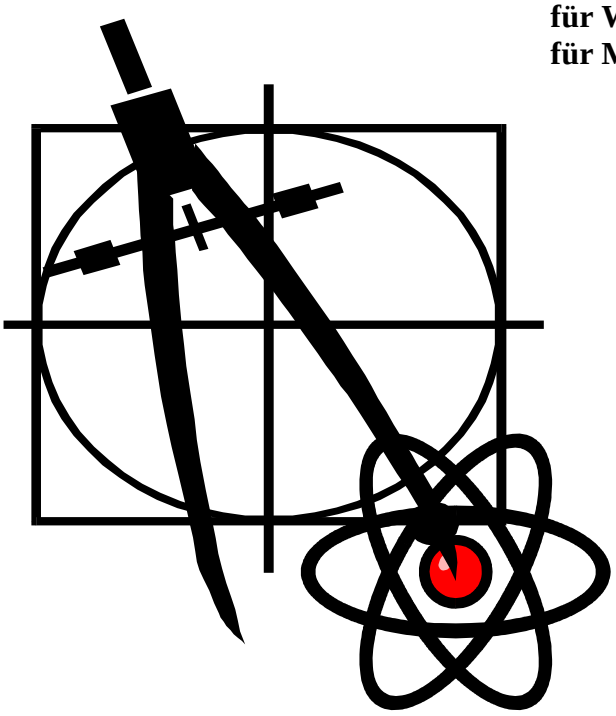
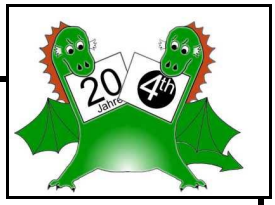
VIERTE DIMENSION

3 / 2004

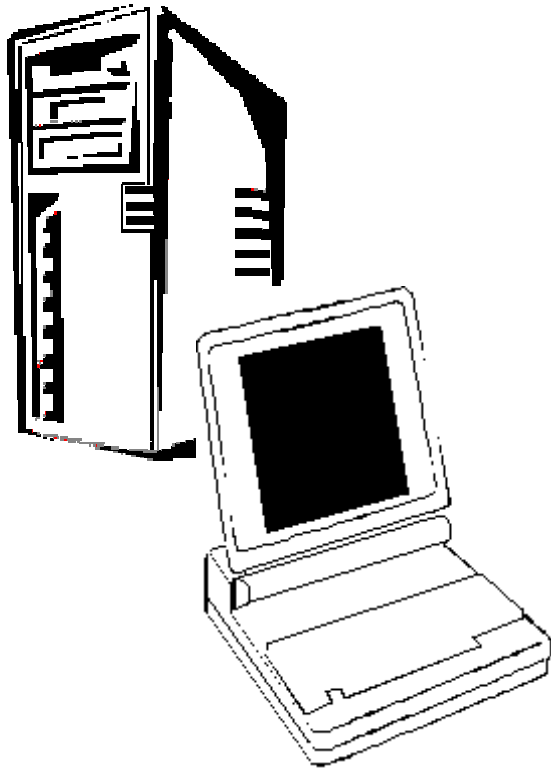
Das FORTH-Magazin

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





In dieser Ausgabe:



20. Jahrgang - **Ausg. 3/2004** - 4,00 €

Organ der FORTH-Gesellschaft e.V.

Dienstleistungen und Produkte fördernder Mitglieder des Vereins

tematik GmbH
Technische Informatik

Feldstrasse 143
D-22880 Wedel
Fon 04103 – 808989 – 0
Fax 04103 – 808989 – 9
mail@tematik.de
www.tematik.de

Gegründet 1985 als Partnerinstitut der FH-Wedel beschäftigten wir uns in den letzten Jahren vorwiegend mit Industrieelektronik und Präzisionsmeßtechnik und bauen z.Z. eine eigene Produktpalette auf.

Know-How-Schwerpunkte liegen in den Bereichen Industriewaagen SWA & SWW, Differential-Dosierwaagen, DMS-Messverstärker, 68000 und 68HC11 Prozessoren, Sigma-Delta A/D. Wir programmieren in Pascal, C und Forth auf SwiftX86k und seit kurzem mit Holon11 und MPE IRTC für Amtel AVR.

LEGO RCX-Verleih

Seit unserem Gewinn (VD 1/2001 S.30) verfügt unsere Schule über so ausreichend viele RCX-Komponenten, dass ich meine privat eingebrachten Dinge nun anderen, vorzugsweise Mitgliedern der Forthgesellschaft e.V., zur Verfügung stellen kann.

Angeboten wird: Ein komplettes LEGO-RCX-Set, so wie es für ca. 230,- € im Handel zu erwerben ist.

Inhalt:

1 RCX, 1 Sendeturm, 2 Motoren, 4 Sensoren und ca. 1.000 LEGO- Steine.

Anfragen bitte an

Martin.Bitter@t-online.de

Letztlich enthält das Ganze auch nicht mehr als einen Mikrocontroller der Familie H8/300 von Hitachi, ein paar Treiber und etwas Peripherie. Zudem: dieses Teil ist ‘narrensicher‘!

Hier könnte IHRE Anzeige stehen!

Wenn Sie ein Förderer der Forthgesellschaft sind oder werden möchten, sprechen Sie mit dem Forth-Büro über die Konditionen einer festen Anzeige.

Secratary@forth-ev.de

Forth Engineering
Dr. Wolf Wejgaard

Tel.: +41 41 377 3774 - Fax: +41 41 377 4774
Neuhöflirain 10
CH-6045 Meggen http://holonforth.com

Wir konzentrieren uns auf Forschung und Weiterentwicklung des Forth-Prinzips und offerieren HolonForth, ein interaktives Forth Cross-Entwicklungssystem mit ungewöhnlichen Eigenschaften. HolonForth ist erhältlich für 80x86, 68HC11 und 68300 Zielprozessoren.

KIMA Echtzeitsysteme GmbH

Tel.: 02461/690-380
Fax: 02461/690-387 oder -100
Karl-Heinz-Beckurtz-Str. 13
52428 Jülich

Automatisierungstechnik: Fortgeschrittene Steuerungen für die Verfahrenstechnik, Schaltanlagenbau, Projektierung, Sensorik, Maschinenüberwachungen. Echtzeitrechnersysteme: für Werkzeug- und Sondermaschinen, Fuzzy Logic.

FORTecH Software
Entwicklungsbüro Dr.-Ing. Egmont Woitzel

Budapester Straße 80 a D-18057 Rostock
Tel.: (0381) 46 13 99 10 Fax: (0381) 4 58 34 88

PC-basierte Forth-Entwicklungswerkzeuge, comFORTH für Windows und eingebettete und verteilte Systeme. Softwareentwicklung für Windows und Mikrocontroller mit Forth, C/C++, Delphi und Basic. Entwicklung von Gerätetreibern und Kommunikationssoftware für Windows 3.1, Windows95 und WindowsNT. Beratung zu Software-/Systementwurf. Mehr als 15 Jahre Erfahrung.

Ingenieurbüro
Dipl.-Ing. Wolfgang Allinger

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

Entwicklung von µC, HW+SW, Embedded Controller, Echtzeitsysteme 1-60 Computer, Forth+Assembler PC / 8031 / 80C166 / RTX 2000 / Z80 ... für extreme Einsatzbedingungen in Walzwerken, KKW, Medizin, Verkehr / >20 Jahre Erfahrung.

Ingenieurbüro
Klaus Kohl

Tel.: 08233-30 524 Fax: - 9971
Postfach 1173
D-86404 Mering

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

Inhalt

Diese Ausgabe der VD wird vermutlich ca. vier bis sechs Wochen nach dem Erscheinen der Druckausgabe im Internet auf der Web-Seite der Forthgesellschaft e.V. veröffentlicht.

http://www.forth-ev.de

Eine PDF-Version dieser Ausgabe wird ab dem Zeitpunkt der Veröffentlichung im Internet ebenfalls zur Verfügung stehen. Bitte wenden Sie sich hierzu über die oben angegebene Adresse an den Webmaster der Forthgesellschaft e.V. oder an die Redaktion der „Vierte Dimension“. *fep*

In der nächsten Ausgabe finden Sie voraussichtlich:

- das Protokoll der Mitgliederversammlung 2004
- die angekündigten Beiträge aus der „Embedded“
- Tagungsbeiträge von Fehmarn
- was immer SIE uns schicken!

Editorial

IMPRESSUM

Name der Zeitschrift

Vierte Dimension

Herausgeberin

Forth-Gesellschaft e.V.
Postfach 19 02 25
80602 München
Tel.: (0 89) 1 23 47 84
E-Mail:
SECRETARY@FORTH-EV.DE
DIREKTORIUM@FORTH-EV.DE

Bankverbindung: Postbank Hamburg
BLZ 200 100 20
Kto 563 211 208
IBAN: DE60 2001 0020 0563 2112 08
BIC: PBNKDEFF

Redaktion & Layout

Friederich Prinz
Hombergerstraße 335
47443 Moers
Tel.: (0 28 41) 5 83 98
E-Mail: ***VD@FORTH-EV.DE***

Anzeigenverwaltung

Büro der Herausgeberin

Redaktionsschluß

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

Erscheinungsweise

1 Ausgabe / Quartal

Einzelpreis

4,00 € + Porto u. Verpackung

Liebe Leser,

die nächste VD zu „machen“ ist jedes Mal eine eigene Herausforderung. Man weiß bis zum letzten Tag der Montage nie, was an Beiträgen zur Verfügung steht.

Einige, wenige Autoren sind immer fleißig. Unser Mann „in den Staaten“, Henry Vinerts, ist mit seinen Leistungen seit langem fest eingeplant. Fred Behringer findet neben den vielen anderen Aktivitäten für unseren Verein auch immer wieder noch Zeit für die Rezensionen unserer „ Schwesternschriften“ und für eigene Arbeiten. Und sogar Menschen, die gar nicht Mitglied der Forthgesellschaft sind, wie Rafael Deliano, geben Ausgabe für Ausgabe Arbeiten zur Veröffentlichung heraus. Andere Autoren schreiben gelegentlich für die VD; gelegentlich und nicht kalkulierbar. Das war in früheren Jahren Material für den „Pool“, in dem sich immer einige Arbeiten befanden, die sich sammeln ließen, und die zur Verfügung standen, wenn die nächste Ausgabe der VD gar zu dünn zu geraten schien. Der Pool ist aber längst leer. Eine Zeit lang war es geübte Praxis, daß die Vortragenden der Jahrestagungen ihre Referate der VD zur Veröffentlichung zur Verfügung gestellt haben.

Im Zeitalter des Beamers ist das aber leider nicht üblich. So fix die Inhalte eines Vortrages in ein beliebiges, auf dem Bildschirm darstellbares Format gebracht werden können, so schwierig und aufwändig scheint es zu sein, diese Inhalte zusätzlich in ein übliches Text– und Bildformat zu überführen und dem Editor der „Vierte Dimension“ zukommen zu lassen. Jedenfalls fließt der VD aus dieser Quelle zur Zeit nichts mehr zu.

Ist das der Preis der Moderne, die über die Bedürfnisse althergebrachter Druckmedien einfach hinweggeht? Wenn das so ist, wird es sehr bald kaum noch möglich sein, den alten und neuen Mitgliedern der Forthgesellschaft viermal im Jahr eine Zeitschrift mit angemessenem Inhalt anzubieten.

Und neue Mitglieder dürfen wir auch in dieser Ausgabe begrüßen. Carsten Strotmann konnte auf dem LinuxTag in Karlsruhe zwei Menschen aus dem Umfeld von OpenBIOS, die auf unserem Stand offenbar eifrig mitgeholfen haben, für Forth und die Forthgesellschaft interessieren. **Patrick Mauritz** und **Stefan Reinauer** (der auch Ansprechpartner auf der Seite http://www.openbios.org ist) sind seit dem 1. Juli 2004 Mitglieder der FG. Herzlich Willkommen!

Ihr

Friederich Prinz

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*

Die Forthgesellschaft wird durch ihr Direktorium vertreten:

Prof. Dr.	Fred Behringer	Kontakte: Direktorium@Forth-ev.de
Dr.	Ulrich Hoffmann	
Dipl.Inf.	Bernd Paysan	

Betreff: Achievement Award
Von: Fred Behringer <behringe@ma.tum.de>

Glückwunsch nach Amerika

Aus dem Forthwrite-Heft 125 erfahren wir, dass Henry Vinerts, "unser Mann aus Amerika", den diesjährigen "2003-Achievement-Award" gewonnen hat. Damit zeigen auch die englischen Forth-Freunde, dass sie Henrys "entertaining and enthusiastic epistles" aus der SVFIG (die in der Forthwrite im Original erscheinen) sehr schätzen. Der Preis ist mit der Nennung in der Web-Ruhmeshalle der FIG UK und mit einer freien Mitgliedschaft für ein Jahr verbunden.
Henry, die Forth-Gesellschaft gratuliert dir!

Fred Behringer

Betreff: Robots
Datum: Tue, 4 May 2004 09:16:48 +0100
Von: Graeme Dunbar <g.r.a.dunbar@rgu.ac.uk>

Hallo,
die Ingenieursschule <an der Graeme Dunbar unterrichtet, der Redakteur> hatte in der vergangenen Woche Besuch von Akademikern der Fachhochschule Frankfurt am Main (Ich hoffe, die Bezeichnung der Schule korrekt wiederzugeben). Unsere Besucher haben einige meiner Kollegen "beschattet", um unsere Unterrichts- und Forschungspraxis zu beobachten.

Am letzten Tag dieses Besuchs habe ich mit Prof.Dr.Ing. Michael Hefter gesprochen, der mir vortrug, wie sehr die Besucher an den Arbeiten unserer Robotikgruppe interessiert sind. Nun, Prof.Dr. Fred Behringer ist einer meiner Freunde und Kollegen, und mein direkter Kontakt zur Forthgemeinde in Deutschland.

Ich weiß, daß ein recht großes Interesse an Robotikprojekten in der europäischen, akademischen Gemeinschaft und in den Forthgemeinschaften seinen Ausdruck gefunden hat. (Ich kenne Forth-Enthusiasten auch in Frankreich mit eigenen Interessen an der Robotik, und wir haben Austauschstudenten aus Deutschland und Frankreich).

Ich frage mich, ob das nicht eine Erwägung bezüglich irgendeiner Art der Kooperation wert ist?

Akademisch muß das Forthelement unglücklicherweise überall zurückstehen. Wenn es darum nicht möglich sein sollte, physische Verbindungen herzustellen, hoffe ich, daß wir trotzdem einen Weg finden werden, unsere Gedanken und Erfahrungen auszutauschen.
Mit freundlichen Grüßen

Graeme Dunbar

[*übersetzt von fep*]

Betreff: Franzosen FORTH + FPGA
Von: Thomas Prinz <topeban@web.de>

Habe ich über eine Französische Seite gefunden :

free.forth.fr

(Leider verstehe ich eher kein Französisch)

Emulatoren und FPGA-Forth !

http://fbroton.free.fr/MiniForth.html#ancre95451

(anscheinend F21 FPGA von Ultratechnology, der dann einen 68000+68881 ersetzt)

Ingenieurschule Fribourg (Ein Herr Schmidt o.ä. war schon mal bei der Tagung in Ludwigshafen zu Besuch)

http://jpb.forth.free.fr/Francais/coeur_forth/coeur_forth.html

PDF-Dateien Schaltplan etc.

http://jpb.forth.free.fr/Francais/encadrement.html

und Tschüß, *Thomas*

Herr Schmidt ist, wenn ich das richtig erinnere, Schweizerbürger und war Mitglied der Forthgesellschaft. Leider hat die FG den Kontakt zu ihm verloren. Vielleicht bietet dies eine Möglichkeit, ihn wieder für uns zu gewinnen.

fep

Betreff: Stefan Schmiedl
Von: Rolf Schöne <rolf@rolf-schoene.de>

Auf die Frage an Rolf Schöne, ob er etwas über das neue Mitglied Stefan Schmiedl zu berichten wüßte, hat Rolf per Mail Kontakt zu Hr. Schmiedl aufgenommen und dabei folgendes erfahren:

[Mail S.Schmiedl an R.Schöne] ...

Ich habe eine Weile gebraucht, bis ich herausbekommen habe, wieso Sie denken, dass ich Mathematiker bin ... bin ich nämlich nicht. Ich bin der einzige nicht-promovierte im Kern-Team der Firma :-), meines Zeichens ausgebildeter Lehrer (M/Ph an Gymnasien) und was die EDV angeht, Autodidakt mit Interesse an allem außer meiner anstehenden Arbeit.
...
Kleinanzeigen in der c't hab' ich noch nie gelesen, große eigentlich auch nicht ... Soweit ich mich erinnern kann, habe ich den Verein im Ergebnis einer Google-Suche "entdeckt".
...
Mit Forth hatte ich das erste Mal vor ca 15 Jahren zu tun, damals auf meinem Mac/SE. So richtig ist aber nichts daraus geworden. Vor ein paar Wochen musste ich herausfinden, warum eine Applikation unter Windows nicht so wollte, wie wir wollten, und weil ich es bis dato geschafft hatte, Windows-Programmierung zu umgehen, war das nicht so einfach.

Ich habe mir dann SwiftForth gekauft, weil es einen ordentlichen Eindruck gemacht hat, einerseits maschinennah genug ist,

um überall hin zu kommen, andererseits aber ausreichend flexibel ist, um von da schnell wieder weg zu kommen :-)

Das Beste an der Sache ist, dass ich auf der Website von Forth Inc. dann auch noch über das Archiv der Mailingliste gestolpert bin, und in der Liste auch einen Programmierer wiedergefunden habe, mit dem ich an einem anderen Open-Source-Projekt schon gearbeitet habe. Beim folgenden E-Mail-Austausch hat sich herausgestellt, dass er Windows-Programmierung noch weniger mag als ich, aber für seine Firma machen muss, und wenn er könnte, würde er ja alles in Forth machen ...

Also machen wir jetzt gerade ein Testprojekt, er baut einen IR-Signal-Analysator und ich schreibe ein GUI dazu. Macht Spaß. Und ist ein hinreichend verrückt: Ein Amerikaner, der eigentlich keinen Forth-Programmierer sucht, lässt bei einem Deutschen, der eigentlich keinen Forth-Auftrag sucht, etwas in Forth entwickeln.

Außerdem sind wir gerade mit Vorbereitungen für ein riesiges Projekt beschäftigt, für das wir auch diverse Hardware (mit entsprechender Software) brauchen werden, wenn es zum Tragen kommt. Und auch hier waren Publikationen aus der Forth-Ecke schon überraschend hilfreich.

S. Schmiedl

Etwas mehr über unser neues Mitglied steht unter http://www.approximity.com zu lesen.

fep

Liebe Freunde,

Es drückt mich ein wenig, daß ich zwei Monate habe verstreichen lassen, ohne Euch ein Lebenszeichen aus dem Silicon Valley zu geben. Und jetzt habe ich tatsächlich die Abgabetermine sowohl für die Vierte Dimension verpasst, als auch für die Forthwrite. Das tut mir leid, aber ich hoffe, es reicht Euch als Entschuldigung, wenn ich Euch sage, dass unser übliches Treffen am vierten Samstag des Monats zumindest teilweise mitschuldig an meinem Versehen ist.

Für mich war Bob Smiths Vortrag das Glanzstück unseres Februar-Treffens. Sein Vortrag hat sich mit musikalischen Skalen und verschiedenen Systemen zur Feinabstimmung (Temperament) von Tasteninstrumenten auseinander gesetzt. Ein von ihm mit Win32For geschriebenes Programm berechnet die Abweichungen von vorgegebenen Temperamenten in Zehnteln. (Die chromatische Skala einer Zwölf-Ton-Oktave entspricht 1200 Zehnteln, ein Zehntel entspricht dem musikalischen Intervall von einem Hundertstel eines Halbtons). Mit Soundfiles, von denen er Stücke in verschiedenen Temperamenten abspielen ließ, demonstrierte Bob uns die Effekte der Feinabstimmung auf unterschiedliche Schemata. Wie ich bereits früher erwähnt habe, ist Bob nicht nur einer der aktivsten „Weiterentwickler“ für F-PC und Win32For, sondern auch ein vollendeter Akkordeonspieler, ein Mann mit großem Wissen, das er bereitwillig mit anderen teilt.

(In der Musiktheorie Bewanderte werden es mir hoffentlich nachsehen, wenn meine Übersetzung fehlerhaft ist. Mein Spaß an Musik ist mehrere Größenordnungen größer als mein Verständnis derselben. fep)

Dr. Ting gab uns eine ausführliche Erläuterung zu einem raffinierten Algorithmus, den er zum Zeichnen von Ellipsen entwickelt hat, natürlich in Forth. Und David Frech kam wieder, um sein muForth noch ausführlicher zu erklären. Einmal mehr ausgezeichnete Unterrichtseinheiten, allerdings wieder grundsätzlich oberhalb der Verständnisebene des ältesten Forthnovizen der Welt. Ich muß Euch aber sagen, dass ich zu diesem Zeitpunkt alles Unbehagen ablegen und mich als Forth- und Computerkenner fühlen konnte, dem Rest der Gruppe gleich. Ich muß mich nur bei jenen freundlichen Individuen entschuldigen, die versucht haben, sich in einer mir unbekannten Sprache mit mir zu unterhalten. Nach seinem Vortrag ahnte David, dieser hilfsbereite Neuling nicht, das seine tiefschürfenden Erklärungen der “tail-recursion” die falschen Ohren erreicht hatte. Ich hoffe, daß er wiederkommt, damit ich ihm noch einmal danken kann.

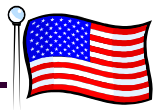
OK, das bringt mich auf den letzten Samstag, zu unserem März-Treffen.

Zuallererst muß ich von meiner Überraschung berichten, die ich empfand, als ich gleich vier neue Gesichter unter den zwölf Leuten ausmachte, die bereits um 10 Uhr am Vormittag dort waren. Wir hatten letztens auch neue Besucher und einen bemerkenswerten Schwund der regulären Teilnehmer, aber das Verhältnis zwischen neuen und alten Besuchern war dieses Mal ungewöhnlich.

Dr. Haydon bat mich, in meinem Brief darauf hinzuweisen, daß alles verbliebene Material der FIG in insgesamt 78 Kartons bei Mountain View Press zum Erwerb zur Verfügung steht. Eine Inventarliste ist in Vorbereitung und sollte demnächst im Web verfügbar sein. (In der Zwischenzeit könnt ihr beliebige Anforderungen direkt an Glenn schicken, sofern ihr ihn bei TheForthSource.com findet.)

Dr. Ting begann den Tag mit der Beschreibung eines Single-Step-Debuggers für den eP32 Forth Simulator. Der Simulator soll Hardwarecode auf einem Chip debuggen, der in Taiwan entworfen wurde. Der Debugger arbeitet anstelle des Maschinencodes Forth-Worte in Einzelschritten ab. „In Software ist Forth sein eigener Debugger“, sagt Ting.

Einige Leute schauten herein und nahmen an verschiedenen unplanmäßigen Diskussionen und Web-Ausflügen teil, aber nur ein gutes Dutzend kam nach einer langen Mittagspause zurück, um John Hall, dem früheren FIG Präsidenten zuzuhören. John hat viele Jahre für Apple gearbeitet und war gekommen,



um uns zu berichten, dass es noch eine Menge für Forth zu tun gibt. Zu tun gibt es nach wie vor in den ewig wachsenden Open-Firmware-Applikationen in allen Macintosh-Maschinen. Tatsächlich hat Apple sogar seinen Programmierern einen Forth-Kurs spendiert. Meinen Lesern möchte ich sagen, dass bei Apple hauptsächlich GForth verwendet wird. Ich möchte dabei gleich die Gelegenheit nutzen und nachfragen, welcher andere Forther Johns Erfahrung teilen kann und seit 1980 ausschließlich für Unternehmen arbeiten durfte, die es ihm erlaubten, mit Forth zu arbeiten. John sagt, dass er gegangen wäre, wenn man ihm das nicht erlaubt hätte, und dass er sich ein neues Unternehmen gesucht hätte, dass ihm die Arbeit mit Forth erlaubt hätte.

Ich habe hier weder genügend Platz, noch verbleibt mir ausreichend Zeit, um all die Fähigkeiten des letzten Redners, Sellam Ismail, aufzuzählen. Dieser energiegeladene junge Mann ist der Inhaber von Vintage Tech und der Organisator des Vintage Computer Festivals, das seit 1997 regelmäßig in diesem Lande stattfindet. Ich meine, dass die europäische Version dieses Festivals im Mai diesen Jahres in München zu erleben sein wird, und ich vermute, dass Sellams Freund, sein europäisches Gegenstück, Hans Franke von der „Gesellschaft für historische Rechenanlagen“ ist, den ich bei zwei unterschiedlichen Gelegenheiten im Silicon Valley treffen konnte.
Sellam begann seine Computer-Sammlung mit einem Mattel Aquarius, 1983, 4K Modell, mit BASIC. Bis heute hat er rund 2.000 Computer gesammelt, von denen 900 Einzelstücke sind. Dazu kommen 7.000 Computerbücher und 20.000 Ausgaben von Computermagazinen. Alles zusammen hat die Erweiterung seiner Lagerkapazitäten notwendig gemacht, von einem Schlafzimmer bis zu einem Warenhaus in Livermore in Kalifornien. Sellam unterhält Interessen und Verbindungen zum Computer-Recycling und Aufbereitung, zur Einrichtung von Computerklassen für Kinder und auch (widerstrebend) zum Verleih von alten Maschinen an die Filmindustrie. Es scheint, als wäre es Dave Jaffes Spende einer größeren Anzahl alten Computermaterials an Sellams Unternehmung, die diesen letztlich zu uns geführt hat.

Unabhängig davon, ob Ihr dies hier noch druckt, oder auch nicht, wünsche ich Euch allen ein frohes Osterfest, und meinen deutschen Freunden ein erfolgreiches und angenehmes Treffen auf der Insel Fehmarn.

Henry

(Übersetzung: fep)

Hallo Friederich, Fred, and Graeme!

Ich habe wenig über die letzten drei Monate zu berichten. Anstelle eines regulären Treffens im April reisten sieben von uns zu Bruce Damers früherer Eichen-Farm in den Santa Cruz Bergen und genossen die vom Gastgeber persönlich geleitete Tour durch seine "Digibarn" (http://www.digibarn.com)Sammlung alter Computer. Ich kann einfach nicht schnell genug schreiben, um all die faszinierenden Eindrücke des Busches aufzuzählen. Ihr könnt aber eine Vorstellung davon auf der Webseite bekommen, die nach Bruces Worten einige Millionen Besucher pro Jahr zählt.
Das Mai-Treffen traf mit einigen Besichtigungstouren zusammen, die meine Frau und ich mit unseren Besuchern aus den Niederlanden unternahmen. Im Juni schliesslich waren, anstelle des SVFIG-Treffens, diejenigen von uns, die Mitglied des Computer History Museum sind (ich gehöre nicht dazu) eingeladen, an einem Ereignis unter dem Titel "Damals und Heute: Computer-Graphik in Spielen" teilzunehmen, zu dem Jordan Mechner, Rand Miller und Will Wright zusammen mit Vince Broady anwesend waren. (Mehr dazu unter http://www.computerhistory.org/nvidia_06102004/)
Das sind die "Lebenszeichen" der Forther des Silicon Valley. Ich werde wahrscheinlich bis Ende Juli, wenn die regulären Treffen wieder beginnen, nichts zu berichten haben. Ich habe mir gedacht, daß, wenn wir nicht genug Zeit oder Material finden, um unsere Leser zu unterhalten, es vielleicht Zeit ist, in den Archiven zu graben und einige Perlen der Weisheit aus den vielen Muscheln herauszusammeln, die die Forth-Gemeinschaft produziert hat. Jemand hat einmal gesagt, dass das beste Kompliment für einen Autor ein Zitat aus seinen Werken ist. Was haltet Ihr davon, wenn wir unsere Leser auf die Suche nach Forth-Zitaten schicken und diese in den Zeitschriften unter einer ." (dot-quote) Rubrik abdrucken? Sobald wir genug davon gesammelt haben, können wir diese als Einführung in ein Buch beutzen, von dem ich hoffe, das es einen Titel wie etwa "Understanding Forth" (Forth verstehen) trägt und nicht "Ending Forth" (Das Ende von Forth). Kann mir jemand sagen, wie lange es her ist, dass einer der großen Buchläden ein Forth-Buch in seinen Regalen hatte?
Lasst mich mit einem der bekanntesten Zitate von Chuck Moore beginnen (aus dem Vorwort von "Starting Forth"): "Simplicity provides confidence, reliability, compactness, and speed." Wolfgang Allinger hat dies schon einmal ins Deutsche übersetzt: "Einfachheit erzeugt Vertrauen, Zuverlaessigkeit, Ueberschaubarkeit und Schnelligkeit."
Für alle die, die etwas derartiges Interessantes gelesen haben, aber das Buch nicht mehr besitzen, kann ich zumindest die angefügte Liste der Bücher, die ich besitze beifügen. Wer weiss, da mag eine gute Anzahl verlorener Schätze unter der Mühsal all der Autoren vergraben sein?
Euch allen einen guten Sommer wünschend,

Henry

(Übersetzer: Thomas Beierlein)

Anhang:
Die Liste mit Henry's Forth Büchern
(\mydocuments\my eBooks\4thbooks.doc, June 2004)

1. "Starting FORTH" by Leo Brodie, 1981, Forth Inc.
2. "Starting Forth" 2nd ed., by Leo Brodie, 1987, Prentice-Hall
3. "Dr. Dobb's Toolbook of Forth" Vol. I, 1987, M&T Books
4. "Dr. Dobb's Toolbook of Forth" Vol.II, 1987, M&T Books
5. "All About Forth" MVP-FORTH Series Vol.1, by Glen B. Haydon, 2nd ed., 1984, Mountain View Press
6. "All About Forth--An Annotated Glossary," by Glen B. Haydon, 3rd ed., 1990, MVP-FORTH SERIES
7. "Real Time Forth" by Tim Hendtlass, 1993, Swinburne

Univ. of Technology, Australia
8. "Thinking Forth" by Leo Brodie, 1984, Prentice-Hall (proofreading copy for 1994 edition)
9. "Using FORTH" by Rather, Brodie, Rosenberg, 2nd ed., 1980, Forth, Inc.
10. "Mastering FORTH" by Tracy & Anderson, Advanced MicroMotion, Inc., 1989, Brady Books
11. "FORTH: A Text and Reference" by Kelly & Spies, 1986, Prentice-Hall
12. "Introduction to FORTH" by Ken Knecht, 1982, Howard W. Sams
13. "The Complete FORTH" by Alan Winfield, 1983, Sigma/Wiley
14. "FORTH-79" ver. 2, Z-80 CP/M edition, by Anderson & Tracy, 1982, = MicroMotion
15. "Discover FORTH" by Thom Hogan, 1982, Osborne/McGraw-Hill
16. "Learning FORTH--A Self-teaching Guide" by Margaret Armstrong, 1985, John Wiley
17. "Threaded Interpretive Languages" by R.G. Loeliger, 1981, Byte Books

Gehaltvolles

zusammengestellt und übertragen
von Fred Behringer

VIJGEBLAADJE der HCC Forth-gebruikersgroep, Niederlande

Nr. 42, Februar 2004

Van de redactie
Albert Nijhof

Der Redakteur des Vijgeblaadjes hatte im Vorstand der Fgg die Frage gestellt, ob Forth sich auch fürs Programmieren großer Systeme eigne. Er bekam fünf Antworten - ob fiktiv oder nicht, ist für den Leser nicht erkennbar.

Is Forth geschikt voor grote programma's? "Antwortensammlung" von Albert Nijhof

(1) Meta-Forth hat 25000 Zeilen. Lennart Benschops 32-Bit-Forth-Emulator hat 10000 Zeilen. Das sind große Systeme. Win32Forth, VFX und SwiftForth sind groß. Das Steuerungssystem auf dem Flugplatz in Riad (siehe diverse Interviews mit Chuck Moore) ist groß (400 Prozessoren, 36000 Sensoren).

(2) Was heißt "groß"? Viel Code, viele Daten, viele Teilsysteme, viele Funktionen, viele beteiligte Programmierer? Byte-Forth kann in vielerlei Hinsicht als groß angesehen werden. Schwierig ist die Zusammenarbeit mehrerer Programmierer an einem "großen" Projekt.

(3) Richtiges (geeignetes) "Faktorisieren" ist in Forth das A und O. "Programmieren" kann man lernen, Faktorisieren ist eine Kunst. Codieren ist das Handwerkzeug, Faktorisieren ist das Management. Auch die Aufgabe des Leiters einer Projektgruppe von Programmierern kann als Faktorisieren bezeichnet werden. Forth ist für große Systeme geeignet. Forth kann alles - nur natürlich nicht zaubern. Mit einer Stradivari in der Hand ist man noch kein Virtuose.

(4) Auch das allergrößte und allerschönste Anwendungssystem taugt nichts, wenn es nicht auf einer soliden Grundlage (Maschinen-Code für I/O-Operationen usw.) aufbaut. Diese solide Grundlage verschafft man sich in Forth im Handumdrehen. Alles andere, mag es noch so komplex sein, geht dann in Forth wie von selbst.

(5) Bei großen Forth-Programmen wird es ein Problem, die vielen Namen auseinander zu halten. Vokabulare sind nicht wirklich hilfreich. (Der Rezensent: Ein "schneller", übereifriger Assembler-Zusatz, in dem das Forth-Wort XOR vorkommt, ohne zu bemerken, dass dieses Wort inzwischen schon seine Assembler-Bedeutung angenommen hat.) DEFER ist auch keine Lösung. Es führt zu unübersichtlichen Strukturen. Vorschlag: "Lokale Worte". Alles, was zwischen { und } steht, ist von außen nicht zugänglich.

Nr. 43, April 2004

Ushi trekt een strep
Willem Ouwerkerk

"Ushi zieht einen Strich" ist gut, Ushi zieht Kreise. Auch bei uns. Die Teilnehmer an unserer Forth-Tagung auf Fehmarn waren von Willems munterem Experiment beeindruckt. Der hier zu besprechende Artikel bildet die Grundlage. Der Roboter Ushi konnte sogar (Anlehnung an Martin?) zeichnen. Sensoren tasten reflektierende Streifenbelege auf den Radscheiben ab und führen die Impulse einer Variablen zu, die die Zähldifferenz misst. Negativ heißt, links muss "ein Zahn zugelegt" werden, positiv heißt, dasselbe rechts. Zum Zeichnen (oder Schreiben von "Forth") muss auch noch die Länge des zurückgelegten Weges in die Rechnung einfließen. Zwei Timer-Interrupts und ein Multitasker erledigen das.

Forth vanaf de grond
Ron Minke

Erfrischend, auch mal andere Autoren zu sehen. Ron beginnt eine Serie von Artikeln, die Forth "von der Pike auf" lehren sollen. Er geht von FIG-Forth aus und fasst den Aufbau einer vir-

tuellen Forth-Maschine über dem AT90S8515 von Atmel ins Auge: Zwei Stacks, Wörterbuch, fünf Pseudoregister, Eingabepuffer, (zunächst) drei Maschinenbefehle (MOV DEC INC), vier Wortfelder, EXECUTE und NEXT.

Nr. 44, Juni 2004

Forth vanaf de grond 2
Ron Minke

Zweiter Teil der Lehrstücks "Forth von Grund auf". Am Ende des Artikels steht "wird fortgesetzt". Ich (der Rezensent) gebe die Vorrede des Autors wieder:

"In diesem Teil versuchen wir, die Register des AVR-Prozessors auf die Register der virtuellen Forth-Maschine abzubilden. Zuallererst müssen wir uns vergegenwärtigen, dass wir dabei sind, ein 16-Bit-Forth auf einen 8-Bit-Prozessor zu zwängen. Wir müssen also alle Forth-Register an Register-PAARE des AVR koppeln. Wir werden sehen, dass das Gott sei Dank möglich ist."

Tijdnood?
Ernst Kouwe

"Ob ich noch etwas Interessantes für das Vijgeblaadje hätte, ein oder zwei Seiten, bis morgen fertig und eingereicht. Tja, so'ne Frage kann man erwarten, wenn man genau einen Tag vor Redaktionsschluss seinen Forth-Guru um Rat über LCD-Displays bittet."

So fängt der Artikel des Autors an. Er berichtet dann über seine Erfahrungen beim Anschluss eines Xiamen Ocular GDM1602A an die AT51-Platine oder an Ushi, den Roboter. Um es kurz zu machen: Zwei Tage lang war er damit beschäftigt: Rumgooglen, Datenblätter vergleichen, Ungereimtheiten entdecken, eigene Überlegungen anstellen, Spannungen messen, ausprobieren und so weiter.

FORTHWRITE der FIG UK, Großbritannien

Nr. 125 Mai 2004

2 Editorial
Graeme Dunbar <g.r.a.dunbar@rgu.ac.uk>

Graeme sucht Freiwillige, die die Kolumnen "Forth News" und "From the Net" übernehmen. "Andernfalls müssen diese eingestellt werden. Dieses Jahr wird die Forthwrite leider nur in fünf Heften erscheinen." Chatten über das IRC-Netz auf Kanal #FIGUK jeden ersten Samstag im Monat.

3 Forth News
Graeme Dunbar

euroFORTH 2004 auf Schloss Dagstuhl - 25 Jahre FIG-UK im November 2004 - 2 Artikel über Forth in den ACM SIGPLAN Notices (übers Internet) - die Forth-CD fast fertig, 6 engl. Pfund, 200 MB Programme, der Rest Videos (von Chuck Moore).

4 Situations Vacant
Graeme Dunbar

Noch einmal detailliert Graemes Ruf nach Freiwilligen für die Kolumnen "Forth News" und "From the Net":

5 Membership Matters
Douglas Neal & Graeme Dunbar

7 neue Mitglieder, einige davon Wiedereinsteiger. Einer von ihnen: "Nachdem ich eine Pause von 20 Jahren gemacht habe, habe ich mir jetzt im Alter von 81 Jahren Forth von MPE bestellt und fange wieder an. Forth ist einfach wonderful."

6 Simple State Machines
Jenny Brien

Erst ein einfaches Beispiel, bei welchem der Zustand einfach ein "Execution Token" ist, das bei Ausführung für jede mögliche Aussprungbedingung einen anderen Zustand auf dem Stack zurücklässt. Schließlich ein komplizierteres Modell, das sich auf Julian Nobles "Endliche Zustandsautomaten" stützt (http://www.jfar.org/article001.html).

10 Regions: into unknown Win32Forth
David R. Pochin

Regions ist nicht die einzige Klasse von Windows-API-Funktionen, an die man "normalerweise" gar nicht herankommt. Dave hat nachgeforscht und herausbekommen, dass COMCTL32.DLL eine große Rolle spielt. Zugriff über WinLib.f --> WinLibrary --> WinLibrary COMCTL32.DLL [ret].

18 Letters
Graeme Dunbar

Brief von Jim Boyd: Mit der modifizierten Definition : FLOAD 1 WORD COUNT INCLUDED ; kann man in Win32For auch Dateien laden, die Zwischenräume im Namen

haben. Brief von Julian Noble: Einige (Fantasie-)Vorschläge zur c.l.f.-Frage, was man in einer Neuauflage von Dr. Dobb's Toolbox of Forth alles aufnehmen würde.

20 Book Review
Paul E. Bennett

Paul bespricht das Buch "The Art of Designing Embedded Systems" von Jack Ganssle: Kein Forth, ein bisschen C, C++ und Assembler, trotzdem auch für den Forthler empfehlenswert.

21 Start Simple
Graham Telfer

"Anfänger brauchen nicht das Gefühl zu haben, draußen in der Kälte stehengelassen zu werden. Graham will sie in weiteren Artikeln mit Antworten versorgen", sagt der Redakteur. Der Autor über sich selbst: "Ich finde es nicht schön, dass Forth nur noch eng mit Embedded Systems in Verbindung gebracht wird. Irgendwann auf dem langen Entwicklungsweg ist die Chance vertan worden, Forth zu einer Universalsprache für Amateure und Fachleute gleichermaßen werden zu lassen."

24 Presenting the FIG UK Awards of 2003

2002 waren es Ed Hersom und Howard Oakford. Für 2003 haben die Preise gewonnen: Henry Vinerts und James Boyd. Über Henry Vinerts freue ich (der Rezensent) mich ganz besonders. Ist doch Henry maßgeblich an der Entwicklung der äußerst guten Beziehungen Amerika-England-Deutschland beteiligt. Sogar am Ausbau unserer sehr guten Beziehungen nach Holland arbeitet Henry fleißig mit. James Boyd hat es mit seinem überragenden Artikel über "Nichtdeterministische Zustandsautomaten“ geschafft. Congratulations!

25 Across the Big Teich
Henry Vinerts <Volvovid@aol.com>

Henrys Berichte vom Februar und März 2004 über SVFIG-Aktivitäten in Originalfassung. Wir kennen sie in der Übersetzung von Thomas Beierlein. Diesmal hat der Redakteur zwei Zeilen der Aufmunterung an Henry hinzugefügt und ihm versichert, dass seine lebendigen Berichte immer gern gelesen werden.

27 Library Notes
Graeme Dunbar

Der Katalog wird neu erarbeitet. Die Ausleihe geht ungestört weiter.

28 Vierte Dimension 4/2003
Joe Anderson <jia@jus.abel.co.uk>

Joe bespricht das Heft 4/2003 unserer Vierten Dimension.

31 Deutsche Forth-Gesellschaft

Unsere Anzeige zur Anwerbung von Mitgliedern für die Forth-Gesellschaft.

32 FIG UK Contacts and Information and Services to Members

Enthält u.a. die E-Mail-Adressen und Postanschriften des "Boards": Vorsitzender, Verwaltung, Redakteur, Schatzmeister, Webmaster, Bibliothek.

Was ist Intelligenz?

Gedanken zu den Fragen,
die in der VD 02/2004
von Ulrich Paul
aufgeworfen wurden.

Friederich Prinz

Eines hat sicher nichts oder vernachlässigbar wenig mit Intelligenz zu tun, nämlich das Wissen über die Bedeutung von Tierspuren, auswendig gelernte Ergebnisse des kleinen Einmaleins, oder das Wissens, daß in Deutschland jeder Schankwirt jedem Durstigen zumindest Trinkwasser kostenfrei geben muß. Sich dargebotenes Wissen anzueignen, ist eine Leistung, die von den meisten Tieren ebenso erbracht wird wie von Homo Sapiens Sapiens. Tiere lernen zum Beispiel, welche anderen Tiere zum eigenen Beutespektrum gehören, wie sie Spuren anderer Spezies kontextbezogen richtig deuten (bin ich das Wild oder der Jäger) und wo die Elterntiere Wasserlöcher, Jagdgründe und sicheren Unterschlupf kennen. Die Bilder des Beutespektrums können sich grundlegend ändern, wenn sich durch Wanderung oder durch rudimentäre Veränderungen in der angestammten Umgebung das Beutespektrum notwendiger Weise verändert. Das Tier ist wie der Mensch bis zu seinem letzten Atemzug lernfähig.

Auch die tatkräftige Bereitschaft das erlernte Wissen anzuwenden, ist nicht notwendiger Weise ein Hinweis auf Intelligenz, sondern schlicht überlebensnotwendig. Der junge Fuchs, der einen hungrigen Wolf trotz der gegenteiligen Bemühungen seiner ihn unterrichtenden Elterntiere zum Spiel auffordert, wird wenig Chancen haben, seine Gene weiterzugeben. Der lernresi-

stente junge Mensch, der alle Bemühungen seiner Lehrer ignoriert und sich schlicht weigert das kleine Einmaleins (und anderes) zu lernen, wird von potentiellen Partnerinnen später ebenfalls wenig bis gar keine Gelegenheit bekommen, seine Gene weiterzugeben.

Intelligenz ist insgesamt so schwer zu fassen, daß man sich ernsthaft fragen muß, warum ein intelligenter Mensch versucht ist, eine künstliche Intelligenz zu schaffen, wo es doch seiner gesamten Spezies bisher nicht gelungen ist, ihre eigene natürliche Intelligenz hinreichend detailliert zu beschreiben.

Seit Immanuel Kant scheint es aber zumindest Hinweise auf das Wesen dessen zu geben, was Homo Sapiens Sapiens stolz glauben lässt, in anderer und vor allem leistungsfähigerer Weise als Tiere über Intelligenz zu verfügen. In der „Kritik der reinen Vernunft“ erklärt Kant uns eine Art Stufenmodell. Zunächst lernen wir, den Tieren gleich, durch Empirik. Unsere fünf Sinne ermöglichen es uns, Dinge zu erleben. Messer sind (meistens) scharf. Herdplatten sind heiß. Das Wasser in der Ostsee ist im April zu kalt zum Baden. Die gottgegebene Fähigkeit empirisch zu lernen, bezeichnet Kant als Verstand. Verstand ist eine Begabung, eine Fähigkeit, die den einzelnen Individuen in unterschiedlichem Maße zur Verfügung steht. Manche Menschen sind lernfähiger als andere Menschen. Nur wenige Hunde lassen sich zu Blindenhunden ausbilden. Eine Katze lässt sich nicht zu einem Blindenhund ausbilden. Das liegt nicht daran, daß eine Katze kein Hund sein kann, sondern vielmehr daran, daß eine Katze zu bestimmten Lernleistungen nicht fähig ist. Nun ließe sich behaupten, daß alle Hunde, die keine Blindenhunde werden können, dumm wie Katzen seien. Das würde aber implizieren, daß Intelligenz allgemein mit Lernfähigkeit gleichzusetzen wäre. Tatsächlich ist die Lernfähigkeit im Bereich der Empirik aber im Wesentlichen von der zur Verfügung stehenden Sensorik abhängig. Einem von Geburt an blindem Fuchs lässt sich nur schwer beibringen, daß der Wolf der Jäger ist, und nicht die Beute. Das macht den Fuchs nicht dümmer; nur kurzlebiger.
Vernunft ist laut Kant (Kundige mögen mir die extremen Vereinfachungen verzeihen) die Fertigkeit und die Bereitschaft alles Erlernte auch wirklich und jederzeit nutzbringend einzusetzen. Das bestätigt uns die Empirik selbst. Jeder kennt unzählige beobachtete Beispiele anderer Menschen und eigenen Verhaltens, in denen gegen vorhandenes Wissen manchmal sogar zu eigenem Schaden gehandelt wurde. Ist das nun Dummheit im Sinne fehlender Intelligenz, oder ist das schlicht die Folge nicht ausreichend geübter (trainierter) Vernunft?

Immerhin ist bezüglich der Vernunft bereits in einem gewissen Rahmen eine Beeinflussung derselben durch das handelnde Individuum möglich. Anders als bei der Fähigkeit des Verstandes, den man in einem gottgegebenen Maße hat oder nicht, lässt sich vernünftiges Handeln üben und in der Qualität der Ergebnisse verbessern. Aus Schaden wird man klug (sofern es eine zweite Chance gibt).

Eine zweite Stufe des Erkenntnisgewinns ist nach Kant die besondere Fähigkeit des menschlichen Verstandes, aus den empirisch ermittelten Fakten auf solche Fakten zu schließen, die sich der zur Verfügung stehenden Sensorik entziehen. Raum und Zeit sind solche Fakten, für die Lebewesen, so wie wir das wissen und verstehen, keine Sensoren haben. Ein Würfel hat Abmessungen, eine ihm eigene Gewichtskraft als Folge seiner Dichte und der Beschleunigung, der er unterliegt, eine zum Zeitpunkt der Beurteilung bestimmte Temperatur, eine oder mehrere Farben und vielleicht einen spezifischen Geschmack und einen eigenen Geruch. Daß der Würfel einen ganz bestimmten, seinen Abmessungen entsprechenden Raum benötigt, -wissen- wir. Wir setzen das a priori als gegeben voraus. Das muß einfach so sein. Und z.B. mit Hilfe der euklidischen Geometrie können wir recht exakt sagen, wie -groß- dieser Raum ist. Die Arbeit mit einer anderen als der euklidischen Geometrie würde uns schon eine andere Auskunft über den vom Würfel beanspruchten Raum geben. Wir -wissen- auch, daß der Würfel, wenn wir ihn von ‚A’ nach ‚B’ versetzen, den gleichen Raum einnimmt. Und wir wissen, daß dies nicht derselbe Raum ist. Aber schon bei einfachen Fragen, ob diese beiden Räume in einer Beziehung zueinander stehen, und wie diese Beziehung beschaffen sein könnte, und ob die Beziehung gleich ist, wenn der Würfel sich in einem dieser Räume befindet oder nicht, versagen wir auf der Suche nach Antworten ziemlich gründlich.
Und Ähnliches gilt für die Zeit. Wir erkennen sie a priori als gegeben an, vermögen aber nichts über ihre Beschaffenheit zu sagen. Wir -erleben-, daß Zeit vergeht, wenn wir den Würfel von ‚A’ nach ‚B’ bewegen. Wir können diese Zeit messen, das heißt mit anderen Zeiten vergleichen. Und das tun wir mit Hilfe einer so kosmologisch relevanten Konstanten wie dem 86.400-tel einer mittleren Umdrehung unseres Planeten um sich selbst. Wirklich beeindruckend scheint mir das nicht zu sein.

Aber es bleibt festzuhalten, daß wir Menschen als einzige Spezies auf unserem Planeten die Fähigkeit zu haben scheinen, aus empirischer Erkenntnis ‚reine’ Erkenntnis (eben nicht empirisch) im Sinne Kants zu gewinnen. Wir können auf der Grundlage von Gelerntem auf Fakten schließen, die wir nicht mehr im ursprünglichsten Sinn des Wortes ‚begreifen’ können. Und die Arbeit mit diesen ‚reinen’ Erkenntnissen hilft der Menschheit in der Entwicklung des menschlichen Kollektivs. Ist das die Intelligenz, nach der die Wissenschaft sucht? Wenn dem so wäre, dann müsste ein Mensch, der dem groben Beispiel mit dem Würfel nicht oder nur schwer zu folgen vermag, als nicht intelligent klassifiziert werden. Dann müssten allerdings alle Menschen als nicht intelligent klassifiziert werden, denen es schwer fällt, sich nicht empirisch prüfbare Erkenntnisse als Wissen anzueignen.

Als Bergmann erlebe ich aber beinahe täglich, wie schwer es den meisten Menschen fällt, sich zum Beispiel ein dreidimensionales Bild von Spannungen in einem gegebenen Gebirgskörper zu machen, obwohl sich diese Spannungen zumindest in ihren Auswirkungen in situ ‚begreifbar’ machen lassen. Selbstverständlich sind nicht alle diese Menschen unintelligent.

Wir wissen nicht erst seit Kant viel über das Lernen, über das Wissen und auch über das Denken. Und nicht nur die Pädagogik lernt stetig mehr über das Lernen und Lehren. Trotzdem wissen wir kaum etwas über die Intelligenz. Wenn aber das Wissen über das Wesen einer ‚Sache' die Voraussetzung für das Schaffen einer funktional gleichwertigen künstlichen Sache ist, dann muß zum jetzigen Zeitpunkt die Suche nach der künstlichen Intelligenz als chancenlos beiseite gelegt werden. Solange niemand die ‚natürliche' Intelligenz eindeutig zu beschreiben vermag, wird es niemandem gelingen, einer künstlichen Intelligenz auch nur nahe zu kommen.

Ulrich Paul fragt in seinen einfachen Überlegungen, was das alles mit Forth zu tun habe. Ich meine, das hat mit Forth genau so viel zu tun, wie ich persönlich das will. Ich bin nicht einmal besonders klug, geschweige denn weise. Aber ich muß niemandem beweisen, daß ich intelligent bin. Ich muß auch niemandem beweisen, daß Forth zur Lösung bestimmter Aufgaben unter bestimmten Voraussetzungen und in den Händen bestimmter Menschen besser als jedes andere Werkzeug geeignet ist. Ich muß, wenn ich das nicht will, nicht einmal fragen, warum Forth unter bestimmten Voraussetzungen ein besseres Werkzeug sein kann, als andere, ähnliche Werkzeuge. Wenn ich Forth und seine besonderen Möglichkeiten nur nutze, ohne sie zu hinterfragen, macht das weder Forth noch seine besonderen Möglichkeiten schlechter oder ‚unfähiger'. Und ebenso wird Forth selbst in seinen Möglichkeiten nicht davon beeinflusst, ob Gegner von Forth dessen Möglichkeiten nicht erkennen oder sogar (dummerweise) bewusst ablehnen.

Das was Forth-Programmierer an den besonderen Konzepten von Forth schätzen, leisten andere Programmierumgebungen nicht oder nur in unterschiedlich ausgeprägten Ansätzen. Ich persönlich bin von der Leistungsfähigkeit der besonderen Konzepte der unterschiedlichen Forth-Systeme überzeugt. Ich sehe aber selbstverständlich auch, daß andere Programmierumgebungen mit ihren jeweiligen Konzepten bestimmte Aufgaben sehr viel leistungsfähiger lösen können als alle mir bekannten Forth-Systeme. Daß ich Forth diesen anderen Programmierumgebungen vorziehe, hängt vor allem damit zusammen, daß ich mich nur selten mit den Aufgaben beschäftige, in deren Lösungen andere Programmierumgebungen ‚besser' sind als Forth. Als ich aber vor wenigen Tagen bei der Erprobung des gerade zuhause eingerichteten WLAN in aller Eile ein Werkzeug zur Durchführung verschiedener Tests programmieren wollte, habe ich nicht Forth als Werkzeug gewählt, sondern Visual Basic. Und als ich vor mehr als 10 Jahren für eine Zeit beruflich mit Datenbanken im dBase-Format zu schaffen hatte, war der damals sehr leistungsfähige und spezialisierte Clipper-Compiler das Werkzeug meiner Wahl. Nur das, was sich mit dem Clipper-Compiler nicht oder nur mit großem Aufwand erledigen ließ, habe ich mit meinen Forth-Werkzeugen dazugegeben.

Vielleicht kommt das der gesuchten Intelligenz am nächsten. Vielleicht ist Intelligenz keine Fähigkeit oder Fertigkeit, sondern ausschließlich eine bestimmte Einstellung. Vielleicht ist Intelligenz lediglich die Bereitschaft, vorurteilslos aus der Menge der gesammelten Erkenntnisse alles zu verwenden, was den Lösungsversuchen zu einer bestimmten Aufgabe am ehesten zum Erfolg verhilft.

Friederich Prinz

Gforth für OS X Panther

Michael Kalus

Apples OS X Panther als „integrierte Entwicklungsumgebung" für Gforth. Oder: Warum man nun das ganze Zeug in Forth selbst nicht mehr braucht!

Mein besonderer Dank geht an Anton Ertl, Bernd Paysan, Jens Wilke und Neal Crook für das Gforth. Und an Ulrich Hoffmann für die klare Anleitung Gforth auf Apples OS X Panther auf einem PowerBook G4 zu ziehen.

Einleitung

Dieser Beitrag ist kurz. So kurz wie es brauchte um Gforth zum Apple zu bringen. Danke für dieses Geschenk an seine Macher!

Hauptteil: „Diesmal kein Drama."

Es kommt schon sehr gut, eine ForthDefinition in der TextEdit Quelle zu markieren, rüber nach gForth ins Terminal zu legen, und sofort prüfen zu können, ob das so geht, oder nicht. Gforth selbst ist da sein komfortabelster Debugger! Als Demonstrationsobjekt füge ich meine kleine Übung mit dem Gforth an. Die Mnemonics für einenH8 CrossAssembler. ***H8ass.txt*** ist das steuernde File. Es nimmt ***H8Mnemonics.txt*** hinzu und lädt einen kleinen ***Test.txt*** für die so definierten Mnemonics. Das Steuerfile meldet, welche Bereiche gerade abgearbeitet werden. Das hilft Fehler einzugrenzen, und der Programmierer bleibt froh. Es schafft den Platz für das Image des crossassemblierten Teiles und liefert einige Definitionen für Tests. Viel Spass beim Ausprobieren.

Gforth wurde geschaffen von Anton Ertl, Bernd Paysan, Jens Wilke, Neal Crook und anderen. Ob ihr Ziel, damit einen Standard zu setzen, inzwischen erreicht ist, kann ich nicht beurteilen. Aber die Apple benutzenden Forthler können den Vieren ein Denkmal setzen - es ist ein Segen, ein solch klares Forth zu haben, dass sich blank an ANS hält. Denn nur weil das so klar eingehalten worden ist, verträgt es sich so gut mit den Möglich-

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 10 € 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: ***w.ouwerkerk@kader.hobby.nl***

Oder überweisen Sie einfach 10 € auf das Konto 525 35 72 der HCC-Forth-gebruikersgroep bei der Postbank Amsterdam. Noch einfacher ist es wahrscheinlich, sich deshalb direkt an unseren Vorsitzenden Willem Ouwerkerk zu wenden.

keiten des OS X Panther. Weil es Forth ist, und nichts anderes. Wunderbar.

Epilog

Herrlich, endlich braucht Forth keine eigenen Editoren mehr, keine eigenen Fenstermechanismen, keine Datei- und Verzeichnisverwaltung, nichts! Panther macht das alles zu einem Kinderspiel, sein Feature namens "Expose" macht es möglich. Wer das einmal kennen gelernt hat, wirft alles andere weg.

Ich bin - angeregt und beflügelt von der Forthtagung in Fehmarn, 20 Jahre Forthgesellschaft - daran gegangen und habe Gforth auf das PowerBook G4 geholt. Seit Apple sein OS X hat, öffnet sich auch für Apple die große weite Welt. Was alle fleißigen Programmierer da global schon zusammen getragen haben, ist, so scheint es, auf einmal auch einfach in dieses neue System zu portieren - und das hat es wahrlich in sich. Wer Panther und Expose noch nicht gesehen hat: Schaut es euch mal an, ein Muss!

Nun, wie ging das, Gforth auf das PowerBook zu holen? Ich habe keine 10 Minuten gebraucht und es lief. Gut, das machen andere Forth-Systeme für den PC inzwischen auch: Downloaden, starten, fertig. Aber was dann auf dem PC und deren Forths folgt, ist zum Abgewöhnen. Bis man endlich verstanden hat, wie da ein File editiert werden kann, ein Projekt gespeichert wird usw., die ganze Handhabung, und bis das dann endlich auch geht - das alles fiel weg mit Gforth und Expose. Denn Gforth ist Forth, und sonst nichts.

Gforth ist schnell geholt. Bei http://fink.sourceforge.net/download/index.php?phpLang=de gibt es viele Tools aus der Unixwelt fertig aufbereitet für den Mac, auch Gforth ist da vertreten.

Unter Panther dann ein Terminalfenster zu öffnen und „ gforth myprojekt.txt" zu tippen, ist für Schriftkundige kein Problem. Im Terminal läuft Gforth mit seinem Zeileninterpreter. Schön ist folgendes: Gforth verlassen mit BYE und im Terminal einmal die Cursortaste-nach-oben drücken und schon steht da die letzte Eingabe, z.B. wieder: „gforth myprojekt.txt", und sofort kann dieses erneut gestartet werden, das ganze Projekt baut sich sauber neu auf. Gforth holt sich aus diesem Projektsteuerfile alles, was es braucht. Und gibt Meldungen aus über den Stand der Entwicklung des Projektes bis zum abschließenden OK - oder bricht ab mit einer Fehlermeldung, die File, Zeile und Fehlerart enthält. Das genügt, um damit sofort ins Expose zu wechseln, und dort die beklagte Stelle aufzusuchen, in Dokumentationen nachzulesen, Notizen zu schreiben, eine Email mit einer Frage zum

(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 Mitgliedsbeitrag beträgt 12 engl. Pfund.
Hierfür bekommen Sie 6 Hefte unserer
Vereinszeitschrift Forthwrite.
Beschleunigte Zustellung (Air Mail)
ins Ausland kostet 20 Pfund.
Körperschaften zahlen 36 Pfund,
erhalten dafür aber viel Werbung.

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

Problem an den nächsten Expertenfreund zu schicken, den Chat mit ihm zu beginnen oder auch per Telefonie gleich zu reden - und dazu einen kleinen Artikel darüber für die VD zu schreiben. Die markierten Stellen bewegen sich von der Maus geführt dahin, wo sie hin sollen, in jedes Fenster nach Belieben. Ach ja, und das Terminal merkt sich alles, speichert und lädt die Sitzung nach Belieben, alles vom letzten Mal wieder da, wenn man will.

Mehrere TextEdit-Fenster offen zu haben, also meine Quellen für ein Projekt, das ist sehr angenehm. Der Finder in Panther ermöglicht es, das Projekt übersichtlich zu verwalten. Es sind viele Finderfenster gleichzeitig da. Das ermöglicht es, in dem einen Ordner zu blättern nach Quellfiles, oder in einem andern die Doku zu finden. Die dann wie im Fall des ANS Forth Standard als Html-Seiten daherkommt - muss ich sagen, dass der Browser auch offen ist und man mal eben darin herum blättern kann, mal eine Seite druckt, um was nachzulesen?

Expose macht es also besonders einfach, in Forth zu entwikkeln. Mit Expose überblickt man immer alle Fenster gleichzeitig, sobald der Mauszeiger in eine Ecke geht. Und man kann jederzeit mit einem Klick den Schreibtisch frei einsehen, das, was dort noch so an Files herumliegt dann duplizieren, archivieren, öffnen oder einräumen usw. Die Wechsel sind blitzschnell, aller einfachste Handhabung. Die Gedanken bleiben sehr frei für das eigentliche Projekt. Man muss sich nicht erst tagelang damit herumplagen, wie die spezifische IDE eines Herstellers XY denn nun wieder funktioniert. Das befreit wirklich!

Michael Kalus

Zitat:

"Gforth is the Forth implementation of the GNU project (Current release 0.6.2, have a look to the User Manual). Binary distributions for popular platforms such as DOS, Windows, Linux, OS/2 <***Der "Sätza": Und OS X nun auch! Bei Fink zu finden***> can be found in Home of Gforth. Gforth uses GCC to compile a fast direct or indirect threaded Forth; Gforth is fully ANS FORTH compliant. Authors of Gforth are Anton Ertl, Bernd Paysan, Jens Wilke, Neal Crook, and others.

The goal of the Gforth Project is to develop a standard model for ANSI Forth. This can be split into several subgoals:

- Gforth should conform to the ANSI Forth standard.

- It should be a model, i.e. it should define all the implementation-dependent things.

- It should become standard, i.e. widely accepted and used. This goal is the most difficult one."

Das folgende Programm-Beispiel soll das Gesagte illustrieren und erhebt keinen Anspruch auf Vollständigkeit oder Fehlerfreiheit.

Das Beispiel

```
." H8/300 Crossassembler Studie - "
   hex   marker neu
   vocabulary h8    h8 definitions

create image 8000 allot
." 8000 Bytes of imagespace allottet, "
here constant endofimage
variable tp
." target pointer build, "


." file handling..."
: tp+ ( -- ) tp @ 1+ tp ! ;
0 value fd-in
0 value fd-out
: open-input ( addr u --- )
  w/o open-file   throw to fd-in ;
: open-output ( addr u --- )
  w/o create-file throw to fd-out ;
: close-input ( -- )
  fd-in close-file throw ;
: close-output ( -- )
  fd-out close-file throw ;
: closefiles ( -- )
  close-input close-output ;
: flushimage  ( -- )
  image tp @ fd-out write-file
  throw fd-out flush-file throw ;
: h8bin ( -- )
  s" H8.bin" open-output flushimage
  close-output ;
." done."
cr
```

```
." Testing - "
: tp. cr ." TP=" tp @ . ;
: ..  tp. image 20 dump  ;
." included" cr

." Cross Image Access Words - "
: ORG ( n -- ) tp ! ;  0 org
\ In Holon wird in ein Ziel z compiliert,
\ daher werden hier auch c,z und ,z
\ definiert.
: c,z ( c -- ) image tp @ + c! tp+ ;
  \ compile one byte into target.
: ,z ( n -- ) dup 100 / c,z c,z ;
  \ compile two bytes into target.
." included" cr

." h8mnemonics.txt - "
  include h8mnemonics.txt  ." included" cr
." test.txt - "
  include test.txt   ." included" cr

\ Ende
```

H8 Mnemonics.txt

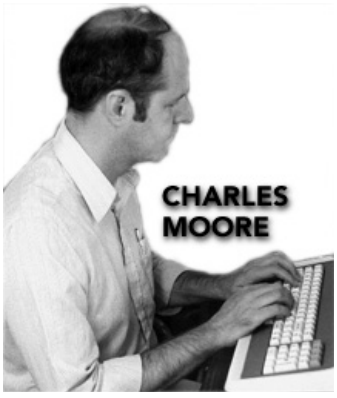
```
( H8 Mnemonics,Beispiel für den Aufbau der
  Befehle, alfabetische Folge.)
```

```
: ADD.B ( Rs Rd -- )
  08 c,z swap 10 * + ( rs_rd ) c,z ;
: ADD.B# ( xx:8 Rd -- )
  80 + c,z ( xx ) c,z ;
: ADD.w ( Rs Rd -- )
  09 c,z 7 and swap 7 and 10 * +
  ( 0rs_0rd ) c,z ;
: ADDS#1 ( Rd -- )
  0B c,z 7 and ( 0-0rd ) c,z ;
: ADDS#2 ( Rd -- )
  0B c,z 7 and 80 + ( 8_0rd ) c,z ;
: ADDX ( Rs Rd -- )
  0E c,z swap 10 * + ( rs_rd ) c,z ;
: ADDX# ( xx:8 Rd -- )
  90 + c,z ( xx ) c,z ;
: AND, ( Rs Rd -- )
  16 c,z swap 10 * + ( rs_rd ) c,z ;
  \ has to be and, in order to avoid
  \ conflict with and in forth.
: AND# ( xx:8 Rd -- )
  E0 + c,z ( xx ) c,z ;
: ANDC ( xx:8 -- )
  06 c,z ( xx ) c,z ;
: BAND# ( xx:3 Rd -- )
  76 c,z swap 7 and 10 * + ( 0x_rd ) c,z ;
: BAND#! ( xx:3 aa:8 -- )
  7E c,z ( aa ) c,z 76 c,z 7 and 10 *
  ( 0x_0 ) c,z ;
: BAND#) ( xx:3 Rd -- )
  7C c,z 7 and 10 * ( 0rd_0 ) c,z 76 c,z 7
  and 10 * ( 0x_0 ) c,z ;
: BCC ( d:8 -- )
  44 c,z ( pp ) c,z ;
: BCLR ( Rn Rd -- )
  62 c,z swap 10 * + ( rs_rd ) c,z ;
: BCLR# ( xx:3 Rd -- )
  72 c,z swap 7 and 10 * + ( 0x_rd ) c,z ;
: BCLR#! ( xx:3 aa:8 -- )
  7F c,z ( aa ) c,z 72 c,z 7 and 10 *
  ( 0x_0 ) c,z ;
: BCLR#) ( xx:3 Rd -- )
  7D c,z 7 and 10 * ( 0rd_0 ) c,z 72 c,z 7
  and 10 * ( 0x_0 ) c,z ;
: BCLR) ( Rn Rd -- )
  7D c,z 7 and 10 * ( 0rd_0 ) c,z 62 c,z
  10 * ( rn_0 ) c,z ;
: BCLRc! ( Rn aa:8 -- )
  7F c,z ( aa ) c,z 62 c,z 10 * ( rn_0 )
  c,z ;
: BCS ( d:8 -- ) 45 c,z ( pp ) c,z ;
: BEQ ( d:8 -- ) 47 c,z ( pp ) c,z ;
: BGE ( d:8 -- ) 4C c,z ( pp ) c,z ;
: BGT ( d:8 -- ) 4E c,z ( pp ) c,z ;
: BHI ( d:8 -- ) 42 c,z ( pp ) c,z ;
( snip... )

: JMP ( aa:16 -- )
  5A c,z 00 c,z ( aa ) ,z ;
: JMP) ( Rn -- )
  59 c,z 7 and 10 * ( 0rn_0 ) c,z ;
: JMP@ ( aa:8 -- )
  5B c,z ( aa ) c,z ;

: JSR ( aa:16 -- )
  5E c,z 00 c,z ( aa ) ,z ;
: JSR) ( Rn -- )
  5D c,z 7 and 10 * ( 0rn_0 ) c,z ;
: JSR@ ( aa:8 -- )
  5F c,z ( aa ) c,z ;
: LDC ( Rs -- )
  03 c,z ( 0_rs ) c,z ;
: LDC# ( xx:8 -- )
  07 c,z ( xx ) c,z ;
: MOV.B ( Rs Rd -- )
  0C c,z swap 10 * + ( rs_rd ) c,z ;
: MOV.B! ( Rs aa:16 -- )
  6A c,z 80 + ( 8_rs ) c,z ( aa ) ,z ;
: MOV.B# ( xx:8 Rd -- )
  F0 + c,z ( xx ) c,z ;
: MOV.B( ( Rs Rd -- )
  68 c,z swap 7 and 10 * + ( 0rs_rd )
  c,z ;
: MOV.B(+ ( Rs Rd -- )
  6C c,z swap 7 and 10 * + ( 0rs_rd )
  c,z ;
: MOV.B) ( Rs Rd -- )
  68 c,z 8 or 10 * + ( 1rd_rs ) c,z ;
: MOV.B-) ( Rs Rd -- )
  6C c,z 8 or 10 * + ( 1rd_rs ) c,z ;
: MOV.B)d ( pp Rs Rd -- )
  6E c,z 8 or 10 * + ( 1rd_rs ) c,z
  ( pp ) ,z ;
: MOV.B@ ( aa:16 Rd -- )
  6A c,z ( 0_rd ) c,z ( aa ) ,z ;
: MOV.Bc! ( Rs aa:8 -- )
  30 swap + c,z ( aa ) c,z ;
: MOV.Bc@ ( aa:16 Rd -- )
  20 + c,z ( aa ) c,z ;
: MOV.Bd( ( pp Rs Rd -- )
  6E c,z swap 7 and 10 * + ( 0rs_rd ) c,z
  ( pp ) ,z ;
: MOV.W ( Rs Rd -- )
  0D c,z 7 and swap 7 and 10 * +
  ( 0rs_0rd ) c,z ;
: MOV.W! ( Rs aa:16 -- )
  6B c,z swap 7 and ( 0_0rs ) c,z ( aa ) ,
  z ;
: MOV.W# ( xx:16 Rd -- )
  79 c,z ( 0_rd ) c,z ( xx ) ,z ;
: MOV.W( ( Rs Rd -- )
  69 c,z 7 and swap 7 and 10 * +
  ( 0rs_0rd ) c,z ;
: MOV.W(+ ( Rs Rd -- )
  6D c,z 7 and swap 7 and 10 * +
  ( 0rs_0rd ) c,z ;
: MOV.W) ( Rs Rd -- )
  69 c,z 8 or 10 * + ( 1rd_rs ) c,z ;
: MOV.W-) ( Rs Rd -- )
  6D c,z 8 or 10 * + ( 1rd_rs ) c,z ;
: MOV.W)d ( pp Rs Rd -- )
  6F c,z 8 or 10 * swap 7 and +
  ( 1rd-0rs ) c,z ( pp ) ,z ;
: MOV.W@ ( aa:16 Rd -- )
  6B c,z swap 7 and ( 0_0rd ) c,z ( aa ) ,
  z ;

: MOV.Wd( ( pp Rs Rd -- )
  6F c,z 7 and swap 7 and 10 * +
  ( 0rs_0rd ) c,z ( pp ) ,z ;
```


```
: MOVFPE@ ( aa:16 Rd -- )
  6A c,z 40 + ( 4_rd ) c,z ( aa ) ,z ;
: MOVTPE! ( Rs aa:16 -- )
  6A c,z swap C0 + ( C_rs ) c,z ( aa ) ,
  z ;
: MULXU ( Rs Rd -- )
  50 c,z 7 and swap 10 * + ( rs_0rd ) c,
  z ;
: NEG ( Rd -- )
  17 c,z 80 + ( 8_rd ) c,z ;
: NOP ( -- )
  00 c,z 00 c,z ;
( snip...)
( Die vollständigen Codes gibt es beim
  Fileservice der FG)
```

Test.txt

```
( Teste die Mnemonics)

\ : test ( -- ) .s depth if abort then ;
\ Stacktiefe muß 0 sein nach compilation
\ in diesem Test.


: test ;

 1 2 ADD.B ( Rs, Rd -- ) test
 1 2 ADD.B# ( xx:8 Rd -- ) test
 1 2 ADD.w ( Rs, Rd -- ) test
   2 ADDS#1 ( Rd -- ) test
   2 ADDS#2 ( Rd -- ) test
 1 2 ADDX ( Rs, Rd -- ) test
 1 2 ADDX# ( xx:8 Rd -- ) test
 1 2 AND, ( Rs Rd -- ) test
 1 2 AND# ( xx:8 Rd -- ) test
   2 ANDC ( xx:8 -- ) test
 1 2 BAND# ( xx:3 Rd -- ) test
 1 2 BAND#! ( xx:3 aa:8 -- ) test
 1 2 BAND#) ( xx:3 Rd -- ) test
   2 BCC ( d:8 -- ) test
 1 2 BCLR ( Rn Rd -- ) test
 1 2 BCLR# ( xx:3 Rd -- ) test
 1 2 BCLR#! ( xx:3 aa:8 -- ) test
 1 2 BCLRc! ( Rn aa:8 -- ) test
 1 2 BCLR#) ( xx:3 Rd -- ) test
 1 2 BCLR) ( Rn Rd -- ) test
   2 BCS ( d:8 -- ) test
   2 BEQ ( d:8 -- ) test
   2 BGE ( d:8 -- ) test
   2 BGT ( d:8 -- ) test
   2 BHI ( d:8 -- ) test
( snip...)


   2 JMP@ ( aa:8 -- ) test
   2 JMP) ( Rn -- ) test
   2 JMP ( aa:16 -- ) test
   2 JSR@ ( aa:8 -- ) test
   2 JSR) ( Rn -- ) test
   2 JSR ( aa:16 -- ) test
   2 LDC ( Rs -- ) test
   2 LDC ( xx:8 -- ) test
 1 2 MOV.Bc@ ( aa:16 Rd -- ) test
 1 2 MOV.Bc! ( Rs aa:8 -- ) test
 1 2 MOV.B) ( Rs Rd -- ) test
 1 2 MOV.B-) ( Rs Rd -- ) test
 1 2 MOV.B( ( Rs Rd -- ) test
 1 2 MOV.B(+ ( Rs Rd -- ) test
 1 2 MOV.B ( Rs Rd -- ) test
 1 2 MOV.B# ( xx:8 Rd -- ) test
 1 2 MOV.B@ ( aa:16 Rd -- ) test
 1 2 MOV.B! ( Rs aa:16 -- ) test
55 1 2 MOV.Bd( ( pp Rs Rd -- ) test
1 55 2 MOV.B)d ( Rs dd Rd -- ) test
 1 2 MOV.W) ( Rs Rd -- ) test
 1 2 MOV.W-) ( Rs Rd -- ) test
 1 2 MOV.W( ( Rs Rd -- ) test
 1 2 MOV.W(+ ( Rs Rd -- ) test
 1 2 MOV.W ( Rs Rd -- ) test
 1 2 MOV.W# ( xx Rd -- ) test
 1 2 MOV.W@ ( aa:16 Rd -- ) test
 1 2 MOV.W! ( Rs aa:16 -- ) test
1 55 2 MOV.W)d ( Rs dd Rd -- ) test
55 1 2 MOV.Wd( ( pp Rs Rd -- ) test
 1 2 MOVFPE@ ( aa:16 Rd -- ) test
 1 2 MOVTPE! ( Rs aa:16 -- ) test
 1 2 MULXU ( Rs Rd -- ) test
   2 NEG ( Rd -- ) test
     NOP ( -- ) test
( snip...)

( Ende)
```



Biographie

von

Charles H. Moore

(Übersetzung: *mka*)

Ausbildung

Geboren in McKeesport Pennsylvania, nahe bei Pittsburg, USA, im Jahre 1938, wuchs er auf in Flint, Michigan und wurde der Abschiedsredner der Central High School (1956). Über ein National-Merit-Stipendium kam er zum MIT und trat der Kappa-Sigma-Bruderschaft bei. Als Doktor der Physik (1960) wurde er mit einer Arbeit über Datenreduktion für den Explorer XI Gammastrahlen-Satelliten beauftragt und ging dann nach Stanford, wo er 2 Jahre Mathematik studierte (1961).

Programmierer

Er lernte Lisp von John McCarthy. Und Fortran II für den IBM 704, um damit Berechnungen für mondbeobachtende Satelliten am Smithsonian Astrophysical Observatory durchzuführen (1958). Dass dann auch in Assembler komprimierte Programm diente für die Bestimmung von Flugbahnen der Satelliten. (1959). An der anderen Küste der USA lernte er Algol für die Burroughs B5500, um damit in Stanford im Linear Accelerator Center Steuerungen für Elektronenstrahlen zu optimieren (1962). Als „Charles H Moore and Associates“ schrieb er ein Fortran-Algol-Übersetzer, um einen timesharing service zu unterstützen (1964). Er programmierte einen real-time Gaschromatographen auf seinem ersten Minicomputer (1965). Für ein Netzwerk in Mohasco lernte er Cobol (1968). Mehr Einzelheiten gibt es in einem unveröffentlichem Papier von ihm selbst. (Kontakt über http://www.colorforth.com)

Forth

Chuck erfand Forth (1968) und sammelte seine persönliche Software-Bibliothek auf einer IBM 1130, welche an dem ersten graphikfähigen Terminal, das er gesehen hatte, angeschlossen war (IBM 2250). Bald schon benutzte er Forth für das 30-Fuß-Teloskop am Kitt Peak für eine nationale Radio Astronomy Beobachtung (1970).

Er setzte bei der Gründung der Forth Inc. 5.000 $ ein. In den nächsten 10 Jahren portierte er Forthsysteme auf viele Mini-, Micro- and Main-frame Computer, und programmierte viele Anwendungen damit, von Datenbanken bis Robotics.

Im Jahre 1980 veröffentlichte das angesehene Byte Magazin ein Sonderheft über "The Forth Language." Das Vorwort von Greg Williams gibt einen der seltenen Eindrücke von Leuten außerhalb der Forthwelt wieder. (Siehe seine Homepage)

Chips

Schließlich beschloss er, einen Forth-Chip zu bauen, um die Architektur, die dem Forth innewohnt, in Hardware zu verwirklichen. Dazu wurde Novix, Inc. gegründet und er implementierte den NC4000 (1983) als Prozessor in einem gate array, entwickelte und vertrieb Bausätze dazu. Ein Derivat davon wurde an Harris Semiconductor verkauft und als RTX2000 für Weltraumanwendungen vermarktet (1988).

Nun als "Computer Cowboy" entwarf er den Sh-Boom Chip aus Standardzelle (1985), welcher noch immer vermarktet wird. Für den MuP21 entwickelte er sein eigenes Entwurfswerkzeug (1990). Der MuP21 hat multiple spezialisierte Prozessoren im Chip. Und sein F21 hat eine Netzwerk-Schnittstelle integriert (1993). Er gründete die iTv Gesellschaft, und entwarf den i21, eine ähnliche Architektur, aber mit erhöhter Leistung, gezielt für Internet Anwendungen (1996).

Nun zurück bei den Computer Cowboys, entwickelte Chuck colorForth, portierte sein VLSI design tool auch dorthin und entwarf den c18 microcomputer (2001), ein sehr einfacher Kern, der viele Male auf einem Chip repliziert werden kann. Alle seine Chips waren für hohe Leitungen und geringen Stromverbrauch ausgelegt.

Auszeichnungen:

- Auszeichnung für Beiträge zur Software Qualität und Computer-Design, 1983.
  (Die Plakette wurde unterzeichnet von Präsident Reagan.)
- Ehrenmitglied der FIG auf Lebenszeit.
- Ehrenvorsitzender der FIG China, 1987
- Footsteps in an Empty Valley, Offete Enterprises, 1988

Patente:

US 05070451 Forth Specific Language Microprocessor, 1991
US 05319757 Forth Specific Language Microprocessor, 1995
US 05440749 High performance,
low cost microprocessor architecture, 1995
US 05530890 High performance,
low cost microprocessor, 1996
US 05604915 Data processing system
having load dependent bus timing, 1997
US 05659703 Microprocessor system with hierarchical stack
and method of operation, 1997
US 05784584 High performance microprocessor using
instructions that operate
within instruction groups, 1998
US 05809336 High performance microprocessor
having variable speed system clock, 1998
EP 0870226 Risc microprocessor architecture, 1997
WO 9715001 Risc microprocessor architecture, 1997

Jüngste Publikationen:

- Renaissance Development,
Embedded Systems Conference, 1992

- The Evolution of Forth; Rather, Coburn, Moore;
History of Programming Languages II, Addison-Wesley, 1996

- Some chip documentation is on-line at UltraTechnology.

- Seine website: http://www.colorforth.com/index.html

mka, 26.2.2004

(Quelle: Biographie von Charles H. Moore auf seiner Website.)

Rätsellösung:

Variablentausch ohne Zwischenspeicherung

Fred Behringer

<behringe@ma.tum.de>

Zusammenfassung:
Das Assembler-Programmstück **(E)AX (E)CX XOR (E)CX (E)AX XOR (E)AX (E)CX XOR** ist eine Lösung. Auch arithmetische Verknüpfungen gehen (**ADD SUB, MUL DIV**). Jede Verknüpfung, die den betrachteten Zahlenbereich in der Sprache der Algebra zu einer Quasigruppe macht, funktioniert.

Es geht um das Rätsel von Arndt Klingelnberg
[Kli] aus dem VD-Heft 1/1994 und um die Lösung von Friederich (Fritz) Prinz [Pri] aus dem Heft 1/2004. Die **XOR-**Lösung ist in meinem "Rätsel" aus [Beh2] und in meiner "Lösung" aus [Beh3], beide aus dem Jahre 2000, implizit enthalten. Fritz Prinz, Michael Major und Martin Bitter haben sich damals bei der Lösung hervorgetan [Beh3] und es würde mich nicht wundern, wenn insbesondere Martin im vorliegenden VD-Heft ebenfalls mit einer Lösung aufwartet: Man betrachte Variablen als "Behälter" und wende die Kofferveranschaulichung aus [Beh3], die ich weiter unten in erweiterter Form wiederholen werde, an. Die am Schluss der vorliegenden Arbeit durchgeführten mathematischen Betrachtungen gleichen denen aus [Beh3]. Arndt Klingelnberg gibt die, wie er sagt,

"unpräzise Fragestellung"
aus dem VDI-Nachrichten-Magazin [Kli] an mit: "Ist es möglich, den Austausch der Werte zweier Variablen ohne eine Hilfsvariable durchzuführen?" Er versucht eine

"exaktere Formulierung für Forthler":
Statt auf Variable, Speicherstellen oder den Stack wird die Aufgabe auf prozessor-interne Register bezogen: Es sind die Werte in zwei Registern auszutauschen, ohne ein drittes Register (oder etwa auch ein bit-breites Register, wie beispielsweise im Prozessor-Flag) oder den möglicherweise vorhandenen XCHG-Befehl oder Ähnliches zu nutzen.

Mein Vorgehen:
Ich gehe vom Befehlssatz des 80486 aus. Der 80286 würde auch schon reichen. Der Pentium wäre noch zulässig. Den AMD64 mit seinen 64 Bits im Long-Mode betrachte ich nicht. Die hinreichend bekannte, allen Ansprüchen genügende Lösung steht oben in der Zusammenfassung. Ich möchte mich aber an eine möglichst vollständige Analyse des Problems wagen und will mich zunächst einmal der Lösung von Fritz [Pri] zuwenden.

Mir ist ein Malheur passiert ;-):
Ich hatte Fritzens Lösung [Pri] auf Lochkarten gestanzt. Mir glitt das Päckchen aus der Hand. Einige Karten waren in den Schmutz gefallen. Ich hob die sauber gebliebenen Karten auf und sortierte sie nach Nummern. Übrig blieb das folgende Programm, das drei noch freie Stackplätze benötigt und das Problem schon löst:

```
VARIABLE Eins  4 Eins !
VARIABLE Zwei 17 Zwei !
: Tauschen ( -- )
  Eins @ Zwei @ Eins ! Zwei ! ;
```

Aber mal im Ernst
gefragt: Wenn ich in einem Forth-Programm durch Streichung von mindestens einer Zeile zu einem Programm gelange, das auf die vorgesehenen Eingabewerte genau dieselbe Funktion ausübt wie das ursprüngliche ("redundante") Programm, habe ich mir dann einen "brauchbaren" Redundanzbegriff erschaffen? Brauchbar im Sinne von: Was kann ich damit Vernünftiges tun? Welche sinnvollen Forderungen muss ich bei der Definition eines solchen Redundanzbegriffs ganz allgemein stellen? Darf ich mich, wie in unserem Fall, über Schleifen und Sprünge großzügig hinwegsetzen und mir nur diejenigen Programmrosinen (unter Wahrung der Reihenfolge der Zeilen) herauspicken, die ich für meine Betrachtungszwecke benötige?

Soweit das "Philosophische"
des vorliegenden Artikels. Jetzt zur Lösung des Rätsels von Arndt Klingelnberg. Eigentlich hatte ich seit 1994 immer die Arndtsche "Präzisierung" und dabei die Intel-Prozessoren vor Augen (in High-Level-Forth gibt es ja kein **XCHG**) und "Variablen", dachte ich, seien allenfalls indirekt, beispielsweise über **0 [BX]**, ansprechbare Stellen im Arbeitsspeicher (wobei Ausdrücke der Form **0 [SI] 0 [DI] MOV** nicht erlaubt sind - Ziel oder/und Quelle müssen "einfache" Register, die Quelle kann auch ein Zahlenwert sein). In den VD-Heften wurde nie eine Auflösung des von Arndt Klingelnberg gestellten Rätsels gegeben. Das Vorgehen über **XOR** (irgendwie erinnert es an die Türme von Hanoi) war wohl doch zu offensichtlich.

Die Schiebeoperation
in Fritzens Lösung hat mich andererseits fasziniert (eigentlich "nur" auf Maschinenebene, da man in High-Level-Forth wohl kaum um die (in der Arndtschen Aufgabe verbotene) Verwendung eines Stacks herumkommt). Was mich dabei interessiert: In **AX** könnte man mit **AH** und **AL** getrennt hantieren, bei **EAX** (beispielsweise) geht das nicht. Dort kommt man nur an den "Lo-Part" **AX** heran, an den "Hi-Part" nicht. In meiner VD-Arbeit [Beh1] bin ich auf die 32-bittige **ROL**-Operation verfallen. Auch dazu mehr ein paar Zeilen weiter unten. Ich versuche im Folgenden, die jeweiligen Voraussetzungen präzise zu formulieren, und gebe Lösungen an. Andere Interpreta-

tionen würden andere Lösungsmöglichkeiten nach sich ziehen.

Meine Voraussetzungen 1:
In einer **2VARIABLE**n soll hi und lo vertauscht werden. (Ich hatte das in sehr starkem Maße in meinem VD-Artikel [Beh1] benötigt, wo ich in einem 16-Bit-System (Turbo-Forth oder ZF) 32-Bit-RAM-Zugriffe in Little-Endian-Manier bereitstellen musste.) Gesucht sei dafür eine **CODE**-Definition. Es sei nur **ein einziges** Register frei, das allerdings für die indirekte Adressierung verwendet werden können muss. O.B.d.A. nehme ich **BX**. Verboten seien die Maschinenbefehle **XCHG MOV POP PUSH**.

Lösung 1 (ROL):

```
HEX  : OP: 66 C, ;

2VARIABLE X  12345678. X 2!

CODE TAUSCH ( -- )
  BX BX XOR
  OP: X [BX] ROL  C1 HERE 4 - C!  10 C,
NEXT END-CODE
```

Ich gehe von einem 16-Bit-System (Turbo-Forth) aus
und verwende gemischte 16/32-Bit-Programmierung: Die Adresse (die von **X** schon beim Assemblieren aufgerufen wird) hat 16 Bits, die Operation **ROL** soll gleich auf 32 Bits wirken. Der Turbo-Forth-Assembler macht da nicht mit. Ich helfe nach: **OP:** stellt (hier und im Folgenden) das Präfix 66h für das 32-Bit-**ROL** bereit. Ein Präfix 67h wird (hier und im Folgenden) wegen der nur 16 Bit breiten Adressen nicht benötigt. **X [BX] ROL** wird (in Turbo-Forth und anderswo) als **ROL**-Befehl mit Verschiebung um nur ein einziges Bit assembliert. Der Befehlscode für **ROL** wird dabei mit **D1** angesetzt. Ich möchte eine (direkt anzugebende) Verschiebung um 10h Bits haben. Dazu muss der **ROL**-Befehl mit dem Code **C1** angesetzt werden. Mit **C1 HERE 4 - C!** wandle ich (beim Assemblieren) das vom Assembler zunächst eingetragene **D1** unmittelbar danach in **C1** um. Schließlich wird an das bis dato erreichte Assemblat mit **10 C,** die Bitverschiebezahl angehängt.

Ich bin Forth dankbar
dafür, dass es mich das alles tun lässt. Ich darf in das Assembliergeschehen eingreifen, ohne dafür mit einer Fehlermeldung bestraft zu werden. In anderen Systemen (soweit ich solche kennen lernen durfte) hatte ich das nicht gekonnt. Dabei spreche ich von Forth als System, nicht als Sprache. Über das Bessersein oder nicht Bessersein als Sprache kann ich nicht urteilen. Als System begeistert es mich immer wieder aufs Neue.

Das eben besprochene TAUSCH benötigt keinen Datenstack, ganz so, wie es die Voraussetzung will. Dafür handelt man sich aber ein, dass dieses **TAUSCH** auf die vorher definierte Variable **X** und nur auf diese zugeschnitten ist. Für jede andere Variable muss man sich ein anderes **TAUSCH** compilieren (assemblieren). (Die von Arndt Klingelnberg erwähnte ursprüngliche Voraussetzung, die er als unpräzise betrachtet, lässt ein solches Vorgehen zu.) Außerdem muss es sich um eine **2VARIABLE** (Begründung siehe gleich) handeln.

Variationsmöglichkeiten (ROR ADD SUB):
Statt **BX BX XOR** geht offensichtlich auch **0 # BX AND** oder **BX BX SUB**, statt **ROL** auch **ROR**. Das sind 6 Variationsmöglichkeiten für diesen Lösungsvorschlag.

Gemischte 16/32-Bit-Programmierung
ist bei den eben besprochenen Vorschlägen wesentlich. In einem 32-Bit-Forth-System mit einem 32-Bit-Intel-Prozessor könnte man hi und lo in einer **2VARIABLE**n in einer wie eben aufgezogenen **CODE**-Definition nicht vertauschen. Der reine Schiebebefehl **SHLD**, der zwar über zwei 32-Bit-Register wirkt, lässt sich dazu nicht verwenden, da er die herausfallenden Bitstellen nicht ins andere 32-Bit-Register einfügt. Mit einem auf 32-Bit-Register angewandten **XOR** geht es aber - sogar wenn die zu vertauschenden 32-Bit-Zellen nicht nebeneinander liegen. Mehr dazu, in Low-Level-Forth etwas später, in High-Level-Forth gleich:

Meine Voraussetzungen 2:
Es soll (interpretativ) in ANS-kompatiblem High-Level-Forth gearbeitet werden. Die (zunächst nur 16-bitigen) **VARIABLE**n **X** und **Y** sollen ihre Werte miteinander vertauschen. Der Returnstack sei nicht zugelassen und der Datenstack habe nur noch **zwei** Zellen frei. (32-Bit-Operationen (in 16-Bit-Systemen) oder entsprechende Überlegungen gleich für 32-Bit-Systeme verlaufen nach gleichem Schema - vorausgesetzt, man hat im **2VARIABLE**n-Fall das Nicht-ANS-Forth-Wort **2XOR** zur Verfügung.)

Lösung 2 (XOR):

```
X @ Y @ XOR   X !
X @ Y @ XOR   Y !
X @ Y @ XOR   X !
```

Mehr zur **XOR**-Technik weiter unten. "**XOR** verwendet man in der Verschlüsselungstechnik, wenn man mit dem Computer arbeitet. In der Zeit vor Erfindung des Computers hat man mit Plus und Minus gearbeitet", sagt [Agu1], S. 23. Nun denn!:

Alternativlösung 2 (+ und -):

```
X @ Y @ +   X !
X @ Y @ -   Y !
X @ Y @ -   X !
```

Oder weniger symmetrisch, aber stärker platzsparend:

```
Y @ X +!
X @ Y @ -   Y !
X @ Y @ -   X !
```


Variablentausch ohne Zwischenspeicherung

Dazu gibt es Varianten, beispielsweise:

```
X @ Y @ -   Y !
X @ Y @ -   X !
X @ Y +!
```

Und noch mehr Platz einsparend, statt 18 nur noch 16 Forth-Worte:

```
Y @ NEGATE X +!
X @ Y +!
Y @ X @ -   X !
```

Hätte man ein -! (naheliegend, aber nicht ANS-Forth) zur Verfügung, könnte man schreiben:

```
Y @ X -!
X @ Y +!
Y @ X @ -   X !
```

und hätte noch ein weiteres Forth-Wort eingespart, statt 16 nur noch 15.

Das additive Verfahren funktioniert immer,
auch wenn die durch **+!** erzeugte Summe über die Zellbreite hinausschießt: **+** und **-** werden (in 16-Bit-Systemen) modulo 65536 ausgeführt. "**+**" führt im Zahlenkreis rechts herum, "**-**" links herum. Beide Verfahren sind zudem von der Zellbreite (Bitzahl) des Forthsystems unabhängig.

Bemerkt werden muss jedoch,
dass die Annahme, nur noch zwei Zellen auf dem Stack seien frei, davon ausgeht, dass alle hier aufgeführten Forth-Worte Primärworte sind und beim Abgearbeitetwerden nicht etwa selbst noch Stackplatz benötigen. Ein System, in welchem **+!** als **: +! DUP @ ROT + SWAP ! ;** definiert wäre, wäre hier nicht geeignet.

In Turbo-Forth und anderen F83-Derivaten,
die sämtliche 64-KB-Segmente auf ein und denselben Platz legen (SS = CS = DS = ES) und für den Forth-Stack-Zeiger den System-SP verwenden, ist es allerdings müßig, bei Primärworten anderes Stackverhalten als bei High-Level-Forth-Worten zu erwarten. Um bei High-Level-Forth-Betrachtungen der Art, wie sie oben geführt wurden, die Voraussetzung "keine Extra-Stack-Bewegungen" aufrecht erhalten zu können, müsste man dafür sorgen, dass alle in den Sekundärworten verwendeten Primärworte ohne die Zuhilfenahme des Systemstacks auskommen. Das könnte man machen, das wäre aber eine Systemveränderung und würde weit über die von Arndt Klingelnberg gestellte Rätselfrage hinausführen.

In BASIC geht das Additionsverfahren
ohne weiteres, zumindest in QBASIC aus DOS 6.2. Auch das "Zahlbereichs-Wrapping" klappt:

```
X = X + Y: Y = X - Y: X = X - Y
```

Auch XOR geht in QBASIC:

```
X = X XOR Y: Y = X XOR Y: X = X XOR Y
```

Auch die Multiplikation geht (sowohl in QBASIC wie auch in Forth), soweit man nicht erwartet, dass ein Überlauf des Multiplikations-Produktes durch "Zahlbereichs-Wrapping" aufgefangen wird. Man muss also durch Beschränkung auf hinreichend kleine Werte dafür sorgen, dass das Produkt wieder in eine Zelle passt. Beschränkt man sich, was in Forth bei Verwendung von "/" selbstverständlich geschieht, auf ganzzahlige Werte, so können Werte, die zu klein sind, keinen Schaden anrichten. Null dürfen allerdings aus offensichtlichen Gründen weder **X** noch **Y** werden. Bei QBASIC, das ja auch nichtganzzahlige Eingabewerte akzeptiert, würde beim Multiplikationsverfahren die Division als Umkehrung der Multiplikation bei zu kleinen Werten unerträgliche Ungenauigkeiten in die Rechnung bringen können, welche beim Additionsverfahren nicht auftreten könnten. Dessen ungeachtet folgen hier meine Programmvorschläge für das Multiplikationsverfahren:

```
X @ Y @ *   X !
X @ Y @ /   Y !
X @ Y @ /   X !   (Forth)

X = X * Y:  Y = X / Y:  X = X / Y   (BASIC)
```

47,11 für X,Y funktioniert in BASIC wie in Forth.
7047,7011 für **X,Y** geht, zumindest im (16-Bit-)Turbo-Forth-System, nicht! Das Produkt passt nicht mehr in eine Zelle (2 Byte). In QBASIC (unter DOS 6.2) geht 7000047,7000011 für **X,Y** (gerade) noch. Auch 70000.47,70000.11 für **X,Y** geht (QBASIC verarbeitet ja (auch) Gleitkommazahlen). Bei 70000047,70000011 für **X,Y** steigt auch QBASIC aus.

Variationen des Multiplikationsverfahrens
sind ähnlich wie beim Additionsverfahren möglich. Man betrachte:

```
X @ Y @ /   X !
X @ Y @ *   Y !
Y @ X @ /   X !   (Forth)

X = X / Y:  Y = X * Y:  X = Y / X   (BASIC)
```

Das geht in Forth nicht (unbedingt)!
In Turbo-Forth funktioniert es beispielsweise bei Eingabe von 47,11 für **X,Y** nicht. 47 geteilt durch 11 lässt einen Rest, und der bringt das Verfahren durcheinander. Die Division kann auch im Bereich der ganzen Zahlen als Umkehrung der Multiplikation betrachtet werden, die Division selbst ist aber im ganzzahligen Bereich ohne Rest nicht durchgehend definiert.

QBASIC macht (cum grano salis) keine Schwierigkeiten
In QBASIC wird nach Bedarf im Gleitkommaverfahren gerechnet (für mich als Benutzer nur an der Wirkung erkennbar, da ich keinen Zugang zu den Quelltexten habe). 47,11 für **X,**

Y geht ohne weiteres. 700047,700011 für **X,Y** geht nicht mehr. Es liefert 700011,700046.9 für **X,Y**. Hier lässt die Genauigkeit der Gleitkommaverarbeitung nach.

EQU als Gegenstück zu XOR:
XOR ist im Sprachgebrauch der Aussagenlogik (von der bitweisen, also vektoriellen Verwendung mal abgesehen) die "Antivalenz". Der Variablentausch funktioniert auch mit deren Gegenstück, der "Äquivalenz". Als bitweise Operation gibt es diese weder in Forth noch in QBASIC. (Man verwechsle sie nicht mit der Gleichheitsoperation "**=**"!) Ich führe die (bitweise) Äquivalenzoperation ein:

```
: EQU ( n1 n2 -- n3 ) XOR NOT ;
```

Man kann **XOR** als bitweise Ungleichheit und **EQU** als bitweise Gleichheit betrachten. Damit erreiche ich den Variablentausch ohne Drittvariable unter den bei **XOR** gemachten Voraussetzungen durch:

```
X @ Y @  EQU  X !
X @ Y @  EQU  Y !
X @ Y @  EQU  X !
```

MIX als Mischung aus XOR und EQU:
Wenn es nun aber in Bezug auf die einzelnen Bits schon egal ist, ob **XOR** oder **EQU,** dann kann man auch eine Mischform aus beiden (ich nenne sie **MIX**) betrachten, bei welcher jedem einzelnen Bit mitgeteilt wird, ob es ge**XOR**t oder ge**EQU**t werden soll. Das bitweise Umschalten oder Nichtumschalten geschieht über ein abermaliges **XOR**en, und zwar mit dem anfangs einzugebenden Wert für die Variable **SEED** (man beachte, dass weder bei **EQU** noch bei **MIX** mehr als zwei noch freie Plätze auf dem Datenstack benötigt werden):

```
VARIABLE SEED

: MIX ( n1 n2 -- n3 ) XOR SEED @ XOR ;
```

Und hier der Variablentausch mit MIX:

```
X @ Y @  MIX  X !
X @ Y @  MIX  Y !
X @ Y @  MIX  X !
```

Meine Voraussetzungen 3:
Gesucht wird eine **CODE**-Definition (in Intel-Assembler). Es stehen zwei Register zur Verfügung, von denen eins zur indirekten Adressierung verwendet werden kann, sagen wir **AX** und **BX**. Die (Adresswerte der) Variablen (nennen wir sie **X** und **Y**) seien schon bei der Compilation der gesuchten **CODE**-Definition bekannt. Die Werte von **X** und **Y** sollen gegeneinander ausgetauscht werden. Verboten seien die Operationen **XCHG MOV POP PUSH** und Verschiebebefehle. Und es darf auch kein Speicherplatz zur Zwischenspeicherung verwendet werden.

Das hier verbotene MOV
verkompliziert den folgenden Lösungsvorschlag gegenüber den obigen High-Level-Vorschlägen, die eigentliche **XOR**-Technik bleibt dieselbe. Ich ersetze **MOV** durch **XOR** nach folgendem Muster:

```
X [BX] AX MOV = AX AX XOR X [BX] AX XOR
```

Die Frage, inwieweit sich **XOR** wiederum durch **EQU** oder **MIX** ersetzen lässt und ob Optimierungen möglich sind, will ich als sofort beantwortbar betrachten und hier nicht weiter verfolgen.

Lösung 3 (Intel-Assembler in CODE-Definition):

```
HEX

  2VARIABLE X 12345678. X 2!
       \ Willkürliche Belegung
  2VARIABLE Y 76543210. Y 2!
       \ Willkürliche Belegung

CODE TAUSCH ( -- )
OP: AX AX XOR  OP: BX BX XOR
    \ eax = 0  ebx = 0
OP: Y [BX] AX XOR
    \ eax = 76543210
OP: X [BX] AX XOR
    \ eax = 12345678xor76543210
OP: AX X [BX] XOR
    \ (X) = 76543210
OP: AX Y [BX] XOR
    \ (Y) = 12345678
NEXT END-CODE
```

Wegen **OP:** siehe oben bei **ROL**. Die mit **XOR** arbeitende **CODE**-Definition **TAUSCH** habe ich absichtlich gleich für 32-Bit-Variablen formuliert, weil das so herrlich einfach geht.

Für 16-Bit-VARIABLEn
sieht es natürlich wie folgt aus:

```
HEX
  VARIABLE X  47 X !
  VARIABLE Y  11 Y !

CODE TAUSCH ( -- )
      AX AX XOR      BX BX XOR
  Y [BX] AX XOR  X [BX] AX XOR
  AX X [BX] XOR  AX Y [BX] XOR
NEXT END-CODE
```

Das Wesentliche an der XOR-Technik
kann man in einer **CODE**-Definition herausarbeiten, wenn man folgende

Voraussetzung 4 (Registertausch)
macht: Gesucht sei eine **CODE**-Definition. Es mögen nur zwei Register zur Verfügung stehen, sagen wir **AX** und **CX.** (Ein bereits erarbeiteter Wert im "Akkumulator" **AX** soll beispiels-

Variablentausch ohne Zwischenspeicherung

weise zum Zählwert im "Count-Register" **CX** gemacht werden, wobei die Inhalte von **AX** und **CX** nach Beendigung der in diesem Zusammenhang zu betrachtenden Schleife in ihren ursprünglichen Werten wieder zur Verfügung stehen sollen.) Verboten seien **XCHG MOV PUSH POP** und die Verwendung von Zwischenspeichern im RAM. Man vergleiche diese Formulierung mit der eingangs erwähnten von Arndt Klingelnberg [Kli].

Lösung 4 (XOR):

```
CODE TAUSCH
   AX  CX XOR      CX  AX XOR
   AX  CX XOR    NEXT END-CODE

CODE TAUSCH
   EAX ECX XOR   ECX EAX XOR
   EAX ECX XOR  NEXT END-CODE

CODE TAUSCH
   OP: AX  CX XOR
   OP: CX  AX XOR
   OP: AX  CX XOR      NEXT END-CODE
```

Das erste Beispiel gilt für 16-Bit-Systeme (Turbo-Forth), das zweite für 32-Bit-Systeme, das dritte wieder für 16-Bit-Systeme (für 80x86-artige Prozessoren mit 32-Bit-Register, die über das Präfix 66h angesprochen werden können). Wegen **OP:** siehe weiter oben bei **ROL**. Zum Ausprobieren baue man geeignete **PUSH**s und **POP**s ein. Natürlich können auch auf diese "einfachere" Aufgabenstellung **EQU MIX ADD SUB MUL DIV** angewendet werden.

Meine Voraussetzungen 5:
Ich betrachte x und y als variable n-komponentige Vektoren im mathematischen Sinn, n natürliche Zahl, mit Komponenten aus {0,1}. Ich führe den Begriff des "Übertragens" ein: u --> w soll heißen, dass der Wert des Vektors u in den Vektor w übertragen wird, ohne dabei aus u entfernt zu werden - mit der offensichtlichen Ausnahmeregelung für f(u) --> u. Ich suche eine vektorwertige Funktion f, derart, dass allein durch genügend oftmaliges Anwenden von f(x,y) --> z, mit z aus {x,y}, die Werte von x und y vertauscht werden.

Lösung 5 (rein mathematisch):
Ich setze für f die komponentenweise Operation des exklusiven Oders, geschrieben "xor", und verwende für diese Vektorverknüpfungsoperation die Postfix-Schreibweise. Es sei x = a und y = b. Ich führe aus:

```
x y xor  -->  x
x y xor  -->  y
x y xor  -->  x .
```

Offensichtlich ist jetzt (nach Ausführung dieser drei Operationen) x = b und y = a. Eine dritte Variable ("Ringtausch") zur "Zwischenspeicherung" wurde nicht benötigt. Der Pfeil kann als "Übergang" (von einem Zustand zum nächsten) in einem (endlichen) Zustandsautomaten gedeutet werden. x und y sind dann Zustandsvariablen, die mit bestimmten Werten belegt sein oder werden können.

Vereinbart man,
dass xor immer auf die am weitesten rechts stehenden beiden Vektoren wirkt und die "Übertragung" immer in den linken dieser beiden Vektoren erfolgt, dann kann man für die Gesamtoperation (unter Einsparung des Übertragungszeichens "**-->**") schreiben: **x y x y xor xor xor** .

Das sieht fast wie ein Forth-Programm aus.
Es ist natürlich noch keins, da das Forth-Wort fürs Abspeichern in die jeweilige Variable, "**!**", fehlt, welches oben noch in der "Vereinbarung" steckt. Man könnte dieses Abspeichern in ein geeignet erweitertes xor einbauen. Es hat keinen Zweck, diesen Gedanken weiterzuverfolgen, da ja ein Forth-Programm, das in der angedeuteten Richtung läuft, ohne die Verwendung des (eines) Stacks nicht auskommt. Und wenn schon Stack, dann natürlich gleich so, wie ganz oben bei den "aufgesammelten sauberen Lochkarten".

Die Schreibweise ohne @ könnte man auch in einem Forth-Programm verwenden, wenn man **x** und **y** als **VALUE**s auffasst und "**-->**" als **TO** interpretiert. (Interessant, dass "**--> x**" nicht etwa "**x TO**" geschrieben werden müsste, wie man es von "**!**" her vielleicht erwarten würde, ja, dass es aus naheliegenden Gründen gar nicht in Postfix-Manier geschrieben werden KANN.) Man beachte auch, dass (die "Bitzahl") n in diese Betrachtung nicht eingeht.

XOR und die Kryptographie:
Die hier verwendete **XOR**-Technik ist vom Verschlüsseln und Entschlüsseln von Geheimschriften her bekannt (siehe [Agu1], [Agu2],[Dix1],[Dix2]). In [Dix1] steht auf Seite 14: "Wir wissen, dass **n1 n1 XOR n2 XOR** uns **n1** als Ergebnis liefert." In [Agu1] steht auf Seite 23: "Man beachte, dass **XOR** sich selbst rückgängig macht: **a b XOR b XOR = a**". Ich wiederum sage: "Man beachte, dass wir bei diesen "Erläuterungen" (in Forth-Notation) ganz gewaltig vom Datenstack Gebrauch machen!"

Tausch ohne Zwischenvariable und Verschlüsselung ohne Schlüsselversand sind von der Struktur her dasselbe. Man kennt das Erläuterungsbeispiel des (Nachrichten-)Koffers mit den zwei Schlössern: X legt die Nachricht in den Koffer, verschließt das linke Schloss und schickt den Koffer an Y. Y verschließt das rechte Schloss und schickt den Koffer zurück. X öffnet das linke Schloss und schickt den Koffer endgültig an Y. Y öffnet das rechte Schloss und entnimmt die Nachricht. Der Koffer ist unterwegs stets verschlossen, jeder hat seinen eigenen Schlüssel, kein Schlüssel braucht ausgetauscht zu werden. Die Gefahr des Schlüsselabfangens ist also gebannt.

Wie viele Wege legt der Koffer zurück?
Geht man davon aus, dass der Koffer wieder an seinen Standort zurückgeschickt werden muss, dann durchläuft er den Weg

viermal. Es wird aber nur eine einzige Nachricht übermittelt, die von X an Y. Nimmt man einen Doppelkoffer, Deckel oben und Deckel unten, dann kann man mit nahezu demselben Aufwand Nachrichten echt "austauschen". X hat den oder (doppelte Sicherheit) "die" Schlüssel zu den linken Schlössern, Y den oder die zu den rechten Schlössern.

Der Doppelkoffer:
X legt seine Nachricht in die eine Kofferhälfte und verschließt das zugehörige linke Schloss. Ab zu Y.
Y legt seine Nachricht in die andere Kofferhälfte und verschließt die beiden rechten Schlösser. Ab zu X.
X öffnet das linke Schloss der einen Hälfte und verschließt das linke Schloss der anderen Hälfte. Ab zu Y.
Y öffnete die beiden rechten Schlösser und entnimmt der einen Hälfte die Nachricht an ihn. Ab zu X.
X öffnet das linke Schloss der anderen Hälfte und entnimmt die Nachricht an ihn.

Wie viele Verschlüsselungsmöglichkeiten gibt es?
Von der Struktur her entspricht der Doppelkoffer offensichtlich dem Variablentausch nach dem **XOR**-, **EQU**- oder **MIX**-Verfahren, der ohne Zuhilfenahme einer dritten Variablen vor sich geht. Von der Verschlüsselung her wären das (bei einem 16-Bit-Forth-System) bei jedem der drei Verfahren 65536 (65534 wesentliche) Möglichkeiten (des Schlüssels, mit dem ge**XOR**t usw. wird). Das **MIX**-Verfahren besteht aber selbst wieder aus einer ganzen Klasse von 65536 möglichen Verfahren, je nach Wert von **SEED** (der "verschlüsselte Schlüssel"). Gibt es etwa 65536*65536 verschiedene Möglichkeiten? Nein. Man kann die Verschlüsselung des Schlüssels ohne weiteres in den Schlüssel selbst mit hineinnehmen - und sich auf das vom Konzept her einfachere (und übliche, bitweise) **XOR**-Verfahren beschränken. Der Möglichkeiten gibt es nach wie vor nur 65536 (mit zwei unwesentlichen).

Jede nicht notwendig kommutative Gruppe
wäre für den in der vorliegenden Arbeit besprochenen Variablentausch ohne Hilfsvariable (oder für die Verschlüsselung) geeignet. Dabei ist es völlig gleich, wie die Gruppenverknüpfung aussieht. Selbst bei **XOR** aus Sicht des Computers haben wir bereits zwei Möglichkeiten der Interpretation: (a) Das "Wertematerial" als (endliche) Menge von Vektoren mit Komponenten aus {0,1} auffassen und die Abbildung **XOR** der Menge auf sich selbst (ähnlich der Vektoraddition) als (der Aussagenlogik fremde) Vektorverknüpfung (bitweise Operation) nehmen oder (b) das "Wertematerial" als Ganzzahlenausschnitt betrachten (`3 5 XOR . [ret] 6 ok`), womit **XOR** zu einer arithmetischen Operation, zu einem eigenartigen "+", wird. Selbst die Endlichkeit könnte man aufgeben: Statt Vektoren könnte man Folgen (Vektoren mit unendlich vielen Komponenten) betrachten. Doch das interessiert hier weniger. Bleiben wir bei endlich vielen 0/1-Komponenten. Es lässt sich leicht einsehen, dass der beispielsweise 16-bittige Ganzzahlenbereich mit der üblichen bitweisen **XOR**-Verknüpfung folgende Eigenschaften hat (ich schreibe x für xor und benutze die Infix-Notation):

(G1) Je zwei Elementen a,b wird eindeutig ein drittes Element c = a x b zugeordnet.
(G2) Für je drei Elemente a,b,c gilt a x (b x c) = (a x b) x c.
(G3) Es gibt ein linksseitiges Nullelement 0 mit 0 x a = a für alle a.
(G4) Zu jedem a existiert mindestens ein Inverses -a mit (-a) x a = 0

Das sind genau die Eigenschaften, die in der Algebra eine Gruppe festlegen [Wae]. Es gilt bei der **XOR**-Verknüpfung sogar stets a x b = b x a. Die Gruppe ist also kommutativ oder abelsch. Nicht, dass der hier besprochene Variablentausch nicht auch bei schon weniger Eigenschaften funktionierte. Bei Gruppen funktioniert er jedenfalls. Beispiele für Gruppen sind: Die Ganzzahlen mit der Addition, die geraden Ganzzahlen mit Addition, die rationalen, reellen, komplexen Zahlen mit der Addition, die rationalen, reellen, komplexen Zahlen mit Ausnahme der Null mit der Multiplikation, die rationalen oder reellen Vektoren mit der Vektoraddition, die rationalen, reellen, komplexen Matrizen mit der Matrizenaddition, reguläre Matrizen mit der Matrizenmultiplikation, Permutationen einer endlichen Menge beliebiger Elemente, Drehungen einer Ebene um einen festen Punkt, eineindeutige Abbildungen einer Menge in sich u.v.m.

Interessant ist der Begriff der Inversen,
wenn man **XOR** entsprechend Punkt (a) im vorigen Abschnitt als (bitweise) logische Operation auffasst: Jedes Element ist dann seine eigene Inverse (a x a = 0)

Interessant ist auch das weiter oben besprochene Additionsverfahren:
Neben der Addition braucht man bei der Gruppenbetrachtung keinen gesonderten Subtraktionsbegriff. Das Abziehen einer Zahl wird als Hinzuzählen des Negativen dieser Zahl aufgefasst. (Hier vermischen sich die Begriffe Minuszeichen und Subtraktionszeichen.) Bei einer Gruppe! Nicht unbedingt so beim obigen Additionsverfahren. Da wurde nichts negiert. Da wurde aber die Subtraktion als eigenständige Operation verwendet, als Umkehrung der Addition. Ähnliches gilt für Multiplikation und Division (bei Aussparung der Null). Das heißt, wir brauchen beim hier diskutierten Variablentausch nicht unbedingt den vollen Begriff der Gruppe.

Eigenartig ist die Betrachtungsweise
beim obigen Punkt (b). Wir bauen uns, wenn wir die Bitmuster als Binärzahlen betrachten, einen recht seltsam anmutenden Additionsbegriff auf: 3 "+" 5 = 6, aber auch 11 "+" 13 = 6. Seltsam an sich, weil ungewohnt, nicht weil die Summe beide Male 6 ist. Bei der üblichen Addition ganzer Zahlen haben wir ja auch 3 + 5 = 8, aber daneben beispielsweise auch 7 + 1 = 8: Das Ergebnis der Addition lässt keinen Rückschluss auf die beteiligten Summanden zu. Aber etwas anderes ist interessant: Statt **XOR** können wir auch **MIX** im oben diskutierten Sinne verwenden. Was bekommen wir (in einem 16-Bit-System)? 65536 verschiedene seltsam anmutende Additionsbegriffe. Ich möchte mich hüten, da schon von "verschiedenen Arithmeti-

Variablentausch ohne Zwischenspeicherung

ken" zu sprechen, denn dazu bräuchten wir zu den verschiedenen Additionen noch "passende" Multiplikationen, die mit den Additionen über Distributivgesetze verknüpft sein müssten. Das würde uns zum Begriff des Körpers führen, ein Begriff, der (in dieser Arbeit) weder zum Variablentausch noch zur Verschlüsselung nötig war.

Kommutativität ist nicht unbedingt nötig.
Allerdings muss dann die (beim obigen Additionsverfahren stark herangezogene) "Umkehrabbildung" in zwei Varianten aufgespalten werden: einmal von links her und einmal von rechts her. Ich gebe ein Beispiel: Man betrachte die Menge aller regulären (reellen) 2x2-Matrizen.

"Regulär" heißt,
es existiert eine Inverse. Man erhält diese beispielsweise, indem man die Matrix selbst und die Einheitsmatrix (in der Hauptdiagonalen Einsen) nebeneinander schreibt und auf beide gleichzeitig solange elementare Zeilenumformungen einwirken lässt, bis aus der ursprünglich gegebenen Matrix die Einheitsmatrix geworden ist. Das, was die Zeilenumformungen dann aus der Einheitsmatrix gemacht haben, ist die Inverse der ursprünglichen Matrix.

Ein konkretes Beispiel:
Ich schreibe 2x2-Matrizen zeilenweise, zuerst die erste Zeile, dann die zweite. Multipliziert man (nach der Methode "Zeile mal Spalte", d.h. Zeile i komponentenweise mit Spalte j malgenommen und das Ganze addiert, ergibt das Matrixelement ij) die Matrix (0120) mit (0110), so erhält man (1002). Multipliziert man (0110) mit (0120), so erhält man (2001). Die Matrizenmultiplikation ist also nicht kommutativ - obwohl beide Matrizen regulär sind! Die Inverse von (0120) ist (0 0.5 10), die von (0110) ist wieder (0110).

Ganz allgemein schreibe ich:
A i* B , wenn B von links her mit der Inversen von A multipliziert wird, und A *i B, wenn A von rechts her mit der Inversen von B multipliziert wird (merke: zwei verschiedene Umkehroperationen, zwei verschiedene "Matrixdivisionen").

Den Variablentausch ohne Hilfsvariable
kann man dann wie folgt vornehmen (x,y seien Variablen, A,B seien reguläre nxn-Matrizen):

```
                  ; x = A      y = B
x *  y --> x      ; x = A * B  y = B
x *i y --> y      ; x = A * B  y = A
y i* x --> x      ; x =     B  y = A
```

Voilà, der Variablentausch ohne Hilfsvariable!

Wozu braucht ein Forthler Matrixprodukte?
Ja, wozu eigentlich? Dass er sich Operationen wie i* und *i ganz schnell selbst bereitlegen kann, ist klar. In Forth geht alles. Auf vorgegebene Datentypen braucht keine Rücksicht genommen zu werden. Man gibt sie sich selbst vor. Und ab ovo braucht niemand zu arbeiten. Für naturwissenschaftliche Berechnungen in Forth schlägt man beispielsweise bei Noble nach [Nob]. Und wenn in der VD vorgeschlagen wird, man möge doch mal, bitteschön, Wirtschaftssimulationsspiele programmieren, in Forth selbstverständlich [Pil], dann wird man leicht vor der Aufgabe stehen, geschachtelte lineare Gleichungssysteme lösen zu müssen. Dazu, beispielsweise, benötigt man Produkte von Matrizen und deren Inversen. Oder aber man denke an das Lösen eines Systems von 100 (vielleicht sogar 1000) linearen Gleichungen in entsprechend vielen Unbekannten. Beispielsweise zur Lösung des europäischen Transportproblems bei Milch, Wein oder Heizöl über die Lineare Optimierung. Wenn man da zwei Koeffizientenmatrizen austauschen möchte ... Ich weiß, die Argumente hinken - an verschiedenen Stellen, aber immerhin! Beim Variablentausch wird man natürlich, wenn es denn schon gleich für Matrizen sein muss, das (oben besprochene) Additionsverfahren nehmen - oder wieder das viel einfachere **XOR**. Was in der vorliegenden Analyse gezeigt werden sollte, war lediglich die Variationsvielfalt des Themas "Variablentausch ohne Hilfsvariable".

Man überlege sich,
wie der Variablentausch nach dem hier beschriebenen Muster bei der Operation "x hoch y", vorsichtshalber nur auf der Menge der positiven rellen Zahlen, aussehen müsste. (Das würde keiner machen, es rundet aber die Analyse ab.) Braucht man die Assoziativität [a(bc) = (ab)c] beim Variablentausch wirklich? Welche Forderungen sind unabdingbar?

Eine erste Antwort:
Man verstehe unter * die Subtraktion, in üblicher Weise von links nach rechts gelesen, auf der Menge der Ganzzahlen, der rationalen, reellen, komplexen Zahlen oder der mxn-Matrizen. Man verstehe unter *i die Addition und unter i* wieder die Subtraktion. Keine Rede von Assoziativität oder Kommutativität bezüglich der Subtraktion. Es gibt kein beidseitiges Nullelement und keine vernünftigen (beidseitigen) Inversen. Überhaupt: keine brauchbare algebraische Struktur (wenn man die Subtraktion als primäre Verknüpfung nimmt). Mit der Subtraktion sind die eben genannten Mengen lediglich Gruppoide (algebraisch abgeschlossen mit einer zweistelligen Verknüpfung) [Heu],[Ihr]. Der Variablentausch ohne Hilfsvariable funktioniert jedoch bestens(!):

```
                  ;  x = A      y = B
x *  y --> x      ;  x = A - B  y = B
x *i y --> y      ;  x = A - B  y = A
y i* x --> x      ;  x =     B  y = A
```

Es ginge sogar wie folgt:

```
                  ;  y = B      x = A
x *  y --> y      ;  y = A - B  x = A
x *  y --> x      ;  y = A - B  x = B
x *i y --> y      ;  y = A      x = B
```

Das geht jedoch nur deshalb,
weil die Subtraktion eben doch ein paar ganz handfeste Eigenschaften hat, die den Variablentausch nach dem eben genannten

Muster ermöglichen. Man müsste untersuchen, welche genau das sind (ich will das hier nicht machen), und die sich so ergebende algebraische Struktur mit einem eigenen Namen belegen! (Was auffällt, ist die starke Symmetrie der obigen Formeln.)

Eine Quasigruppe
wäre andererseits eine in der Literatur schon bekannte [Ihr] Struktur, mit der man für den Variablentausch ohne Hilfsvariable auskäme. Das ist eine Menge mit drei Verknüpfungen, die den folgenden Forderungen (Axiomen) genügen:

```
(Q1)  x \ (x * y) = y
(Q2)  (x * y) / y = x
(Q3)  x * (x \ y) = y
(Q4)  (x / y) * y = x
```

Und selbst hier wäre noch "die Hälfte" der Struktur überflüssig. Wir haben das oben bei dem Beispiel mit den regulären nxn-Matrizen gesehen. Da hatte es sich angeboten, **`*i`** und **`i*`** zu schreiben, da eine links- oder rechtsseitige Matrixdivision kaum anders definiert werden kann als über inverse Matrizen. An sich kommt es aber bei unserem Variablentausch nur auf eine links- und eine rechtsinverse Verknüpfungsoperation an, auf ein Rückgängigmachen der Ausgangsverküpfung **`*`** von links und von rechts her. Man sieht, man kommt dazu mit (Q1) und (Q2) allein schon aus (**`/ = *i und \ = i*`**).

Literatur

[Agu1] Aguilar, Hugh: Ein Programm zum Knacken von polyalphabetischen Codes, Teil1 (Übers. Fred Behringer). Vierte Dimension 1/2000, S. 22-29.

[Agu2] Aguilar, Hugh: Ein Programm zum Knacken von polyalphabetischen Codes, Teil2 (Übers. Fred Behringer). Vierte Dimension 2/2000, S. 7-11.

[Beh1] Behringer, Fred: Real-Mode-32-Bit-Erweiterung für Turbo-Forth. Vierte Dimension 2/1998, S. 10-15.

[Beh2] Behringer, Fred: Der verschlüsselte Schlüssel oder X S 2XOR. Vierte Dimension 2/2000, S. 24-25.

[Beh3] Behringer, Fred: Des Rätsels Lösung. Vierte Dimension 4/2000, S. 16-17.

[Dix1] Dixon, Glenn: Reed-Solomon-Fehlerkorrektur, Teil 1 (Übers. Fred Behringer). Vierte Dimension 4/1999, S.13-16.

[Dix2] Dixon, Glenn: Reed-Solomon-Fehlerkorrektur, Teil 2 (Übers. Fred Behringer). Vierte Dimension 1/2000, S.34-36.

[Heu] Heuser, Harro und Wolf, Hellmuth: Algebra, Funktionalanalysis und Codierung. Verlag B.G. Teubner, 1986

[Ihr] Ihringer, Thomas: Allgemeine Algebra. Verlag B.G. Teubner, Stuttgart 1988.

[Kli] Klingelnberg, Arndt: Preisrätsel aus VDI-Nachrichten-Magazin 1/1994, S.30. Vierte Dimension 1/1994, S. 8.

[Nob] Noble, Julian V.: Scientific Forth. Mechum Banks Publishing, Charlottesville, USA (1992).

[Pil] Pilot, M.: Leserbrief. Vierte Dimension 4/1999.

[Pri] Prinz, Friederich: Ringtausch ausgeschlossen. Vierte Dimension 1/2004, S. 17.

[Wae] Waerden, B.L. van der: Algebra, Teil 1. Springer-Verlag 1966.

Ushi zieht einen Strich

Willem Ouwerkerk

<w.ouwerkerk@kader.hobby.nl>

Übersetzt und bearbeitet von
Fred Behringer
<behringe@ma.tum.de>

Im Folgenden wird ein Vortrag wiedergegeben, den Willem Ouwerkerk von der holländischen Fgg auf unserer kürzlichen Forth-Tagung 2004 auf Fehmarn gehalten hat. Sein Vortrag wurde von einer Vorführung begleitet, die von den Zuhörern mit Aufmerksamkeit verfolgt wurde. Der hier wiedergegebene Text ist eine Übersetzung des ursprünglich im Vijgeblaadje-Heft 43 veröffentlichten Artikels "Ushi trekt een streep". In die Übersetzung wurden einige mit dem Autor abgesprochene Änderungen eingearbeitet. Wir freuen uns, dass die internationale Zusammenarbeit (Fgg/FG) inzwischen soweit gediehen ist, dass wir auf den Tagungen "mir nichts, dir nichts" gegenseitige "Arbeitsbesuche" unternehmen.

Einleitung

In Ushis noch zarter Entwicklung wird hier ein weiterer Schritt vorwärts getan. Der Roboter lernt nämlich zeichnen und schreiben. Hardware und Software dafür werden beschrieben. Auf beiden Hinterrädern wurden reflektierende Scheiben mit einer Anzahl von Streifen darauf montiert. Diese Streifen werden von einem mit einer Brille bewaffneten Reflexionssensor gelesen. Ein paar Teilsysteme von Ushi setzen die Signale in zwei Ströme von elektrischen Impulsen um. Extra-Software zählt die Impulse und passt bei Bedarf die Geschwindigkeit der Motoren über einen Regelkreis an.

Die Software-Anpassungen

Um eine gerade Linie oder einen Bogen zu zeichnen, muss Ushi seine Räder exakt in der gewünschten Spur halten. Das Geschwindigkeitsverhältnis der beiden Motoren muss dann immer

gleich bleiben. Welche Anpassungen sind dazu nötig?

1. Jeder Motor kriegt eine eigene Variable, die die Geschwindigkeit des Motors festhält.
2. Die Interrupt-Routine, die die Motoren steuert, wird über diese Variable ausgeführt.
3. Die Reflexionssensoren tasten die Streifen auf den Rädern ab.
4. Mit diesen Sensordaten bestimmen wir den Unterschied des zurückgelegten Weges der beiden Räder. Das Ergebnis wird in der Unterschiedsvariablen aufbewahrt.
5. Es gibt eine Korrigier-Routine, die in festen Intervallen aufgerufen wird und nach Maßgabe der Unterschiedsvariablen die Geschwindigkeit der Räder anpasst.

Wie arbeitet die Regelung?

Die Impulse des linken Rades erhöhen die Unterschiedsvariable, die vom rechten Rad erniedrigen sie. Wenn Ushi schnurstracks geradeaus läuft, hält sich der Unterschied nahe bei null. Das ist jedoch selten der Fall. Das Korrigieren läuft wie folgt:

Unterschied negativ?	Das rechte Rad geht vor. Geschwindigkeit links = Geschwindigkeit + 2 * (ABS von Unterschied).
Unterschied positiv?	Das linke Rad geht vor. Geschwindigkeit rechts = Geschwindigkeit + 2 * (ABS von Unterschied).

Bögen (Kreise) erzeugen

Wenn beide Räder die Unterschiedsvariable um dieselbe Zahl (z.B. 1) vermindern oder erhöhen, lässt diese Regelung Ushi geradeaus fahren. Sobald man aber beispielsweise die vom rechten Rad (in gleichen Zeitabständen) gezählten Impuls erst mit dem konstanten Wert 2 malnimmt, ehe man sie auf die Unterschiedsvariable aufspielt, dreht sich das linke Rad bei dieser Ausgleichsregelung (der Unterschiedsvariablen auf 0) zweimal so schnell wie das rechte. Ushi fährt dann einen perfekten Rechtsbogen. (Jede andere Konstante, abhängig vom gewünschten Kreisradius, tut es auch.) Die praktische Erfahrung hat gezeigt, dass die Zählwerte von 1 bis 5 gehen dürfen. Außerhalb dieses Bereiches behält Ushi die Unterschiedsvariable nicht mehr im Griff. Es entsteht ein Überlauf und dadurch ein Regelungsfehler. Durch weitere Feinabstimmung kann das vielleicht noch ein bisschen verbessert werden.

Software-Aufbau

Die Demonstrations-Software läuft auf einem AT90S2313 und verwendet, wie bekannt, einen Hardware- und Timer-Interrupt für den RC5-Decoder. Die Motorregelung verwendet den zweiten Timer-Interrupt. Dieser führt gleichzeitig einige Timer-Variablen mit sich. Da der Hardware-Interrupt vom RC5-Decoder belegt ist, wird der im Vijgeblaadje 32 beschriebene Multitasker eingespannt.
Unter diesem Multitasker laufen zwei Tasks, zum einen zur Behandlung der RC5-Steuerung, zum anderen zum Zählen der Impulse für die Unterschiedsvariable, deren Inhalt über die Leuchtdioden angezeigt wird.
Im zweiten Timer-Interrupt wird die hierfür beschriebene Korrigier-Routine so eingehängt, dass sie zehnmal pro Sekunde aufgerufen wird. Die Motorgeschwindigkeit wird so bei Bedarf zehnmal pro Sekunde angepasst.

Beispielscode

```
REGISTER UNTERSCHIED
   ( zwischen den beiden zurueckgelegten
     Radwegen )
REGISTER TEMPO
   ( gewuenschte Geschwindigkeit )
REGISTER R-TEMPO
   ( Geschwindigkeit rechtes Rad )
REGISTER L-TEMPO
   ( Geschwindigkeit linkes  Rad )
```

RECHTS? und **LINKS?** sind keine einfachen Bitflags, sondern Routinen, die dafür sorgen, dass jeder Impuls nur ein einziges Mal gezählt wird.

```
: ZAEHL-IMPULSE      ( -- )
    BEGIN
      RECHTS? IF  -1 +TO UNTERSCHIED  THEN
      LINKS?  IF   1 +TO UNTERSCHIED  THEN
      UNTERSCHIED TO LEDS
      PAUSE        (Taskwechsel )
    AGAIN ;
```

Die Korrigier-Routine ist nur 28 Bytes lang und wird zehnmal pro Sekunde aufgerufen.

```
CODE KORRIGIER       ( -- )
    R16 ADR UNTERSCHIED MOV,
    R16 TST,
    ZLE IF,           ( rechts vor )
       R16 NEG,       ( mach positiv )
       R16 LSL,       ( * 2 )
       R16 ADR TEMPO ADD,  ( + TEMPO )
       ADR L-TEMPO R16 MOV,
                      ( links kraeftiger )
        ADR R-TEMPO ADR TEMPO MOV,
    ELSE,
       R16 LSL,             ( 2 * )
       R16 ADR TEMPO ADD,   ( + TEMPO )
       ADR R-TEMPO R16 MOV,
                      ( rechts kraeftiger )
        ADR L-TEMPO ADR TEMPO MOV,
    THEN,
    RET,      END-CODE
```

```
CODE SERVO-INT       ( -- )
    ...
\ Ruf zehnmal pro Sekunde KORRIGIER auf
    ' KORRIGIER GCALL,
    ...
    RETI,  END-CODE  T1-OVERFLOW
```

Finale

Mithilfe einer weiteren Variablen können wir Ushi Linien und Bögen von vorgegebener Länge zeichnen lassen. Damit versetzen wir Ushi dann in die Lage, genau zu zeichnen.

```
VARIABLE LAENGE
  ( Laenge des zu ziehenden Strichs oder
    Bogens )

\ Vermindere die Laenge, bis sie null ist.
: ZAEHL-AB      ( -- )
    FROM LAENGE IF  DECR LAENGE  THEN ;
```

Warte, bis **LAENGE** null ist. **STOP** dann die Motoren. **PAUSE** sorgt dafür, dass das auch in einer Multitasking-Umgebung korrekt funktioniert.

```
: WARTE         ( -- )
    BEGIN  PAUSE  FROM LAENGE 0= UNTIL
    STOP ;

\ Folge den Impulsen geradeaus weiter,
\ d.h., ziehe einen Strich!

: STRICH        ( u -- )
  TO LAENGE WEITER WARTE ;
```

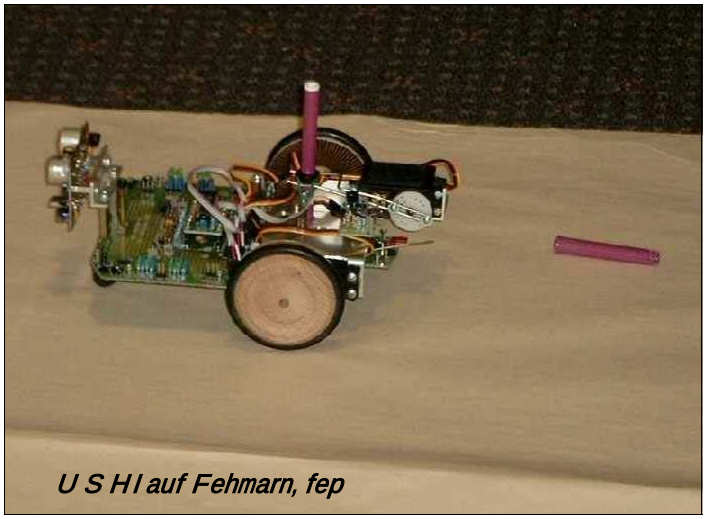
Das Wort **ZAEHL-AB** wird an entsprechender Stelle in **ZAEHL-IMPULSE** eingefügt. **WARTE** lässt die eingestellte Bewegung solange andauern, bis **LAENGE** null ist.

Anmerkungen des Übersetzers

Es wurden hier einige Forth-Worte verwendet, deren Erklärung im vorliegenden Zusammenhang kaum etwas zur Sache beitragen. Man findet sie in den Handbüchern von ByteForth und Ushi. Diese wiederum sind leicht auf der Homepage der niederländischen Forth-Vereinigung Fgg zu finden. Und die niederländische Homepage findet man über ein Link auf unserer eigenen FG-Homepage. Insbesondere haben sich in den Arbeiten unserer niederländischen Forth-Freunde folgende Entsprechungen eingebürgert:

```
FROM VAR  = VAR @
TO VAR    = VAR !
+TO VAR   = VAR +!
INCR VAR  = 1 VAR +!
DECR VAR  = -1 VAR +!
CLEAR VAR = 0 VAR !
PUSH VAR  = VAR @ >R
POP VAR   = R> VAR !
```

Das heißt also insbesondere, dass gewöhnliche **VARIABLE**n nicht nur per **@** und **!** aufgerufen werden können, sondern auch über Präfixe, ähnlich wie die mit dem ANS-Forth-Wort **VALUE** erzeugten Konstruktionen.



U S HI auf Fehmarn, fep

Tingel-Tangel

Albert Nijhof

<a.nijhof@kader.hobby.nl>

Vortrag, gehalten auf der FG-Forth-Tagung 2004.
Übersetzt von
Fred Behringer

1993 kam Marcel Hendrix auf die Idee, für die HHC-Tage (die alljährliche Computer-Börse im November in Utrecht) ein mit Forth gesteuertes Orchester zu bauen. Wir sind auf zweieinhalb Orchestermitglieder gekommen: Die zwei Tingel-Tangels und eine einsaitige Gitarre. Die Gitarre ist nie fertig geworden.

Die Tingel-Tangels, eine Art Xylofone mit kleinen Hammern, aber mit Rohrstückchen aus Aluminium und Kupfer anstelle von Holzplättchen, sind je ungefähr einen Meter lang und einen halben Meter breit und produzieren einen durchdringenden glockenspielartigen Klang.

Es wurde von verschiedenen Mitgliedern der Forthgebruikersgroep daran gebaut, und das war ein ganz schönes Stück Arbeit. Selbst die Rohrstückchen wurden von uns selbst

auf die richtige Länge gesägt (abgezwickt). Die Tingel-Tangels müssen noch stets weitergepflegt werden und sind gegenüber der Temperatur und dem Feuchtigkeitsgrad der Umgebung sehr empfindlich. Sie waren auf der (ersten) Vorstellung auf Fehmarn noch etwas nervös und hatten überdies mit einer von der Reise von Arnhem nach Burg verursachten Umstellung zu kämpfen, aber allmählich ging es dann besser.

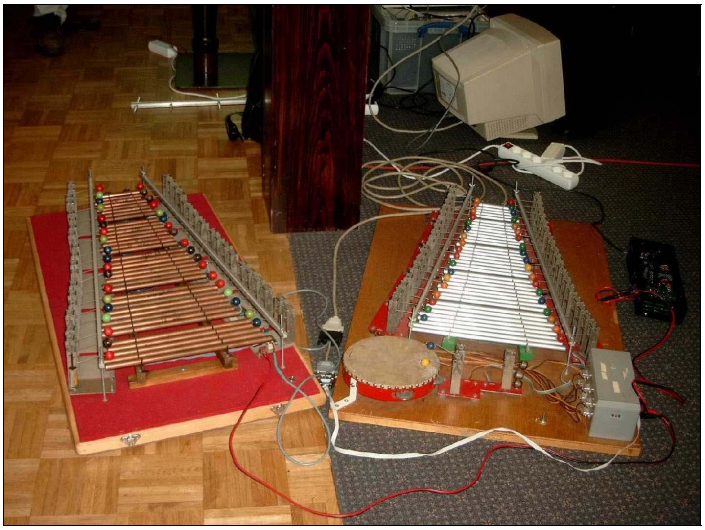
Auf dem Foto ist das kupferne Tingel-Tangel zu sehen. Das Aluminium-Tingel-Tangel hat außer den Rohrstückchen noch ein bisschen Extra-Schlagwerk an Bord. Das Foto wurde im November 2003 anlässlich des 10-jährigen Tingel-Tangel-Jubiläums auf dem Treffen im Hause von Marcel gemacht.

Die Software "MANX" stammt von Marcel Hendrix, mit Ratschlägen von Albert Nijhof für die musikalische Eingabesprache. Ausführliche Informationen darüber findet man auf Marcels Website.

http://home.iae.nl/users/mhx/manxgeneral.html
http://home.iae.nl/users/mhx/manxhlp.html
http://home.iae.nl/users/mhx/midi.html

Zur Vorführung wurde eine abgespeckte MANX-Version (von Albert van der Horst) verwendet, die ausschließlich für die Tingel-Tangels gedacht ist.

Anmerkung des Übersetzers: Das "carillonachtig geluid" (aus dem Original), das wirklich sehr "doordringend" war, hat uns Zuschauer und Zuhörer über alle Maßen erfreut. Nicht nur während des Vortrags, sondern auch und ganz besonders am Abend der Festveranstaltung, als die Tingel-Tangels sich lautstark Gehör verschafften und der "Vollen Spielmannswucht" (Drei-Mann-Bänkelsänger-Band) ernsthaft Konkurrenz machten. Danke, Albert. Wir freuen uns wirklich, dass das Verhältnis zu unseren Forth-Freunden aus den Niederlanden derart herzlich geworden ist.



Musikspaß mit dem Tingeltangel, Photo: fep

Anschluss eines Mikroprozessors ans Ethernet

Hans Eckes

hanseckes@addcom.de

Mit dem allmählichen Aussterben von Comport und paralleler Druckerschnittstelle steht der Hobbyelektroniker vor der Frage, wie er auch weiterhin eigene Hardware ohne allzuviel Aufwand an einen üblichen PC anschließen kann.

Dieser Artikel beschreibt, wie viel bzw. wenig Aufwand nötig ist, einen Mikroprozessor mit 10/100 MBit und TCP/IP-Protokoll ans Ethernet anzuschließen.

Das Ansprechen eines Ethernetcontrollers mit anschließendem Programmieren der üblichen Protokolle wäre sicherlich eine reichlich masochistische Herangehensweise an dieses Problem. Im Zuge der „Kühlschrank ans Internet“-Bewegung sind mittlerweile Bauteile erhältlich, die sich selbstständig um die Datenübertragung kümmern. Der Programmieraufwand beschränkt sich dann auf die Initialisierung und den Tansport der Daten.

Im Weiteren wird genau dieser Weg beschrieben.

Was ist eigentlich Ethernet?

Die Geburt des Ethernets war 1973. 1986 wurde der 10Base2-Standard veröffentlicht (10 Mbit im Koaxkabel), 1991 der 10BaseT-Standard (10 MBit im Twisted-Pair-Kabel). Seit 1995 gibt es den 100BaseX-Standard mit 100 MBit, seit 1998 ist man beim Gigabit-Ethernet-Standard angekommen.

Aus elektrischer Sicht ist Ethernet lediglich eine serielle Datenübertragung. Die Geschwindigkeit liegt bei z.B. 100 MBit allerdings etwa den Faktor Tausend über einer Comportverbindung. Die Daten werden auch nicht kontinuierlich in einem Strom übertragen, sondern in einzelne Pakete aufgeteilt, die dann der Reihe nach gesendet werden.

Die Datenpakete sind folgendermaßen aufgebaut: Zuerst kommt der Header, das sind 8 Bytes lang eine Folge von Nullen und Einsen, dann 6 Bytes die MAC-Adresse des Empfängers, 6 Bytes die MAC-Adresse des Absenders, 2 Bytes Typinformation für das verwendete Protokoll, dann kommt das Datenfeld mit bis zu 1518 Bytes Daten und zum Schluss 4 Bytes Checksumme.

Bei Verwendung des TCP/IP-Protokolls besteht dann das Datenfeld aus den Headern des **I**nternet-**P**rotokolls und des

Transmission **C**ontrol **P**rotokolls und schließlich und endlich auch den Daten, die man übertragen will.

Das TCP/IP-Protokoll kümmert sich um eine fehlerfreie Datenübertragung. Die Details dazu beschäftigen ganze Bücher. In diesem Rahmen soll folgende Information ausreichen:
Die Datenpakete werden so oft gesendet, bis sie fehlerfrei beim Empfänger angekommen sind. Wurden die Daten auf mehrere Datenpakete aufgeteilt, so werden sie vom Empfänger wieder in der richtigen Reihenfolge zusammengesetzt. Der W3100A kann das alles selbstständig erledigen.

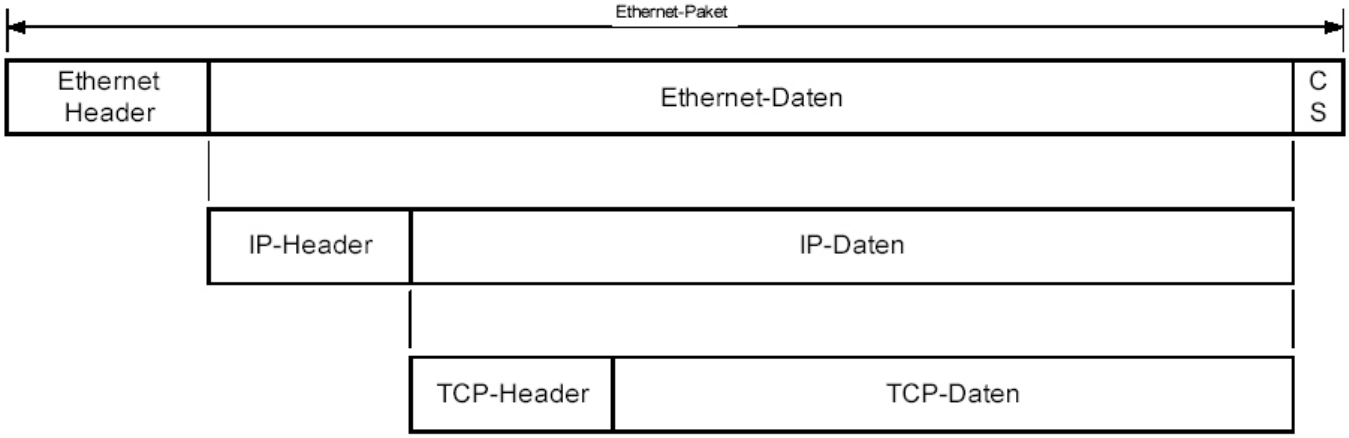


Abbildung 1: Geschachtelte Protokolle

Wieviel Elektronik wird benötigt?

Die vorgestellte Lösung arbeitet mit den heute üblichen 10/100 MBit auf einer twisted-pair-Leitung mit RJ45-Steckverbindern. Die Verarbeitung der Protokolle wird dem Baustein W3100A von WIZnet überlassen. Der W3100A beherrscht TCP, IP Ver.4, UDP, ICMP und ARP.

Der W3100A enthält 16 KB dual ported RAM, das sich auf bis zu vier voneinander unabhängigen Verbindungen im full-duplex-Mode aufteilt.

Richtung Mikroprozessor gehen 15 Adressleitungen, 8 Datenleitungen, je eine Chipselect-, Schreib- und Leseleitung, ein Reseteingang und ein Interruptausgang.

Richtung Ethernet wird noch ein sog. PHY benötigt. Dieser PHY kümmert sich um die Aufbereitung der Daten von und zum Ethernet und von und zum Ethernetcontroller, in diesem Fall der W3100A. Es werden z.B. die mit 100 MBit ankommenden Signale auf vier Signale zu je 25 MBit aufgeteilt, beim Senden geschieht dies in die andere Richtung. Ebenso handelt der PHY mit der Gegenstelle die Übertragungsgeschwindigkeit aus und ob Duplexbetrieb möglich ist.

Nach dem PHY kommt nur noch eine galvanische Trennung in Form von Übertragerdrosseln und schließlich die RJ45-Buchse. Abbildung 2 zeigt die gesamte Elektronik, die zum Anschluss eines Mikroprozessors ans Ethernet benötigt wird. Wie man sieht, genügen zwei ICs und etwa 20 passive Bauteile.

Die Versorgungsspannung beträgt 3,3 Volt. Der W3100A, der ja die Schnittstelle Richtung Mikroprozessor darstellt, akzeptiert auch 5-Volt-Pegel an seinen Eingängen. Die Buchse für das Ethernetkabel hat die galvanische Trennung bereits eingebaut. Somit entfällt ein zusätzlicher Überträger.

Wie wird der W3100A initialisiert?

Ein Gerät in einem Netzwerk muss einige Voraussetzungen erfüllen.

- Eine sog. MAC-Adresse (Media Access Control Adresse) wird benötigt. Eine MAC-Adresse ist 6 Bytes lang und weltweit jeweils nur einmal vergeben und identifiziert damit den Absender eines Datenpakets. Dies gilt natürlich auch für Einzelstücke eines Hobbyelektronikers.

- Eine IP-Adresse identifiziert das Gerät für das Internet Protokoll. Eine IP-Adresse ist 4 Bytes lang und beginnt bei lokalen Netzwerken typischerweise mit 192.168.xxx.yyy. Xxx und yyy sind jeweils 1 Byte lang und können damit Werte von 0 bis 255 annehmen.

- Die Subnetz-Maske teilt dem Gerät mit, zu welchem Netzwerk es gehört. Große Netzwerke bestehen üblicherweise aus mehreren Subnetzen. Die Subnetze sind dann über Router miteinander verbunden. Alle Bits, die Eins sind, kennzeichnen das Subnetz, die übrigen Bits kennzeichnen die Nummer des Geräts im Subnetz. Die übliche Subnetzmaske ist 255.255.255.0.

- Das Gateway kümmert sich um den Weitertransport der Datenpakete im Netzwerk. Mit der Gateway-Adresse teilt man dem Gerät mit, wo sich das Gateway befndet. Für den PC

erfährt man die IP-Adresse des Gateways durch Eingeben von **ipconfig** am DOS-Prompt.

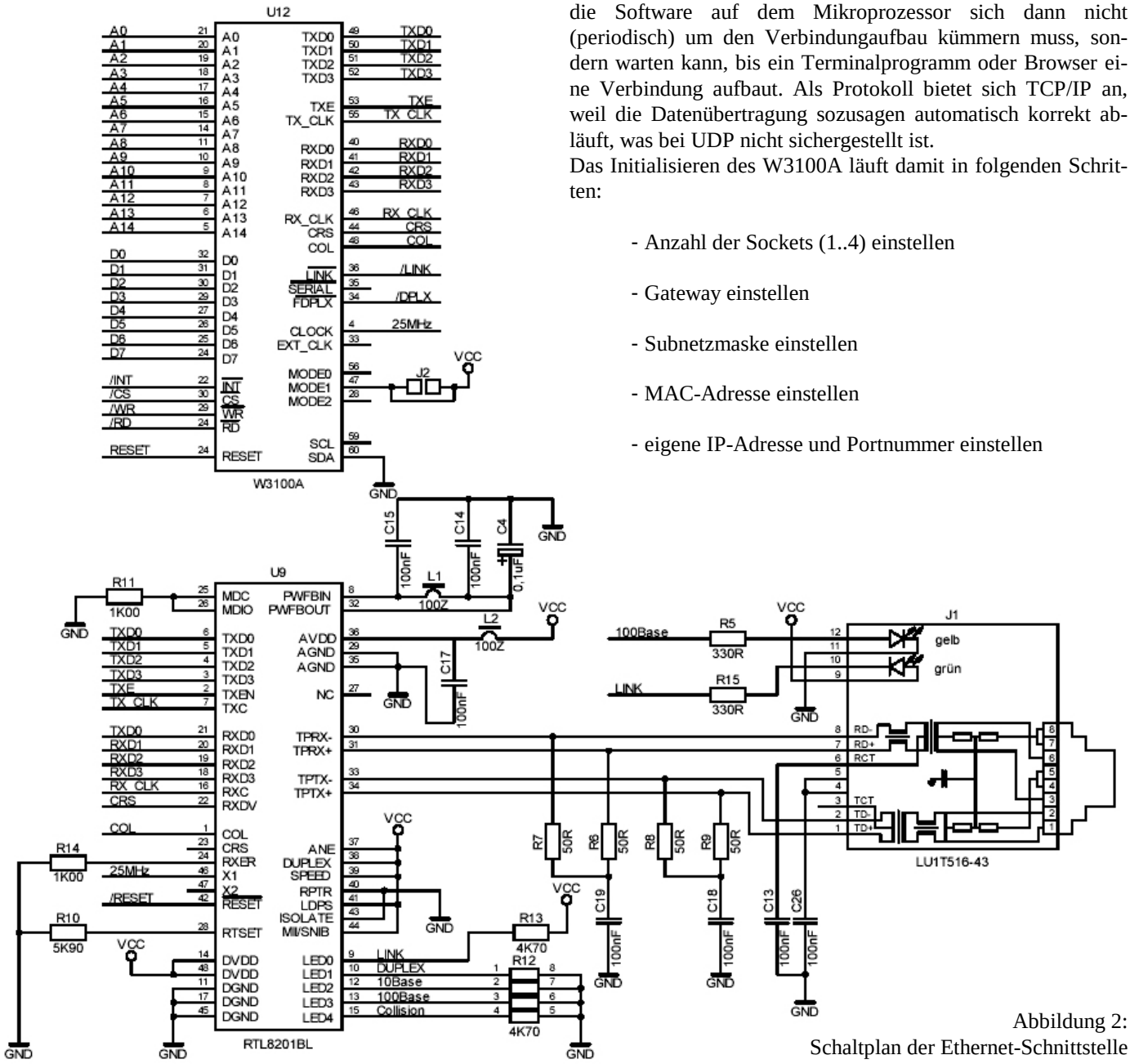


Abbildung 2:
Schaltplan der Ethernet-Schnittstelle

Am einfachsten gibt man am DOS-Prompt **ipconfig** ein und verwendet die angezeigten Adressen von Gateway und Subnetz-Maske. Die IP-Adresse, die man für den W3100A wählt, muss sich im gleichen Subnetz befinden wie der PC und darf nicht bereits anderweitig in Verwendung sein.

Beim Initialisieren des W3100A muss man nicht nur MAC-Adresse und die Netzwerkadressen mitteilen, sondern auch wie der W3100A arbeiten soll. Zur Auswahl steht, ob die Datenübertragung per UDP oder TCP/IP stattfinden soll, ob 1, 2 oder 4 Sockets gleichzeitig geöffnet werden können und ob der Sokket als Client oder Server arbeiten soll.

Arbeitet der W3100A als Server, so muss er warten, bis ein Client eine Verbindung aufbaut. Dies ist eine gute Lösung, weil die Software auf dem Mikroprozessor sich dann nicht (periodisch) um den Verbindungaufbau kümmern muss, sondern warten kann, bis ein Terminalprogramm oder Browser eine Verbindung aufbaut. Als Protokoll bietet sich TCP/IP an, weil die Datenübertragung sozusagen automatisch korrekt abläuft, was bei UDP nicht sichergestellt ist.
Das Initialisieren des W3100A läuft damit in folgenden Schritten:

- Anzahl der Sockets (1..4) einstellen
- Gateway einstellen
- Subnetzmaske einstellen
- MAC-Adresse einstellen
- eigene IP-Adresse und Portnummer einstellen

Wie kann eine Verbindung zum W3100A aufgebaut werden?

Alle Verbindungen laufen über sog. Sockets, d.h. zum Senden oder Empfangen von Daten benötigt der Prozessor erst einmal einen Socket.

Die Software fordert also einen der vier Sockets an und sagt gleichzeitig, welcher Port verwendet werden soll.

Steht der Socket dann zur Verfügung, geht das Kommando zum Warten auf einen Verbindungsaufbau an den W3100A.

Damit sind alle Voraussetzungen zum Verbindungsaufbau erfüllt. Der W3100A wartet jetzt, bis sich z.B. ein Terminalprogramm meldet. Die Gegenseite muss jetzt nur noch IP-Adresse und Portnummer wissen, dann kann sie eine Verbindung aufbauen.

Wie werden Daten gesendet und empfangen?

Der W3100A verfügt über 16KB internes, dual-ported RAM. Diese 16 KB Puffer können damit vom W3100A und auch vom angeschlossenen Prozessor benutzt werden.

Der Ablauf beim Senden ist damit: Man prüft bzw. wartet, bis genügend Platz im Puffer ist, füllt die Daten dann hinein, sagt dem W3100A, wo sie stehen und wieviel Bytes es sind, und gibt dann das Kommando zum Senden.

Beim Empfangen läuft es ähnlich: der W3100A löst einen Interrupt aus, sobald die Daten zur Verfügung stehen. Die Software liest daraufhin aus, wo die Daten stehen und wieviele Bytes es sind, und holt sich dann die Daten.

Wie geht's in Forth?

Der folgende Quelltext ist nicht ganz vollständig, sondern stellt „nur“ den Teil dar, den die übrige Applikation (bzw. der Interpreter) zur Kommunikation verwendet. Die nicht erklärten Wörter greifen hauptsächlich auf die einzelnen Register im W3100A zu und sind hoffentlich durch die Namensgebung, bzw. den Kommentar, selbsterklärend.

```
: INIT.W3100A ( -- flag )
INIT.W3100A-INTERRUPT
            \ Interrupt für W3100A starten
0 CHANNEL ! \ Socket 0 verwenden
80 C0 CR ETH-C!
            \ S/W-Reset Kommando geben
55 55 INIT.BUFFER
            \ 4 Channels zu je 2K Größe
GATEWAY-IP @ SET.GATEWAY
            \ Gateway einstellen
SUBNET @ SET.SUBNET
            \ Subnetz einstellen
MAC-ADDRESS 2@ SET.MAC-ADDRESS
            \ MAC-Adresse einstellen
SOURCE-IP @ SET.SOURCE-IP-ADDRESS
            \ eigene IP-Adresse einstellen
7D0 SET.RETRY-TIMEOUT 5 SET.RETRY-COUNT
1 C0 CR ETH-C!
           \ Sys-Init Kommando geben
0 I-STATUS !
           \ Status-Bits zurücksetzen
@g15
  \ g15 zählt im Millisekundentakt abwärts
  BEGIN
    DUP @g15 - ABS 50 >
               \ warte 50 ms lang
    I-STATUS @ #INIT OK AND 0<> OR
               \ auf das Bit #INIT OK
  UNTIL
DROP           \ ist das #INIT OK gekommen,
I-STATUS @ 1 AND 0<>
       \ dann ist der W3100A initialisiert
;

: GET.SOCKET
        ( protocol port socket -- flag )
\ Anwendung z.B.: #TCP 5000 0 GET.SOCKET
3 AND CHANNEL ! \ Socketnummer einstellen
SWAP !SOPR      \ Protokoll an den W3100A
                \ übertragen
FFFF AND SET.SOURCE-PORT
                \ den eigenen Port in den
                \ Socket schreiben
0 I-STATUS !    \ Interrupt-Status für
                \ diesen Socket zurück-
                \ setzen
2 CHANNEL @ ETH-C! \ Sock Init-Kommando
                   \ geben
@g15
  BEGIN
    DUP @g15 - ABS 50 > \ warte 50 ms lang
    I-STATUS @ #SINIT OK AND 0<> OR
               \ auf das Bit Socket_init_ok
  UNTIL
DROP
I-STATUS @ #SINIT OK AND 0<> \ war es ein
                \ Sock Init OK Interrupt?
IF TX-BUFFER @ \ dann initialisiere die
DUP !TX WR PTR WAIT.1,6us
               \ Pointer-Register
DUP !TX RD PTR WAIT.1,6us
!TX ACK PTR WAIT.1,6us
0 !RX RD PTR WAIT.1,6us
0 !RX WR PTR WAIT.1,6us
TRUE
ELSE
FALSE
THEN
;

: MAKE.SOCKET.LISTEN ( -- )
               \ damit läuft der W3100A im
0 I-STATUS !  \ Server Mode und wartet auf
8 GIVE.COMMAND \ eine Verbindung
;

DECIMAL

: SYNCRONIZE ( -- )
0 CHANNEL !
INIT.W3100A NOT IF CR
." INIT.W3100A failed " THEN
\ #TCP 80 0 GET.SOCKET \ Port für den
\ Explorer
```


```
#TCP 2000 0 GET.SOCKET ( flag )
       \ Port 2000 fÄur Socket 0
FLAG.SYNCRONIZED !
MAKE.SOCKET.LISTEN
;

: SYNCRONIZED? ( -- flag )
FLAG.SYNCRONIZED @ 0<>
;

: COMMUNICATE ( -- )
\ kümmert sich ums Senden und Empfangen
SYNCRONIZED? NOT IF SYNCRONIZE THEN
@SOCKET-STATUS
CASE
  #SOCK ESTABLISHED
  OF
    CALC.RECEIVED.CHARS 0<>
         \ sind Zeichen im W3100A?
        IF GET.RECEIVED.CHARS THEN
         \ dann hole sie in den Ringpuffer
    ETH-TX-PUFFER CALC.CHARS.IM.RINGPUFFER
         \ sind Zeichen im Sendepuffer?
    CALC.FREE.BUFFER-SIZE MIN
 \ davon so viele wie in den W3100A passen
   DUP 0> IF TRANSMIT.CHARS ELSE DROP THEN
 \ und schreibe sie in den W3100A
    ENDOF
    #SOCK CLOSED
    OF
      SYNCRONIZE
  ENDOF
ENDCASE
;

: ETH-EMIT ( c -- )
ETH-TX-PUFFER RINGPUFFER.VOLL?
               \ Sendepuffer voll?
IF             \ dann
  BEGIN        \ warte, bis er das nicht
    COMMUNICATE \ mehr ist
    ETH-TX-PUFFER RINGPUFFER.VOLL? NOT
  UNTIL
THEN
ETH-TX-PUFFER ADD.CHAR.RINGPUFFER DROP
\ schreibe das Zeichen in den Sendepuffer
SEND.IN.PROGRESS? NOT
IF COMMUNICATE THEN
;


: ETH-?TERMINAL ( -- flag )
COMMUNICATE
\ ist ein Zeichen im W3100A, dann hole es
ETH-RX-PUFFER RINGPUFFER.LEER? NOT
\ in den Ringpuffer
;


: ETH-KEY ( -- c )
BEGIN ETH-?TERMINAL UNTIL
ETH-RX-PUFFER GET.CHAR.RINGPUFFER
;
: >ETH ( -- )
INIT.W3100A-INTERRUPT
            \ starte den Interrupt
SYNCRONIZE   \ und hole Socket
SYNCRONIZED? \ hat es funktioniert?
IF CR ." schalte um auf Ethernet"
['] ETH-EMIT vEMIT !
['] ETH-?TERMINAL v?TERMINAL !
['] ETH-KEY vKEY !
THEN
;
```

Die nachfolgenden

... Impressionen aus Fehmarn ...

Wurden von verschiedenen Photographen aufgenommen, die allesamt den Lesern der VD viel Spaß beim Betrachten wünschen.

Tassen zur Tagung, mit dem SWAP und einem Hinweis auf Fehmarn im Jahr 2004 – kunstfertig und liebevoll hergestellt von Martin Bitter.

Ulrich Hoffmann bei der Anmeldung.

Spaziergänge am Strand und im Hafen von Burg – Kurzweil auf Fehmarn

OVER heißt der kleine „Computer aided Programmer“, den Klaus Schleisiek auf die Insel gebracht hat.

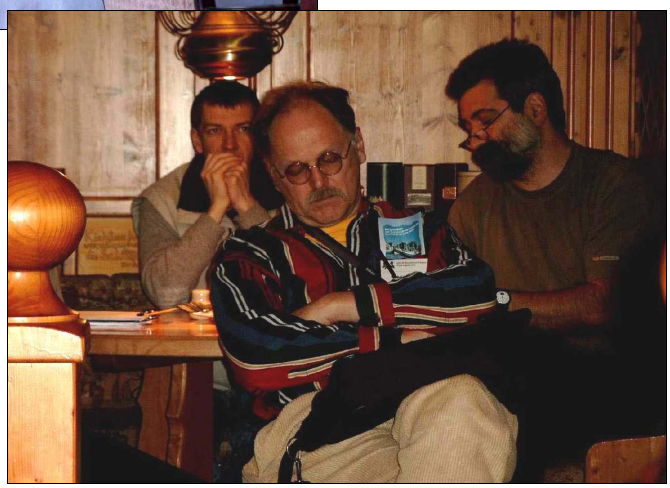
Eine Zigarettenpause vor dem Schützenhof...

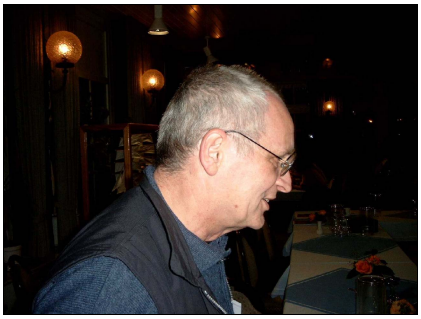
...nutzt der SWAP zu einem Flirt.

Ein ganz besonderer Drache, den Ulrike und Heinz Schnitter dem Editor der VD geschenkt haben – „Let Execute“ wird der kleine Editor-Assistent in Moers genannt.

Impressionen von Fehmarn

Aufmerksame und weniger aufmerksame Teilnehmer der Tagung und der Mitgliederversammlung werden unterschiedliche Eindrücke von Fehmarn mit nach Hause genommen haben.





Zwei der ehemaligen „ Macher“ der VD:

Rolf Kretschmar

und...

Claus Vogt.

Bänkelsang, vorgetragen von einem international agierenden Trio, das den Editor der VD Wochen später am Niederrhein (auf dem Flachsmarkt der Krefelder Wasserburg Linn) messerscharf als „einen von dem Computergeheimbund“ identifizieren konnte.

Schön war es, wie immer.

Wir freuen uns auf 2005 in Sachsen.

Forth-Gruppen regional

Moers

Friederich Prinz
Tel.: (0 28 41) - 5 83 98 (p) (Q)
(Bitte den Anrufbeantworter nutzen!)
(Besucher: Bitte anmelden!)
Treffen: 2. und 4. Samstag im Monat
14:00 Uhr, **MALZ, Donaustraße 1**
47441 Moers

Mannheim

Thomas Prinz
Tel.: (0 62 71) - 28 30 (p)
Ewald Rieger
Tel.: (0 62 39) - 92 01 85 (p)
Treffen: jeden 1. Mittwoch im Monat
Vereinslokal Segelverein Mannheim e.V.
Flugplatz Mannheim-Neuostheim

München

Jens Wilke
Tel.: (0 89) - 8 97 68 90
Treffen: jeden 4. Mittwoch im Monat
Ristorante Pizzeria Gran Sasso
Ebenauer Str. 1
80637 München

Hamburg

Küstenforth
Klaus Schleisiek
Tel.: **(0 40) - 37 50 08 03 (g)**
kschleisiek@send.de
Treffen 1 Mal im Quartal
Ort und Zeit nach Vereinbarung
(bitte erfragen)

Gruppengründungen, Kontakte

Hier könnte Ihre Adresse oder Ihre Rufnummer stehen – wenn Sie eine Forthgruppe gründen wollen.

µP-Controller Verleih

Thomas Prinz
Tel.: **(0 62 71) - 28 30** (p)
micro@forth-ev.de

Forth-Hilfe für Ratsuchende

Jörg Plewe
Tel.: **(02 08) - 49 70 68** (p)

Jörg Staben
Tel.: **(0 21 03) - 24 06 09** (p)

Karl Schroer
Tel.: **(0 28 45) - 2 89 51** (p)

Spezielle Fachgebiete

FORTHchips
(FRP 1600, RTX, Novix)

Klaus Schleisiek-Kern
Tel.: **(0 40) - 37 50 08 03** (g)

KI, Object Oriented Forth,
Sicherheitskritische
Systeme

Ulrich Hoffmann
Tel.: **(0 43 51) - 71 22 17** (p)
Fax: **- 71 22 16**

Forth-Vertrieb
vlksFORTH
ultraFORTH
RTX / FG / Super8
KK-FORTH

Ingenieurbüro **Klaus Kohl**
Tel.: **(0 82 33) - 3 05 24** (p)
Fax :**(0 82 33) - 99 71**
mailorder@forth-ev.de

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 - oder schicken Sie uns eine E-Mail!

Hinweise zu den Angaben nach den Telefonnummern:

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

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

Hinten von links nach rechts, stehend:

Friedel Amend, Arndt Klingelnberg, Bernd Paysan, Klaus Krieger, Rolf Schöne, Ulrich Hoffmann, Klaus Zobawa, Claus-Werner Vogt, Joachim Soll, Jörg Völker, Carsten Strotmann, Johannes Reilhofer, Michael Kalus, Heinz Schnitter, Elisabeth Rohrmayer, Adolf Krüger, Rolf Kretzschmar, Malte Bitter, Hans Eckes, Thomas Beierlein, Klaus Schleisiek, Thomas Prinz

Vorne hockend:

Ewald Rieger, Fred Behringer, Egmont Woitzel, Friederich Prinz, Martin Bitter, Wolfgang Allinger, Karsten Roederer