaille für seine Beiträge, und das trifft sicher auch noch auf eine Anzahl anderer zu, deren Artikel ich mit Interesse in den letzten 5 Jahren gelesen habe.
Ich wünschte, daß Du die Namen der Leute auf den Fotos auf der Rückseite der 3/'98er Ausgabe hinzugefügt hättest. Ich erkenne nur eine kleine Anzahl von ihnen von Bildern, die ich zuvor gesehen habe. Und wo ist denn Klaus Schleisiek? So ein langer Kerl kann unmöglich im Hintergrund verborgen sein. Ich bin neugierig mehr über Eduard, die Stoffmaus, und SWAP's Bilderbuch zu hören. Und ich bedauere sicher, daß ich das ganze, gute MALZ verpaßt habe.
Unser Juli SVFIG Treffen am letzten Samstag zog erneut eine überraschend große Zahl Leute an (um die 30, die gesamten 6 Stunden, in denen wir den Raum im Cogswell College nutzen können). Es ist möglich, daß jetzt, seit die SVFIG den Ablauf der jeweils kommenden Treffen auf ihrer Web-Seite veröffentlicht, mehr Leute herkommen. Und die Sommerzeit reduziert die Anwesenheit nicht, da, wie ich schon vorher einmal sagte, die Sommerferien für Lehrer, Schüler und Leute mit kleinen Kindern da sind. Die meisten unserer Forther gehören nicht in diese Kategorie.
Ich sagte das letzte Mal, daß wir eine Besprechung des neuen "Forth Programmer's Handbook" hören wollten. Aber der Redner war nicht da. Vielleicht das nächste Mal.
Stattdessen hielt John Bumgarner einen sehr informativen Vortrag über die Programmierung von SPS mittels Kontakt-

plänen. Offensichtlich wird die Masse industrieller Steuerungsaufgaben nach wie vor mittels SPS erledigt, wobei diese schneller und leistungsfähiger geworden sind. Dennoch wußte Niemand im Raum etwas von irgendeinem ANSI oder anderen
Standards, die sich mit Kontaktplänen befassen; und somit gibt es eine Reihe Unterschiede zwischen den verschiedenen proprietären Sprachen, die sich dafür entwickelt haben.
Oh, ich habe nicht eher daran gedacht, aber ein Teil der Gründe für die gute Anwesenheit muß wohl das Grillen gewesen sein, das Dr. Ting zum Mittagessen arrangiert hatte. (Wer weiß, wenn wir noch Bier dazu gegeben hätten, wären wohl doppelt so viele Leute gekommen.)
Am Nachmittag gaben drei Leute, die dort waren, einen Bericht über die letzte Rochester-Forth-Konferenz. Sie haben das Gefühl, daß es die letzte Konferenz dort war. Nur 20 Leute waren erschienen. Dr. Haydon bemerkte, daß jemand den Gedanken zum Ausdruck brachte, daß der ANSForth-Standard "Forth töten" könnte. Vielleicht liegt etwas Wahrheit darin, da der Standart die Vielfalt unterdrückt und Vielfalt war immer die "Nahrung" von FORTH.
Das erinnert mich an zwei Bemerkungen von Charles Moore, die ich auf seinem Vortrag auf dem SVFIG-Treffen am 23. Mai 1992 aufschrieb:

"Sobald zu viele Leute beteiligt sind, kann man keine Standardisierung erreichen" und "Sobald Standards auftauchen, ist das Feld reif und es ist Zeit weiterzugehen."

John Rible machte darauf aufmerksam, daß die ANSI Regeln es erfordern, daß ihre Standards alle 4 Jahre ratifiziert werden; somit muß sich die FIG noch vor Ende dieses Jahres an ANSI wenden, entweder mit Vorschlägen für irgendwelche Änderungen des ANSI-Forth-Standards, oder mit der Aufforderung, den existierenden Standard zu verlängern. Wenn nichts gemacht wird, wird der Standard verworfen.
Als nächster kam Bill Ragsdale mit einem Aufruf an alle "Old-Timer" zur 20. FORML-Konferenz im Herbst nach Asilomar, Kalifornien, zu kommen. Er forderte auf, Vorschläge für interessante Tagesordnungspunkte für die Konferenz zu unterbreiten. Besonders interessant wären Themen mit historischem Inhalt über Forth. Falls jemand alte Forth'er kennt, die in der weiten Welt verlorengegangen sind, schickt bitte eine Information an Bill via **RAGSDALE@NETCOM.COM**.
Dann war es Zeit für Dr. Ting, seine Story über sein Arbeitsprojekt mit den sechs Robotern zur Handhabung von Wafern fortzusetzen; wie sein EFORTH auf den Phillips 80C51XA es fertigbringt, sie alle in ein Netzwerk einzubinden.
Soviel über SVFIG bis Ende August. Ich muß mich wieder für meinen unpolierten Schreibstil entschuldigen und für die Benutzung des amerikanischen Slangs, dessen Übersetzung ins Deutsche hart sein kann.
(Dazu nochmal: Meine Bemerkung daß "Mother Nature has to show that she is more powerful than City Hall" ist inzwischen nach Köln gegangen, wo meine Tochter versucht hat, mir mit der Übersetzung zu helfen. Hier sind zwei mögliche Versionen:
"Allerdings müssen Naturkräfte beweisen, daß sie gewaltiger als die Herren im Rathaus sind." "Also müssen Naturkräfte beweisen, daß sie mächtiger sind als die Gesetzgeber.")
Ich habe anzumerken vergessen, daß ich dieses Mal auf dem Treffen nichts mehr über die Idee gehört habe, ein auf Forth basierendes Betriebssystem zu bauen. Allerdings hatte Dr. Haydon ein paar gute Worte über eine Linux-Konferenz, die wie er sagte, eine Woche vor dem Treffen in Silicon Valley stattfand. Soweit ich verstanden habe, wird Linux einzig von den Ideen von Linus Torvald reguliert - eine Standardisierung durch Komitees hatte bisher keine Chance sich zu entwickeln. Ich weiß nicht viel über Linux, aber ich sehe eine Menge Bücher darüber in den Buchläden. Forth-Bücher jedoch sind nur sehr schwer zu finden.
Schließlich, zum Schluß, möchte ich Euch mit einem Gedanken verlassen (abgeleitet von dem was ich über Wissen in einem Buch der Weisheit, zusammengestellt von Tolstoy, gelesen habe)
"Forth is a tool, not a purpose."
("Forth ist ein Werkzeug, nicht der Zweck.")
Alles Gute,

Henry

- übersetzt von Thomas Beierlein

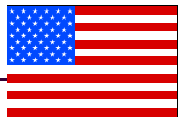
Grüße!

Das Juni-SVFIG-Treffen startete mit 6 Leuten und der angesagte Sprecher, Chris Passauer, der den Kreuzzug zu einem neuen OS/Forth fortsetzen sollte, war nicht da. Unser Tagungsleiter hatte eine Mitteilung erhalten, daß Chris nach Minnesota gefahren war, angeblich auf der Suche nach besseren Jobmöglichkeiten.

Wie üblich kam Dr. Ting als Rettung und füllte den Morgen mit einem Vortrag über seine Arbeit mit Wafer-Transport-Robotern, unter Nutzung von eForth auf dem Philips 80C51XA. Jetzt, da er die unterlagerte Steuerung der individuellen Roboterarme entwickelt hat, arbeitet er an der Idee, eForth so zu erweitern, daß es 'LabView' ersetzen kann, welches zur Gesamtsteuerung des Systems von Roboterarmen genutzt wird.

Die Morgensitzung wuchs bis auf 14 Teilnehmer; über den ganzen Tag zählte ich etwa 20 Leute. Sie kommen und gehen zu verschiedenen Zeiten, und es sind viele, die nur mal hereinschauen, wenn ein Vortrag zu einem Thema auf der Tagungsordnung steht, welches für sie von Interesse ist.

So war dieses Treffen möglicherweise weniger populär, da der Enthusiasmus zur Entwicklung eines neuen, kommerziell lebensfähigen Betriebssystems auf Basis von Forth von den meisten nicht geteilt wird. Nach dem Essen gab Alan Furman eine Zusammenfassung des Betriebssystems, welches Chris vorgeschlagen hat. Aber er mußte hinzufügen, daß nicht abzusehen ist, wieso Händler sich für irgendetwas entscheiden sollen, was nur den Verkauf existierender Betriebssysteme schmälert. Wir wissen das Händler wollen, daß alles auf 'Windows' läuft. Eine andere Haupthürde, angemerkt von Peter Milford, ist das Problem der Vielfalt von Treibern für die verschiedenen Peripheriegeräte, die nur sehr schwierig für ein neues Betriebssystem bereitzustellen sind.



Dr. Ting merkte an, daß "Wir nur auf andere Systeme aufsetzen können, wie wir es mit WinForth getan haben. Wir sollten unser Betätigungsfeld einschränken und das, was wir im Sinn haben, möglichst schnell tun. Wie wäre es z.B. mit einem Zeichensystem für Ingenieure, im selben Sinn wie Chuck Moore's OKAD?"

Doug Hammed schlug vor, ein Forth-Betriebssystem für die Zukunft zu machen, z.B. für die Steuerung eines TV. Es gibt eine Menge guter Ideen, aber wo sind die Ressourcen? Nur das Geld zählt. Jemand kam mit der fürchterlichen Neuigkeit, daß sehr bald jedes neue Auto Windows CE eingebaut hat. (Nicht allzu liebevoll sprechen manche Leute in diesem Zusamenhang von "wince", zu deutsch 'zusammenzucken'.)

Ich las in der heutigen Zeitung, daß sich Microsoft weigert, Windows ins Isländische zu übersetzen, sie lassen es nicht einmal die Isländer selbst machen

Am Ende des Tages gab John Carpenter Beispiele für objektorientierte Programmierung in Forth. "Es ist nicht schwierig in diesem Stil zu programmieren."

Ich fragte die Gruppe, ob irgendjemand eine Kopie des neuen "Forth Programmer's Handbook" von Conklin und Rather gekauft hat und was sie davon halten. Ja, zwei Leute hatten es gekauft. Doug Hammed wurde überredet, das nächste Mal eine entsprechende Buchbesprechung zu halten. John Carpenter zeigte uns seine Kopie und bemerkte, daß es zumeist ANSForth in einer ansprechend repräsentierten Form ist, aber daß das Buch nicht als Lernmittel für Forth konzipiert ist. Ich stimme zu, daß es gut aussieht, das Layout ist leicht lesbar. Aber für ein Buch, daß ohne eine Diskette kommt, teile ich das Gefühl mit einer Anzahl anderer Leute -- der Preis ist etwas hoch. Natürlich, je weniger Exemplare gedruckt werden, um so höher ist der Preis.

Man sieht nicht ein einziges Forth-Buch in den Buchläden hier. Ich habe aber letzte Woche in einem größeren Buchladen nahe der Stanford Universität zehn verschiedene Bücher über Smalltalk gesehen. Gut, da waren auch keine Bücher über Wordstar. Auch nicht über Multiplan, in das ich immer noch vernarrt bin. Haben wir die Ära schon hinter uns gebracht, in der einzelne Gehirne, wie die von Charles Simonyi, Gary Killdall und Chuck Moore in der Lage waren die Welt mit ihren Erfindungen zu erschüttern? Sind die Bäume, die diese Leute gepflanzt haben, in einem, vom Big Business, von Komitees und dem menschlichen Drang nach Spaß und Komfort, anstatt nach Arbeit und Lernen, erzeugten Dschungel verloren gegangen?

Ich höre eigentlich nicht gern mit einem 'sauren' Ton auf, aber ich habe genug geschrieben, zu viel, wirklich. Bitte, streicht was nicht in Eure Veröffentlichung paßt.

Mit den besten Grüßen bis zum nächsten Mal,

Henry.

- übersetzt von Thomas Beierlein

Guten Tag!

Vor einiger Zeit hat Friederich mich gefragt: „Gibt es bei Euch keine Ferien ?“

Es scheint, als gingen unsere Forthtreffen das ganze Jahr immer gleich umher, ohne besondere, traditionelle Unterbrechungen durch Ferien – in denen die meisten Menschen versuchen, die Arbeit so weit wie nur möglich hinter sich zu lassen.

Nun, wie ich bereits schon früher geantwortet habe, sind längere Ferien nur den wenigen Glücklichen vergönnt, unter denen sich natürlich Schullehrer und vermutlich die meisten jüngeren Schüler und Studenten wiederfinden.

Worauf ich hinaus will ist, daß dieses Mal unser Treffpunkt, das Cogswell College, wegen der Ferien geschlossen war. Und hätte Dr. Ting uns nicht allesamt überaus generös in sein eigenes Haus eingeladen, dann hätte es in diesem Monat wohl kein Treffen der SVFIG gegeben.

Aus diesem Grunde traf sich ein gutes Dutzend von uns in Dr. Tings Haus, wo wir, welche glückliche Wendung, mit einem Mal sogar mehr Sprecher als Zuhörer und überdies eine angenehme Zeit hatten.

Und als ich Friederichs Frage in der Gruppe wiederholte, bekam ich zur Antwort: „Unsere Treffen sind unsere Ferien !“. Von meiner Seite aus hierzu ein weitere, umgangssprachlicher Ausdruck: „I‘ll buy that.“ (***‚Gekauft !‘*** *- ??? Der Übersetzer*)

Dr. Ting hatte gerade Taiwan besucht, und brachte uns von dort ein Video mit, in welchem preiswerte, modularisierte Einrichtungen zur Herstellung von Chips vorgestellt wurden.

Weil der Recorder gerade warm war, sahen wir uns noch einen Film über Dr. Tings aktuelles Projekt an, wafertransfer-system robots, die mit eForth gesteuert werden.

Wir stimmten Dr. Tings Statement zu, daß Forth die Sprache ist, die am besten zur parallelen Verarbeitung von Prozessen geeignet ist. Das führte uns zu Dr. Tings Ideen über das Aufgreifen unterschiedlicher Datenbasen, wie man sie bei Spielen wie GO oder SCHACH findet, deren parallele Verarbeitung zu Expertensystemen, zu mini-max Algorithmen usw. führen könnte.

Was dieses Treffen so interessant machte, war die Tatsache, daß wir dieses Mal keinem angekündigten Programm folgen mußten, sondern statt dessen in den unterschiedlichen Präsentationen und Diskussionen gleichsam durch eine Fabrik des Wissens geführt wurden. Das war für die ganze Gruppe attraktiv, weil wir uns nicht mit esoterischen Vorträgen langweilen mußten. Wir mußten nicht einmal die sonst üblichen Pausen einlegen, die notwendig sind, um einzelne Interessengruppen bilden zu können, deren individuelle Diskussionen häufig die Tendenz haben, das gesamte Programm erheblich zu verzögern. Ich denke, dieses Treffen war ein gutes Beispiel für die Richtigkeit der Aussage :“Small is beautiful“.

Ich denke, ein anderes, großes Plus dieses Treffens war die Anwesenheit von Skip Carter, dem Präsidenten unserer FIG. Es ist nicht gerade so, daß es mir ‚warme Gefühle‘ verursacht, wenn ich unseren Führungspersönlichkeiten die Hände schütteln darf. Aber ist eine Tatsache, daß Skip ein stets sehr informierter, fähiger Vortragender ist, stets bereit unsere Diskussionen zu erweitern und zu bereichern.

Skip hatte sein neues Spielzeug mitgebracht – ein Toshiba Libretto Laptop, auch als Baby-Laptop bekannt. Das wiegt weniger als ein Kilogramm, also weniger als Dr. Tings Mathe-

buch und hat die gleiche Größe wie ein gewöhnliches Schreibheft, wie man es an den Colleges nutzt.

Skip hat berichtet, daß es er zwei Tage gebraucht hat, um das mit dem Laptop mitgelieferte Windows von dem Gerät restlos zu entfernen und Linux zu installieren. Das führte natürlich sofort zu Fragen und Antworten zu Linux, „headless“ Maschinen (ohne Tastatur und Monitor), zukünftige Betriebssysteme usw.. Skip berichtete weiter, daß er für das Laptop ein WinCE ‚Development Kit‘ bekommen hat, das er aber sofort zur Seite gelegt hat, als er herausfand, daß diese ‚Kit‘ aus 12 CDs besteht, die insgesamt 4 Gigabyte Festplattenspeicher belegen wollen. (Glücklicherweise brachte Niemand das Gespräch auf ein forthbasiertes Betriebssystem zurück)

In den Buchläden gibt es keine Forthbücher mehr. Im Embedded System Magazin war eine Rezension (von Crenshaw) die mit Forth zu tun hatte. Aber das Wort Forth wird nirgendwo und von Niemandem mehr erwähnt, solange das nicht absolut notwendig ist. Warum ist das so ? Skip meint, daß die Marketing Experten gar nicht erst versuchen würden, irgendetwas zu verkaufen, das den Ruf von Renegaten hat. John Carpenter scherzte darüber, daß wir uns C-Programmierer gerne als Köter vorstellen, daß wir dann aber nur eine Bande verwilderter Straßenkater wären. Und diese Biester sind nun einmal nicht sehr populär. John fügte hinzu, daß ihm sein Professor während eines Javakurses gesagt habe, daß die objektorientierte Programmierung als Inspiration von Forth ausgegangen ist.

Wir mußten Skip darin zustimmen, das gute C-Programmierer aus der Programmierdisziplin von Forth heraus erwachsen sind. C ist eine Sache, aber C++ Programmierer müssen auch heute planen, faktorisieren und strukturieren.

Skips Zusammenfassung der positiven Aspekte an Forth umfaßt im Wesentlichen 5 Punkte:

Nutze Forth:

1. ...als Weg ‚über das Denken zu denken.
2. ...als Weg, über die Arbeit mit Computern zu denken,
3. ...als Weg, über das Design von Computern zu denken,
4. ...als Weg, Programme zu entwerfen
5. ...als Programmiersprache um Applikationen zu schreiben.

Skip berichtete auch, daß die Mitgliedschaft der FIG ganz allmählich wächst. Die magische Grenze von 1.000 Mitgliedern haben wir aber noch nicht erreicht. Die Arbeit des FIG-Büros wird überwiegend von Volontären erbracht. Und es gibt Zeiten, in denen es sehr schwer ist, alle Anfragen und Wünsche der Mitglieder zu bearbeiten. Die jeweils nächste Ausgabe der Forth Dimensions ist immer etwas spät in der Post.

Zum Beginn des Nachmittags kam das Thema dann auf die Musik. Dr. Ting stellte uns sein privates Projekt zur Synthese, zur Edition und zum Studium der Musik vor

(selbstverständlich alles in Forth). Meine Eltern waren beide Musiker – und ich muß zugeben, daß ich an diesem Nachmittag eine Menge über Musikgeschichte und Instrumente gehört habe, was mir zuvor unbekannt gewesen war.

George Perry, „Zeremonienmeister‘ und Vorsitzender unserer Gruppe, verfügt über viele Talente - und gab uns ‚freihändig‘ eine ausgezeichnete Kostprobe seines musikalischen Talentes auf dem Piano des Hauses.

Ich habe vergessen zu erwähnen, daß unser generöser Gastgeber sich selbst auf den Weg gemacht hat, uns ein besonderes Mittagessen von einem chinesischen Restaurant zu besorgen, dessen Rufnummer er nicht im Telefonbuch finden konnte, weil er gar nicht wußte, wie sich dieses Restaurant auf englisch nennt.

Da bleibt mir an dieser Stelle nur, Dr. Ting für den schönen und interessanten Tag in sei-

Forth Interest Group International (FIG USA)

Wollen Sie mit der ganzen Welt verbunden sein und dabei Ihr Englisch perfektionieren? Amerika ist ein wesentlicher Teil der ganzen Welt. Zumindest, was Forth betrifft. Über tausend Mitglieder aus allen Ländern sind bei uns.

Werden auch Sie Mitglied in der Amerikanischen Forth-Gesellschaft (FIG USA).

Für 45 Dollar im Jahr (Studenten zahlen 18 Dollar) bekommen Sie 6 Hefte unserer Vereinszeitschrift Forth Dimensions und genießen auch sonst verschiedene Vorteile. In den Heften erfahren Sie Forth-Neuigkeiten aus aller Welt, neue Produkte, Literatur, Forth-Ideen, fundiertes Wissen, Artikel auch für Einsteiger, Projekte, Leser-Diskussionen, Quelltexte, Hinweise auf Internet-Verbindungen, kostenlose Forth-Systeme und vieles mehr. (Für Übersee-Porto müssen wir leider noch 15 Dollar hinzurechnen).

Unmittelbare Informationen über uns bekommen Sie, wenn Sie auf der Homepage der Deutschen Forth-Gesellschaft "Links zu anderen Forth-Organisationen" und dann "Forth Interest Group (USA)" anklicken.

Ansonsten bekommen Sie Auskünfte über das amerikanische Forth-Büro:

Forth Interest Group
100 Dolores Street, suite 183
Carmel, California 93923
USA

oder auch vom Redakteur, Marlin Ouverson, unter der E-Mail-Adresse:

E-Mail: **office@forth.org**

FIG UK

(Englische Forth-Gesellschaft)

Treten Sie unserer Forth-Gruppe bei.
Verschaffen Sie sich Zugang zu unserer umfangreichen Bibliothek.
Sichern Sie sich alle zwei Monate
ein Heft unserer Vereinszeitschrift.
(Auch ältere Hefte erhältlich.)
Suchen Sie unsere Webseite auf:
www.users.zetnet.co.uk/aborigine/Forth.htm
Lassen Sie sich unser Neuzugangs-Gratis-Paket geben.
Der Mitgliedsjahresbeitrag beträgt 10 engl. Pfund.
Hierfür bekommen Sie 6 Hefte unserer
Vereinszeitschrift Forthwrite.
Wenden Sie sich an:

Dr. Douglas Neale
58 Woodland Way
Morden Surrey
SM4 4DS
Tel.: (44) 181-542-2747

E-Mail: dneale@w58wmorden.demon.co.uk

nem Haus im Namen aller Teilnehmer herzlich zu danken.
Tschüß, bis zum nächsten Mal

Henry

- übersetzt von Friederich Prinz

Claus Vogt hat folgende E-Mail, die nicht nur für AMIGA User interessant ist, an die Redaktion weitergeleitet. Interessant ist der Text nicht nur wegen seiner Beschreibungen zu den Features von JForth. Die Tatsache, daß JForth ab sofort kostenlos zu haben ist, wird allerdings nur den Amiga User aufhorchen lassen.

Faszinierend ist aber der enorme ‚Run', der offensichtlich auf das JForth stattgefunden hat. Es gibt sie also noch, die Amigas. Und dieser Rechner hat tatsächlich auch noch Nutzer. Und davon sind nicht wenige an einem Forth interessiert, daß auf dem 83er Standard aufbaut ! In der Welt von Bill Gates bewirkt eine Pressemitteilung über ein Forth, daß in zwei Tagen mehr als 2.500 Downloads auf dieses Forth stattfinden. Das gibt Hoffnung...

fep

Nachricht vom 16.08.98 weitergeleitet
Ursprung : **clv**@clvpoint.forth-ev.de
Ersteller: **marrandy**@tampabay.rr.com

Am Dienstag, den 11. August 1998 um 23:30 Uhr (EST), brach der JForth Download aufgrund mehrerer hundert Download Requests zusammen. Das System stand für 8 Stunden nicht mehr zur Verfügung !

In den ersten 24 Stunden des offenen Downloads wurden auf der primären ftp-Seite unglaubliche 1.649 Zugriffe aufgezeichnet. Aufzeichnungen über Zugriffe auf der sekundären ftp-Seite sind nicht möglich.

In den zweiten 24 Stunden wurde die noch phantastischere Zahl von 1.969 Zugriffen registriert.

—

Diese Zugriffe auf JForth waren offensichtlich eine Folge der nachstehenden Pressemitteilung vom 10. August.

—

JForth wird als Freeware freigegeben – May the Forth be with you...Always !!!

Upgraded to OS3.1

Mit großem Dank an die primären Autoren, Mike Haas und Phil Burk, die sich großzügiger Weise entschieden haben, dieses Produkt als Freeware herauszugeben (vorher: 179 $), stellen wir fest, daß der Amiga einen neuen ‚Stachel im Fleisch' hat.

***** Hinweis: Drucken Sie das ausgezeichnete Handbuch aus. Es enthält nicht nur wertvolle Informationen über die Vielzahl der von JForth angebotenen Tools, sondern darüber hinaus eine exzellente, dreistufige Einführung.

***** Nutzen Sie die JForth Mailing-Liste !!!

Schreiben Sie an **MDaemon@ChaosSolutions.com** mit 'subscribe JForth-list' als einzigem Text im Betreff.

Worum geht es ?

JForth ist eine Programmiersprache, die Ihnen direkte Interaktion mit Ihrem Amiga ermöglicht. Wenn Sie in JForth programmieren, dann befinden Sie sich ‚innerhalb' der Sprache selbst. Sie können beliebig auf Datenstrukturen zugreifen, beliebige Routinen einzeln testen, oder beliebige Entwicklungstools direkt von der Tastatur aus steuern.

Diese direkte Kommunikation mit dem Computer kann Ihre Produktivität steigern, während sie Ihnen gleichzeitig freie Zeit verschafft, die Sie benötigen, um die Qualität Ihrer Softwareprodukte zu verbessern.

JForth basiert auf dem Standard der Sprache Forth, der 1983 definiert wurde. Forth wurde von Charles Moore entwickelt und eingesetzt, als er eine neue Sprache benötigte, um Teleskope zu steuern.

Er entwickelte eine Sprache, die auf einem Wörterbuch. Dieses Wörterbuch kann durch die Definition neuer Worte erweitert werden, die jeweils auf die bereits bekannten Worte aufbauen.

JFORTH ist Freeware...

Seit der ersten Version von Mr. Moore, wurde Forth auf beinahe alle Computer portiert, vom größten Mainframe bis zum kleinsten Microcomputer.

Weil ein minimales Forth mit nur wenigen Kilobyte Code implementiert werden kann, wird es häufig in sehr kleinen embedded Systemen zur Prozeß– oder Robotersteuerung genutzt. Selbstverständlich ist Forth ebenso für den Einsatz auf größeren, fähigeren Computern wie den Commodore Amiga geeignet.

Forth ist eine sehr flexible Sprache und kann an größere Computer angepaßt werden, ohne daß die besonderen Vorteile für kleinere Systeme dabei verloren gehen. JForth ist eine Implementierung, die speziell für den Amiga vorgenommen wurde.

JForth arbeitet mit einem 32-Bit Stack und kompiliert 68000er Maschinencode. Das macht JForth schneller als die meisten anderen Forth-Systeme.

Weiterhin bietet JForth einen umfangreichen Satz Tools an, mit deren Hilfe Sie auf die speziellen Features des Amiga zugreifen können.

Sie können jede Bibliotheksroutine des Amiga über deren Namen erreichen und auf die internen Strukturen des Amiga über Konstrukte zugreifen, die denen in C ähnlich sind. Dazu bietet JForth einige spezielle Toolboxen, die einfache Graphiken unterstützen, intuitive Menüs, IFF-Files und anderes. Mit diesen Toolboxen können Sie ohne Umwege die Arbeit mit Ihrem Amiga vereinfachen.

Die Quellen zu all diesen Toolboxen stehen selbstverständlich zur Verfügung, so daß Sie, falls Ihnen das notwendig erscheint, Änderungen vornehmen, oder einfach die Programmierung der Toolboxen als Programmierbeispiele verwenden können. Darüber hinaus bekommen Sie mit dem JForth mehr als ein Dutzend kleiner Beispielprogramme, die für Leute wie mich gemacht sind, die am besten bei der Arbeit lernen.

JForth ermöglicht es Ihnen, Dinge zu tun, die in der FORTH-Welt einzigartig sind. Von besonderer Dramatik ist dabei CLONE. Dieses außergewöhnliche Werkzeug erlaubt es, vollkommen von der Entwicklungsumgebung unabhängige Standalone-Versionen Ihrer Programme zu erstellen, bei minimaler Größe dieser Programme.

Amiga Library Calls - die Zugriffe auf beliebige Bibliotheksroutinen erfolgen einfach über die Nennung des Namens einer Routine und der Parameterübergabe auf dem Stack, in der Reihenfolge, wie die Parameterübergabe in der Dokumentation zum Amiga beschrieben wird. Eine Anzahl von Toolboxen unterstützt spezielle Teile der Amiga-Bibliothek, wie zum Beispiel EZMENUS, Graphik,die serielle Schnittstelle, ANSI-Codes und Vieles mehr.

Amiga Structure Support - JForth arbeitet mit dem Äquivalent der „.h“ Include-Dateien in C. Diese „.j“-Dateien enthalten alle Definitionen zu notwendigen Strukturen, sowie Konstanten zur Übergabe an die Bibliotheksroutinen.
Strukturen können interaktiv ‚gedumpt‘ werden. Einschließlich aller Substrukturen bekommen Sie diese sowohl mit dem Namen als auch mit dem zugehörigen Wert angezeigt, wenn Sie DST nutzen. Dieses Debugwerkzeug ist für das OS3.1 aufgearbeitet worden !

ARexx Support - ist eine Sprache, die es unterschiedlichsten Applikationen ermöglicht, miteinander zu kommunizieren. Zum Beispiel könnte eine Tabellenkalkulation mit einem Datenbankprogramm kommunizieren. Die entsprechenden Tools werden Ihnen dabei helfen, ARexx-kompatible Programme zu schreiben.

Assembler - JForth unterstützt gleich zwei 68000er Assembler, von denen einer mit der UPN arbeitet, und der andere in einer Motorola ähnlichen Syntax. Der UPN-Assembler kann dazu genutzt werden, Makros zu definieren.
Der Motorola Assembler ist allerdings einfacher zu lesen.
Selbstverständlich bietet JForth auch einen Disassembler.

Block Support and SCRED - für Jene, die immer noch die alte BLOCK und SCREEN Umgebung bevorzugen, werden LIST und LOAD und ein Standard Zeileneditor angeboten.
Selbstverständlich bietet JForth aber auch einen WYSIWIG Editor mit dem Namen SCRED.

Clone - kann dazu genutzt werden, kleine, exekutable Images Ihrer Programme zu erzeugen. CLONE extrahiert aus einem bereits kompilierten JForth Programm alle Codes und Daten, die zur Laufzeit des Programms benötigt werden. Aus diesem Extrakt erzeugt CLONE ein reassembliertes, kleineres Image Ihres Programms. Alle JForth Entwicklungstools, Name– und Linkfelder und andere unbenutzte Worte werden beiseite gelassen. Das Endergebnis ist durchaus vergleichbar mit Images, die von einem C-Compiler und Linker erzeugt wurden.

Die Images können selbstverständlich mit einer Symboltabelle gespeichert werden, damit Ihnen, bei Bedarf, Debugger wie WACK oder andere zur Verfügung stehen.

Der Forth-SourceLevel Debugger kann natürlich auch mit geklonten Programmen arbeiten. Die meisten Ihrer Programme lassen sich völlig problemlos klonen, wenn Sie einige, wenige und einfache Regeln beachten, wie zum Beispiel die Laufzeit-
initialisierung von Variablen und Arrays mit Forthadressen.

Debugger - JForth bietet Ihnen einen SourceLevel Debugger, mit dessen Hilfe Sie in einzelnen Schritten durch Ihre Programme ‘steppen‘ können. Zu jedem Schritt können Sie dabei die Stackinhalte kontrollieren, Speicher ‘dumpen‘, Haltepunkte setzen, Ausführungen einzelner Worte kontrollieren oder einfach Forth-Kommandos dazugeben.

Floating Point - JForth unterstützt beides: das ‚Fast Floating Point‘ mit einfacher Genauigkeit und die doppelte Genauigkeit der IEEE-Spezifikation. Die entsprechenden Worte

sind konform mit dem Standard der Forth Vendors Group.

IFF Support - bietet JForth Ihnen selbstverständlich auch. Zusätzlich besitzt JForth eine Toolbox, die speziell für ILBM Graphikfiles entwickelt wurde. Das ermöglicht es Ihnen, Pictures, Brushes, Anims, Animbrushes in Malprogrammen oder anderen Quellen für Ihre eigenen Programme zu nutzen. Darüber hinaus bietet JForth Ihnen Tools zur Erstellung eigener Animationen und Präsentationen - Funktionen wie Blit, Wipe, FadeIn, FadeOut usw.. Mit Hilfe dieser Tools lassen sich machtvolle Graphik– und Animationsprogramme erstellen.

Lokale Variablen - können die Definition komplexer Worte sehr vereinfachen, weil sie viele Stackoperationen überflüssig machen. Lokale Variablen sind schnelle, auf sich selbst zugreifende Speicherworte, mit denen die Definition von reentranten, rekursiven Codes möglich wird. Der Gebrauch von ‚gewöhnlichen' Variablen macht Codes meist nicht-reentrant.

Module - Die von JForth vorkompilierten Module stellen eine besondere Methode dar, mit deren Hilfe Ihr System während der Kompilationszeit besonders schnell auf Code zugreifen kann. Dies betrifft hauptsächlich den Assembler und den Disassembler, sowie die Amiga Include-Dateien. Diese Module werden dynamisch – bei Bedarf – in das Wörterbuch eingebunden. Das spart Ihnen viel Platz im Wörterbuch.

Multi-Standard - Das System ermöglicht es Ihnen, einfach zwischen JForth und den Hauptstandards (FIG, ‚79 und , 83) hin und her zu schalten. Damit können Sie auch Codes kompilieren, die unter anderen Forth-Systemen definiert wurden ! Ein ANSI kompatibles Modul wird zur Zeit erarbeitet !

ODE - ist eine objektorientierte Entwicklungsumgebung, ähnlich dem Konzept von SmallTalk. Sie können Klassen intelligenter Datenstrukturen definieren und von diesen Klassen soviele Kopien erzeugen, wie Sie benötigen. Mit dieser Technik können Sie sich das Programmieren immens vereinfachen und Ihre Codes wirklich wiederverwendbar gestalten.

Profiler - Dieses Optimierungswerkzeug beobachtet Ihre Programme bei der Arbeit – und meldet Ihnen, womit Ihre Programme die Zeit verbringen. Die meisten Programme verbringen die überwiegende Laufzeit in kleinen Teilen ihrer Codes. Wenn Sie diese Teile kennen, dann können Sie sich bei Optimierungsaufgaben auf diejenigen Stellen Ihrer Programme konzentrieren, bei denen eine Optimierung wirklich lohnt.

Textra - ist ein mächtiger, aber einfach zu benutzender Texteditor für Programmierer. Textra arbeitet mit der Maus, bietet die Funktionen Cut, Copy und Paste und erlaubt Operationen zwischen den einzelnen Instanzenfenstern. Zusätzlich können Sie aus Textra heraus auf Arexx-Funktionen zugreifen, so daß Sie auch unter Textra die auf Ihrem Amiga vorhanden Makros nutzen können (...oder schreiben Sie sich Ihre eigenen Makros...).
Textra kann direkt mit JForth kommunizieren. Sie können direkt aus dem Editor heraus kompilieren. Falls der Compiler einen Fehler meldet, markiert Textra Ihnen die entsprechende Stelle im Quelltext.

Tutorials - Sie werden vom Anfänger bis zum fortgeschrittenen Nutzer durch das System geführt.

Miscellaneous debugger tools - stehen Ihnen zur Verfügung; Analyse des Amiga auf binärer Ebene, Dateianalysen, Graphik und Vieles mehr !

Vergessen Sie nicht, das ausgezeichnete Handbuch auszudrucken. Es gibt Ihnen nicht nur unschätzbare Informationen über die vielen Tools in JForth, sondern bietet Ihnen zusätzlich eine dreistufige Einführung

Nutzen Sie die JForth Mailing Liste. Schreiben Sie an:

MDaemon@ChaosSolutions.com

Mit dem Text

subscribe JForth-list

im Betreff.

Sie können JForth herunterladen von:

http://www.softsynth.com/JForth

May the Forth be with you...always !!!

- übersetzt von Friederich Prinz

...immer wieder Interessantes aus dem Netz...

Private Homepages kostenfrei einzurichten, bieten heute die meisten Provider Ihren Privatkunden an. Die Telekom stellt Ihren Nutzern dafür bis zu 1 MByte Plattenspeicher zur Verfügung. Viel Gebrauch machen die Nutzer von diesem Service (noch) nicht. Es ist auch nicht Jedermanns Sache, sich selbst darzustellen. (Und Vielen, die das versuchen, gelingt es nicht so recht.)
Eine schöne Homepage – mit einem noch schöneren Hinweis auf FORTH – hat unser Mitglied

Ulrich Richter.

Möchten Sie das ‚Schiebespiel' (WIN32FOR) gerne einmal völlig selbsttätig in Aktion sehen ? Dann sehen Sie doch einfach einmal dort hinein:

http://home.t-online.de/home/Ulrich-Richter/Uli00.htm

d.Red.

Kennen Sie weitere, sehenswerte Homepages von Forthern, Forth-Anbiertern, Instituten...? Schreiben Sie uns, rufen Sie uns an, oder ‚mailen' Sie uns bitte die Adresse, zwecks Veröffentlichung

CRC für Dummies

Dipl.-Ing. Wolfgang Allinger
Ingenieurbüro Allinger
Brander Weg 6
42699 Solingen
All@business.forth-ev.de

0. CRC? Ist das was zum essen? (Nää... zum k....)

CRC ist die Abkürzung für Cyclic Redundancy Check. Dabei handelt es sich um ein (zyklisches) Verfahren, bei der an eine zu prüfenden Gruppe von binären Informationen noch eine zusätzliche Information angehängt wird, damit man (Übertragungs) Fehler erkennen kann. Bei den hier betrachteten CRC wird von 1 Byte langen Informationen (Zeichen) ausgegangen. An eine Gruppe von Zeichen werden bei einem 16bit CRC zwei (Prüf) Zeichen angehängt. Diese Prüfzeichen werden auch Frame Check Sequence (FCS) genannt. CRC wird u.A. benutzt bei Floppy Disk Controllern, Xmodem, Ymodem u. Zmodem, Ethernet...

1. Wie kommt man an CRC?

Für ein Projekt brauchte ich einen bestimmten CRC-Algorithmus. (Die serielle Datenübertragung mit einem µC wurde mit CRC abgesichert. Und der antwortete nur dann, wenn man ihn mit der richtigen Zeichenfolge incl. einer richtigen FCS beschimpft hatte, ansonsten schmollte der und gab absolut nichts von sich, kein Echo und nicht den Hauch eines Hinweises auf irgendwas. Sehr bedienungsfreundlich, man mußte Baudrate, Zeichenlänge, Paritybit, Stopbits, Kommando String und CRC richtig beisammen haben. Ansonsten tiefstes Schweigen! Und ellenlange eklige Flüche meinerseits. Obendrein wurde natürlich auch keine Mustersequenz mit einer richtigen CRC dargestellt und zum krönenden Abschluß wurde das Low Byte des FCS zuerst geschickt. Darum konnte man auch nicht einfach auf $0000 (s.u.) für eine korrekte Übertragung prüfen.)
In der Literatur fand ich nur hochtrabende Betrachtungen, finsterstes (pseudo) mathematisches Geschwafel und so gut wie keine Programme und erst recht keine Beispiele. Je mehr ich da rumstöberte, umso mehr wurde ich verwirrt (ist bei meinem kleinen Geist eh ganz einfach :-)
Ich hab dann in Internet rumgesucht, nach stundenlanger Suche wurde ich etwas fündiger. Aber wieder nur riesige Staubentwicklung und unverständlichen Kram. Jedoch ein kleiner Lichtblick: die Uni Berkeley hatte ein Beispiel für eine kurze C-Routine [BER98].Die hatte der Author nach einem (x86) Assembler Beispiel umgesetzt, das Beispiel leider aber nicht veröffentlicht und auch keine Quelle dazu angegeben. Die eigentliche Funktion des Programmes war sehr undurchsichtig. Glücklicherweise waren sehr viele Querverweise auf andere Artikel zu CRC enthalten.
Wieder war dort nur weiteres unverständliches Zeug vorhanden. Aber ein Artikel war die ganze bisherige Mühe wert: Terry Ritter “The Great CRC Mystery” [RIT98]

Terry beschreibt darin einigermaßen verständlich, was CRC ist und wie das funktioniert.

2. Langsam verzieht sich der Pulverdampf

Auf einen Hilferuf im usenet unter /comp/lang/forth erhielt ich am 16.4.96 von Wil Baden einen Auszug aus einem etwa 10 Jahre alten Artikel von ihm: “CRC Polynominals made Plain”. Hierin beschreibt Will, daß viele Programmierer was von CRC gehört haben und es sogar programmiert haben, aber sehr wenige nur verstanden haben, wie das funktioniert. Er schreibt weiter, daß es leicht war, herauszufinden, warum CRC nicht verstanden wurde:

Die Hauptursache für die Verwirrung sei, daß man aus einer Mücke einen Elefanten machte, also eine einfachen Sache so kompliziert wie möglich darstellt. (Also klarer Fall von: sollen die anderen vor Bewunderung ob des gewaltigen Geistes in ergriffenes Schweigen verfallen!)

Ein weiterer Grund sei, daß Programmierer durch schwülstige mathematische Darstellungen abgeschreckt werden, in dem man den Begriff eines Polynoms fälschlicherweise auf die Darstellung des CRC-Algorithmus anwendet. Eine Polynom-Formel hat aber nichts mit der Darstellung des CRC-Algorithmus zu tun. Bei der CRC-Darstellung wird ein synthetisches Polynom benutzt, was lediglich eine Liste von 0 und 1 Folgen ist, die aber absolut nichts mit der binären Darstellung einer Zahl zu tun haben. Um die Sache noch weiter zu verschleiern, wird diese Folge dann noch als HEX-Zahl dargestellt und die führende 1 weggelassen.

2 synthetische Polynome werden sehr häufig benutzt:

```
CRC-16   1  1000  0000  0000  0101
         ( IBM fan club )
         in Hex $8005

CRC-CCITT 1  0001  0000  0010  0001
         ( CCITT und ISO )
         in Hex $1021
```

Aus diesen synthetischen Polynomen werden auch die folgenden (Verwirrungs) Polynom Formeln dummerweise so dargestellt:

CRC-16 **x^16+x^15+x^2+x^0**
oder **x^16+x^15+x^2+1**

CRC-CCITT **x^16+x^12+x^5+x^0**
oder **x^16+x^12+x^5+1**

3. Wat nu?

Und nun kommt das, was ich glaube verstanden zu haben:

Als Hardware muß man sich ein Schieberegister Ring vorstellen, welches eine Stufe weniger hat, wie Ziffern in dem Synthetischen Polynom sind. In diesem Schieberegister sind noch Rückkoppelungen mit XOR-Gattern eingebaut, bei denen ein Eingang immer mit dem Ausgang des höchstwertigsten Bits und der zweite Eingang mit dem Ausgang einer vorherigen Stufe bzw. des einzuschiebenden Bit verbunden sind. So ein XOR-Gatter befindet sich vor jeder mit 1 im synthetischen Polynom markierten Stufe.

Für das CRC-CCITT sieht das (hoffentlich) so aus:

([n] Register bit-n, X- XOR)

```
synth. Polynom  0    0    0    1        0    0   0   0   0   0   1       0   0   0   0   1
Input---->-X-<[15]-[14]-[13]-[12]-<-X-<[11]-[10]-[9]-[8]-[7]-[6]-[5]-<-X-<[4]-[3]-[2]-[1]-[0]-<-
          |                       ^                                   ^                       ^
          v                       |                                   |                       |
          --------------->---------*------------------>--------------*------------>------------
```

Zu Beginn der Operation werden alle Register gelöscht. Alle Bits der zu prüfenden Folge werden über das oberste XOR in die unterste Stufe eingeschoben. Wenn alle zu prüfenden Bits reingefummelt sind, enthält das Schieberegister die 16bit FCS.

Nun haben CRCs noch zwei Eigenheiten:

1. Wenn man die FCS einer CRC codierten Folge auch noch mit den oberen 8 bit zuerst ebenfalls noch in das CRC-Register reinschiebt, so muß das Register komplett Nullen enthalten, wenn kein Fehler vorliegt.
2. Wenn man eine beliebige 16bit Zahl invertiert und dann in das CRC-Register einschiebt, kommt immer dieselbe (Kenn) Ziffer heraus. Diese Kennziffer ist angeblich typisch für das jeweilige CRC (polynom :-)

CRC	Kennziffer:	sog. "Polynom"
CRC-CCITT	$1D0F	x^16+x^12+x^5+x^0
CRC-16	$B001	x^16+x^15+x^2+x^0

Den Hinweis auf die Kennziffer habe ich in einem Dallas Datenbuch gefunden [DAL94]

Die erste Eigenheit ist der Grund für die Beliebtheit des CRC für Hardware. So werden z.B. bei einer Floppy alle bits eines Sektors inclusive der 16 angehängten FCS bits eingeschoben. Danach muß das CRC-Register =0 sein, was sehr leicht mit Hardware abzufragen ist.

Die zweite Eigenheit kann man ausnutzen, um CRC-Routinen zu überprüfen. (So prüft DALLAS offensichtlich die Hardware...)

4. Wie macht man CRC in Software?

Ansatz 1: alles in der brute-force Methode. Man emuliert die Hardware Lösung und bastelt sich das Schieberegister mitsamt den XOR Gattern bit für bit zu Fuß. So machen es die meisten.
Ziemlich lang und langsam.
Ansatz 2: man generiert sich Tabellen, in denen (Teile) des CRCs abgelegt werden und groffelt da mehr oder weniger schnell drin rum. (so macht es z.B. Will Baden auch)
Ziemlich länger aber schneller

Ansatz 3: Man macht das byteweise wie Terry Zimmer [RIT98] in Listing Two Method 3 in Pascal:

```
Dx      contains the 16bit CRC
        also called
        Frame Check Sequence  FCS
Data    contains the char
        to make the new FCS
```

```
PROCEDURE crcitta ( var dx: INTEGER; data: INTEGER )
BEGIN
 dx  :=  Swap (dx) XOR data ;
 dx  :=  dx XOR (Lo(dx) shr 4) ;
 dx  :=  dx XOR (Swap(Lo(dx)) Shl 4)
         XOR (Lo(dx) Shl 5) ;
END ;
```

mit den Spezial-Funktionen

```
pascal:  description  Forth (na endlich):
Lo(dx)   get low byte  $FF AND or 255 AND
Swap(dx) swap bytes  FLIP  or ><
```

Elegant, kurz und bündig (bloß wie ist der da drauf gekommen?)

Ansatz 4 Klar, daß im Abschnitt 4 und Ansatz 4 was mit Forth kommen muß:

```
\ CRC for DUMMIES or bytes bite the CRC
\ ALL980422+
\ or how I understand the byte shuffle
\ algorithm
\ from Terry Ritter METHOD 3
\
\ a) FCS rotate left 8
\ b) XOR byte to FCS, i.e. the char is
\    'shifted' in thru gate x^0
\ c) Though char is shifted in w/o any
\    'correction' of the '3 XOR gates', so:
\     1. correct the XOR propagation from
\        2^15 to 2^0, i.e. missed the x^0 gate:
\        mask 2^12..2^15 shift to 2^0, XOR it
\        to FCS
\     2. correct the XOR propagation
\        from 2^0 to 2^5, i.e. missed
\        the x^0 gate:
\        mask 2^0..2^7 shift to 2^5, XOR it
\        to FCS
\     3. correct the XOR propagation from
\          2^0 to 2^12, i.e. missed
```

```
\         the x^0 gate:
\         mask 2^0..2^7 shift to 2^12,
\         XOR it to FCS
\     done
\ ALL980422-
:CRC-ALL        ( uFCS char -- uFCSnew )
\OK ALL980422
\applicate char to FCS w/
\CRC-CCITT = x^16 + x^12 + x^5 + x^0
    SWAP    ( -- char uFCSold )
    FLIP                    \ = 8bit ROTATE
    XOR     ( -- u )        \ char 'shifted' in
    DUP  255  AND  4 RSHIFT
            ( -- u u1 )     \ 1. correct
    XOR                     \     the
    DUP  255  AND  5 LSHIFT
            ( -- u u1 )     \ 2.    ...
    XOR     ( -- u )        \       ...
    DUP  255  AND 12 LSHIFT
            ( -- u u1 )     \ 3.    result
    XOR     ( -- u )        \ UFF!
 ;
```

Exclusiv, kurz, schnell, einfach und sehr leicht als ultraschnelles Assembler-Programm umzusetzen.

Wer Genaueres wissen/sehen möchte, schaue bitte in mein Programm **crc-ccit.seq** .

5. Ende

Mich würde interessieren, ob meine ”Bastelanleitung” für alle CRC’s gilt und ob man so von den synthetischen Polynomen auch auf effiziente 32bit Algorithmen kommt. Ich würde mich sehr freuen, wenn dazu einige Versuche, Anregungen und Kommentare kommen würden.

Und helft mit, den verdammten Sandstreuern und Staubentwicklern das Handwerk zu legen.

Wie Chuck Moore sagt:**KEEP IT SIMPLE**....nur was einfach ist, ist auch schön und gut!!!!

Quellen:

[BER98] 15.4.98: **http://ozma/ssl.berkeley.edu/~dbb/crc-code.htm**

[RIT98] 15.4.98: **http://www.io.com/~ritter/ARTS/CRCMYST.htm**

[DAL94] Feb94: Dallas Semiconductor: **Book of DS19xx Touch Memory Standards**

Und wer mal sehen will, wie man das auch beschreiben kann:

15.4.98: **http://bbs.uniinc.msk.ru/tech1/1994/er_cont/err_ctl.htm ....**

16.4.98: **http://bugs.wpi.edu:8080/EE535/**


```
\\ crc-ccit.seq        16bit CRC here      CRC-CCITT
                                           ALL980607

ALL980607 v0.02 final(?) clean up
ALL980417 v0.01 proofed it against
    ...\taygeta\...FSL-lib...algorithm44
    same result for CRC-CCITT, but this seems:
       1.) much faster on compiling,
       2.) not much slower runtime,
       3.) very much shorter code/list space,
           improved comments...
           LSHIFT and RSHIFT instead of
     * / ... it's very easy converted
           to a CODE so it is guaranteed very
           much faster at runtime
ALL980416  v0.00 1st try after a lot of
           searching in internet
           based on
 http://www.io.com/~ritter/ARTS/CRCMYST.HTM
          and
 http://www.io.com/~ritter/ARTS/CRCLIST2.HTM
  "The Great CRC Mystery" by Terry Ritter

thank you Terry Ritter
thanks to Skip Carters taygeta server

converted to F-PC (thank you Tom Zimmer)

released to public domain by
Dipl.-Ing. Wolfgang Allinger   'ALL'
Brander Weg 6
D-42699 Solingen
Germany
eMail: all@business.forth-ev.de

{
 anew --crc-ccitt--
 DECIMAL           \ default base for this file

} Terry Ritter gave some examples in the file
        LISTING TWO  (crclist2.htm)

 I translated METHOD 3  pascal

It uses a 16bit CRC with the polynom
x^16 + x^12 +x^5 + 1
it's also called the CRC-CCITT because it's used by
CCITT and this is used by Ward Christensens famous
XMODEM, disk controller, SDLC, ....

Dx      contains the 16bit CRC   also called
        Frame Check Sequence  FCS
Data    contains the char to make the new FCS

PROCEDURE crcitta
    ( var dx: INTEGER; data: INTEGER );
 BEGIN
   dx :=  Swap (dx) XOR data ;
   dx :=  dx XOR (Lo(dx) Shr 4);
   dx :=  dx XOR (Swap(Lo(dx)) Shl 4) XOR
          (Lo(dx) Shl 5);
 END;

with special functions:
 pascal:  description:   Forth:
 Lo(dx)   get low byte   $FF AND  or  255 AND
 Swap(dx) swap bytes     FLIP     or  ><

From different places of CRC descriptions I
learned, that there are 2 special properties of the
CRC:

1st) a 16bit num split in two bytes and then
     the high and low byte feed through the
     CRC algorithm results in 0000h. This is
     very handy for hardware, i.e. feed the
     whole data including the appended 2 FCS
     through will allways result in 0000h
In other words (Forth words :-)
```

```
DUP SPLIT ROT SWAP CRC-CCITT SWAP CRC-CCITT
   ( u -- 0 )

2nd) if the inverted number is feed as 2
     bytes, then the result will allways be a
     characteristic number for the CRC
     polynom.

In other words (Forth words :-)

DUP -1 XOR SPLIT ROT SWAP CRC-CCITT
SWAP CRC-CCITT
   ( u -- usecret )

The secret for CRC-16 I found at DALLAS Touch Memory
Standards book. The secret for CRC-CCITT I assumed
by applicate it to the routine. ( So it's not a real
proof, please inform me if it's incorrect :-)
( but gives the same result as crc of FSL lib from
taygeta... ALL980417 :-)

CRC:      secret #:   polynom:
CRC-CCITT $1D0F       x^16 + x^12 + x^5 + 1
CRC-16    $B001       x^16 + x^15 + x^2 + 1

{
\ ------------------

CODE CRC-CCITT  ( uFCS char -- uFCSnew )
   POP AX          \ AX: char
   POP DX          \ DX: uFCS  ( -- )
\  dx := Swap (dx) XOR data ;
\  FLIP   XOR                    ( -- uFCS' )
   XCHG    DH, DL
   XOR     DX, AX

\  dx := dx XOR (Lo(dx) Shr 4);
\  DUP 255 AND 4 RSHIFT XOR     ( -- uFCS" )
   MOV     AL, DL
   SHR     AL, # 1
   SHR     AL, # 1
   SHR     AL, # 1
   SHR     AL, # 1
   XOR     DX, AX

\  dx := dx XOR (Swap(Lo(dx)) Shl 4) XOR
\                (Lo(dx) Shl 5);
\  DUP 255 AND FLIP 4 LSHIFT    ( -- uFCS" u1 )
   MOV     AH, DL          \ dup 255 and flip
   SHL     AH, # 1         \ 4 lshift
   SHL     AH, # 1
   SHL     AH, # 1
   SHL     AH, # 1
   XOR     DH, AH

\  OVER 255 AND 5 LSHIFT    ( -- uFCS" u1 u2 )
   MOV     AL, DL          \ over 255 and
   SUB     AH, AH          \ AH=0
   SHL     AX, # 1         \ 5 lshift
   SHL     AX, # 1
   SHL     AX, # 1
   SHL     AX, # 1
   SHL     AX, # 1
   XOR     AX, DX        \ RESULT -> AX !!!

\  XOR XOR                ( -- uFCSnew )
\                          xor-ed already I hope
   1PUSH                  ( -- AX )
   END-CODE

\ \S      ---------- test sample ------------
: TEST-CRC ( u -- 0 )
   DEPTH 1 < ABORT"  need a number "
   DUP >R
   DUP SPLIT ROT SWAP           ( -- uL u uH )
   CRC-CCITT SWAP               ( -- u' uL )
   CRC-CCITT  ABORT"  implementation bad "
   R>
   DUP -1 XOR SPLIT ROT SWAP    ( -- uL u uH )
   CRC-CCITT SWAP                ( -- u' uL )
   CRC-CCITT
   ."  secret# is "  H.
 ;

\ --------------------
 defined LSHIFT nip 0= #IF

CODE LSHIFT  ( u1 u -- u2 )
              \ logical Left SHIFT
      POP CX
      POP AX
      SHL AX, CL
   1PUSH  END-CODE

CODE RSHIFT  ( u1 u -- u2 )
              \ logical Right SHIFT
     POP CX
     POP AX
     SHR AX, CL
   1PUSH END-CODE
 #THEN

\ \S      ---------- test sample ------------

: TCRCA         ( u -- 0 )
   DEPTH 1 < ABORT"  need a number "
   DUP >R
   DUP SPLIT ROT SWAP          ( -- uL u uH )
   CRC-ALL   SWAP              ( -- u' uL )
   CRC-ALL  ABORT"  implementation bad "
   R>
   DUP -1 XOR SPLIT ROT SWAP   ( -- uL u uH )
   CRC-ALL   SWAP               ( -- u' uL )
   CRC-ALL
   ."  secret# is "  H.
 ;

\ CRC-ALL ... siehe vorstehenden Aufsatz

\ --------------------

\ END of All's garbage
```

Fehlt Ihnen ein Listing, von dem Sie wissen, daß es in einer früheren Ausgabe der VD stand ?
Oder erinnern Sie sich an einen der vielen ausgezeichneten Artikel in früheren Ausgaben unserer Zeitschrift, können aber die entsprechende Ausgabe einfach nicht finden ?
Oder würden Sie gerne Ihre private Sammlung der VD vervollständigen ? Dann fragen Sie doch einmal im FORTHBÜRO nach. Dort stehen noch ältere Ausgaben der VD ,**zum Abgeben‘** zur Verfügung.

Mitglieder der Forthgesellschaft können im Forthbüro gegen Erstattung der Versandkosten (Porto und Verpakkung) auch ältere Jahrgänge und Ausgaben der VD anfordern.

Bitte rufen Sie an, schreiben Sie, oder senden Sie dem **Forth-Büro** eine entsprechende E-Mail !

Bericht über die Jahrestagung 1998

Forth-Tagung '98

Bernd Paysan

Die Jahrestagung der Forth-Gesellschaft fand dieses Jahr in Moers bzw. Neukirchen-Vluyn statt. Das liegt in einer Ecke Deutschlands, wo Städte öfter Doppelnamen haben, wie Castrop-Rauxel oder Wanne-Eickel, wo noch vereinzelt Bergbau betrieben und Stahl gekocht wird. Die dortige Forth-Szene läuft deshalb auch im bestbesuchten öffentlichen Ort (dem Arbeitsamt) ab.

Eigentlich hatte der Fritz für den Freitagmorgen ja die Besichtigung eines Bergwerks versprochen, für die, die so früh schon aufgestanden waren, gab's dann aber (wegen übergroßen Andrangs) nur ein Schulungsbergwerk zu sehen. Darüber müssen aber andere berichten, da war ich noch nicht da.

Die Zahl der Vorträge hielt sich diesmal in Grenzen, so hatte jeder Vortragende viel Zeit.

Uli Hoffmann fing mit dem MD5-Algorithmus in ANS Forth an. Dieser Algorithmus liefert eine 128-Bit-Prüfsumme, indem er die Eingabedaten blockweise durchknetet (ähnlich wie Blätterteig herstellen), und das mit der bisherigen Summe vermischt. Das alles ist ein alter Hut, in C gibt's den Algorithmus schon lang. Uli zeigte, daß man das in Forth auch machen kann. Mehr noch, der Code wird kompakter und der Stack kommt auch optimal zu Ehren.

Thomas Beierlein zeigte einen Prefix-Assembler für den 68k. Wie bei anderen Prefix-Assemblern wird auch hier der Befehl verzögert, so daß jeder Befehl den jeweils vorherigen assembliert.

Man kann sich damit das Parsen sparen. Angesichts der vielen Adressierungsarten des 68k hat man aber mehr Registerkonstanten, und es darf diskutiert werden, ob \texttt{Ax -)} ein angemessener Ersatz für \texttt{-(Ax)} ist, oder ob man nicht besser noch 8 zusätzliche Konstanten spendiert.

Nach dem Abendessen gab's noch einen Workshop zur Messezeitung „Echtzeit", in der die Forth-Gesellschaft die Gelegenheit bekommen hat, sich zu präsentieren. Verschiedene Forth-Advokaten steuerten hier ihre Standard-Sätze bei.

Danach wurde an den Computern gehackt, was aber schon kurz nach 23 Uhr dadurch ein Ende fand, daß der Tagungsraum abgeschlossen wurde.

Am nächsten Morgen war beim Frühstück der Bundestagsabgeordnete Peter Enders und die Presse (in Gestalt eines Journalisten) am Tisch des Direktoriums zu Gast. Auch hier muß ich anderen die Berichterstattung überlassen. Peter Enders blieb noch zum ersten Vortrag.

Stefan Lange stellte darin eine Überwachungskamera für Atomanlagen vor. Solche Überwachungskameras plant die IAEO[1] in den überwachten Staaten einzusetzen. Herrn Enders schienen Leute wie Saddam Hussein überwachungswürdig, angesichts evtl. vertuschter Störfälle in deutschen Nuklearanlagen (Nukem) können wir uns da auch an der eigenen Nase fassen.

Eine solche Kamera soll ereignisgesteuert Bilder aufnehmen (etwa wenn sich etwas bewegt), und speichert das dann auf einer PCMCIA-Karte. Natürlich muß man sicherstellen, daß an der Kamera nicht manipuliert wird, indem man digitale Signaturen verwendet.

Wolf Wejgaard zeigte eine neue Variante der unendlichen Geschichte HOLON: JAVALON. Wie kann man die Ideen von HOLON und Java vereinen?

Egmont Woitzel führte comForth 4 vor. Das basiert auf dem letztes Jahr schon vorgestellten stdcall-Threading, bei dem Forth-Wörter so aufgerufen werden, wie Prozeduren aus einer DLL. So fügt sich comForth 4 noch nahtloser in Windows ein. Es gibt auch eine leistungsfähige objekt-orientierte Forth-Erweiterung, die auch Mehrfachvererbung kann, und trotzdem performant ist (über virtuelle Methoden-Tabellen, wie C++). ComForth 4 verwendet objektorientierte Aspekte auch im Kernel, etwa den Vokabularen, bei der Ein/Ausgabe, und beim Input für den Interpreter. Anders als frühere comForth-Versionen wird bei comForth 4 nicht der Code, sondern der Service verkauft. Da noch einige Features und Feinschliff fehlt, ist comForth 4 noch nicht fertig.

Bernd Paysan präsentierte nach dem Mittagessen, was aus MINOS im letzten Jahr geworden ist. Anders als damals kann es jetzt mehr als nur einen Taschenrechner zusammenklicken: Man kann OpenGL[2]-Anwendungen programmieren, mit SQL[3] Datenbanken abfragen, und MIDI-Dateien abspielen.

MINOS ist eine in objekt-orientertem Forth geschriebene Widget-Bibliothek; dazu gibt es einen Oberflächen-Editor (Theseus), mit dem man sich Dialogboxen zusammenklicken kann. Primäre Heimat für MINOS ist Linux/X, aber auch unter Windows 95/NT läuft es inzwischen (mehr schlecht als recht). Auch die Kombination bigFORTH+MINOS ist frei verfügbar.

Klaus Schleisiek regte abschließend noch zur Diskussion über objekt-orientiertes Forth an. Er präsentierte eine Art „ One-Shot"-Vokabular, mit dem man das Information-Hiding für den Forth-Interpreter verständlich (und ohne state-smarte Wörter) machen kann. Egmont hat sowas auch in comForth 4 eingebaut, aber statt den Interpreter an ein paar kleinen Stellen zu ändern, basiert seine Lösung auf besonderen Vokabularen, die sich nach Durchsuchung selbst aus dem Vokabular-Stack austragen. Die anschließende Diskussion brachte noch tiefere Einblicke in das OOF von comForth 4.

Nach dem Abendessen gab's dann noch einen Workshop: „ Ein gutes Forth(buch)". Ziel dieses Projekts ist es, Nachwuchs zu bekommen. Nachdem alle wichtigen und guten Bü-

cher zu Forth nicht nur total veraltet sind, sondern auch vergriffen sind[4], muß ein Buch geschrieben werden. Als verwendete Plattform einigten sich die Teilnehmer schnell auf bigFORTH+MINOS, weil das genügend aufregende Dinge bietet, die heute noch interessieren. Allerdings muß MINOS unter Windows noch richtig laufen; bis Linux das Ziel „total world domination“ erreicht hat, vergeht der Forth-Gemeinde zuviel Zeit.

Danach zog sich der Drachenrat zurück, und beschloß, den Drachen dieses Jahr dem Autoren dieses Artikels für seine Leistungen für Forth zu verleihen.

Fußnoten:

1 Internationale Atom-Energie Organisation
2 Eine 3D-Grafik-Sprache von Silicon Graphics
3 Structured Query Language
4 Arndt verteilte am Anfang der Tagung noch
3 Restexemplare von „Starting Forth“

Richtigstellung:
Die Moerser Forther treffen sich NICHT allwöchentlich im Arbeitsamt der Stadt, sondern, wie schon während der Tagung ausführlich erklärt, im MALZ (Moerser Arbeitslosenzentrum). Das MALZ ist ein von staatlichen Stellen unabhängiger Verein (e.V. – wie die FG) Moerser Bürger (erwerbslose UND erwerbstätige Menschen), der vom DGB und seinen Einzelgewerkschaften unterstützt wird. Ausführliche Informationen findet der geneigte Leser im Netz unter ***WWW.MALZ.DE****. Diese Seite zu besuchen, möchten wir an dieser Stelle ausdrücklich empfehlen.*

Die Redaktion

Gehaltvolles

zusammengestellt und übertragen
von Fred Behringer

VIJGEBLAADJE der HCC Forth-gebruikersgroep, Niederlande

Nr. 10, Juni 1998

Treffen am 13.6.98 wieder in der Volkssternwarte Utrecht: Es stand unter dem Motto "Finite State Machines".

Eindige Toestanden Mechanismen (FSM)
Leendert C. van den Heuvel

Das Forth-Programm in diesem einzigen, dafür aber umso ausführlicheren diesmaligen Artikel (132 Zeilen, 16 Colon-Definitionen) liefert ein Beispiel (Zeiteingabe über die Tastatur) für einen endlichen Zustandsautomaten nach Mealy (getrennte Tabellen für Steuerung von Zustand und Ausgabe).

Forth-gg produkten

13 Artikel mit Preisangabe, die man über die angegebene Adresse beziehen kann. Ich habe einige davon in der Rezension des Vijgeblaadje 9 aufgezählt.

Namen und Adressen

Die sechs offenbar wichtigsten Aktiven in der HCC Forth-gebruikersgroep. Mir fällt auf, daß zwar bei jedem einzelnen die Telefonnummer angegeben ist, aber keine E-Mail-Adresse.

Forth-Programmer's Handbook
Albert Nijhof

Der Autor setzt sich auf 60 Zeilen mit dem neuen Buch von Edward K. Conklin und Elizabeth D. Rather kritisch auseinander. Auch mit den Versandkosten. Eine wertvolle Hilfe, denn auch wir hier in Deutschland, zumindest in München, müssen ja solche Bücher über die Buchhandlung unbesehen, in reinem Vertrauen auf die Namen der Autoren bestellen - wobei man dann gar zu leicht nach dreimonatigem vergeblichen Warten aufgibt und stornieren läßt.

Das Interesse an dem Buch von Rather und Conklin scheint groß zu sein – auch in der BRD. Ich prüfe deshalb gerade die Möglichkeit einer Sammelbestellung, in der Hoffnung, zumindest die Transportkosten reduzieren zu können. Interessenten mögen sich bis zum 15 November bei mir melden, zur Absprache der „Modalitäten“.
(F.PRINZ@MHB.GUN.DE oder
FRIEDERICH.PRINZ@T-ONLINE.DE oder
02841-58398 (bitte Anrufbeantworter nutzen !))

fep

Holländisch ist gar nicht so schwer. Es ähnelt sehr den norddeutschen Sprachgepflogenheiten. Und außerdem ist Forth sowieso international. Neugierig? Werden Sie Förderer der

HCC-Forth-gebruikersgroep .

Für 20 Gulden pro Jahr schicken wir Ihnen 5 oder 6 Hefte unserer Vereinszeitschrift 'Het Vijgeblaadje' zu. Dort können Sie sich über die Aktivitäten unserer Mitglieder, über neue Hard- und Softwareprojekte, über Produkte zu günstigen Bezugspreisen, über Literatur aus unserer Forth-Bibliothek und vieles mehr aus erster Hand unterrichten. Auskünfte erteilt:

Willem Ouwerkerk
Boulevard Heuvelink 126
NL-6828 KW Arnhem
E-Mail:

Willem Ouwerkerk <w.ouwerkerk@kader.hobby.nl>

Oder überweisen Sie einfach 20 Gulden auf das Konto 5253572 der HCC-Forth-gebruikersgroep bei der Postbank Amsterdam. Noch einfacher ist es wahrscheinlich, sich deshalb direkt an unseren Vorsitzenden, Willem Ouwerkerk, zu wenden.

Gehaltvolles

zusammengestellt und übertragen
von Fred Behringer

Forth Dimensions der Forth Interest Group, USA

März/April 1998

2 Office News
Trace Carter

Eine beachtliche Menge von Neuigkeiten aus dem amerikanischen Forth-Büro: Für Studenten beläuft sich der Jahresbeitrag auf nur noch 18 Dollar. Allerdings zuzüglich 15 Dollar an Übersee-Versandkosten. Über die FIG-Homepage kann man sich jetzt (als FIG-Mitglied) in eine oder mehrere der E-Mail-Diskussionsgruppen der 11 speziellen Arbeitsgemeinschaften (Special Interest Groups) einklinken; Diskussionsgruppen, die man per E-Mail ins Haus geschickt bekommt und an denen man sich per E-Mail beteiligen kann.

4 Around the World
Marlin Ouverson

Eine gute Idee unseres Redakteurs, Friederich Prinz, den vollen Wortlaut dieses bemerkenswerten Editorials von Marlin Ouverson, dem Redakteur der Forth Dimensions, über Forthens Internationalität in Übersetzung in der VD 3/98 zu bringen. An sein Editorial hat Marlin eine Anzeige unserer Forth-Gesellschaft zur Mitgliederwerbung angehängt. Zumindest ein Mitglied aus Amerika konnten wir bereits begrüßen: Dr. Glen Haydon - bei der FIG USA und anderswo kein Unbekannter.

6 Stack Gymnastics Made Easy
Ronald T. Kneusel

Keiner weiß auf den ersten Blick, was beispielsweise DUP ROT SWAP DROP bewirkt. Noch schwieriger ist es, sich komplizierte Stackoperationen vom Programm her auszudenken. Der Autor schlägt einen generell einsetzbaren Stack-Umordner, einen 'Stacker', also einen 'Stapler', ==> , für Pokket Forth vor. Die Idee dazu stammt aus einer anderen, forthähnlichen Sprache. Was auffällt, ist, daß ==> als Immediate-Wort zum unmittelbaren Einsatz innerhalb einer Colon-Definition aufgefaßt wird.

8 URLs - a selection of Web-based Forth resources

24 gut kommentierte http- und ftp-Adressen, über die man Forth-Systeme beziehen kann.

9 Building a Remote Target Compiler
Dave Taliaferro

Dritter und letzter Teil einer Artikelserie über Forth-Compilation in den Speicher eines getrennten Zielprozessors. Ein derartiger Compiler läuft auf einem Wirtsrechner mit einem Embedded-Entwicklungssystem und erlaubt die Erzeugung von ausführbaren Forth- und Assembler-Routinen für einen Zielprozessor. Sobald die Tastatureingaben oder Quelldateien vom Wirts-Forth interpretiert sind, wird der übersetzte Code und die zugehörigen Daten zum Zielprozessor übertragen, wo sie dann zum sofortigen Austesten und zur Weiterverarbeitung bereitstehen.

18 Mushroom Identification
Charles Samuels

Ein Pilzbestimmungsprogramm sollte her: Der Benutzer gibt ein paar Bestimmungscharakteristika in eine Datenbank ein und bekommt gesagt, um welchen Pilz es sich handelt. Der Autor hat 12 Jahre lang nicht mehr ernsthaft programmiert und Windows war an ihm vorbeigerauscht. Mit WinForth von LMI und dem Resource Workshop von Borland hat er es geschafft. Leider sagt er uns nicht, wo und wie man sein Programm erhält.

FIG Mail Order Form
(Einlageblatt)

Allerlei schöne Forth-Literatur, die man per Post (Übersee-Porto 15 Dollar) beziehen kann: Forth-Dimensions-Bände 1 (1979/80) und 6-18 (1984/85-1996/97), jeweils 35 Dollar. FORML-Tagungsbände 1981-89, 1992-95, 30-45 Dollar. 17 Forth-Bücher 20-90 Dollar. Liste der Artikel aus Forth Dimensions 1-15, ebensolche über FORML 1980-92. Das Forth Programmer's Handbook von E.K. Conklin and E.D. Rather für 57 Dollar. 22 Disketten mit den Forth-Systemen (Programme und Quelltexte) der verschiedenen Autoren, jeweils zum Selbstkostenpreis (8-30 Dollar). More on Forth Engines, Band 10-17, jeweils 15 Dollar, Band 18-20, jeweils 20 Dollar. 1900 Forth-Quellennachweise für 18 Dollar. Dr. Dobb's Journal September 1982, September 1983, September 1984 (3 Ausgaben) für 10 Dollar.

19 An Extensible User Interface
John J. Wavrik

Der Autor lehrt Mathematik (Abstrakte Algebra) an der University of California. Er verwendet Forth seit 1980 für seine Forschungsarbeiten. Er möchte seine Ergebnisse anderen Mathematikern und allgemein einem Zuhörerkreis präsentieren, von dem nicht verlangt werden kann, daß die Hörer eine Computersprache, hier Forth, lernen, nur um herauszufinden, ob sie sich für den dargebotenen Stoff (Beispiel: alle Gruppen bis zur Odnung 32) überhaupt interessieren. Kurzum: Das (an sich interaktive, aber doch viel zu komplizierte) dem eigentlichen Programm zugrundeliegende Forth-System muß so auf-

bereitet werden, daß der Benutzer bloß noch Kommandos einzugeben braucht, und schon legt (im angegebenen Beispiel) das Gruppenerzeugungsprogramm los. Forth macht das ja an sich schon. Mehr Interaktivität kann man doch gar nicht verlangen. Wenn ich den Autor richtig verstanden habe, stört die Kollegen das "OK" und vielleicht der kryptische Klammeraffe. Das kenne ich! (Kampfparole: Man benenne den Klammeraffen anders, und schon herrscht eitle Akzeptanz.) Das Ganze läuft doch wohl, nicht in diesem Artikel, aber von der Ideenwelt her schon, auf lichtbalkengesteuerte Programmbedienung hinaus. Bei OS/2, Windows x oder Linux XWindows wären das dann die Buttons (Knöpfe). Solche Programmiervereinfachungen werden also überall angedacht, nicht nur bei uns. War das nicht bei PC-TOOLS, CONTEXT, Norton und dem Midnight Commander schon immer so? Der Quelltext des Programms liegt auf ftp://ftp.forth.org/pub/Forth/FD/1998 .

28 Runstk - Stack Utility
Warren Heath

Ein Text-Editor, der das jeweilige Stackabbild mit anzeigt. An sich wohl nicht neu. Interessant aber, wie der Autor das Thema angeht. Die Stacknotation wird fest vorgegeben. Verstöße gegen diese Vorgabe sind unzulässig. Eine ganze Reihe von Worten des Forth-Systems muß umdefiniert werden. Herzerfrischend ist es außerdem, den Forth-Werdegang des Autors ab 1979 zu lesen. Man merkt, daß er mit Liebe bei der Sache war und ist.

32 Character Literals
Wil Baden

Das neue Standard-Forth scheint [CHAR] x und CHAR x da zu verwenden, wo F83 und die Nachfolgesysteme (Turbo-Forth etc.) ASCII x einsetzen. Der Autor scheint die beiden CHAR-Worte nicht zu mögen. Interessant! Dann darf man sich wohl auch erlauben, ganz unschuldig zu fragen: Was war denn eigentlich an ASCII so schlecht? Leicht ist es, Forth zu schreiben. Schwer ist es, sich zu einigen! CHAR x ist für den Interpreter-Modus konzipiert. Der Autor möchte als kürzere Form C x einsetzen, und zwar im Compiler-Modus genauso wie im Interpreter-Modus. Die Variable STATE abzufragen, scheint in Forth-Expertenkreisen verpönt zu sein. Er schlägt eine Definition für [LITERAL] (zur Verwendung in C x) vor, die ohne eine solche Abfrage auskommt. Natürlich alles in High-Level-Forth, ohne Rückgriff auf CODE-Definitionen. ASCII erledigt das, worüber der Autor spricht, schon immer. Allerdings erkundigt sich ASCII in den mir geläufigen Implementationen nach dem Wert von STATE.

33 Double Number Arithmetic
Wil Baden

Nummer 19 der Serie des Autors zur Erweiterung von Forth (Stretching Standard Forth). Wirklich nur 'Forth', reine Colon-Definitionen. Addition und Multiplikation in doppelter Genauigkeit sind nicht besonders schwer. Es ist die Division, die überraschenderweise Schwierigkeiten macht. Der Autor greift auf einen Algorithmus von Knuth zurück. Den Quelltext kann man sich per E-Mail schicken lassen: wilbaden@netcom.com . Ich habe das getan und habe ihn postwendend bekommen. Dankeschön Wil Baden!

Mai/Juni 1998

2 Office News
Trace Carter

Die amerikanische Forth Interest Group geht jetzt ins zwanzigste Jahr ihres Bestehens. Die Mitgliederzahl wächst. Das Forth-Büro hat seine Ausrüstung dank großzügiger Spenden aufstocken können.

4 Change is the Constant
Marlin Ouverson

"Eines bleibt sicher: Alles verändert sich". In seinem Editorial ruft Marlin dazu auf, Artikel für die Forth Dimensions zu schreiben, um auf die sich ständig verändernden Gegebenheiten in der Forth-Gemeinschaft aufmerksam zu machen. Das "Journal of Forth Applications and Research", sagt Marlin, erscheint seit dem 7. Jahrgang in elektronischer Form (siehe www.jfar.org). Es bringt nur Artikel, die durch die Hände von Gutachtern gegangen sind. Die Forth Dimensions wird jetzt regelmäßig die eine oder andere Arbeit daraus übernehmen.

5 ANS Forth Update
Elizabeth Rather

Nach den ANSI-Statuten muß das ANS Forth Technical Committee (TC) vier Jahre nach Veröffentlichung der Standardisierungsvorschläge über "Überarbeiten", "Endgültig bestätigen" oder "Zurückziehen" abstimmen. Die vier Jahre sind um. Bei entsprechendem Wahlausgang sind zwei diesbezügliche Konferenzen angekündigt. Mitglied des TC kann jede(r) werden, die (der) nachweislich irgendwie mit Forth verbunden ist, auch Bürger(innen), die nicht aus den Vereinigten Staaten kommen. Der Mitglieder-Jahresbeitrag beträgt 300 Dollar.

6 eForth for Java
Michael A. Losh

Das Beste vorweg: Alle nicht kommerziellen Forth-Freunde können das Paket ohne Gewissensbisse verwenden und beziehen über: http://www.amsystech.com/mlosh/ Es war verhältnismäßig einfach, die 154 kB einzuladen. Nur ein paar Sekunden und die Datei jeforth.java war auf meiner Platte. eForth für Java (oder jeForth) ist eine in High-Level-Java ge-

schriebene eForth-Implementation für die Java Virtual Machine (JVM). Zugriff hat man in Internet-Sitzungen über den Netscape Navigator (geeigneter Version) oder den Microsoft Internet Explorer. eForth wurde für die vorliegenden Zwecke um DO ... LOOP und ähnliche heute übliche Konstrukte erweitert. Der Autor suchte für Lehrzwecke einen leichten Internet-Zugang zu einem "lebendigen" Forth, ohne allzusehr auf Effizienz zu achten. Über das vorliegende System hinaus hat er fest umrissene weitere Pläne.

Für diejenigen, die sich nicht so sehr für den Quelltext interessieren, sondern nur mal schnell in das System hineinschnuppern wollen: Man rufe die Webseite auf und schon steht das jeForth-System, blau eingerahmt, auf dem Bildschirm. Ich habe es ausprobiert. Wirklich prima Sache! Und das Allerbeste: Das System steht ab dann (mit der immer wieder aufrufbaren Webseite) im Cache-Speicher. Natürlich braucht man dazu einen java-fähigen Browser. Ich verwende den Netscape Navigator Gold 3.01. Achtung: Die Worte müssen in Großbuchstaben eingegeben werden.

13 Temperature Monitoring
Ken Merk

Der Autor verwendet F-PC zur Programmierung der Schnittstelle von einem digitalen Thermometer-Sensor zum Parallel-Port des PCs. Er setzt den DS1620 von Dallas Semiconductor ein. Dieser hat ein 3-Punkt-Seriell-Interface in einem 8-Pin-DIP-Chip eingebaut. Der Autor geht auch auf schaltungstechnische Einzelheiten ein (zwei Schaltbilder).

21 LOAD" Module"
Dave Edwards

Dave Edwards, Inhaber der Firma Jarrah Computers (Embedded Systems), verwendet UR/FORTH (Forth-83) von LMI (screen-orientiert). Die erste Zeile (Zeile 0) eines jeden Schirms (screens) steht normalerweise für Anmerkungen zur Verfügung und wird vom Programm nicht benutzt. Dave zieht sie zur besseren Organisation des Einladens von Programmteilen heran. Auch zum Aufbau eines einfachen Help-Mechanismus kann seine Überlegung verwendet werden.

24 Local Macros
Wil Baden

Globale Makros können in ANS-Forth über (das auf Strings wirkende) EVALUATE definiert werden (der Rezensent: $EXECUTE in Turbo-Forth). Wil Baden schlägt diesmal in seiner Reihe "Standard-Forth-Werkzeuggurt" ("Standard Forth Tool Belt") Lokale Makros vor: "Lokale Makros sind keine Forth-Definitionen. Sie nehmen keinen Platz im Dictionary ein. Man definiere sie, benutze sie und werfe sie weg."

27 What's a Character?
Wil Baden

Wil Baden beschäftigt sich diesmal in seiner Reihe "Standard-Forth ausdehnen" ("Stretching Standard Forth") mit Worten zur Umwandlung von Klein- in Großbuchstaben, zur Abfrage, ob eine solche Umwandlung vorgenommen werden soll, usw. Praktisch jedes Forth-System, sagt Wil, hat solche Worte, aber überall werden sie anders bezeichnet. Er schlägt vor, einfach die Bezeichnungen von C zu übernehmen (inalpha, inalnum usw.). Mit der Bemerkung "Bezeichnungen wie in C" hätte man dann eine Art von Standardisierung. Er gibt Implementationen an, die Übersetzungstabellen und lokale Makros (siehe seinen anderen Artikel) verwenden.

31 Adaptive PID
Skip Carter

Zweiter Teil einer Serie von Skip Carter, dem Präsidenten der Forth Interest Group USA, über die Simulation von PID-Controller-Umgebungen ((P)roportional-, (I)ntegral-, (D) ifferential-Anteil). Dieser Artikel konzentriert sich auf die mathematischen Grundlagen (Beweise oder gute Literaturhinweise). Eine Forth-Implementation wird für den nächsten Teil versprochen. Stichworte: Software-Paket Mathematica, Differentialgleichung, Approximation über Differenzengleichung, Optimierung, Ausgleich nach der Methode der kleinsten Quadrate, Lineare Algebra, Matrizenschreibweise.

Skip leitet auch das Forth-Scientific-Library-Projekt (eng verbunden mit dem Namen Julian V. Noble) und betreibt www.taygeta.com , wo viele Forth-Systeme zum Herunterladen bereitstehen (mein Transputer-Forth F-TP 1.00 liegt auch dort).

Gehaltvolles

zusammengestellt und übertragen
von Fred Behringer

FORTHWRITE der FIG UK, Großbritannien

Nr. 96, Mai 1998

1 Editorial
Chris Jakeman

"Ich bin der Überzeugung, daß jemand, der das Programmieren erlernen möchte, das in Forth viel besser kann (oder können sollte) als in irgendeiner anderen Sprache", sagt Chris. Er kündigt ein Einsteigerpaket als Projekt an, das bei der FIG UK gerade in Angriff genommen wird. Diese Überlegungen kommen mir, dem Rezensenten, von unserer letzten Forth-Tagung her bekannt vor.

2 Forth News
Chris Jakeman

Berichte aus der Nachrichtengruppe comp.lang.forth: Forth-Kurs auf der FIG-UK-Webseite, euroForth diesmal bei Saarbrücken, Rochester-Conference war im Juni, MPE-Forth für DOS und Windows, Public-Domain-Open-Firmware-Forth für Micro-Controller, ANS-Forth wird neu festgelegt, GUI-Design für Win32Forth, Forth von Klaus Kohl für den SIEMENS-C167, OCCAM-ähnliches Forth auf Java, eForth für Java, Anton Ertl: Müllabfuhr (garbage collection), comp. lang.forth im Archiv.

5 Reading the World (1)
Paul E. Benett

Erster Teil einer Serie von beabsichtigten Artikeln aus der Regelungstechnik, die "den Software-Orientierten zeigen soll, worüber die Hardware-Typen eigentlich reden". Chris Jakeman meint im Editorial, diese Serie könne vielleicht sogar in eine FIG-UK-Entwicklungskarte münden. Als weitere Themen sind vorgesehen: "Abfrage der Tastatureingabe", "Analoge Eingabeschaltungen", "Auslesen analoger Eingaben". Paul scheint einen Posten bei der Londoner U-Bahn (25000 Volt in der Oberleitung) zu haben und konzentriert sich im vorliegenden Artikel auf Störeinflüsse, mit denen sich der Entwicklungsingenieur herumschlagen muß.

11 Welcome Disk - Bons Mots
Chris Jakeman

Diesmal ein 'gutes Wort' von Leo Wong, Mitglied der FIG UK aus den USA. Chris wünscht sich mehr Einsendungen.

12 Vierte Dimension '98 No 1
Alan Wenham

Es ist eine wahre Freude, auf Englisch nachlesen zu können, was wir da alles Schönes gemacht haben. Mit Alan hat sich ein wirklich guter und gewissenhafter Rezensent gefunden. Und die Zusammenarbeit mit ihm ist ausgezeichnet. Seinen Lesern teilt er mit, daß er noch frühere Ausgaben der VD hat, aus denen er eventuell und gern Xerokopien einzelner Artikel anfertigen und weiterreichen würde. Vielleicht könnten wir in Einzel- und Spezialfällen diese Bemühungen von uns aus unterstützen?

15 A Solution And Another Puzzle
Chris Jakeman

Anthony Jordan hat das Rätsel aus dem letzten Heft gelöst. Und schon wieder kommt ein neues Rätsel. Welches ANS-Wort versteckt sich hinter: : A OVER MIN >R SWAP R@ + SWAP R> - ; ?

16 Cubic Roots Without Division
Fred Behringer

Ein Beitrag des Rezensenten. (Diese Art der rekursiven Rezension wird ab jetzt häufiger vorkommen.) Nichts Besonderes. Uralte Methode und nicht sehr effektiv: Dritte Wurzel mit Hilfe der Intervallhalbierung (funktioniert nach dem Zwischenwertsatz).

19 A Story
Ray Allwright

"Wir müssen den Rest der Welt von der Überlegenheit der Sprache Forth überzeugen. Mit religiösem Fanatismus allein schaffen wir das nicht. ..." usw. Das schrieb Mark Ditta kürzlich in comp.lang.forth. Ray hat das zu einem kleinen Drama verarbeitet, in dem nichts geschieht und immer wieder alles beim alten bleibt. Trotzdem: Ringsherum entsteht viel Neues und vergeht auch wieder. Unsere Helden aber jammern immer noch darüber, daß nichts geschieht und alles beim alten bleibt. Das Jammern macht sie glücklich.

22 4tH - the alternative Forth compiler
Hans Bezemer

Ich habe den Artikel gelesen. (Ich habe noch nie in meinem Leben so viele Artikel gelesen wie seit der Zeit, da ich begann, anderssprachige Forth-Zeitschriften zu rezensieren.) Der Autor ist mir von diversen anderen Schriften her als Forth-Autor bekannt. Den Artikel finde ich deswegen so interessant, weil er einen Einblick in die Werkstatt eines Forthlers an vorderster Front gewährt. Wie muß ein Mensch beschaffen sein, der Forth-Systeme entwickelt? Was bewegt ihn? Was treibt ihn? Wie geht er vor? 4tH ist anders als andere Forths. Aus dem Inhalt: "Anfänger greifen gern zu 4tH, da es ihnen das Erlernen von Forth leicht macht. Forth hat viele Gesichter. Eines davon ist 4tH." Und nun gebe ich noch das wieder, was Chris Jakeman zu diesem Artikel sagt: "Hans Bezemer hat ein effizientes, ausgetestetes, dokumentiertes ANS-Forth geschaffen, das für den Zugriff von einem C-Programm aus gedacht ist. Seine Web-Seite ist ebenfalls einen Besuch wert: http://visitweb.com/hansoft . Alle zweisprachigen (Forth & C) Programmierer sollten es sich überlegen, ob sie nicht 4tH in ihr Repertoire aufnehmen wollen."

30 Library News
Chris Jakeman

Mitteilung, daß die amerikanischen FORML-Proceedings von 1994 und 1995 endlich erschienen sind. FORML ist das Akronym für Forth Modification Laboratory. Die Bibliothek der FIK UK, die den Mitgliedern zum Ausleihen offensteht, ist gut sortiert. Es werden bereitgestellt: Mehrere Jahrgänge Byte und Dr. Dobb's Journal mit Forth-Artikeln, 18 einleitende Forth-Bücher, 13 für Fortgeschrittene, 5 MVP-Bände, 14 Beschreibungen einzelner Forth-Systeme, 5 Anleitungen zum Selbstbau eines Forth-Systems, eine Riesenauswahl an Tagungsberichten (FORML, euroForth, JOFAR).

33 Forth Programmer's Handbook

Eine Anzeige der Forth, Inc. Dort kann man das Buch offensichtlich bekommen.

Gehaltvolles

34 Forth Programmer's Handbook
Jeremy Fowell

Eine ausführliche Besprechung des neuen Buchs von Edward Conklin und Elizabeth Rather. Nicht ganz so kritisch wie die Besprechung, die im Vijgeblaadje 10 erschien. Aber: Nichts für Anfänger. Zum Nachschlagen für jemanden, der die ANS-Dinge nicht gar so trocken in den "offiziellen" Dokumenten nachlesen will. 67 Dollar ab FIG USA bis zur Bibliothek der FIG UK.

36 Deutsche Forth-Gesellschaft

Eine Anzeige zur Anwerbung von Mitgliedern, an der der Rezensent nicht ganz unschuldig ist. Der Rezensent dankt dem Redakteur der FIG UK, Chris Jakeman, für dessen aufgeschlossene Haltung bei unseren wechselseitigen Bemühungen um eine Intensivierung der internationalen Zusammenarbeit. Frage: Sollen wir in solchem Zusammenhang (auch beim Link auf uns auf der Webseite der FIG UK) an 'Deutsche Forth-Gesellschaft' festhalten oder auch hier die 'offizielle' Bezeichung 'Forth-Gesellschaft e.V.' verwenden?

37 Genetix - The Inside Story
Chris Jakeman

"Genetix von Prof. Bernard Hodson von der Universität Ottawa ist ein Programmiersystem, wird gesagt, das die Software-Welt revolutionieren wird" - usw. Die wiedergegebenen Informationen stammen von Neal Bridges, der das Buch "The Gene Machine" (95 engl. Pfund) für Forthwrite gelesen hat. 35 Jahre Entwicklung, früher als Charles Moore, knüpft unmittelbar an das Konzept der Turing-Maschine an, viel effizienter, viel leichter zu programmierende virtuelle Maschine, viel kompakterer Code - das alles wird, so Neal, im Buch behauptet. "Keine Ahnung von der Existenz von Forth, kein fundierter Vergleich mit dem, was es schon gibt. Viel Wind, keine Zukunft", sagt Neal.

40 Letters
Chris Jakeman

Dankbrief des bisherigen Redakteurs der Forthwrite und Mitbegründer der FIG UK, Gil Filbey, für die Ehrungen, die ihm zuteil, und die Aufmerksamkeiten, die ihm entgegengebracht wurden.

Nr. 97, Juli 1998

1 Editorial
Chris Jakeman

Chris begrüßt 6 neue Mitglieder, darunter zwei aus Frankreich und Schweden.

2 Forth News
Chris Jakeman

Berichte aus dem Nachrichtenforum comp.lang.forth: euro-Forth '98, Swift-Forth, PolyForth, 8031-Forth-ROM, Aztec 1.07, Win32Forth update 5, Mops 3.2, GForth, Quartus 0.3.0b, ASM68K, SPICE in Forth.

5 Reading the World (2)
Paul E. Benett

Zweiter Teil einer Serie. Diesmal bespricht der Autor Digitaleingabe über Tastatur und Zifferntastatur und Analogeingabe über AD-Wandlung. Er beschränkt sich auf das Wesentliche: 4x4-Tastenmatrix mit Dioden gegen Rückwirkungen bei doppelt gedrückten Tasten. Entprellung, software- oder hardware-mäßig, und dergleichen gehört nicht mehr zur Besprechung. Die Scancode-Abfrage, interrupt-gesteuert oder nicht, wird von einem kleinen Forth-Programm erledigt.

11 Interfacing SSBUV, a Scientific Instrument, to the Space Shuttle
Robert T. Caffrey

Ein Auszug aus einem anderswo erschienenen Artikel. Eine Anwendung des chipFORTH-Hard-und-Software-Entwicklungssystems zur Erstellung zuverlässiger Software für die STS-45-Space-Shuttle-Expedition.

13 Forth in Space
Joe Anderson

Joe Anderson war im Flugradarwesen beschäftigt und ist jetzt im Ruhestand. Ein sehr interessanter Hinweis-Artikel auf Forth-Aktivitäten bei der NASA. Er empfiehlt die Web-Seite groucho.gsfc.nasa.gov/forth , hier insbesondere eine umfassende Liste von James L. Rash über NASA-Projekte, in denen Forth eine Rolle spielt. Sehr interessant ist im Zusammenhang mit den Betreibern dieser Projekte auch der Hinweis auf die Deutsche Satelliten-Radio-Amateur-Gesellschaft, AMSAT-DL. Joe empfiehlt sie wegen des hohen Niveaus ihrer Untersuchungen, obwohl es sich doch um eine reine Amateur-Vereinigung handelt. Er zählt acht Gebiete auf, in denen AMSAT-DL Forth einsetzt.

Der Rezensent: Nach dem Krieg habe ich mich mit Radiobasteln auf mein späteres Hinwenden zum Computer und zu Forth vorbereitet. Die Kreise schließen sich. Vielleicht könnte man mal was über Forth im Radioamateurwesen in der VD lesen?

17 Vierte Dimension '98 No 2
Alan Wenham

Nicht schlecht, wie Alan seine Rezensionen unserer VD angeht: Fett gedruckte Stichworte statt Überschriften. Vielleicht sollte ich das auch machen?

19 Finite State Machines - 1/3
Graeme Dunbar

Ein "Tutorial", ein Lehrstück. Endliche Automaten, mit und

ohne Gedächtnis. Mealy-Automat, Moore-Automat. Was ist das? Beispiel Cola-Automat: Geld rein, aus endlich vielen Zuständen (Zustandsvektoren) einen wählen, zweimal eine Mark oder einmal Zweimark, Pepsi oder Apfelschorle, draufdrücken, Klappe heben, Dose entnehmen. Wenn nichts kommt, kräftig aufs Blech hauen. Wenn immer noch ohne Erfolg, entrüstet von dannen ziehen.
Chris Jakeman hatte recht, als er mir schon vorher schrieb, daß dieser Artikel wesentlich besser zu lesen sei als manche andere Schrift zum selben Thema. In den nächsten Ausgaben folgen noch zwei Artikel. Graeme liefert interessanterweise einen Internet-Hinweis auf ein Kapitel zu diesem Thema aus einem "elektronischen" Buch über Digitalelektronik: http://www.cs.berkeley.edu/~randy/CDL/chapter08.doc.htm .

23 Quad (Fixed-Point) Arithmetic
Ed Hersom

Ed Hersom gehört mit zu den ersten, die nach dem Krieg das entwickelt haben, was wir heute den Computer nennen. In seinem Programm macht er von der Formel sin(3A) = 3sin (A)-4(sin(A))^3 Gebrauch und berechnet sin(A) für gegebenes A bei vorgegebener Genauigkeit rekursiv. Er verwendet Pygmy Forth von Frank Sergeant. Er arbeitet in Festkomma-Darstellung, ohne Gleitkomma-Prozessor. Zum Vergleich mit Gleitkomma-CPU-Angaben macht er Experimente mit zwei 80186-Maschinen. Ed Hersoms Artikel paßt gut zu den Scientific-Forth-Library-Bestrebungen von Julian V. Noble auf Taygeta zur Förderung der Akzeptanz von Forth als seriöse Programmiersprache.

28 FORML Conference Proceedings '94 & '95
John Payne

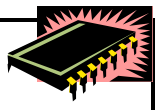
Eine sehr lebendige Besprechung der beiden Tagungsbände, gespickt mit humorvollen, beißend ironischen, scharfsichtigen und manchmal doch recht eigenen Bemerkungen. John läßt aber jedem das Recht, "seine eigene Meinung zu haben, auch wenn sie von seiner abweicht und damit falsch ist". Für uns interessant ist seine höchst positive Meinung über Bernd Paysan und Anton Ertl: "Diese Burschen (guys) wissen eine ganze Menge über Forth, über Portierbarkeit und darüber, wie man aus einem Prozessor auch noch das Allerletzte herausholt". Es geht dabei um GForth. Mitglieder der FIG UK können sich die beiden Bände gegen Portoerstattung aus den Beständen der Forth-Bibliothek ausleihen.

32 More of the Lighter Side
Chris Jakeman

Bemerkungen über Computer-Speicher, aus dem Netz, nicht ganz ernst zu nehmen. .... ARAM (Audio RAM) = ähnlich wie VRAM (Video RAM), beschreibt jedoch das Bild in Worten .... usw. Schön ist auch die Abbildung mit dem breiten Hintern, der sich gerade von der Parkbank erhoben hat, auf der er sich ausruhte, nachdem er sie frisch gestrichen hatte.

34 Wondrous Numbers
Graham Telfer

Der Autor lebt seit über zwei Jahren in Japan. Seine Forth-Bücher hat er in England zurückgelassen. Von Chris Jakeman hat er Aztec-Forth bekommen. Jetzt ist er dabei, Forth neu zu erlernen. Was ist besser dazu geeignet als zahlentheoretische Fragen? Gesagt, getan, beschäftigt er sich mit einem Forth-Programm zur Bestimmung von wundersamen Zahlen: Gegeben eine natürliche Zahl; wenn gerade, dann durch 2 teilen, wenn ungerade, dann mit 3 multiplizieren und 1 hinzuaddieren. Wundersame Zahlen sind solche, die sich auf 1 reduzieren lassen. Er bezieht sich auf das Buch "Gödel, Escher, Bach" von D.R. Hofstadter. Frage: Gibt es unwundersame Zahlen? Weitere Fragen: 1-4-2-1 ist ein Zyklus. Gibt es andere (nichtleere) Zyklen (deren Elemente also alle unwundersame Zahlen sind)? Kann man (in Abhängigkeit von der Ausgangszahl) eine obere Schranke für die nötige Anzahl von Schritten zur Reduzierung auf 1 angeben? Kann man zeigen, daß all diese Fragen grundsätzlich nicht beantwortet werden können? Kann man zeigen, daß man das nicht zeigen kann? Hofstadters Buch gibt keinen Hinweis. (Die Schildkröte bleibt Achilles die Antwort schuldig.) Mit den paar Zeilen Forth von Graham Telfer kann man sich aber der Beantwortung experimentell nähern.



MARC4

Ausgabe 4 „Transponder“ von R.Delianos Newsletter ist ab sofort verfügbar.

Nicht nur für die ‚Hardware‘-ker ist die 4. Ausgabe von MARC4 interessant. R. Deliano geht in seinem Newsletter über 20 Seiten auf IDIC-Transponder ein. Er beschreibt Anwendungen, das Funktionsprinzip, ein einfaches Lesegerät, den Baustein TK5530 und Optimierungen in der Ansteuerung des Bausteins.

Der Leser wird in dieser Ausgabe durch Details geführt, wie die Kopplung zwischen Transponder und Lesegerät, wie den Aufbau eines einfachen Lesegerätes und die Wicklung einer selbst herzustellenden Spule auf einer M5-Schraube. Blockschaltbilder und Rechenbeispiele illustrieren die jeweiligen Aussagen.

Ein kurzes Forth/Assembler-Listing zu den vorgestellten Bausteinen und Schaltungen vervollständigt eine gut zu lesende Arbeit.

Und wer nicht selbst mit Transpondern arbeiten und/oder experimentieren will, der erfährt hier zumindest – mit einfachen Worten erklärt – was ein Transponder ist, wie er arbeitet und wozu man ihn braucht.

Zu ‚beziehen‘ ist der Newsletter bei:
Rafael Deliano
Steinbergerstraße 37
82110 Germering
Tel./Fax 089-8418317

fep

Patchen ohne den Beigeschmack des Halbfertigen

von

Fred Behringer
Planegger Str. 24 , 81241 München

Die einfachste Art, ein Wort nachträglich zu ändern (z.B. Zur Anpassung von F-PC an die ANS-Gegebenheiten) besteht darin, das Wort gleich von vornherein zu DEFERieren. Das Apriori-DEFERieren widerspricht aber der Nachträglichkeit der beabsichtigten Anpassung. Im folgenden wird darüber gesprochen, wie man praktisch jedes Wort eines bestehenden Systems ohne neuerliche Metacompilation nachträglich, ich betone nachträglich, DEFERieren kann. Damit wird das Patchen zum Kinderspiel und das Patchen selbst verliert den Ruch des Unfertigen, Provisorischen, nicht ganz Hieb- und Stichfesten.

Stichworte: Turbo-Forth, Compiler, Metacompiler, Transputer-Forth

"Klassiker"? Diese Arbeit erhebt keinen Anspruch auf Neuheit oder Vollständigkeit. Es ist alles schon mal dagewesen. Quellen, die ich nicht angebe, sind mir entfallen. Hinweise werde ich aufmerksam studieren.

"Patch as Patch can", fordert Arndt Klingelnberg [15]. "Ein Patchen in Ehren kann niemand verwehren", räumt Ronald Zech [27] ein. Jedenfalls gibt es Systeme, an die man nicht rankommt oder nicht ran will und bei denen ein moderates Patchen angebracht erscheint.

Flickwerk? Patchen heißt soviel wie flicken, ausbessern, überbrücken, umleiten, modifizieren, nachträglich ändern. Patchen hat den Ruch der Flickschusterei. Leitet man ein **DEFER**iertes (ein vektorisiertes) Wort (per **IS**) nachträglich um, so spricht man normalerweise nicht von Patchen. Das "Patchen" des vorliegenden Artikels besteht darin, Worte, die eigentlich nicht vektorisiert sind, nachträglich zu vektorisieren. Ein solches "Patchen" kann als ein Patchen höherer Art, als ein "sauberes" Patchen, betrachtet werden.

Motive: In meinem Transputer-Forth-System F-TP 1.00 (es liegt auf ftp://ftp.taygeta.com/pub/forth/compilers/native/dos/transputer/) ist der (IBM-)Host-Server in Turbo-Forth (16 Bit) programmiert. Die Metacompilation ist eigentlich eine Cross-Metacompilation, der Assembler im Server ist ein Cross-Assembler, der Transputer wird (beim Metacompilieren) über den PEEK/POKE-Mechanismus angesprochen [3]. Den Übergang von den 16 Bits in Turbo-Forth zu den 32 Bits des Transputers bewältige ich für den Metacompilierungsvorgang weitestgehend durch ein zeitweiliges Patchen von **CONVERT** : Zahlen, die normal (einfachgenau) eingegeben werden, werden als doppeltgenau interpretiert.

Bei der gegenwärtigen Überarbeitung von F-TP strebe ich eine Art von inkrementeller Metacompilation an. Das System wird (durch Verschieben seines Endes während des Laufs) maximal erweitert und erzeugt ein neues System innerhalb seines eigenen freien Dictionary-Raums. Alle Worte des neuen Systems sind (im Gesamtsystem alt+neu) sofort nach ihrer Definition bereits wirksam. Die Idee ist folgende: Der übliche Forth-Compiler ist ein "inkrementeller" Compiler: An jeder Stelle des Compiliervorgangs kann das Programm unterbrochen und das gerade compilierte Wort ausprobiert und eventuell verändert werden. Warum also nicht eine Stufe höher gehen und auch das Metacompilieren inkrementell erledigen?

Das neue System wird vom alten abgenabelt, sobald es fertig ist. Viele der alten Worte (beispielsweise **STATE**) müssen beim Aufbau des neuen Systems schon von Anfang an im neuen System laufen, obwohl sie im alten System (dem Gesamtsystem) noch in ihrer alten Bedeutung agieren. Ich definiere sie im neuen System neu und leite das jeweilige alte Exemplar auf das neue um (ich vektorisiere das alte Exemplar und hänge das neue dran). Die Idee ist durchaus nicht neu. Grotke and Kelly [11],[12] metacompilieren auf ähnliche Art, und Ulrich Hoffmann ist mit seinem "Einfachen Forth" [13] auch nicht weit davon entfernt. Hier geht es mir nur um ein sinnvolles Patchen zur Verwirklichung dieses Ideenkreises.

Patchen bei anderen: Bei Ronald Zech [27] auf Seite 243 geht es um das nachträgliche Einfügen des Opcodes 72 (für **U**>=) anstelle des Opcodes 7C (für das ungenügende >=) an einer bestimmten Stelle in der **CODE**-Definition **UM/MOD**, um in F83 einen schwerwiegenden Fehler zu beheben. Ein solches Patchen könnte man als echtes Patchen bezeichnen. Es kann leicht Verwirrung stiften, wenn es nicht extrem gut dokumentiert wird. Das im vorliegenden Artikel besprochene Patchen über ein nachträgliches **DEFER**ieren kann als ein Patchen auf höherer Ebene betrachtet werden.

Claus Vogt patcht in seinem Zeichensatz-Programm [24] **BIOSKEY** und **BIOSKEY?** , indem er in die cfa einen Sprung zu eigenem Code hin einbaut.

Die Probleme von Egmont Woitzel und Udo Schütz in ihrem "Totalen Patch" [25] scheinen ähnlich zu liegen wie bei mir: Alte Definitionen sollen bei Neudefinition auf die Neudefinitionen umgeschaltet werden. Die Ausführung in [25] ist von meiner hier diskutierten sehr verschieden.

Grotke and Kelly [11],[12] suchen für Unterrichtszwecke einen Metacompiler, der durchsichtiger erscheint als der von F83 (ohne **HASH**-Technik usw.). Sie bauen das neue System innerhalb des alten auf und lassen Vorwärtsreferenzen nur in (schon initialisierter) vektorisierter Form zu. Speziell **LITERAL** muß bei ihnen neu definiert und an das alte **LITERAL** (das normalerweise nicht vektorisiert ist) gepatcht werden.

Feierbach and Thomas [9] beschreiben in ihrem Abschnitt über das Patchen das unmittelbare (nachträgliche) Einfügen des neuen Wortes als ersten Eintrag in das Parameterfeld der Colon-Definition des alten Wortes und das anschließende Dranhängen der cfa von **EXIT** in der zweiten (so überhaupt vorhandenen) Zelle des Parameterfeldes des alten Wortes. Siehe auch weiter unten.

Steve Davies [8] gibt in Forthwrite das Bearbeitungswort **PATCH** an, um ein ähnliches Problem zu lösen, wie ich es habe: Er möchte ein langes Programm nicht neu compilieren, wenn nur ein einziges, aber tief vergrabenes Wort neu definiert werden soll. (Es geht dabei um Compilieren, nicht Metacompilieren. Ich würde in diesem Fall neu compilieren, aber 1987 scheinen die Rechenzeitverhältnisse noch anders gelegen zu haben.) Steve Davies verwendet in etwa die Methode, die ich weiter unten in "Colon weniger einfach" beschreibe, also die Methode von Feierbach and Thomas [9]. Er schlägt vor, auch **CODE**-Definitionen damit zu bearbeiten.

FORBID , wie man es bei Marc Petremann [18] und anderen findet, kann als Patch-Werkzeug betrachtet werden. Es klinkt ein Wort aus dem System aus, indem es die Linkadreßkette an entsprechender Stelle überbrückt. Ich mache davon jetzt beim "inkrementellen Metacompilieren" (in Verbindung mit **ALLOW**) ausgiebig Gebrauch. Motto: Patchen ist nicht mehr gar so anrüchig, wenn man die eigentliche Handlung verschleiert, indem man den Vorgang "anständig" in ein Wort verpackt und ihm dadurch einen seriösen Anstrich verleiht. Anders ausgedrückt: Es wird immer und überall gepatcht, man sieht es nur nicht immer gleich.

In Turbo-Forth wird die Idee des Patchens beim Debugger **DEBUG** verwendet. Auch bei der Anbindung des von Charles Curley stammenden und von Michel Zupan [28] für Turbo-Forth bearbeiteten 8086-Disassemblers an den Decompiler **SEE** . Im einen Fall wird **NEXT** zeitweilig gepatcht, im anderen **SEE** ständig. **DEBUG** habe ich in derselben Technik für F-TP 1.00 übernommen. Vom 8086-Disassembler war natürlich für den Transputer T800 nichts zu verwenden. Ich habe aber für den T800 in ähnlicher Technik gleich zu Anfang meiner Arbeiten an F-TP 1.00 (ab 1989) einen Disassembler und Assembler entwickelt [2]. Diesen habe ich mit kleinen Abänderungen in F-TP 1.00 übernommen.

Alles im vorliegenden Artikel Gesagte kann als eine Art von Programm-Aufruf über Vektoren (vector execution) betrachtet werden. Was mich interessierte, war das Ziel, die Vektorisierung nachträglich an einem schon existierenden System vorzunehmen. Vector-Execution wird beispielsweise behandelt in [5],[7],[16],[21],[22],[23],[26] . Interessant ist das **EXECUTE:** in Laxens Artikel [16]. Das scheint ein Vorläufer des späteren **DEFER** zu sein.

ALIAS aus volksForth-83, das Ulrich Hoffmann in seinem "Kleinen Forth" [13] verwendet, kann als eine Umkehrung des hier verfolgten Ziels betrachtet werden: Die neue Definition soll dort die Funktion der alten übernehmen, bei mir übernimmt die alte (nachträglich) die der neuen. Wird ein solches **ALIAS** in der Form verwendet, wie sie bei Tracy and Anderson [22] vorgeschlagen wird, so kann man das als ein Zwischending auf dem Weg zum Patchen betrachten: Überall, wo das neue Wort (in einer weiteren Colon-Definition) aufgerufen wird, wird an dessen Stelle das alte (in die weitere Colon-Definition) eingesetzt. Das ist jedoch nicht das Ziel (siehe "Motive"), das ich in der vorliegenden Arbeit verfolge. **ALIAS** wird auch besprochen in [4],[6],[20].

Patchen ist das, was der Heimwerker macht, wenn er die durchgeschmolzene Haussicherung mit einer Sicherheitsnadel überbrückt. Patchen tut der Wartungsingenieur, der Fernsehreparateur und der Automechaniker. Auch des Arztes Handlung kann man so verstehen. Patchen hat was mit Diagnose, Fehlersuche, und Zuverlässigkeit [1] zu tun. Auch mit Kannibalisation [14]. Gentechniker sind, dem "Bild der Wissenschaft" zufolge, vorerst aufs intelligente Patchen der DNS angewiesen. Vom Analogrechner [10] her kennt man den Begriff des Patchboards. Patchen ist ein zentraler Begriff. Auch und vor allen Dingen in Forth. Patchen wird behandelt in: [3],[6],[8],[9],[15],[17],[23],[24],[25],[27].

CODE ganz einfach: Beliebige Worte, ob Colon- oder **CODE**-Definitionen, **VARIABLE**n oder **STRING**s, kann man (nachträglich oder gleich von vornherein) so vektorisieren, daß sie bei Aufruf eine (neudefinierte oder nicht neudefinierte) **CODE**-Definition (Colon-Definition und dergleichen geht nicht !) wirken lassen: Man braucht nur die pfa (Parameterfeldadresse) der (neuen) **CODE**-Definition in die cfa (Codefeldadresse) des "alten" Wortes zu schreiben, und schon ist das "Patchen" vollbracht.

Colon weniger einfach: Ist das neue Wort eine Colon-Definition (und das alte dann sinnvollerweise auch), wird es schwierig, wenn das alte leer ist, z.B. **: XXX ;** Es fehlt dann für ein unmittelbares Vorgehen (siehe gleich) eine Zelle im Wort. Natürlich erscheint eine leere Colon-Definition unsinnig. Es kann aber Situationen geben, in welchen sie doch sinnvoll ist. Bei dem Vorschlag weiter unten mit dem nachträglichen **DEFER**ieren funktioniert es auch bei leeren Worten.

Hat man eine Colon-Definition in der Form **: XXX NOOP ;** bereitgestellt, seiner Funktion nach ebenfalls eine leere Colon-Definition, dann kann man diese ohne weiteres als **DEFER**iertes Wort ansehen und darf die Konstruktion mit **IS** anwenden. Schreibt man nachträglich ´ **YYY IS XXX** , dann wird aus dem obigen das "gepatchte" Wort **: XXX YYY ;** (ich hatte bei der Metacompilation meines Transputer-Forth-Systems F-TP 1.00 an bestimmter Stelle Schwierigkeiten mit **DEFER XXX** und behalf mich mit eben dieser Konstruktion). Natürlich hätte statt **NOOP** auch irgendein anderes Wort in **XXX** stehen können: **: XXX ZZZ ;**

Will man eine Colon-Definition mit mehr als einem Wort nachträglich auf ein neues Wort umleiten, so kann man (siehe aber auch Lösung weiter unten) wie folgt vorgehen: Aus **: XXX WWW ZZZ ;** mache man (nachträglich) **: XXX YYY ;** (wenn **YYY** das inzwischen schon hinzudefinierte neue Wort ist). Platz ist genügend da. Es ist alles nur eine Frage der Organisation. Das ist genau die Vorgehensweise von Feierbach and Thomas [9]. Feierbach and Thomas verwenden Forth79 und hier speziell "Tick" in einer mir ungewohnten Weise. Auch das **FIND** in dem Patch-Wort **PATCH** in [9] kommt mir fremd vor. Für Turbo-Forth und die anderen Derivate von F83 (F-TP 1.00 gehört auch dazu) darf ich folgenden Vorschlag machen:

Syntax: **' YYY IS-NOW XXX**

```
: IS-NOW ( <name> cfa -- )
      ' >BODY      \ pfa von XXX
      ['] UNNEST   \ cfa von UNNEST
                   \ (im wesentlichen ;)
      OVER CELL+ ! \ nach pfa+Zelle von XXX
      ! ;        \ cfa von YYY nach pfa von
XXX
```

Für Turbo-Forth (und F83 usw.) beachte man, daß ich hier das ANS-Wort **CELL+** verwende, das in Turbo-Forth (16 Bit) gleichbedeutend mit **2+** ist und in F-TP 1.00 (32 Bit) gleichbedeutend mit **4+** . In F-TP 1.00 habe ich das eingebaut. Außerdem verwende ich weiter unten das ANS-Wort **CELLS**, in Turbo-Forth gleichbedeutend mit **2 *** und in F-TP 1.00 gleichbedeutend mit **4 *** .

Ein späteres erneutes Umleiten, z.B. nach **ZZZ**, kann dann wieder nach dem gewohnten Schema **' ZZZ IS XXX** geschehen.

VARIABLE: Ich betrachte einfachgenaue **VARIABLE**n. Im Prinzip sind die Vorgänge bei **2VARIABLE**n, **CONSTANT**en und **2CONSTANT**en dieselben und werden de(m)r Leser(in) überlassen. Über **CONSTANT**en sage ich gleich noch etwas. In dem weiter unten gemachten allgemeinen Vorschlag des nachträglichen **DEFER**ierens sind **VARIABLE**n, **2VARIABLE**n, **CONSTANT**en und **2CONSTANT**en mit enthalten.

Will man bei **VARIABLE**n das eben unter "Colon" besprochene Verfahren anwenden, so hat man (ähnlich wie bei leeren Colon-Definitionen) eine Zelle zu wenig. **VARIABLE**n hören mit der einen und einzigen Zelle (pfa) für ihren Inhalt auf. Ein abschließendes "**;**" gibt es nicht. Man kann aber wie folgt vorgehen:

Man mache die **VARIABLE** (z.B. **XXX**) nachträglich zur **CONSTANT**en, die die pfa der anzuhängenden neuen **VARIABLE**n (z.B. **YYY**) als (bei Aufruf auf den Stack zu legenden) Wert (nachträglich) zugeteilt bekommt. Das folgende Programm leistet diese Arbeit. (Man beachte, daß ich aus Gründen der Harmonie immer wieder die Bezeichnung "**IS-NOW**" verwende, von Mal zu Mal aber in anderer Bedeutung.)
Syntax: **' YYY IS-NOW XXX**

```
: IS-NOW ( <name> cfa -- )
      >BODY       \ pfa von YYY
      '           \ cfa von XXX
      ['] CONSTANT 4 CELLS + @\ Adresse des
                  \ Codeteils einer CONSTAN-
Ten
      OVER !       \ nach cfa von XXX spei-
chern
      >BODY ! ; \ pfa von YYY nach pfa von
XXX
```

Es ist gar nicht so leicht, an **DOCONSTANT**, wie der Codeteil von **CONSTANT**, den man hier braucht, beim Metacompilieren heißt, zu gelangen. In Turbo-Forth (und anderswo) wird er nicht explizit (über einen eigenen Namen) herausgeführt. Hat man eine per **CONSTANT** erzeugte Konstante, in Turbo-Forth ist **FILES** eine solche, dann kann statt des obigen **['] CONSTANT 4 CELLS + @** auch **['] FILES @** verwendet werden. Überraschenderweise wurden alle im F83-Buch von Zech [27] aufgeführten Konstanten von Forth 83 (**BL** , **BS** usw.) in der mir vorliegenden Version von Turbo-Forth von Marc Petremann als **CODE**-Definitionen ausgelegt und sind damit in dem hier besprochenen Sinn nicht brauchbar.

Ein erneutes Umdirigieren muß wieder über **' ZZZ IS-NOW XXX** vorgenommen werden. Ein **' ZZZ IS XXX** würde die cfa von **ZZZ** in die nunmehrige "Konstante" **XXX** plazieren, verlangt wird aber die pfa.

CONSTANT: Das Umleiten einer **CONSTANT**en scheint auf den ersten Blick überflüssig, da man ja einfach nur der alten **CONSTANT**en denselben Wert wie der neuen **CONSTANT**en zu geben braucht. Beim Metacompilieren (siehe oben unter "Motive"), um ein Beispiel zu nennen, ist aber immerhin der Fall denkbar, daß eine **CONSTANT**e im Verlauf des Prozesses, also nachträglich, einen neuen Wert zugeordnet bekommen soll (was natürlich eine Form des "Patchens" darstellt). Mit dem folgenden Programm, das bis auf ein zusätzliches "**@**" dem Programm unter "VARIABLE" gleich ist, kann der alten **CONSTANT**en der Wert der neuen (der von dem der alten verschieden sein kann) nachträglich zugeordnet werden.

Syntax: **' YYY IS-NOW XXX**

```
: IS-NOW ( <name> cfa -- )
      >BODY @     \ Wert in der pfa von YYY
      '           \ cfa von XXX
      ['] CONSTANT 4 CELLS + @\ Adresse des
                  \ Codeteils einer CONSTAN-
Ten
      OVER !      \ nach cfa von XXX spei-
chern
      >BODY ! ;   \ Wert in pfa von YYY nach
```

```
                \ pfa von XXX
```

Ein erneutes Umdirigieren muß wieder über **' ZZZ IS-NOW XXX** vorgenommen werden. Ein **' ZZZ IS XXX** würde die cfa von **ZZZ** in die pfa der Konstanten **XXX** plazieren, verlangt wird aber der Wert aus der pfa von **ZZZ**. Aber auch eine einfache "Neubelegung" der **CONSTANT**en **YYY** zieht nicht automatisch eine Neubelegung der **CONSTANT**en **XXX** nach sich. **XXX** muß per **IS-NOW** an **YYY** angeglichen werden. Überhaupt kann das obige **IS-NOW** auch als generelles Werkzeug zur Neubelegung einer "Konstanten" angesehen werden.

DEFER als Lösung: Worte wie **KEY** werden normalerweise gleich von vornherein **DEFER**iert ausgelegt (**DEFER KEY**). Damit können sie dann nachträglich zu beliebigen Eingabeworten hin umgeleitet werden. Üblich ist zunächst einmal **´ (KEY) IS KEY** , und in **(KEY)** steckt dann das eigentliche Tastatureingabewort. Man beachte, daß ich weiter unten **' KEY @** aufrufe, um an den für vektorisierte Worte zuständigen Codeteil zu gelangen. Sollte im System des geschätzten Lesers **KEY** nicht vektorisiert sein, dann nehme man irgendein anderes vektorisiertes Wort **WWW** und rufe **' WWW @** auf.

Nachträgliches DEFERieren: In der Methode, auf die der vorliegende Artikel abzielt, wird in die cfa des nachträglich zu vektorisierenden Wortes die Adresse des für vektorisierte Worte zuständigen Codeteils geschrieben. In die pfa wird die cfa des Zielwortes geschrieben. Das zu vektorisierende Wort muß also außer der cfa noch mindestens eine Zelle enthalten. Bei Colon-Definitionen ist das immer der Fall, da dort ja mindestens noch die cfa von **UNNEST** für den Rücksprung im Parameterfeld auftritt. Bei **VARIABLE**n besteht auch keine Schwierigkeit, da hier auf die cfa noch die Zelle mit dem Inhalt folgt. Bei **CONSTANT**en ebenso. **CODE**-Definitionen haben neben der cfa zwar mindestens noch ein Byte für den Rücksprung zu **NEXT** , es wird aber eine ganze Zelle benötigt. In meinem Transputer-Forth-System sind die **CODE**-Definitionen (und fast alles andere auch) wortorientiert (aligned). Bei per **ALIGN** ausgerichteten Systemen (bei INTEL-Systemen nicht immer) funktioniert die hier zu besprechende Methode des nachträglichen **DEFER**ierens also auch bei **CODE**-Definitionen. Mit anderen Worten, die Methode funktioniert bei solchen Systemen zumindest bei Colon- und **CODE**-Definitionen und bei **VARIABLE**n und **CONSTANT**en. Bei **STRING**s funktioniert sie in 16-Bit-Systemen wegen des Vorhandenseins von Max-Byte und Count-Byte auch immer, in meinem F-TP 1.00 dagegen nicht unbedingt. Hier muß die String-Variable für mindestens zwei Byte maximal ausgelegt sein. Interessanterweise können die Worte, auf die bei einer solchen nachträglichen Vektorisierung umgeleitet werden soll, einer beliebigen Wortklasse angehören. Es folgt mein Vorschlag zur nachträglichen Vektorisierung.

```
: REDEFER ( <name> -- )
      '          \ cfa von <name> holen
      ['] KEY @  \ Adresse des Codeteils zur
                 \ Vektorisierung
      OVER !      \ nach cfa(<name>) spei-
chern
      ['] CRASH SWAP  \ Adresse der
                      \ Fehlermeldung
      CELL+ ! ;   \ nach pfa(<name>) spei-
chern
```

Syntax: **REDEFER XXX ' YYY IS XXX**

Der eben gemachte Vorschlag eignet sich für Fälle, bei denen ein an sich nicht vektorisiertes Wort nachträglich vektorisiert werden, aber erst später wirklich auf ein anderes Wort geleitet werden soll. Die obenerwähnte Fehlermeldung weist auf ein eventuell vergessenes Umleiten hin.

Auch hier kann man das nachträgliche Vektorisieren und das erstmalige Umleiten unter Einsparung eventuell erforderlicher Fehlermeldungen wiederum zu einem einzigen Vorgang zusammenfassen. Ich verwende wieder **IS-NOW** , diesmal aber in anderer Bedeutung; genauer gesagt, mit derselben Wirkungsweise, die aber auf anderem Wege erzielt wird:

Syntax: **' YYY IS-NOW XXX**

```
: IS-NOW ( <name> cfa -- )
      '          \ cfa von XXX
      ['] KEY @\ Adresse des Codeteils
               \ zur Vektorisierung
      OVER !   \ nach cfa von XXX speichern
      >BODY ! ;\ cfa von YYY nach pfa von XXX
```

State-Smartness: Ein neuerliches Umleiten kann in gewohnter Weise über **' YYY IS XXX** vorgenommen werden. Es schadet aber auch nichts, grundsätzlich **' YYY IS-NOW XXX** zu verwenden. In einem solchen Fall wäre zu überlegen, ob man nicht gleich **IS** im Sinne von **IS-NOW** umdefinieren sollte. Man beachte, daß **IS** noch eine Art von Zustandsabhängigkeit (state-smartness) eingebaut hat, die dann bei genauerer Vorgehensweise in **IS-NOW** noch zu berücksichtigen wäre.

HOOK-ON: Zum Abschluß noch ein Wort, das meine eigentliche Absicht (nachträgliche Übernahme der Funktion des neuen Wortes durch das schon vorher definierte alte Wort) besser verwirklicht. **HOOK-ON** ist hier so eine Art nachgeschobenes Modifikationswerkzeug, das ähnlich wie **IMMEDIATE** auf das zuletzt definierte Wort wirkt. Man beachte die Syntax: Erst wird die cfa des alten (zu ersetzenden) Wortes verlangt, dann wird das neue Wort (das bei meinem Vorhaben dieselbe Bezeichnung tragen soll wie das alte, was aber ansonsten durchaus nicht erforderlich ist) definiert und schließlich wird das neue Wort so an das alte gehängt, daß bei Aufruf sowohl des alten wie auch des neuen Wortes das neue in Funktion tritt. In bezug auf die zulässigen Wortklassen gilt das, was unter "Nachträgliches DEFERieren" gesagt wurde.

Syntax: **' XXX : YYY … ; HOOK-ON**

```
: HOOK-ON ( cfa -- )
      ['] KEY @       \ Adresse des Codeteils
                      \zur Vektorisierung
      OVER !          \ nach cfa von XXX
      LAST @ NAME>    \ cfa von YYY
      SWAP >BODY ! ;  \ nach pfa von XXX
```

Die Anpassung an die weiter oben gemachten Vorschläge bleibe den Leser(inne)n überlassen.

HOOK-ON-VARIABLE: Und schließlich dasselbe nochmal für **VARIABLE**n, bei denen die neu definierte (im laufenden System) ja zuerst auch noch den Wert der alten übernehmen soll. Um nicht durcheinander zu geraten, habe ich auch hier wieder eine Syntax mit dem "Tick" gewählt, obwohl das einfache Aufrufen von **XXX**, was ja unmittelbar die pfa liefert, vielleicht natürlicher gewesen wäre.

Syntax: **' XXX VARIABLE YYY HOOK-ON-VARIABLE**

```
: HOOK-ON-VARIABLE ( cfa -- )
      DUP >BODY @    \ Inhalt von XXX
      LAST @ NAME>   \ cfa von YYY
      TUCK >BODY !   \ Inhalt von XXX nach pfa
                     \ von YYY
      OVER >BODY !   \ cfa von YYY nach pfa
                     \ von XXX
      ['] KEY @      \ Adresse des Codeteils
                     \ zur Vektorisierung
      SWAP ! ;        \ nach cfa von XXX
```

Die Anpassung an die weiter oben gemachten Vorschläge bleibe wiederum den Leser(inne)n überlassen.

[1] Barlow, R.E., and F. Proschan: Statistische Theorie der Zuverlässigkeit, Frankfurt/M. 1978.

[2] Behringer, F.: An Assembler/Disassembler for the Transputer T800 on an IBM-AT Host, CompUser 3 (1990), Heft 1, S.23-33.

[3] Behringer, F.: Umlaute in den Namen von Forth-Worten, Vierte Dimension 11 (1995), Heft 4, S.37-40.

[4] Brien, J.: ALIASING, Forthwrite 62 (1991), S.16.

[5] Brodie, L.: Programmieren in Forth, Übersetzer Bernd Steinbach, Hanser-Verlag München 1984.

[6] Charlton, G.: Retyping Local Variables with an Alias, Forthwrite 50 (1989), S.15-16.

[7] Dale, F.: Vectored Execution in the Real World, Forthwrite 25 (1985), S.25-27.

[8] Davies, S.: Speaking of Patches, Forthwrite 34 (1987), S.14.

[9] Feierbach, C., and P. Thomas: Forth Tools and Applications, Reston Publishing Company, Inc. 1985.

[10] Feilmeier, M.: Hybridrechnen, Birkhäuser-Verlag Basel 1974.

[11] Grotke, G.T., and G.M. Kelly: Un métacompilateur simple, JEDI 35 (1987), S.7-9.

[12] Grotke, G.T., and G.M. Kelly: A Simple Metacompiler, Forthwrite 24 (1985), S.10-18.

[13] Hoffmann, U.: Ein Weg, wie man Forth klein macht, Vierte Dimension 7 (1991), Heft 1, S.10-11.

[14] Kaufmann, A.: Zuverlässigkeit in der Technik, München 1970.

[15] Klingelnberg, A.: KEYB8B, Vierte Dimension 9 (1993), Heft 1, S.17-20.

[16] Laxen, H.: A Techniques Tutorial: Execution Vectors, Forth Dimensions 3 (1982), Heft 6, S.174.

[17] Payne, J.: Simpler Forth, Forthwrite 57 (1990), S.25-27.

Objektorientierte Programmierung in comFORTH 4

Teil – 2 –

Dr.-Ing. Egmont Woitzel
FORTech Software Entwicklungsbüro
Joachim-Jungius-Straße 9 – 18059 Rostock
EWOI@FORTECH.DE

Im ersten Teil seines Aufsatzes stellte Egmont Woitzel die Sichtbarkeitsregeln, Instanzvariablen, einfache Methoden, typisierte Zeiger und Klassendefinitionen in comFORTH 4 vor. Der zweite, hier vorliegende Teil befaßt sich mit den Themen Vererbung und Virtuelle Methoden.

„Das Objektmodell von comFORTH 4 orientiert sich sehr stark an C++. Einer der wichtigsten Gründe dafür ist die angestrebte (teil)automatische Übersetzung von comFORTH-Klassen in COM-Objekte. Ein weiteres Argument für diese Wahl besteht darin, daß viele , mehrsprachige' Programmierer mit dem C++ Objektmodell vertaut sind."

Gelbe Birnen

Wie im richtigen Leben trifft es uns unerwartet: Die vermuteten gelben Äpfel sind in Wirklichkeit gelbe Birnen! Das konnte keiner vorhersehen. Unser Auftraggeber lenkt ein und gibt uns zusätzliche Zeit für eine komplette Überarbeitung unseres Programms. Grüne Birnen gibt es ja schließlich auch. Unser bisheriger Ansatz platzt aus allen Nähten. Nutzen wir also die zweite Chance.
Wie es aussieht, benötigen wir für Äpfel und Birnen jeweils einen eigenen Zeiger auf einen der aktuellen Farbe zugeordneten Zähler. Außerdem sollten wir gleich dafür sorgen, daß wir Äpfel und Birnen getrennt und zusammen zählen können. Das wird sowieso gebraucht und hilft beim Testen. Nach

den bisherigen Erfahrungen ist vermutlich auch noch mit Pflaumen oder Pfirsichen zu rechnen, wir sollten uns also nicht auf zwei Früchte festlegen. Das läuft auf eine Liste für die verschiedenen Früchte hinaus, die jeweils eine Liste aller zugehörigen Zähler und einen Zeiger auf den davon aktiven enthalten. Es wäre gar nicht schlecht, wenn man die Listen nur einmal programmieren müßte...

Vererbung

... heißt das Zauberwort, mit dem man beim objektorientierten Programmieren Code in unterschiedlichen Situationen wiederverwenden kann. Im comFORTH 4 Objektsystem kann eine Klasse von beliebig vielen Elternklassen erben. Dazu wird zwischen CLASS: und ;CLASS das Wort INHERIT benutzt:

classname INHERIT parentname

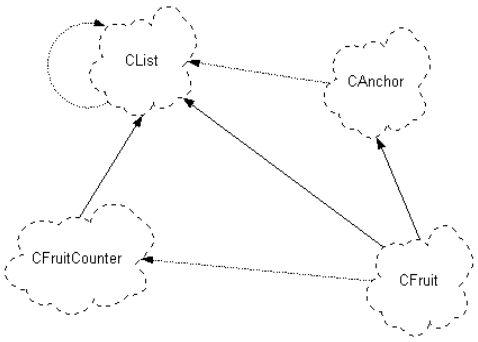
Als classname wird der Name der gewünschten Elternklasse angegeben. Hinter INHERIT folgt der Name, unter dem das der Elternklasse zugeordnete Teilobjekt identifiziert werden soll. Soll eine Klasse von einer anderen erben, muß man das veranlassen, bevor Instanzvariablen oder die später eingeführten virtuellen Methoden definiert werden. Klassen, die von keiner anderen erben, werden auch Basisklassen genannt. Eine Klasse, die von einer anderen geerbt hat, wird auch als von dieser abgeleitet bezeichnet.

Das Erben von einer Klasse ähnelt der Definition einer Instanzvariable mit dem Typ dieser Klasse. In Bezug auf den Aufbau des Objekts ist es sogar identisch. Ebenso benimmt sich ein per INHERIT definiertes Teilobjekt genauso wie eines, das durch OBJECT definiert wurde. Die Wirkung von INHERIT reicht jedoch weiter. Durch das Erben von einer Klasse werden die Objekte der abgeleiteten Klasse buchstäblich auch Objekte der Elternklasse. Das bedeutet, daß alle Methoden, die auf Objekte der Elternklasse angewandt werden dürfen, auch auf Objekte der abgeleiteten Klasse angewandt werden dürfen. Daher darf man einem Zeiger nicht nur Objekte der Klasse des Zeigers, sondern auch von allen Klassen zuweisen, die von dieser geerbt haben.

Einer abgeleiteten Klasse besitzt gegenüber der Elternklasse eine gewisse Vertrauensstellung. Daher kann man ihr auch ein etwas weitergehendes Interface zur Verfügung stellen. Dazu dient die Sichtbarkeitsklasse PROTECTED. Als PROTECTED definierte Worte sind im Inneren von abgeleiteten Klassen zusätzlich zu den öffentlichen Definitionen sichtbar. Nur die in der Elternklasse privat definierten Worte sind auch für die Erben nicht sichtbar.

Bunte Früchte zum Ersten

Zurück zu unseren Früchten. Ausgerüstet mit unserem neuen Werkzeug Vererbung können wir die Behandlung verketteter Listen ein für alle mal erledigen. Eine kurze Analyse bringt zutage, daß eine Trennung in eine Klasse für Listenelemente und eine für einen Listenanker sinnvoll ist. Unsere Klasse für die Apfelzähler benennen wir passender als Früchtezähler. Ein Früchtezähler ist gleichzeitig auch ein Listenelement, daher leiten wir diese Klasse von der Klasse der Listenelemente ab. Neu einführen müssen wir eine Klasse für die einzelnen Früchte. Diese Objekte müssen gleichzeitig Elemente der Liste der Früchte und Anker für die Listen der Früchtezähler sein, daher müssen wir sowohl von den Listenelementen als auch von den Ankern erben. Es ergibt sich folgendes Bild:
Die durchgezogenen Linien symbolisieren dabei eine Vererbungsbeziehung (lies: CFruitCounter erbt von CList), die gepunkteten eine Verweisbeziehung (lies: Objekte der Klasse



CAnchor verweisen auf Objekte der Klasse CList).

Beginnen wir mit der Klasse für die Listenelemente. Glücklicherweise müssen wir unsere Listen nur erweitern, so daß wir mit einer Initialisierungsmethode und einer Methode für die Erweiterung der Liste auskommen.

```
CLASS: CList
  PROTECTED
  CList POINTER Tail
  ;CLASS

ALSO CList DEFINITIONS
PUBLIC

M: Prepare ( -- )( prepare for first use )
  0 TO Tail ;M

M: Link ( 0 | addr -- addr )( append the given list to THIS )
  TO Tail THIS ;M

PREVIOUS DEFINITIONS
```

Die rekursive Natur der Liste ist an der Definition der einzigen Instanzvariablen zu erkennen, die als Zeiger auf ein Objekt des eigenen Klasse implementiert wird. Die Methode zum Einketten des Elements reflektiert dies auch noch einmal. Link erhält eine unter Umständen leere Liste, trägt diese als eigenen Schwanz ein und gibt das Element selbst als neue Liste zurück. Dieses Interface wird erst im Zusammenhang mit der Klasse für die Listenanker plausibel.

```
CLASS: CAnchor
  PROTECTED
  CList POINTER Head
  ;CLASS

ALSO CAnchor DEFINITIONS
PUBLIC

M: Prepare ( -- )( prepare for first use )
  0 TO Head ;M

M: Insert ( addr -- )( insert a list element )
  Head  SWAP CList Link  TO Head  ;M

PREVIOUS DEFINITIONS
```

Das ging schon flott von der Hand. Kommen wir jetzt zur Definition der Klasse für die Früchtezähler. Diese enthält im Gegensatz zur früheren Apfelzählerklasse keine Anzeigefunktion mehr. Die Anzeige bleibt der Früchteklasse vorbehalten. Anstelle der Anzeigemethode kommt eine neue Methode für das Auslesen des Zählers in das Interface. Damit bereinigen wir nebenbei eine Schwachstelle der Vorgängerversion. Dort wurde nicht nur beim Auslesen, sondern auch beim Zusammenzählen aller Äpfel per @ auf die Zählervariable zugegriffen. Das kapseln wir jetzt in der Methode Get.

```
CLASS: CFruitCounter ( apple counter class )
  CList INHERIT CList:: \ list of colors
  VARIABLE #Fruits
  ;CLASS

ALSO CFruitCounter DEFINITIONS
PUBLIC

M: Prepare ( addr -- )( reset count and insert into list )
  CAnchor PTR anchor
  CList:: Prepare
  #Fruits OFF
  THIS anchor Insert ;M

M: Add ( n -- )( add the given number of fruits )
  #Fruits +! ;M

M: Get ( -- n )( return the current number of fruits )
  #Fruits @ ;M

PREVIOUS DEFINITIONS
```

Interessant ist vor allem die Definition von Prepare. Hier wird zunächst die gegebene Adresse des Listenankers in einen lokalen Zeiger gerettet und dann als erstes das von CList ererbte Teilobjekt initialisiert. Die nach CList:: gerufene Methode Prepare ist die in der Klasse CList definierte! Erst nach dem Rücksetzen der Zählvariablen wird dann der Zähler in die Liste des gegebenen Ankers eingehängt. In unserer Anwendung wird das ein Objekt der Klasse CFruit sein. Deren Definition ist zwar ein bißchen länger, sollte uns aber über weite Strekken bekannt vorkommen. Der bisherige Zeiger auf die aktuelle Farbe und der Listenanker für die existierenden Zähler sind jetzt als Instanzvariablen wiederzufinden. Für die Liste der Früchte vereinbaren wir jetzt ein Objekt der Klasse CAnchor innerhalb von CFruit. Für die Auswahl des aktuellen Zählers brauchen wir jetzt eine eigene Methode, das eigentliche Zählen delegieren wir an das angezeigte Zählerobjekt.

```
CLASS: CFruit ( represents one special kind of fruits )
  PRIVATE
  CList INHERIT CList::                    \ list of fruits
  CAnchor INHERIT CAnchor::                \ list of colors
  CFruitCounter POINTER CurrentCounter \ current color
  ;CLASS

ALSO CFruit DEFINITIONS
PRIVATE

CAnchor OBJECT FruitList ( -- addr )( the list of fruits )
  FruitList Prepare

PUBLIC

M: Get ( -- u )( total count of fruits in the color list )
  Head                        \ get the list head
  CFruitCounter PTR counter   \ it is a list of counters
  0                           \ start with zero fruits
  BEGIN                       \ walk through the counter List
    counter
  WHILE                       \ there is a current counter
    counter Get +
    counter Tail TO counter
  REPEAT ;M

M: Prepare ( -- )( and add the object to the counter list )
  CList:: Prepare
  CAnchor:: Prepare
  0 TO CurrentCounter
  THIS FruitList Insert ;M

M: Select ( addr -- )( select the current counter )
  TO CurrentCounter ;M

M: Add ( n -- )( delegate counting )
  CurrentCounter Add ;M

M: ? ( -- )( display the current fruit count )
  CurrentCounter Get . ;M

M: All? ( -- )( display the total count of these fruits )
  Get . ;M

: GetTotal ( -- u )( return the total count of all fruits )
  FruitList Head      \ get the list head
  CFruit PTR fruit    \ it is a list of fruits
  0                   \ start with zero fruits
  BEGIN               \ walk through the fruit List
```

```
    fruit
  WHILE                \ there is a current counter
    fruit Get +
    fruit Tail TO fruit
  REPEAT ;
```

PREVIOUS DEFINITIONS

Auf dieser Basis können wir das Programm um die eigentlichen Objekte vervollständigen. Das ist bloß noch Fleißarbeit.

```
CFruit OBJECT Apples ( -- addr )( apple pointer )
  Apples Prepare

CFruitCounter OBJECT RedApples ( -- addr )( apple counter )
  Apples RedApples Prepare

CFruitCounter OBJECT GreenApples  ( -- addr )
                                  ( apple counter )
  Apples GreenApples Prepare

CFruit OBJECT Pears ( -- addr )( pear pointer )
  Pears Prepare

CFruitCounter OBJECT GreenPears  ( -- addr )( pears counter )
  Pears GreenPears Prepare

CFruitCounter OBJECT YellowPears( -- addr )
                                  ( pears counter )
  Pears YellowPears Prepare
```

Bei der Steuerung der aktuellen Farben gehen wir erst mal einen Kompromiß ein. Da wir wissen, welche Früchte in welcher Farbe vorkommen, vereinbaren wir die Auswahl der Zähler fest. Der Bauch sagt uns aber schon, daß dies keine Lösung auf Dauer sein kann.

```
: Red ( -- )( activate red apples )
  RedApples Apples Select ;

: Green ( -- )( activate green apples and pears )
  GreenApples Apples Select
  GreenPears Pears Select ;

: Yellow ( -- )( activate yellow pears )
  YellowPears Pears Select ;

Green ( -- )( initialize fruits )

: Total? ( -- )( syntactic sugar )
  CFruit GetTotal . ;
```

Nach dieser Operation sind wir froh, wenn unser Patient überhaupt noch lebt. Ein Test beruhigt da ungemein:

```
3 Apples Add  ok
7 Red Apples Add  ok
4 Yellow Pears Add  ok
Apples ? 7  ok
Red Apples ? 7  ok
Apples All? 10  ok
Pears ? 4  ok
Total? 14  ok
```

Wie nicht anders zu erwarten, ist die Implementation insgesamt aufwendiger geworden. Aber zur Bestimmung der Gesamtanzahl an Früchten müssen wir eben rekursiv durch eine Liste der Listen laufen. Immerhin ist es uns gelungen, für das Eintragen von Zählern und Früchten denselben Code zu benutzen, nämlich das in CAnchor definierte Insert (das seinerseits Link aus CList aufruft). Ganz zufrieden sind wir allerdings noch nicht. Gerade das aufwendige Durchlaufen einer Listen zum Aufsummieren der einzelnen Werte existiert zweimal in fast identischer Fassung, nämlich in GetTotal und in Get in der Klasse CFruit. Dummerweise laufen wir einmal entlang der Zählerliste und einmal entlang der Früchteliste, und das Get der Früchte ist nicht mit dem Get der Zähler zu verwechseln. Enden hier unsere Möglichkeiten?

Virtuelle Methoden

Das was wir jetzt brauchen, sind sogenannte virtuelle Methoden. Virtuelle Methoden verhalten sich eigentlich wie ganz normale Methoden. Der wesentliche Unterschied zu ihnen besteht darin, daß innerhalb einer Vererbungshierarchie virtuellen Methoden mehrfach Code zugewiesen werden darf. Falls sich zur Übersetzungszeit nicht eindeutig feststellen läßt, welcher Code aufgerufen werden muß, erfolgt seine Auswahl erst zur Laufzeit. Dies nennt man “späte Bindung” oder “*late binding*” im Gegensatz zur “frühen Bindung” oder “*early binding*” zur Übersetzungszeit. Immer wenn man ein Problem ausschließlich mit früher Bindung lösen kann, benötigt man keine virtuellen Methoden. Wir brauchen diese aber, da beim Aufsummieren zwei verschiedene Varianten von Get abgearbeitet werden sollen.

Virtuelle Methoden werden ebenso wie Instanzvariablen zwischen CLASS: und ;CLASS definiert. Soll eine Klasse erben, so muß das erste INHERIT vor der ersten virtuellen Methode stehen. Zur Definition dient das Wort VIRTUAL. Virtuelle Methoden unterwerfen sich denselben Sichtbarkeitsregeln wie alle anderen Definitionen einer Klasse.

```
VIRTUAL virtualname
```

Um einer virtuellen Methode Leben einzuhauchen, wird die Konstruktion

```
V: virtualname ... ;V
```

verwendet, die ansonsten genauso wie das Pärchen M: und ; M funktioniert. Solange in einer Klasse eine virtuelle Methode existiert, der noch kein Code zugewiesen wurde, darf man

genaugenommen kein Objekt von dieser Klasse instantiieren. Beim Aufruf einer solchen freigelassenen Methode ist nicht mehr als ein Fehlerabbruch zu erwarten. Aus diesem Grund werden Klassen, die virtuelle Methoden ohne zugewiesenen Code enthalten, auch abstrakte Klassen genannt.

Der Aufruf virtueller Methoden ist im comFORTH 4 Objektmodell durch die Verwendung von sogenannten virtuellen Methodentabellen nur wenig langsamer als der Aufruf normaler Methoden. Die Sequenz zur Bestimmung der Adresse des aufzurufenden Forth-Codes würde man etwa wie folgt hinschreiben:

... DUP @ xxx + @ ...

xxx steht dabei für den der Methode zugeordneten Offset in der Methodentabelle. Es erfogt also bei später Bindung im Gegensatz zu anderen Objektmodellen keine Suche im Wörterbuch. Laufzeit sollte daher kein Grund für die Ablehnung der späten Bindung sein.

Bunte Früchte zum Zweiten

Der richtige Platz für unsere Summierung ist die Klasse CAnchor, da wir in jedem Fall eine ganze Liste aufsummieren wollen. Dazu müssen wir aber davon ausgehen, daß alle Listenelemente einen aufzusummierenden Einzelwert liefern können. Also müssen wir in der Klasse CList eine virtuelle Methode Get definieren, die in den von CList abgeleiteten Klassen spezifiziert werden (wir haben Get ja bereits als normale Methode in beiden von CList abgeleiteten Klassen definiert). In CList selbst ist keine Implementation von Get sinnvoll. CList wird damit zur abstrakten Klasse.

Hier folgt die neue Definition der Elementeklasse:

```
CLASS: CList
  PUBLIC
  CList POINTER Tail
  VIRTUAL Get ( -- n )( retrieve the element value )
  ;CLASS
```

Es ist zu beachten, daß sowohl Tail als auch Get öffentlich definiert werden, da sie in CAnchor verwendet werden, die ja kein Erbe von CList ist. Unsere Summation verschieben wir per *cut&paste* in die neue Methode Sum von CAnchor.

```
M: Sum ( -- n )( return the sum of all element values )
  Head PTR element       \ element points to CList objects
  0
  BEGIN                  \ walk through the element chain
    element
  WHILE                  \ end of chain not reached yet
    element Get +
    element Tail TO element
  REPEAT ;M
```

Die übrigen Änderungen sind ebenso einfach. In der Klasse CFruitCounter muß die Methode Get nicht mehr per M: definiert werden, sondern nur mit V: genau denselben Code zugewiesen bekommen.

```
V: Get ( -- n )( return the current number of fruits )
  #Fruits @ ;V
```

Richtig spürbar werden die Vereinfachungen natürlich erst in CFruit. Hier verschwinden die beiden Summationsschleifen. Get wird ebenfalls per V: Code zugewiesen, nun aber nicht mehr die bisherige Schleife, sondern nur noch ein Aufruf auf die von CAnchor ererbte Methode Sum. Analog reduziert sich GetTotal auf das Anwenden von Sum auf den Anker der Früchteliste.

```
V: Get ( -- u )( return the total count of fruits )
  Sum ;V
```

Absender : michael@malente.forth-ev.de (**Michael Kalus**)
Betreff : Forth und grafische Oberflächen

Hi

Wie ist Eure geschätzte Meinung zu folgendem:

Forth ist eine Allzweckssprache. Ihr Standard-Wortschatz ist unabhängig von der darunter arbeitenden Maschine. So wurde es möglich, Code auf unterschiedliche Plattformen zu bringen und trotzdem bei Bedarf die speziellen Eigenschaften einer Maschine zu nutzen. In Forth entwickelte sich ein Wortschatz der häufige Operationen der Datenverarbeitung beschreibt - die wordsets. Es ist eine Art 'lebendige' Sprache die sich da entwickelt indem sie benutzt wird.
Bei den graphischen Oberflächen scheint mir ein ähnliches Problem vorzuliegen. Betriebssysteme auf unterschiedliche Maschinen bedienen auf unterschiedliche Weise die graphischen Oberflächen.

Ich könnte mir vorstellen, daß ein SWAP den Ort zweier grapischer Objekte auf dem Schirm vertauscht, ein DROP ein Objekt löscht usw. Oder ein SIZE (n --) die Größe des Objektes verändert, wobei n eine Angabe in % sein könnte. Auch ein MOVE (x1 y1 x2 y2 --) kann ich mir vorstellen. Nun, das soll mal genügen, um zu illustrieren was ich meine.

Bisher hörte ich auf den Tagungen wohl, das die jeweiligen Forthsysteme der PC's auch deren graphische Oberflächen bedienen können. Dazu wurden recht aufwendig deren Anweisungen in Forth nachgebildet. Das kommt mir so vor, wie ehedem die Nachbildung des Assemblers für einen bestimmten Prozessor mittels Forth - mühselig aber notwendig und auch nützlich, aber keine höhere Programmiersprache wie die wordsets selbst.

Bisher habe ich noch nicht davon gehört – oder hier gelesen – daß im Forth die wichtigen und wiederkehrenden Operationen an graphischen Objekten sich auch zu so etwas wie 'graphischen primitives' entwickelt hätten.

Kommentare dazu? *fep*

Forth-Gruppen regional

Moers
Friederich Prinz
Tel.: 02841-58398 (p) (Q)
(Bitte den Anrufbeantworter nutzen !)
(Besucher: Bitte anmelden !)
Treffen: (fast) jeden Samstag,
14:00 Uhr, MALZ, Donaustraße 1
47443 Moers

Mannheim
Thomas Prinz
Tel.: 06271-2830 (p)
Ewald Rieger
Tel.: 06239-8632 (p)
Treffen: jeden 1. Mittwoch im Monat
Vereinslokal Segelverein Mannheim e.V.
Flugplatz Mannheim-Neuostheim

München
Jens Wilke
Tel.: 089-89 76 890
Treffen: jeden 4. Mittwoch im
Monat, China Restaurant XIANG
Morungerstraße 8
München-Parsing

µP-Controller Verleih

Thomas Prinz
Tel.: 06271-2830 (p)

Gruppengründungen, Kontakte

Fachbezogen 8051 ... (Forth statt Basic, e-Forth)
Thomas Prinz
Tel.: 06271-2830 (p)

Forth-Hilfe für Ratsuchende

Forth allgemein Jörg Plewe
Tel.: 0208-49 70 68 (p)

Jörg Staben
Tel.: 02103-24 06 09 (p)

Karl Schroer
Tel.: 02845-2 89 51 (p)

Spezielle Fachgebiete

Arbeitsgruppe MARC4	Rafael Deliano Tel./Fax: 089-841 83 17 (p)
FORTHchips (FRP 1600, RTX, Novix)	Klaus Schleisiek-Kern Tel.: 040-375 008 03 (g)
F-PC & TCOM, Asyst (Meßtechnik), embedded Controller (H8/5xx// TDS2020,TDS8092 Fuzzy	Arndt Klingelnberg, Consultants akg@aachen.forth-ev.de Tel.: ++32 +87 - 63 09 89 Fax: ++32 +87 - 63 09 88
KI, Object Oriented Forth, Sicherheitskritische Systeme	Ulrich Hoffmann Tel.: 04351 -712 217 (p) Fax: -712 216
Forth-Vertrieb	volksFORTH / ultraFORTH RTX / FG / Super8 / KK-FORTH Ingenieurbüro Klaus Kohl Tel.: 08233-3 05 24 (p) Fax : 08233-99 71
Forth-Mailbox (KBBS)	0431-533 98 98 (8 N 1) Sysop Holger Petersen hp@kbbs.org Tel.: 0431-533 98 96 (p) bis 22:00 Fax : 0431-533 98 97 Helsinkistraße 52 24109 Kiel

Möchten Sie gerne in Ihrer Umgebung eine lokale Forthgruppe gründen, oder einfach nur regelmäßige Treffen initiieren ? Oder können Sie sich vorstellen, ratsuchenden Forthern zu Forth (oder anderen Themen) Hilfestellung zu leisten ? Möchten Sie gerne Kontakte knüpfen, die über die VD und das jährliche Mitgliedertreffen hinausgehen ?

Schreiben Sie einfach der VD - oder rufen Sie an -

Hinweise zu den Angaben nach den Telefonnummern:

Q = Anrufbeantworter
p = privat, außerhalb typischer Arbeistzeiten
g = geschäftlich

Die Adressen des Büros der Forthgesellschaft und der VD finden Sie im Impressum des Heftes.

Hätten Sie mal wieder Lust auf einen Trip in die Berge ?

Der Termin für die FORTH-Tagung 1999 steht fest !

Vom 23. bis zum 25. April 1999

ist die Chance recht hoch, dabei auch noch eine Portion Schnee zu erwischen !

Anmeldeformulare und weitere Informationen werden Sie in der VD 1/1999 finden.

Ulrike und Heinz Schnitter freuen sich auf Ihr Kommen !