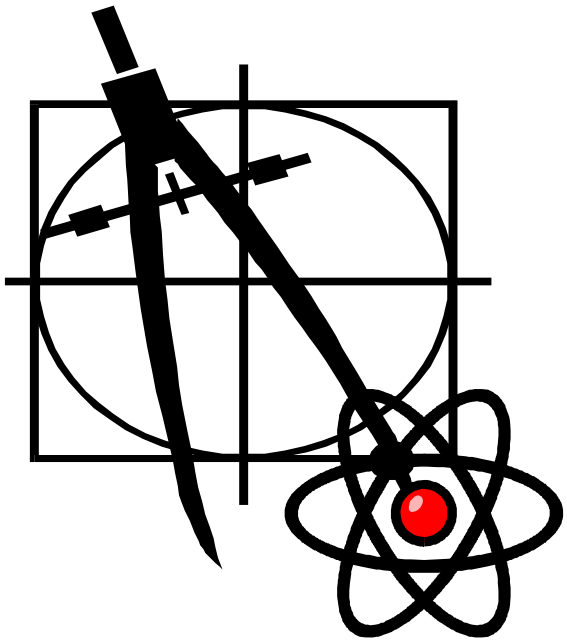
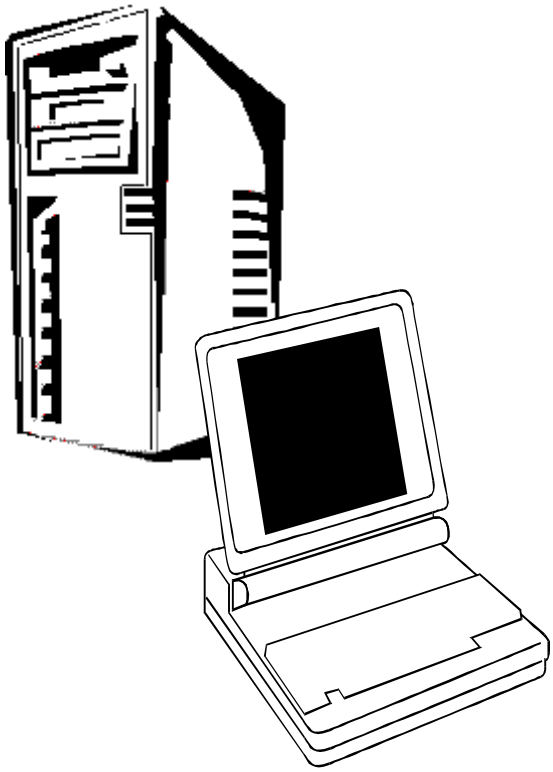
VIERTE DIMENSION

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





In dieser Ausgabe:

15. Jahrgang - **Ausg. 1/’99** - 10,- DM

Organ der FORTH-Gesellschaft e.V.

Dienstleistungen und Produkte fördernder Mitglieder des Vereins

Editorial

IMPRESSUM

Name der Zeitschrift

Vierte Dimension

Herausgeberin

Forth-Gesellschaft e.V.
Postfach 161204
D-18025 Rostock
Tel.: 0381-4007828
E-Mail:
SECRETARY@ADMIN.FORTH-EV.DE

Bankverbindung: Postbank Hamburg
BLZ 200 100 20
Kto 563 211 208

Redaktion & Layout

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

Anzeigenverwaltung

Büro der Herausgeberin

Redaktionsschluß 1999

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

Erscheinungsweise

1 Ausgabe / Quartal

Einzelpreis

DM 10,- zzgl. Porto u. Verp.

Liebe Leser,

unter der Überschrift „Fehler und Freunde“ kann sicher jeder von uns seine eigenen Anekdoten erzählen. Irgendwie passiert immer irgendetwas und das Fatale an Fettnäpfchen ist sicher, daß die Dinger nicht mit großen Warntafeln gekennzeichnet am Wegesrand stehen. Positiv ist aber, daß man aus jedem Fettnäpfchen – das man einmal ‚erwischt‘ hat – auch etwas lernt. Vermeidbar ist das Hineintreten oder das Umwerfen nur, wenn man den Weg genau kennt, oder auf die Begleitung eines Kundigen vertrauen kann, oder sich erst gar nicht bewegt. Letzteres kommt natürlich für die VD und ihr ‚Editoriat‘ überhaupt nicht in Frage. Und darum bleiben auch uns gelegentliche Fehltritte nicht erspart. So konnte es passieren, daß trotz der – wie wir meinten – eindeutigen Kennzeichnung des Übersetzers, für die Leser der VD nicht mehr klar erkenntlich war, welchen redaktionellen Ursprung der Text zu dem „AMIGA FreeForth“ (VD 4/98) tatsächlich hatte. Die Folge davon waren eine Reihe von E-Mails, die sowohl zwischen der Redaktion und mehreren Lesern hin und her gingen, als auch über de.com.lang.forth ausgetauscht werden mußten. Wir werden uns zu ähnlichen Fragestellungen zukünftig noch größere Mühe geben – versprochen.

Ein Fehler anderer Art hat sich in unsere Arbeit eingeschlichen, als wir die englischsprachigen Forther im Forum comp.lang.forth animieren wollten, sich selbst direkt an die Redaktion zu wenden, um uns Interessantes und Wissenswertes mitzuteilen. Daraufhin hat uns zunächst die Anfrage erreicht, ob das ein Hinweis auf die Isolation sein, die deutsche Forther fühlten, weil sie sich vom ‚Mainstream‘ der Forthbewegung abgekoppelt sähen. Das erschien uns wenig verständlich. Erst die private E-Mail eines freundlichen Forthers (Bruce McFarling) brachte Aufklärung. Im englischen Sprachraum gilt ‚VD‘ als Abkürzung für ‚venereal disease‘, für eine beliebige Geschlechtskrankheit. Eine Sprache reden, hören, lesen und schreiben zu können ist eine Sache – eine ganz andere Sache ist es, diese Sprache auch zu verstehen ! Hier tat der freundschaftliche Hinweis wirklich Not ;-)

Freundschaftlich entwickeln sich auch die Kontakte, die unser , Auslandskorrespondent‘ Fred Behringer für den Verein und die VD pflegt, und die weit über die in der VD sichtbare Arbeit (Rezensionen) hinausgehen. Ein reger Kontakt besteht zu Chris Jakeman, dem Editor der englischen FORTHWRITE und Alan Wenham, der unsere VD in der Forthwrite regelmäßig bespricht. Und mit Marlin Ouverson, dem Editor der Forth Dimension, steht Fred Behringer ebenso in ständiger Verbindung, wie mit Willem Ouwerwerk, der für das Vijgeblaadje unserer niederländischen Nachbarn verantwortlich ist.

Henry Vinerts Briefe aus dem FIG Chapter des Silicon Valley sind Ihnen, den Lesern der VD, natürlich ebenso bekannt, wie die Tatsache, daß Thomas Beierlein, der , freundliche Professor‘, diese Briefe für uns alle übersetzt.

Mit anderen Worten gesagt: die VD geht in das zweite ‚Moerser Jahr‘. Wir haben im ersten Jahr Anekdoten gesammelt und Freunde gewonnen – Freunde für die Forthgesellschaft, für die VD und für uns selbst. Das bringt uns neben der Arbeit auch viel Spaß, umso mehr, als wir wissen, daß wir ‚im Hintergrund‘ von vielen fleißigen und freundlichen Menschen dabei unterstützt werden, Ihnen regelmäßig eine interessante und ‚gut gefüllte‘ VD auf die Tische zu legen. In diesem Sinne wünschen wir Ihnen, dem Verein und uns selbst ein interessantes, anekdotenreiches und glückliches Jahr 1999.

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

Leserbriefe

Lieber Herr Prinz,

ich habe soeben die VD 4/98 erhalten. Ein dickes Lob Ihnen, dem Redakteur, und der Moerser Gruppe, Martin Bitter, Ulrich Richter, Michael Major usw.!
Die VD ist gut. Wie seit längerem, gut dosiert, von allem etwas, viele kurze und viele interessante Mitteilungen. Der Druck ist ausgezeichnet. Wenn man es nicht besser wüßte, würde man es nicht glauben, daß das alles ehrenamtlich und ohne große Kosten für den Verein zustande kommt.
Ich habe in letzter Zeit viele Hefte der anderen Forth-Gruppen (FIG USA, FIG UK, Forth-gebruikersgroep) intensiv (wegen der von mir anzufertigenden Rezensionen) gelesen: Die VD kann sich in diesem erlauchten Kreis ganz bestimmt sehen lassen.

Fred Behringer

Vielen Dank, Herr Behringer. Wir geben uns – selbstverständlich – große Mühe. Das gilt für das ‚Editoriat' unserer Zeitschrift ebenso wie für Thomas Beierlein, der den Druck, bzw. die Vervielfältigung organisiert. Und das Forthbüro legt sich zu Anfang des Quartals ‚richtig ins Zeug', um den Lesern die VD auf die Schreibtische zu legen.

fep

Frage an die Redaktion:
Ich habe mit Interesse die Beiträge über das freigegebene Jforth gelesen. Wie stellt man eigentlich als Programm-Autor fest, wieviele downloadende Zugriffe auf einem nicht ganz unbekannten FTP-Server (leo) oder sogar auf einem weltweit bekannten Server (taygeta) auf das Programm erfolgt sind?

Fred Behringer

Diese Frage geben wir weiter an die Leser der VD. Die Angaben in der von uns übersetzten und abgedruckten ‚Meldung' wurden mehrfach angezweifelt. Antworten bitte an die Redaktion der VD.

fep

Literatur

In Dr. Dobb´s Journal vom Oktober 1998 steht ein Artikel von William M. Stein über einen Echtzeit-Controller in einer Forth-Implementation von New Micros in einem System, das auf Motorolas MC68HC11 aufbaut.

- beh -

Michael A. Losh hat sein Forth für Java (jeForth) auf **http://www.amsystech.com/mlosh/** gelegt. Die Homepage enthält die allerersten Erklärungen für den Umgang mit jeForth. Wenn man diese Webseite aufruft, hat man sofort das auf Java aufbauende eForth-System betriebsbereit vor sich. (Natürlich nur, wenn der Browser java-fähig ist.) Den Quelltext muß man sich noch extra herunterladen (150 KB). Den braucht aber der "Normalverbraucher" gar nicht. Ich habe für Michael A. Losh eine deutsche Übersetzung dieser Homepage angefertigt. Er hat das auf eine weitere Webseite gelegt. Klickt man auf den Link "Deutsch", so kommt man dorthin. (Von dort natürlich über "English" wieder zurück) Die Adresse für den direkten Zugang zur deutschen Webseite lautet: **http://www.amsystech.com/mlosh/jeForth_Deutsch.htm**

Fred Behringer

Betreff: Was also ist dran am JForth?

Im Forth Magazin "Vierte Dimension" Nr.4, 1998, stellte Friederich Prinz im Beitrag 'JForth ist Freeware' die verwunderte Frage: "Was also ist dran am JForth?" als er gewahr wurde das viele Amiga-Leute sich dieses Forth für ihre Maschine geholt haben sollen.
Einiges daran verwundert mich auch:
Claus Vogt's email behauptete erstmal, daß am 11.August 1998 auf der "primären ftp-site" zum download des Jforth in 24 Stunden 1.649 Zugriffe erfolgt seien sollen und in den darauf folgenden 24 Stunden seien es 1.969 Zugriffe gewesen. Als 'Ersteller' wurde marrandy@tampabay.rr.com angegeben, aber es erscheint eine deutschsprachige email? Und Claus, wo hast du das eigentlich aufgegabelt?
Zu den Zahlen selbst: So eine site summiert doch gewöhnlich nicht 24-Stundenweise. Also nach 48 Stunden waren es schließlich 1.969 x downloads?
Und, lieber Friederich, in deinem Kommentar zu Martin Randall's Ankündigung der JForth Freeware sprichst du schließlich sogar von "mehr als 2.500 downloads von JForth in 2 Tagen". Ja, wie kamst du denn jetzt auf diese Zahlen?
Und du vermutetest, diese Zugriffe auf JForth seien die Folge einer "Pressemitteilung vom 10. August" gewesen. Ja was für eine Pressemitteilung denn? Printmedium oder auch email oder war es eine newsgroup? Ja wer schreibt denn in welchen Blättern sowas zu Forth und dann noch für den Amiga? Das konnte ich als Leser dem Artikel nicht mehr entnehmen - hätte mich aber brennend interessiert :-(
Könnte es sich bei der Sache auch um eine 'Ente' handeln?
Soweit meine kritischen Anmerkungen.

Was das Phänomen Forth selbst betrifft, fiel mir dazu folgende Textstelle von Alexander Mitscherlich in "Studien zur psychosomatischen Medizin I, 1969" kürzlich in die Hände :-)
(Es geht um die Ich-Identität.)
"Sicher ist eine der wichtigsten Bedingungen für das entstehen dieser Ich-Identität ... die Möglichkeit, sich in 'originalen' Leistungen ausdrücken zu können. In Leistungen also, in denen das Individuum den Erfahrungsstoff seiner Gesellschaft aufgreift, in der Darstellung aber seine 'Handschrift' wiedererkennt. Eine auf normierte Massenproduktion und Massenkonsum eingerichtete Zivilisation ist aber gerade darauf eingestellt, solche Spuren der individuellen Existenz als Schwankungen der Fertigkeit auszulöschen und die Werkstücke nach dem Muster des ersten Entwurfs zu reproduzieren. ... Der Ingenieur, der ein neues Getriebe konstruiert, ist in der Lage, dieses Modellstück als das seine zu erkennen - nicht aber die zahllosen Arbeiter, die diese Apparatur am Fließband zusammenstellen. Sie arbeiten quantitativ, aber spurlos. Das heißt, daß eine solche Tätigkeit (gesteigert nochmals in der Aktenarbeit der Bürokratie) einen zentralen Sinnbezug eingebüßt hat - sie gerinnt zu einer Zweckleistung per se, durch welche auf dem Tauschwege die Fristung des Existenzminimums und der Genußmarge angestrebt wird. Damit wird sowohl die Leistung wie die Lebensfristung wie die Genußbefriedigung 'primitiv'."
Soviel mal dazu ;-)

Grüße aus Malente, Michael (Kalus)

Inhalt

In der nächsten Ausgabe finden Sie voraussichtlich:

Beiträge zu ‚klassischem OO-Forth von Friederich Prinz & Michael Major, und Chris Jakeman

Eine Arbeit zu einem ‚Stringpaket‘ für ANS-Forth von Nils Eilers

Eine weitere „Werkstattarbeit“ zum F21 MISC von Soeren Tiedemann

Eine Arbeit über die Auslagerung des Stacks in eine Datei, von Fred Behringer

Leserbriefe

Hinweis in eigener Sache

In den Ausgaben 3/98 und 4/98 der VD war versehentlich im Impressum für das **Forthbüro** in Rostock eine falsche Telefonnummer angegeben. Die korrekte Rufnummer lautet :

0381-4007828

Unter dieser Nummer ist das Forthbüro sowohl telefonisch erreichbar, als auch per FAX.

Die Redaktion der VD ist ab sofort ebenfalls telefonisch und per FAX erreichbar, unter der Rufnummer:

02841-58398

fep

Auszug aus einer E-Mail: Antwort auf Michael Kalus‘ Anfrage auf Seite 4...

Hier der Versuch, Deine Anfrage zu beantworten ;-)

> Claus Vogt's email behauptete erstmal das am 11.August 1998 auf der "primären ftp-site" zum download des Jforth in 24 Stunden 1.649 Zugriffe erfolgt seien sollen und in den darauf folgenden 24 Stunden seien es 1.969 Zugriffe gewesen. Als 'Ersteller' marrandy@tampabay.rr.com angegeben, aber es erscheint eine deutschsprachige email? Und Claus, wo hast du das eigentlich aufgegabelt?

Da steckt der 'Fehlerteufel' in mehreren Details.

1.) Claus Vogt hat mir die originalen E-Mails gepostet, bzw. diese an mich weitergeleitet. Die Originale waren in englischer Sprache verfasst.

2.) Ich habe die Originale übersetzt, was ich m.E. auf der Seite 13, unten, links kenntlich gemacht habe "- übersetzt von Friederich Prinz". Das war offensichtlich nicht ausreichend gekennzeichnet. Sorry...

3.) 1.649 Zugriffe in den ersten 24h und 1.969 Zugriffe in den nächsten 24h sind DEUTLICH mehr als 2.500 Zugriffe in zwei Tagen.

WIE der Server diese Zugriffe gezählt hat, kann ich nicht sagen.

4.) Die 'Pressemitteilung' ist der zweite, ebenfalls von mir übersetzte Teil des Beitrages, der das JForth beschreibt. Das muß zumindest auch in elektronischer Form in einem Netz vorgelegen haben, sonst hätte Claus es nicht an mich weiterleiten können. Ob es darüber hinaus in einem Printmedium erschienen ist, weiß ich nicht.

> Könnte es sich bei der Sache auch um eine 'Ente' handeln?

Möglich ist immer alles ! Aber warum vermutest Du das ?

> Was das Phänomen Forth selbst betrifft, fiel mir dazu folgende Textstelle von Alexander Mitscherlich in "Studien zur psychosomatischen Medizin I, 1969" kürzlich in die Hände :-)

DAS soll ICH verstehen ? ;-)

fep

...und was uns jetzt DRINGEND fehlt, ist der Bericht eines AMIGA-Nutzers, der uns beschreibt, was WIRKLICH dran ist, am jForth !
Entsprechende Beiträge bitte an die Redaktion der VD !

fep

JForth ist ein native-code Forth fuer den Amiga. Es hat existiert, lange bevor Java einen Namen hatte, der mit J anfing. Nach allem, was man so lesen konnte, war's ein sehr gutes System. Einer der Autoren, Mike Haas, war frueher ein regular in comp.lang.forth (hat in jedem blocks/files-Flamewar eine Hauptrolle gespielt:-).

> > Zu den Zahlen selbst: So eine site summiert doch gewöhnlich nicht 24-Stundenweise. Also nach 48 Stunden waren es schließlich 1.969 x downloads?

Ob das jetzt 1000, 2000, oder 4000 waren, ist doch egal, es sind fuer diesen kurzen Zeitraum jedenfalls unglaublich viele.

Zum Vergleich die Zahlen fuer http://www.complang.tuwien.ac.at/forth/gforth/ seit 1.7.1998:

```
39  /forth/gforth/gforth-0.3.0.tar.gz
26  /forth/gforth/gforth-0.3.0-i586-unknown-linux.tar.gz
 6  /forth/gforth/gforth-0.3.0-binonly-i586-unknown-linux.tar.gz
 7  /forth/gforth/gforth-0.3.0-i486-unknown-linuxaout.tar.gz
 4  /forth/gforth/gforth-0.3.0-binonly-i486-unknown-linuxaout.tar.gz
15  /forth/gforth/gforth.lsm
```

Allerdings ist die letzte Release schon 1.5 Jahre her, es sind nur die http-Zugriffe gezaehlt (nicht ftp), und die meisten Leute werden's wohl ueber einen GNU-Spiegel kriegen; und bei der Suse ist's auch dabei.

Anton Ertl

Übersetzung einer E-Mail von Jim Schneider
im Forum comp\lang\forth

Jim Schneider Enterprises (http://www.find-a-bargain.com)

Ich habe vor, einige Monate an meinem Assembler zu arbeiten. Und ich beabsichtige, eine Version zu Anfang des (nächsten) Jahres herauszugeben.
Für diejenigen von Ihnen, die mich nicht kennen, oder nicht wissen, wovon ich rede: Ich habe den Assembler geschrieben, den Tom Zimmer und Andrew McKewan zusammen mit dem WIN32FOR verteilt haben (Tom, nutzt Du den noch ?). Es ist schon eine Weile her, daß ich daran gearbeitet habe.

Folgende Änderungen/Erweiterung habe ich jetzt vor:
1) MMX, PPro, und PII Instruktionen hinzufügen.
2) Alle systemabhängigen Codes sollen in eine separate Datei.
3) Hinzufügen von Code, der Cross-Assemblierung und mehrfaches Laden unterstützt.
4) Bereinigen von Fehlern..

Wenn Sie irgendwelche Anregungen haben, besonders zu den Punkten 2 und 4, lassen Sie mich das bitte wissen:

Jschneid@fire.he.net

Übersetzung eines Beitrages von Jerry Avins
Im Forum com\lang\forth

The Hungry Computer (A true story)

1987, als ich eine 16-bit Erweiterung für mein Z80-basiertes Steuersystem brauchte, erwarb ich mehrere Gespac 68000 Karten. An Software standen mir OS-9 von Microware und MACH 2 Forth von der Palo Alto Vertriebsgesellschaft zur Verfügung. Das war eindeutig keine Software für ein Steuerungssystem. Weil die Not alle Kosten rechtfertigte, bat ich Forth,Inc darum, mir ein cross-development System von MAC nach Gespac zu schreiben. Daraus wurde leider nichts.
MACH 2 war interessant. Es basierte größtenteils auf F83, und es war das erste Subroutinen-gefädelte Forth, mit dem ich arbeitete. Namen, Codes und Variablenspeicher waren voneinander getrennt, und es war relativ einfach in ROM zu bringen. Vorhandene Vokabulare waren FORTH, OS-9, ASSEMBLER und MATH. Es hatte ‚locals', aber Fließkommaoperatoren kannte es noch nicht. Optimierungen beruhten auf den Ersatz von CALLs durch JUMPs und durch In-Line Worte, die zu kurz waren, als das sich selbst ein Sprung dorthin gelohnt hätte. Im Vergleich zum Z90-polyForth war es unglaublich schnell.

Es war aber gar nicht so einfach, MACH 2 ans Laufen zu bekommen. Es gab irgendwo innen drin Änderungen, die nicht zu dem Code paßten, auf dem es getestet worden war. Und das hat MACH 2 aus dem Speicher geblasen.

Eines der ersten Dinge die ich zur Übung tat, um dem Mach 2 auf die Spur zu kommen, war ein DUMP auf noch nicht initialisierten Speicher. Dabei habe ich folgendes zu sehen bekommen:

HERE 800 + 80 DUMP

FEED C0DE FEED C0DE FEED C0DE FEED C0DE
FEED C0DE FEED C0DE FEED C0DE FEED C0DE
FEED C0DE FEED C0DE FEED C0DE FEED C0DE
FEED C0DE FEED C0DE FEED C0DE FEED C0DE
FEED C0DE FEED C0DE FEED C0DE FEED C0DE
FEED C0DE FEED C0DE FEED C0DE FEED C0DE
FEED C0DE FEED C0DE FEED C0DE FEED C0DE
FEED C0DE FEED C0DE FEED C0DE FEED C0DE

Als ich mich endlich ausgelacht hatte, tat ich, wie von mir verlangt wurde. Ich hätte das allerdings jemandem zeigen sollen, denn Ähnliches habe ich nie wieder zu sehen bekommen.

Jerry

Andere Forther HABEN Ähnliches gesehen, wie eine Vielzahl von Mails beweist, die auf Jerry Alvins Beitrag geantwortet haben. Eine diese Antworten soll den Lesern der VD auf keinen Fall vorenthalten bleiben...

fep

Passend zu Jerry Avins Beitrag – Ich habe vor Jahren Speicher mit der folgenden 32-Bit Zahl initialisiert: 3735928559. Ein DUMP auf diese Zahl zeigt den Text "DEAD BEEF." Es hätte natürlich ein beliebiges, anderes Muster sein können, aber dieses hier produziert eben ein sicheres Kichern. Vegetarier werden sicher andere Muster wählen.
Und JAVA Fans werden sich an eine Zeichenfolge in den magic headers ihrer Klassendateien erinnern „0xCAFEBABE".

John Passaniti

Betreff: **DBMS tools for Win32forth**
Datum: Sat, 14 Nov 1998 22:26:06 -0500
Von: John Whitt <whitt@ibm.net>
Firma: **IBM.NET**
Foren: comp.lang.forth
Ich habe eine Sammlung von Forth-Worten abgelegt, die es ermöglichen, auf dBase IV Datenbankfiles zuzugreifen und Indexe zu erstellen. Diese Worte arbeiten aktuell nur unter WIN32FOR, können aber leicht an Iforth angepaßt werden. Die benötigte Bibliothek, auf die die Worte zugreifen, liegt als DLL vor, die zusammen mit den anderen Dateien abgelegt wurde. Die Worte arbeiten unter Windows 95 oder NT.
Alle benötigten Dateien sind ab sofort von **ftp.forth.org incomming/forth** herunterladbar. Sie sind im Archiv **CODEBASE.ZIP** verpackt.
Bitte lesen die README.TXT Datei und die Hinweise in CODEBASE.F. Dort finden Sie alle notwendige Informationen zur Arbeit mit meiner Wortsammlung.

Wenn Sie zusätzliche Fragen haben, kontaktieren Sie mich bitte unter :

whitt@ibm.net

Internet Relay Chat

...was Sie zu dem Artikel auf der Seite 32 wissen sollten...

Chatten, zu deutsch etwa: schwatzen oder raatschen, ist die direkte Online-Kommunikation via Tastatur. Praktisch ‘beliebig' viele Menschen können sich in einem der vielen IRC Dienste, und dort in einem ‚beliebigen' Kanal treffen und drauf los “plaudern„. Was Sie, neben einem Internetzugang, benötigen, um mitschwatzen zu können, ist nicht viel.

Im einfachsten und bequemsten Fall besuchen Sie die Adresse **http://chat.web.de**. Wenn Ihr Browser javafähig ist, können Sie direkt von dort aus in den Dienst **IRCNET** einsteigen und dort zu den forthigen Kanälen **#forth-ev.de** oder **#figuk** ‘connecten'.

Kaum weniger bequem ist es, wenn Sie sich vor dem ersten Chat einen IRC-**Client** aus dem Netz herunterladen. Das Wenige, was Sie dann noch tun und beachten müssen, finden Sie ab der Seite 32 dieser Ausgabe beschrieben.

Wir hoffen, schon sehr bald im IRC mit Ihnen ‚plaudern' zu können.

Thomas Beierlein, Egmont Woitzel, Friederich Prinz

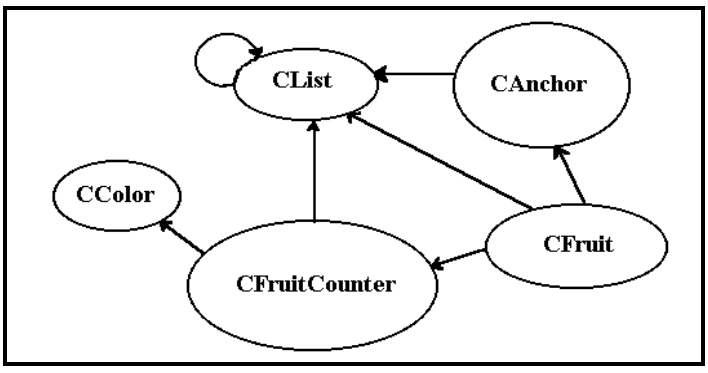
Objektorientierte Programmierung in comFORTH 4 Teil – 3 -

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

Jung, dynamisch und erfolg...

Wir hätten auf den Bauch hören sollen. Die neue Version unseres Programms fiel beim Auftraggeber glatt durch. Gelbe Äpfel und rote Birnen sind natürlich auch möglich. Dafür kann es auch mal sein, daß gar keine grünen Früchte dabei sind. Da unser Auftraggeber selbst nicht genau sagen kann, unter welchen Umständen er was braucht, ist er auf die Idee gekommen, daß sich das Programm automatisch anpassen soll, also die nötigen Zähler nach Bedarf anlegen muß. Außerdem sind ihm die Worte zur Anzeige der aktuellen Werte zu unübersichtlich. Er möchte jetzt lieber einen richtigen Report mit Zwischensummen und plausiblen Texten. Bei der Gelegenheit ist ihm auch aufgefallen, daß man den Report auch mit einem Rücksetzen der Zähler verbinden kann. Das war vorher völlig ungelöst. Schon wieder ein neues Programm? Mal sehen, was wir wiederverwenden können.

So schlecht ist unsere Struktur ja gar nicht. Unser Verfahren zur Bildung der Zwischensummen sollten wir eigentlich auch auf das Schreiben des Report anwenden können. Wir müssen nur unsere Fleißarbeit beim Aufbauen der Zählerlisten automatisieren und außerdem die Listen auch wieder löschen können. Unsere bisherige Farbsteuerung müssen wir aber wohl wegwerfen. Spätestens im Hinblick auf den Report erscheint eine eigene Klasse für die Farben sinnvoll. Diese könnte auch den Programmzustand “aktuelle Farbe” kapseln. Die Früchte könnten dann anhand dieses Zustands den richtigen Zähler in der Liste suchen und gegebenenfalls einen neuen Zähler anlegen. Die Zähler müssen als neue Eigenschaft daher ihre Farbe als Verweis auf das zugehörige Farbobjekt mitführen. Das führt dann zu einer solchen Struktur:



Dynamische Objekte

Unserer Werkzeugsammlung fehlt eigentlich nur noch ein Verfahren zum dynamischen Anlegen und Zerstören von Objekten im Heap. Man könnte natürlich ALLOCATE und FREE verwenden, aber dazu müßte man auch wenigstens die Größe des Objekts ermitteln können. comFORTH 4 erlaubt das natürlich. Das Wort zur Ermittlung der für die Instantiierung eines Objekts notwendigen Anzahl Adreßeinheiten heißt SIZE. Für das dynamische Anlegen eines Objekts sollte man jedoch lieber das Wort NEW verwenden.
classname NEW (-- addr)

Das wird letzten Endes auch auf ALLOCATE zurückgeführt, übernimmt aber zusätzlich die notwendige Initialisierung des Speichers, die sonst von OBJECT bzw. OBJ ausgeführt wird. Diese Initialisierung betrifft nicht die selbst definierten Instanzvariablen sondern nur diejenigen Zellen des Objekts, die für die Verbindung zu ihrer Klasse sorgen. Letzten Endes ist ohne diese Initialisierung kein Aufruf einer virtuellen Methode über diesem Objekt möglich. Eventuelle Laufzeitfehler von ALLOCATE werden auf THROW umgelenkt.

Als Pendant zu NEW wird für das Löschen von Objekten das Wort DELETE benutzt. Einziger Unterschied zur FREE ist die Fehlerbehandlung per THROW.

DELETE (addr --)

Bunte Früchte zum Dritten

Jetzt sollte es uns gelingen, das ultimative Früchtezählprogramm zu schreiben. Wir beginnen am besten mit der dynamischen Generierung der Zähler, da hier die am weitesten reichenden Änderungen nötig sind. Da die Entscheidung für das Anlegen eines Zählers anhand der Farbe getroffen wird, benötigen wir als erstes eine Klasse für die Farben. Dabei bereinigen wir gleichzeitig ein Problem der vorhergehenden Versionen. Dort wurde in jeder Fruchtsorte ein Zeiger für die aktuelle Farbe gehalten, was dafür sorgte, daß zur Verwirrung des Anwenders grüne Äpfel und gelbe Birnen gleichzeitig aktiv sein konnten. Jetzt speichern wir nur noch die aktuelle Farbe. Im Hinblick auf den Report geben wir den Objekten einen Zeiger auf eine (konstante) Zeichenkette als Instanzvariable und eine Methode für ihre Anzeige mit.

```
CLASS: CColor
  VALUE Text
  ;CLASS

ALSO CColor DEFINITIONS
PUBLIC

0 CColor POINTER ActiveColor

M: Prepare ( addr -- )( initialize it with a counted string )
  TO Text ;M
```

```
M: Display ( -- )( display the text string )
  Text COUNT TYPE ;M
```

PREVIOUS DEFINITIONS

Die Definition der Farbobjekte können wir auch gleich innerhalb der Klasse definieren, da im Benutzerinterface ohnehin nur Worte zur Aktivierung der Farbe sichtbar sein sollen.

```
ALSO CColor DEFINITIONS

CColor OBJECT RedColor      ," red "      RedColor
                                          Prepare
CColor OBJECT GreenColor    ," green "    GreenColor
                                          Prepare
CColor OBJECT YellowColor   ," yellow "   YellowColor
                                          Prepare
```

PREVIOUS DEFINITIONS

Die eigentliche Farbsteuerung wird außerhalb der Klasse erledigt und setzt nur den Zeiger auf das aktive Farbobjekt. Sollten irgendwann neue Farben hinzukommen, brauchen wir nur ein neues Farbobjekt in der Klasse definieren und ein zusätzliches Wort zur Aktivierung.

```
: Red ( -- )( activate red apples )
  CColor RedColor TO ActiveColor ;

: Green ( -- )( activate green apples and pears )
  CColor GreenColor TO ActiveColor ;

: Yellow ( -- )( activate yellow pears )
  CColor YellowColor TO ActiveColor ;

Green ( -- )( initialize fruits )
```

Kommen wir jetzt zum Umbau der Früchtezähler. Diese müssen eine neue Instanzvariable bekommen, welche die Farbe anhand eines Zeigers auf das zugehörige Farbobjekt bestimmt. Dadurch muß natürlich auch die Initialisierung verändert werden. Da wir die Implementierung der Farbgebung vor den anderen Klassen des Programms möglichst gut verstecken wollen, definieren wir den Zeiger auf die Farbe privat. Benutzern der Klasse stellen wir nur die Funktion FindCounter zur Verfügung, die einen passenden Zähler aus einer gegebenen Liste heraussucht. Da FindCounter auch auf leere Listen anwendbar sein soll, definieren wir sie nicht als Methode, sondern als gewöhnliche Doppelpunktdefinition.

```
CLASS: CFruitCounter ( apple counter class )
       CList INHERIT CList::   \ list of colors
       VARIABLE #Fruits        \ number of fruits
       CColor POINTER Color    \ color of the fruits
       ;CLASS

ALSO CFruitCounter DEFINITIONS
PUBLIC

M: Prepare ( addr-color addr-anchor -- )( initialization )
       CAnchor PTR anchor
       CList:: Prepare
       #Fruits OFF
       THIS anchor Insert
       TO Color ;M

: FindCounter  ( ds: addr-color addr-list -- addr-count | 0 )
       ( try to locate a counter for the given color )
       ( be aware of an empty list: we receive a null pointer )
       CFruitCounter PTR counter
       BEGIN    \ walk through the counter chain
          counter
       WHILE    \ there is one
          DUP counter Color =
          IF       \ found
             DROP counter EXIT
          THEN
          counter Tail TO counter
       REPEAT
       DROP 0 ;
```

PREVIOUS DEFINITIONS

Jetzt können wir in der Klasse CFruit die Zähler nach Bedarf anlegen. Betroffen ist eigentlich nur die Methode Add. Dort muß vor der eigentlichen Addition überprüft werden, ob schon ein Zähler der aktuellen Farbe vorhanden ist. Wenn das nicht der Fall ist, muß Add mit NEW ein neues Objekt anlegen und initialisieren.

```
ALSO CFruit DEFINITIONS
PUBLIC

M: Add ( n -- )( delegate counting )
       0 CFruitCounter PTR counter
       CColor ActiveColor  CAnchor:: Head
       CFruitCounter FindCounter TO counter
       counter 0=
       IF  \ we must create a new counter
          CFruitCounter NEW TO counter
          CColor ActiveColor THIS
          counter Prepare
       THEN
       counter Add ;M
```

PREVIOUS DEFINITIONS

Nachdem wir diesen Schritt hinter uns haben, können wir erst mal ein wenig aufräumen. Dazu gehört, daß wir die im Wörterbuch definierten Objekte RedApples, GreenApples, GreenPears und YellowPears ersatzlos streichen könnnen. Die Zählerobjekte entstehen ja jetzt dynamisch. Außerdem fallen die Methode Select und die Instanzvariable CurrentCounter aus der Klasse CFruit weg, da ja jetzt die Klasse CColor die aktuelle Farbe kontrolliert.

Als nächstes können wir uns dem Report widmen. Hier wollten wir das schon für das Zählen der Früchte verwendete Verfahren übernehmen. Das bedeutet im wesentlichen, die Klasse CList in Analogie zur Methode Get um eine neue virtuelle Funktion Display zur Anzeige des Zustands eines Elements zu erweitern und in CFruit und CFruitCounter zu implementieren. Das Ablaufen der Listen legen wir analog zu Sum als einfache Methode List in die Klasse CAnchor. Zur Anzeige der Fruchtsorte erhalten die Früchte analog zu den Farben eine Zeichenkette, die bei der Initialisierung ersetzt wird. Im Gegenzug können wir einige nicht mehr benötigte Worte ersatzlos streichen, nämlich ?, ALL?, GetTotal und Total?. Die folgenden Ausschnitte aus dem Quelltext zeigen nur die geänderten Passagen, zunächst die neue Definition von CList.

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

CAnchor implementiert das Ablaufen der Liste. Leider haben wir hier wieder eine Duplikation von Code, der sich nur durch die aufgerufene Methode unterscheidet. Aber hier können wir mit unseren jetzigen Hilfsmitteln nichts machen. Die Anzeige eines Objekts ist einfach etwas anderes als die Beschaffung seines Werts.

```
ALSO CAnchor DEFINITIONS
PUBLIC

M: List ( -- )( display all list elements )
      Head PTR element
      BEGIN \ walk through the element chain
          element
      WHILE \ end of chain not reached yet
          element Display
          element Tail TO element
      REPEAT ;M

PREVIOUS DEFINITIONS
```

Früchtezähler stellen sich selbst durch die Anzeige der aktuellen Anzahl dar und delegieren die Anzeige der Farbe an das gegebenen Farbobjekt.

```
ALSO CFruitCounter DEFINITIONS
PUBLIC

V: Display ( -- )( display the number and color )
        CR Get 5 U.R SPACE Color Display ;

PREVIOUS DEFINITIONS
```

Die Früchteklasse bekommt eine neue Instanzvariable für die Adresse der der Frucht zugeordneten Zeichenkette.

```
CLASS: CFruit ( represents one kind of fruits  )
        CList INHERIT CList::   \ list of fruits
        CAnchor INHERIT CAnchor::  \ list of counters
        VALUE Text        \ name of fruit
        ;CLASS
```

Eine Frucht stellt sich durch die Anzeige der Teilsumme (früher durch All?) dar und delegiert die Anzeige der Details an den Anker der Zählerliste weiter. In Analogie zum früheren GetTotal implementieren wir den Gesamtreport als normales Doppelpunktwort mit dem Aufruf von List über der Früchteliste.

```
ALSO CFruit DEFINITIONS

V: Display ( -- )( display a report of this fruit )
      CR Sum . Text COUNT TYPE ."  counted."
      List CR ;V

: Report ( -- )( report all )
      CR ." Total of " FruitList Sum . ." fruits. "
      FruitList List ;

PREVIOUS DEFINITIONS
```

Bleibt nur noch das Benutzerinterface mit den vordefinierten Objekten für Äpfel und Birnen.

```
CFruit OBJECT Apples ( -- addr )( apple pointer )
  ," apples" Apples Prepare

CFruit OBJECT Pears ( -- addr )( pear pointer )
  ," pears" Pears Prepare

: Report ( -- )( syntactic sugar )
  CFruit Report ;
```

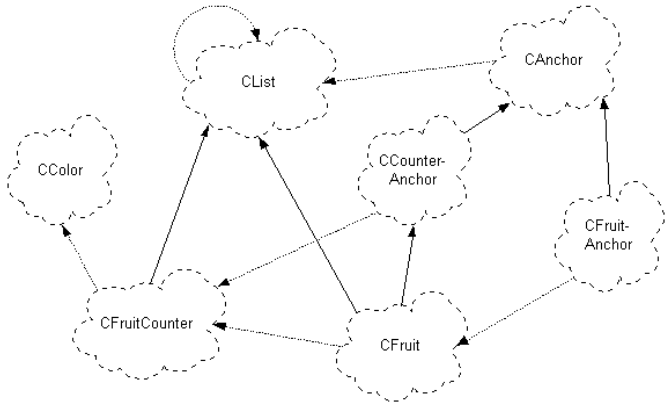
An dieser Stelle sollten wir erst mal eine Atempause einlegen und uns vergewissern, daß das Programm auch noch funktioniert:

```
4 Red Apples Add  ok
5 Yellow Pears Add  ok
3 Apples Add  ok

Report
Total of 12 fruits.
5 pears counted.
   5 yellow

7 apples counted.
   3 yellow
   4 red
ok
```

Das sieht schon so aus, wie wir es haben wollen. Haben wir unseren Job erledigt?

Ungeliebtes Aufräumen

Nur beinahe! Im Report müssen nämlich noch die Zähler zurückgestellt werden, was bedeutet, die mit NEW angelegten Zähler per DELETE wieder wegzuwerfen. Das klingt wieder mal danach, rekursiv Listen abzulaufen und Elemente zurückzusetzen oder zu löschen. Noch ein Clone der Pärchen Get/Sum bzw. Display/List? Leider liegen die Dinge nicht ganz so einfach. Das Problem besteht darin, daß wir mit den Elementen der Früchteliste etwas anderes tun müssen als mit denen der Zählerlisten. Nur die letzteren müssen per DELETE zerstört werden, die Früchte selbst bleiben erhalten. Für die Listenanker bedeutet das, daß der Anker einer Zählerliste zurückgesetzt werden muß, der der Früchteliste bleibt unverändert. Beim Zerstören der Elemente einer Liste muß man auch anders vorgehen als bei Sum und List. Nach der Ausführung von DELETE kann man beispielsweise Tail nicht mehr benutzen, um zum nächsten Element zu gelangen. Also müssen wir neu überlegen. Der wesentliche Unterschied liegt in der Behandlung der Listenanker. Eine Lösung wäre also die zweifache Spezialisierung der Klasse CAnchor für Zähler und Früchte. CList muß um eine Funktion zur Selbstzerstörung erweitert werden. Das führt zu einer neuen Klassenstruktur:



Die Änderungen selbst sind aber gar nicht so aufwendig. Beginnen wir mit der neuen Methode Destroy in CList. Hier nutzen wir wieder die rekursive Natur der Listen aus. Solange das Element einen Nachfolger besitzt, zerstören wir erst diesen und danach das Listenelement selbst. Für die Rekursion können wir hier das gewöhnliche RECURSE verwenden. Wäre Destroy eine virtuelle Methode, müßten wir an Stelle von RECURSE auch Destroy schreiben.

```
ALSO CList DEFINITIONS
PUBLIC

M: Destroy ( -- )( perform a recursive suizide )
      Tail
      IF  \ there is a tail
          Tail RECURSE
      THEN
      THIS DELETE ;M

PREVIOUS DEFINITIONS
```

Mit Hilfe dieser Methode können wir das Rücksetzen des Ankers einer Zählerliste definieren. Die Definition der Klasse ist simpel.

```
CLASS: CCounterAnchor
      PROTECTED
      CAnchor INHERIT CAnchor::
      ;CLASS

ALSO CCounterAnchor DEFINITIONS
PUBLIC

M: Reset ( -- )( destroy the fruit counter list )
      Head
      IF  \ there is a list
          Head Destroy
          0 TO Head
      THEN ;M

PREVIOUS DEFINITIONS
```

Analog verfahren wir für den Anker der Früchteliste. Allerdings implementieren wir hier nach dem Muster von Sum und List eine Schleife. Die Implementierung von Reset müssen wir uns aber trotzdem genauer ansehen. Ein markanter Unterschied besteht darin, daß wir hier die normale Methode Reset aus der Klasse CFruit (die diese aus CCounterAnchor erbt) aufrufen, nicht mehr eine virtuelle Methode der Klasse CList. Das ist nur deshalb zulässig, weil wir genau wissen, daß in dieser Liste nur Früchte vorhanden sind. Anderenfalls würden die Zuweisungen zum lokalen Zeiger fruit Laufzeitfehler provozieren.

```
CLASS: CFruitAnchor
      PROTECTED
      CAnchor INHERIT CAnchor::
      ;CLASS

ALSO CFruitAnchor DEFINITIONS
PUBLIC

M: Reset ( -- )( free all fruit counters )
      Head CFruit PTR fruit
      BEGIN  \ walk along the fruit list
            fruit
      WHILE \ there is another fruit
            fruit Reset
            fruit Tail TO fruit
      REPEAT ;M

PREVIOUS DEFINITIONS
```

Die Benutzung normaler Methoden verursacht eine weitere Irregularität. Um die Methode Reset überhaupt übersetzen zu können, muß die Klasse CFruit bereits definiert worden sein. Wir müssen unseren Quelltext also neu sortieren. Die Definition der Klasse CFruit muß genau hinter die Definition der Klasse CFruitAnchor vorgezogen werden. Der einzige Unter-

schied zur bisherigen Definition ist die Ableitung von CCounterAnchor anstelle von CAnchor.

```
CLASS: CFruit ( represents one kind of fruits  )
        CList INHERIT CList::                   \ list of fruits
        CCounterAnchor INHERIT CCounterAnchor::
                                                \ list of counters
        VALUE Text                              \ name of fruit
        ;CLASS
```

Die Methoden von CFruit werden erst nach der Methode Reset von CFruitAnchor definiert. Die nötigen Änderungen halten sich in Grenzen. Nur der Anker der Früchteliste ist jetzt als Instanz von CFruitAnchor zu definieren und Report erweitern wir um ein unscheinbares Reset der Früchteliste.

```
ALSO CFruit DEFINITIONS
PRIVATE

CFruitAnchor OBJECT FruitList ( -- addr )( list of fruits )
      FruitList Prepare

: Report ( -- )( report all )
      CR ." Total of " FruitList Sum . ." fruits. "
      FruitList List
      FruitList Reset ;

PREVIOUS DEFINITIONS
```

Das war‘s. Als letzte Herausforderung unseres Egos bleibt die Code-Duplikation in Sum und List in CAnchor übrig. Sollte es uns nicht doch noch gelingen, die Schleife auszufaktorisieren? Das wäre das i-Tüpfelchen. Auch wenn es in Bezug auf unseren Job eher als sportliche Leistung einzustufen wäre.

Indirekte Methodenaufrufe

Würden wir in Standard-Forth programmieren, wäre das Verfahren klar. Wir würden der Schleife einfach den *execution token* des gewünschten Worts übergeben und EXECUTE aufrufen. Mit Methoden ist das leider nicht ganz so einfach. Durch die mögliche Mehrfachvererbung müssen zusätzliche Informationen zu jeder Methode mitgeführt werden. Außerdem müssen virtuelle und gewöhnliche Methoden unterschiedlich behandelt werden. Daher kann man leider nicht einfach ‘ bzw. [‘] und EXECUTE verwenden. An deren Stelle kann man in comFORTH 4 aber für gewöhnliche Methoden die Worte METHOD‘ bzw. [METHOD‘] und METHOD-CALL verwenden. Statt eines *execution token* wird ein doppeltgenauer Wert verwendet, der als *method token* bezeichnet werden kann. Für virtuelle Methoden wird ebenfalls ein zwei Zellen großer *virtual method token* benutzt. Zu seiner Ermittlung bzw. Benutzung enthält comFORTH 4 die Worte VIRTUAL‘, [VIRTUAL‘] und VIRTUAL-CALL. Da die Zahlenwerte eines *virtual method token* anders interpretiert werden als die eines normalen *method token*, darf man sie nicht miteinander verwechseln. Zudem sind die Werte der *method token* vom Kontext abhängig, in dem sie gesucht werden. Das führt zu der zusätzlichen Randbedingung, daß ein *method token* nur in dem Kontext ausgeführt werden darf, in dem er herausgesucht wurde.

Bunte Früchte zum Letzten

Die eben kennengelernten indirekten Methodenaufrufe erlauben uns, jetzt die letzte Iteration in Angriff zu nehmen. Wir faktorisieren in der Klasse CAnchor die Schleifen in Sum und List in eine neue Methode Iterate und übergeben dieser den *virtual execution token* einer im Schleifeninneren auszuführenden Methode als Parameter. Dazu müssen wir die durch Sum benutzte virtuelle Methode Get aus der Klasse CList durch die stapelneutrale Methode AddCount ersetzen, die die Addition gleich miterledigt. Das müssen wir in den von CList abgeleiteten Klassen CFruitCounter und CFruit berücksichtigen.

Beginnen wir also mit CList, hier wird nur Get durch AddCount ersetzt.

```
CLASS: CList
   PUBLIC
   CList POINTER Tail
   VIRTUAL AddCount ( n1 -- n2 )( add the element value )
   VIRTUAL Display ( -- )( display the element )
   ;CLASS
```

Unser neues Wissen setzen wir in CAnchor ein. Nach dem Muster von List bauen wir die Methode Iterate. Der übergebene *virtual method token* wird in einer doppeltgenauen lokalen Variable gespeichert und in jedem Zyklus per VIRTUAL-CALL ausgeführt. Die Methoden Sum und List werden jetzt sehr einfach. Sie legen nur noch die Methode fest, die Iterate verwenden soll.

```
ALSO CAnchor DEFINITIONS
PUBLIC

M: Iterate ( .. vt -- .. )( apply the given virtual )
      ( method to all elements in the list )
      2VAL method
      Head PTR element
      BEGIN                 \ walk through the element chain
         element
      WHILE                 \ end of chain not reached yet
         element method VIRTUAL-CALL
         element Tail TO element
      REPEAT ;M

M: Sum ( -- n )( return the sum of all element values )
      0 CList [VIRTUAL'] AddCount Iterate ;M

M: List ( -- )( display all list elements )
      CList [VIRTUAL'] Display Iterate ;M

PREVIOUS DEFINITIONS
```

Es ist zu beachten, daß auch trotz dieser Indirektion eine späte Bindung zum Methodencode erfolgt. Bei einem klassischen Einsatz von EXECUTE hätte man vermutlich anstelle von zwei Aufrufern von Iterate vier. Jetzt bleibt nur noch die Definition von AddCount übrig. In der Klasse CFruitCounter definieren wir einfach Get als normale Methode und führen AddCount auf Get zurück. So stören wir den bisherigen Code nicht.

```
ALSO CFruitCounter DEFINITIONS
PUBLIC

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

V: AddCount ( n1 -- n2 )( add the current number of fruits )
      Get + ;V

PREVIOUS DEFINITIONS
```

In CFruit wird Get nirgendwo direkt benötigt, so daß wir es streichen können. AddCount bleibt aber trivial.

```
ALSO CFruit DEFINITIONS
PUBLIC

V: AddCount ( n1 -- n2 )( add the total count of fruits )
      Sum + ;V

PREVIOUS DEFINITIONS
```

Damit sind wir am Ziel. Weder neue Farben noch zusätzliche Früchte können uns in Zukunft erschüttern. Einige "Verbesserungen" wären zwar noch vorstellbar – zum Beispiel Iterate so umzubauen, daß man auch FindCounter und Reset darüber implementieren kann – das ist aber doch mit größeren Änderungen verbunden. Wir sind gut beraten, unser Produkt auszuliefern solange es noch funktioniert...

Fazit

Unser fiktiver Programmierjob zeigt ziemlich plastisch, welche Möglichkeiten und welche Probleme der Einsatz objektorientierter Techniken mit sich bringt. Auf der einen Seite kann sich aufgrund der Möglichkeiten zur internen Wiederverwendung von Code der Aufwand für die Implementierung einer neuen Funktion stark reduzieren. Andererseits ist gerade dadurch der Programmfluß im System nicht mehr so offensichtlich. Gleichzeitig wächst die Gefahr, die Abhängigkeiten zwischen den einzelnen Klassen eines Systems so stark werden zu lassen, daß schon kleine Änderungen die Stabilität des Gesamtsystems gefährden können. Die Regeln eines strukturierten Systementwurfs werden also keinesfalls durch irgendeine "Objektorientierung" abgelöst, sondern gewinnen eher an Bedeutung. Man merkt sehr schnell, daß ein gut funktionierendes objektorientiertes System eines gründlichen Entwurfs bedarf. Papier und Bleistift sind in dieser Phase nützlicher als ein schneller Rechner.

Ihre volle Leistungsfähigkeit entfalten objektorientierte Programmiersysteme allerdings erst zusammen mit Klassenbibliotheken, die es erlauben, allgemein verwendbaren Code von einem Projekt zum anderen "mitzunehmen". So sollte es bei Verwendung einer solchen Bibliothek nicht mehr nötig sein, in einem Projekt allgemeine Listenfunktionen wie das Einfügen oder Suchen von Elementen zu implementieren. Vererbung und virtuelle Methoden erlauben eine einfache Anpassung an die konkrete Aufgabe.

Der Austausch derartiger Bibliotheken ist an eine Standardisierung der Sprache gebunden. Gegenwärtig konkurrieren eine ganze Reihe recht unterschiedlicher Objektmodelle für Forth miteinander. Ganz sicher haben alle Vor- und Nachteile. Um diese auch im Hinblick auf eine zukünftige Standardisierung besser miteinander vergleichen zu können, würde ich es sehr spannend finden, in den nächsten Ausgaben der VD unter Verwendung anderer Objektmodelle implementierte Versionen des Früchezählers zu finden. Ein nicht objektorientiertes ANS-Standardprogramm würde diesen Vergleich sicher abrunden.

Die *Objektorientierte Programmierung* gehört mittlerweile zu den etablierten Programiertechniken und zum Repertoire der meisten neueren Programmiersprachen. Verwendet man den Begriff der Objektorientierten Programmierung in Bezug auf eine Programmiersprache, wird er üblicherweise mit drei Hauptmerkmalen verbunden: Kapselung, Polymorphie und Vererbung. Alle drei Techniken sind unabhängig voneinander in verschiedenen Formen entwickelt worden und besitzen auch einzeln ihre Daseinsberechtigung.

Unter *Kapselung* versteht man die Einführung von "Sichtbarkeitsgrenzen" im Zusammenhang mit Softwaremodulen. Eine Definition im Inneren eines Moduls wird nur dann für die Außenwelt sichtbar, wenn sie zum Interface des Moduls gehört. Durch diese Einschränkung soll garantiert werden, daß Änderungen an der Implementierung eines Moduls nur dann Auswirkungen auf andere Moduln haben, wenn sein Interface von dieser Änderung betroffen ist. Besonders geeignet für eine Einkapselung sind dabei Datenstrukturen. Durch die Datenelemente wird der "Zustand" des Moduls bestimmt. Kontrolliert man alle Zugriffe auf die Datenstruktur durch wenige Interfacefunktionen, so hat man auch die Kontrolle über die möglichen Zustandsänderungen des Moduls.

Mit dem Merkmal *Polymorphie* wird im allgemeinen das Konzept der datengesteuerten Programmausführung verbunden. Das bedeutet, daß die durch den Aufruf einer Aktion tatsächlich ausgeführte Programmsequenz durch die Daten bestimmt wird, auf die sich diese Aktion bezieht. Einer der direkten Effekte dieser Technik besteht in der globalen Verringerung der Anzahl von Entscheidungsstrukturen in Form von IF- oder CASE-Anweisungen und damit einer effektiven Verringerung der Anzahl der Programmpfade. Darüber hinaus wird es jedoch auch möglich, Algorithmen in allgemeiner Form zu implementieren. Findet sich ein gleiches Muster an Aktionen über verschiedenen Daten, so kann man die Diffe-

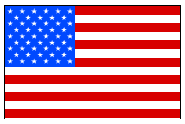
renzen in den datengesteuerten Aufruf von Teilaktionen faktoriseren und das “Muster” als gemeinsam verwendetes Rahmenprogramm implementieren.

Das Merkmal *Vererbung* bezieht sich auf eine Programmiermethodik in Form von Differenzbeschreibungen. Besitzt man eine Lösung eines ähnlichen Problems, kann man die Lösung des eigenen Problems auch durch die Beschreibung der Unterschiede zur gegebenen Lösung formulieren. Das lohnt sich natürlich nur dann, wenn diese Differenzbeschreibung viel einfacher ist als die Beschreibung der vollständigen Lösung. In der Praxis findet man hier vor allem zwei Strategien. Am weitesten verbreitet ist die Ableitung angepaßter Lösungen aus allgemeinen auf dem Weg der Spezialisierung. Eine andere Strategie besteht in der Synthese komplexer Lösungen durch das Verbinden mehrerer Einzelkonzepte.

Der Einsatz der Objektorientierten Programmierung zielt damit auf eine höhere Softwarequalität durch eine Verbesserung der inneren Struktur der Programme sowie eine bessere Effektivität des Programmierens durch Wiederverwendung vorhandenen Codes. Ein Wundermittel ist sie aber doch nicht.

Neues aus den USA

Hallo, Friederich,



Das wird mein Bericht über das September Treffen der SVFIG. Wir waren wieder im üblichen Quartier am Cogswell College und die Teilnahme war gut -- um die 20 Personen zu Beginn, um 24 zu Mittag, aber nicht unbedingt die gleichen Gesichter. Sie kommen und gehen und genießen sicher die Plaudereien, die zwischen den Vorträgen stattfinden.
Wir begannen spät, da die Kaffekanne umgekippt war, vielleicht aus Protest gegen das Java Thema, das den ersten Vortrag bildete. George Shaw von Patriot Scientific Corporation (www.ptsc.com) kam und berichtete uns, daß an Chuck Moore's originalem ShBoom Chip nichts verändert wurde, um daraus eine Java Maschine zu machen, den PSC1000. Er kostet $25 als Einzelstueck, $10 in 10000er Stueckzahl. Ein Artikel von Tom Cantrell in der 1998er Maerz-Ausgabe des "Circuit Cellar" Magazine enthält alles über diesen Chip.
Java ist eine Art bereinigtes C++. Die virtuelle Java Maschine schaut fast wie Forth aus, da sie eine Stackarchitektur besitzt (Als Bezug darauf zitierte Cantrell Goethe: "Alles wurde schon zuvor gedacht, aber das Problem besteht darin noch einmal darüber nachzudenken.")
Nach dem Essen brachte unser letzter Präsident, John Hall, einen richtigen iMac-Computer mit, komplett mit seinem feinen Netzkabel und "voice fonts" (der Übersetzer muß passen ;-)), und natürlich, Rudy, die neue Maus. Monitor und Gehäuse sind in Korea hergestellt. John's Open Firmaware Forth sitzt in einem winzigen Chip, aber expandiert sich zu ungefähr 1,5MB bevor es das Betriebssystem bootet. Der iMac hat keine Floppy, keine serielle Schnittstelle, unterstützt aber den USB (Universial Serial Bus). Der Computer wird hier in den USA fuer $1300 verkauft.
Skip Carter, Al Mitchell, und Bob Nash waren mit ihren Produkten und Projektbeispielen da. Dr. Ting krönte das Treffen mit der Vorführung seines selbstgebauten 81-Tasten Orgel-Keyboards, welches uns 16 gleich-temperierte Noten per Oktave liefern wird -- das nächste Mal, nachdem er das 8051-Board mit den Lautsprechern gekoppelt hat. Wir sind alle gespannt zu hören, wie die Musik klingen wird.
Zufällig hat die versprochene Präsentation des neuen "Forth Programmer's Handbook" immer noch nicht stattgefunden. Der Rezensent wurde wohl von anderen Dringlichkeiten abgehalten. Ebenso gab es keinerlei Gespräche über das Forth OS, dessen Fürsprecher uns auf der Suche nach "grüneren Weiden" scheinbar verlassen hat. Das mag betrüblich sein, aber Forth wird weiterleben, genau wie Fortran, Cobol, Lisp und DOS(?). Denkst Du nicht auch?
Weiterhin wurde angekündigt, das Forth, Inc. einen Stand auf der Embedded System Conference in San Jose, November 3-5, 1998 haben wird.
Bis zum nächsten Mal,

Henry

übersetzt von Thomas Beierlein

Was Neues...

aus **de.comp.lang.forth**

Hallo,

weil es in diesem Brett so leer ist und ich heute am Kaffeetisch eine Idee hatte, wollte ich sie mal zum Besten geben:

DAS NEUE IF{

```
: IF{ ( compiling: xxx -- xxx' ret ) \ ein neues IF
      (   running: flag -- )          \ nix neues, daher
                                      \ kein Kommentar
      compile ?branch      \ heißt wohl jetzt POSTPONE ?
      here 0 ,             \ Platz für branch-Adresse
 ON}  ( ret -- )           \ Die ON} -Klausel
      here swap !          \ sowas machte bisher das THEN
      ; IMMEDIATE

: }   ( xt --- ) execute ; IMMEDIATE
```

Damit kann man jetzt das IF ersetzen:

```
     IF      ...     THEN
 wird: IF{     ...     }
```

und wenn es klappt, kann man noch ein wenig basteln und ersetzen:

Forth Interest Group International (FIG USA)

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

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

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

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

Ansonsten bekommen Sie Auskünfte über das amerikanische Forth-Büro:

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

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

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

```
        BEGIN   ...    AGAIN
wird:   BEGIN{  ...    }

        : name  ...    ;
wird:   { name  ...    }

        IF  ...  ELSE  ...  THEN
wird:   IF{ ...  }{  ...  }
```

Alle anderen Kontrollstrukturen werden auch ersetzt.

Jedesmal wäre das } immer genau so wie oben definiert. Nur das IF{ und das BEGIN{ und so weiter muß jeweils einen zweiten Teil, nämlich die ON} -Klausel enthalten, damit das } weiß, was es machen soll.
Vielleicht kommt noch ein dritter Teil dazu, sagen wir eine {ON} Klausel, damit das }{ (z.B. das bisherige ELSE) auch noch weiß, was es tun soll.
Die Vorteile liegen auf der Hand:
- Es fallen ziemlich viele Wörter weg und verbrauchen keinen Platz mehr im Wörterbuch und im Kopf.
- Die 'Compiler Security' wird ungemein vereinfacht. Wenn alle schließenden Klammern gleich aussehen, kann man nie die falsche hinschreiben.
- Es macht also alles einfacher und das ist doch das Wichtigste.

Das Wort ON} ist der einzige Nachteil, der mir spontan einfällt.

Das Wort ON} ist mindestens dreimal so systemabhängig wie lang.

Das Wort ON} braucht eine ganz scheußliche Erläuterung. Das Glossar zum Wort ON} ist mindestens dreimal so schwierig zu schreiben und fünfmal so unverständlich wie das Produkt aus Codelänge und Systemabhängigkeit, nämlich:

--- Glossar: ----

ON} (-- xt) IMMEDIATE

Typische Benutzung:

```
: name ..words1... ON} ...words2...
; IMMEDIATE
....
: name2  .....  NAME .... } ... ;
```

ON} unterteilt die Definition von name in zwei Teile words1 und words2. Wenn name später ausgeführt wird, werden zunächst lediglich die words1 ausgeführt. ON} wirkt hier also wie ein EXIT . Allerdings läßt es zusätzlich einen Execution Token xt auf dem Stack, der bei späterer Ausführung - meist durch das Wort } - die words2 zur Ausführung bringt.

--- Glossar Ende ---

Bei so traditionellen Systemen wie dem volksForth (indirekt gefädelt, 16 bit-Adressen für alles und überall) ließe sich ON} oben in der Definition von IF{ durch die folgende Sequenz ersetzen:

ON} in einer Definition wirkt wie:

```
[ here 6 + ] Literal  exit  ['] : @ ,
```

Es hinterläßt vorm exit den cfa des folgenden Teils. Der cfa einer Colondefinition sieht immer gleich aus, daher guckt man ihn am besten beim Colon ab, man könnte statt ['] : @ z.B. auch ['] words @ schreiben.

In Forth83 oder volksForth ließe sich das so definieren:

```
: ON}   ( compiling: -- xt )
        (   running: -- )
        here 6 +
        [compile] Literal
        compile exit
        [ ' : @ ] Literal ,
        ; IMMEDIATE
```

Noch einfacher ginge es so:

```
\ Das neue IF{                          clv 08okt98
: }   ( ret -- )  >R ; IMMEDIATE

: ON} ( -- ret )  R> ;    \ diesmal wirklich ohne IMMEDIATE\
\
: IF{  ( compiling: -- adr ret )     \ ein neues IF
      (   running: flag -- )         \ nix neues, kein Kommentar
       compile ?branch               \ heißt wohl jetzt POSTPONE
       here 0 ,                        \ Platz für branch-Adresse
  ON}   ( adr -- )                   \ Die ON} -Klausel
       here over - swap !            \ wie bisher das THEN
       ; IMMEDIATE

\ test für das neue IF{                  clv 08okt98

: test
  ."  Anfang "
  IF{  ."  Ja  "  }
  ."  Ende "
;

  0 test    \ druckt:  Anfang  Ende
  1 test    \ druckt:  Anfang  Ja  Ende
```

Und weils so schön war noch was Ungetestetes:

```
: { : ON} [compile] ; ;                          \ The new Colon
{ test2 ."  Anfang " IF{  ."  Ja  "  }  ."  Ende " }    \ The test
{ .{  compile (.") Ascii } word c@ 1+ allot } immediate
                                                  \ The new ."
{ test3 .{  Anfang } IF{  .{  Ja  }  }  .{  Ende } }  \ The test
```

Beim F-PC muß man erheblich mehr Heckmeck mit Codespace und relativen Jmps und so Zeug treiben. Das wird zwar kein längerer Code, aber noch unverständlicher.

Vielleicht probiere ich mal eine ANS-Version (kann ich nicht gut):

```
: ON}   ( compiling: -- xt )
       (   running: -- )
       0 POSTBONE LITERAL  POSTBONE IF
       \ unconditional branch forward
       0 POSTBONE Literal
       postbone ;
       :noname
```

---- nein, ANS kann ich nicht!
Ciao,

- Claus (Vogt)-

Das ist eine ‚pfiffige' Idee von Claus Vogt. ‚Spielereien' in den inneren Strukturen der Worte machen nicht nur Spaß, sondern oft genug auch Sinn. Und meisten lernt man was dabei ;-)

fep

... auch aus de.comp.lang.forth

Hallo Claus,
...
Du willst uns zu strukturierten Kontrollstrukturen erziehen?

Ich will aber BEGIN ... WHILE ... UNTIL ... ELSE ... THEN, wie in

```
: test ( -- )
   10
   ." You have " dup . ." keys to press the space bar "
   BEGIN
      KEY DUP  BL -
   WHILE
   EMIT
   1-
   dup 0=
   UNTIL
    DROP CR ." you missed it."
   ELSE
    DROP CR ." you had " . ." keys left."
   THEN ;
```

schreiben :-) Nur so kann ich Worte schreiben, die kein anderer versteht, ich als Guru dastehe und alle mich bewundern :-)

P.S. Und dann waren da noch die Fleischerhaken:

IF ... ELSE ... THEN	?[...][...]?
BEGIN ... WHILE ... AGAIN THEN	[[... ?[...]]]?
BEGIN ... UNTIL	[[... ?]

Die haben auch schön kurze Namen....

Ulrich Hoffmann

Von: john sadler <john_sadler@hp.com>
Firma:hewlett-packard bioscience products

Ficl 2.02 is now available for download from the web. (Thanks to Skip Carter at Taygeta) This release fixes a couple of bugs in release 2.01, and adds FORGET, MARKER, and Johns Hopkins style locals, among other stuff. See below for the URL...

*** About Ficl ***
Ficl is the ANS Forth for embedding into other programs.
- Written in ANSI C
- Simple to port
- ROMable
- Export C functions to Ficl
- Execute Ficl code from C
- Object oriented (for consenting adults)

Find out more at this URL:
http://www.taygeta.com/ficl.html

Or download the zipped release from:
ftp://ftp.taygeta.com/pub/Forth/Compilers/native/misc/ficl202/

Gehaltvolles

zusammengestellt und übertragen
von Fred Behringer

FORTHWRITE der FIG UK, Großbritannien

Nr. 98, Oktober 1998

2 Forth News
Chris Jakeman

Peter Knaggs ist jetzt Herausgeber des Journal of Forth Application and Research; in USA werden Forth-Programmierer gesucht; PFE darf in Bernd Paysans gForth eingebaut werden; Philip Preston baut ein Forth als Java Bean; Forth für den PSC1000 und den PIC, für den Palm Pilot; Version 0.5 von DELTA FORTH; JForth für AMIGA jetzt gratis; Version 3.1 von Forthmacs für das Acorn-RISC-OS. Das Wrack der Titanic wurde 1985 mit Hilfe eines Forth-gesteuerten Unterwasserfahrzeugs geortet. Jahrestagung der FIG UK am 14.11.98 in Morden (nahe London).

6 Writing the World (1)
Paul E. Benett

In zwei vorausgegangenen Artikeln hat Paul E. Bennett die Input-Seite betrachtet. Jetzt beschäftigt er sich mit dem Output: Es reicht nicht, eine Output-Leitung einfach nur "anzustoßen" und zu hoffen, daß sie die gewünschte Wirkung hervorruft - man muß auch überprüfen. Man braucht einen Rückmelde- und Nachfaß-Mechanismus. Forth mit 8 Colon-Definitionen.

9 A Hard Day Garbage Collecting
Chris Flynn

Chris Flynn setzt sich kritisch mit Anton Ertls (öffentlich zugängigem) Garbage Collector ("Müllbeseitigung" mit automatischer Speicherneubereitstellung) auseinander. Eigentlich berichtet der Autor über Schwierigkeiten im Umgang mit GForth für einen, der keine Erfahrung im Umgang mit GForth hat und es "nur mal schnell" aufrufen möchte - um (auf Chris Jakemans Bitte hin) den Garbage Collector für die Forthwrite-Leser testen zu können. Chris Flynn hat langjährige Erfahrung im Programmieren von Steuerprogrammen für den Kohle-Bergbau. Er wollte den Garbage Collector nur mal versuchsweise ganz schnell in sein System einbauen: Geht nicht ... viel zu wenig Dokumentation beim Code ... Riesenmaschinerie mit viel zu viel vorausgesetztem Hintergrundwissen ... - sagt Chris.

14 Vierte Dimension '98 No 3
Alan Wenham

Wieder mal gut, wie Alan seine Übersicht über den Inhalt unserer VD hinbekommen hat. "Ob das wohl einer liest?", fragt er sich und mich in einer kürzlichen E-Mail. Na, wir hoffen doch sehr ...

16 Deutsche Forth-Gesellschaft

Macht sich prima, unsere Anzeige zur Mitgliederwerbung in der Forthwrite.

17 Finite State Machines: 2
Graeme Dunbar

Zweiter Teil des Artikels der letzten Forthwrite über Endliche Zustandsautomaten. Als Diskussionsbeispiel werden magnetische Aufzeichnungsverfahren gewählt, extern getaktet oder intern getaktet. Sie können als endliche Zustandsautomaten aufgefaßt werden. Der Autor gibt Simulationsmodelle in Forth an: Return to Zero (RZ) , Non Return to Zero (NRZ), NRZ Inverse (NRZI), Phasenmodulation, Frequenzmodulation, Synchronisationsimpulse von den Datenbits getrennt, mit ihnen vermischt ...

23 Getting Together
Chris Jakeman

Internet Relay Chat (IRC) in "Party Mode" über den Kanal #FIGUK. Eine ausgezeichnete Idee, meine ich, der Rezensent, von Chris Jakeman (auf die ihn sein Sohn gebracht hat). Ich bin jedenfalls gespannt, mehr über diese Internet-Rundgesprächsmöglichkeit zu erfahren. - Wäre das eigentlich nicht auch was für uns? Käme da nicht auch noch das noch modernere ICQ-Verfahren ("I seek you") in Frage? Eine schöne Beschreibung findet sich in der c´t 22/98, S.92-95: "Wo laufen sie denn?" von Axel Eschenburg.

24 ANS File Words for Pygmy Forth
Leo Wong

15 Dateibehandlungsworte nach dem ANS-Standard zur Anpassung von Pygmy Forth an die ANS-Empfehlungen. Alles Colon-Definitionen.

26 More from The Lighter Side
Chris Jakeman

Eine geballte Ladung an witzigen Einfällen zum unsterblichen Thema "Why did the chicken cross the road?" (Warum lief denn das Huhn über die Straße?). BASIC-Huhn, Forth-Huhn usw.

27 jeForth
Chris Jakeman

Chris berichtet kurz über das eForth für Java (eben jeForth) auf der Homepage von Michael A. Losh (www.amsystech.com/mlosh/), das im heimischen Browser-Cache steht, sobald man diese Homepage aufgerufen hat. Chris erwähnt auch die deutsche Beschreibung (Extra-Homepage), an welcher der Rezensent nicht ganz unschuldig ist.

29 Letters
Chris Jakeman

C.H. Tings ZEN-Floating-Point-Vorlage für den 80486 zieht weitere Kreise: David Abrahams bietet den Lesern der Forthwrite die Zusendung einer Ausarbeitung für F-PC mit Erweiterungen an. (Man erinnere sich auch an Martin Bitter in VD 2/98, S.6, der herausfand, daß die Turbo-Forth-Version bis auf Kleinigkeiten auch für ZF verwendbar ist. Wo bleibt Pygmy Forth? Wo bleiben die anderen?)

John Passaniti verwendet einen in C programmierten minimalen Forth-Interpreter, um C-Programme schnell entwickeln und austesten zu können, ohne in einer forth-ignoranten Umgebung, die auf C schwört, anzuecken.

Elko Borissov Tchernev hat an der University of Memphis eine Diplomarbeit (Master of Science) über Genetische Programmierung (GP) in Forth geschrieben. Stack-Crossovers scheinen gegenüber Tree-Crossovers Vorteile zu bringen, was für Forth in GP spricht.

VIJGEBLAADJE

der HCC Forth-gebruikersgroep, Niederlande

Nr. 11, Oktober 1998

Treffen am 10.10.98 wieder in der Volkssternwarte Utrecht. Motto: "Forth für die B+-Karte (Forth für den 80C535)".

Overhoren
Leendert C. van den Heuvel

Beschreibung eines von Albert Nijhof unter Mitwirkung von Willem Ouwerkerk geschriebenen Forth-Programms zum Vokabel-Abfragen Deutsch-Holländisch und Holländisch-Deutsch. Zuerst erscheint die (zufallsmäßig ausgewählte) Frage am Bildschirm, dann kommt per Tastendruck die Antwort (in der jeweils anderen Sprache). Abgedruckt ist auch das vollständige Programm-Listing.

Vierte Dimension

Eine Anzeige unserer Forth-Gesellschaft zur Mitgliederwerbung und zur Lenkung der Aufmerksamkeit der niederländischen Forth-Freunde auf unsere Aktivitäten. Willem Ouwerkerk danke ich (der Rezensent) sehr für die Übertragung ins Holländische.

Poort besturing op het B+ bord
Willem Ouwerkerk

Beschreibung und Listing eines Programms zur zufallsmäßigen Ansteuerung von 8 Relais, die dort angeschlossene Lampen schalten. Das Programm ist in 8051-ANS-Forth geschrieben, ein Forth, das auf die B+-Karte mit ihrem 80C535 zugeschnitten ist.

Activiteiten

Nur Mitglieder des übergeordneten Hobby Computer Club (HCC) können Mitglieder der Forth-gebruikersgroep (Forth-gg) sein. Das Vijgeblaadje (sechsmal pro Jahr) kann man sich aber als "Donateur" der Forth-gg für 25 Gulden pro Jahr sichern. Probeexemplare bei Willem Ouwerkerk.

Die Forth-gg veröffentlicht Anzeigen in der holländischen Computerzeitschrift Computer!Totaal.

Namen und Adressen

Postadressen der 6 Vorstandsmitglieder der Forth-gg. 4 davon haben auch eine E-Mail-Adresse, die ebenfalls angegeben ist.

So kriegt man Forth auch klein

von Fred Behringer
Planegger Str. 24, 81241 München

Geben Sie als Forth-Kursleiter Ihren Kursteilnehmern frühmorgens eines der folgenden beiden Programme mit einer Liste der von Ihnen ausgewählten Forth-Worte und stellen Sie ihnen die Aufgabe, unter alleiniger Verwendung dieses Materials dieses oder jenes (von Ihnen noch zusätzlich anzugebendes) Forth-Wort zu definieren.

Stichworte: Turbo-Forth, FORBID, LEARN, volksFORTH-83, ALIAS, Minimalforth, Forth-Erzeugendensystem

Ulrich Hoffmann [6] hat in einem sehr interessanten Artikel anhand seines volksFORTH-83 gezeigt, wie man aus einem für den Anfänger unübersichtlichen Forth (mit viel zu vielen dem Benutzer zugängigen Worten) für Instruktionszwecke ein "kleineres" Forth macht. Seine echt forthgemäße Lösung ist äußerst elegant und verlangt vom Bearbeiter nur noch, eine Wunschliste von Worten einzugeben, die und nur die allein der Benutzer "sehen" kann. Ulrich Hoffmanns Werkzeug ist die Möglichkeit der Neudefinition der gewünschten Worte in neuen Vokabularen über den **ALIAS**-Mechanismus aus volksFORTH-83.

Die Eleganz in Ulrich Hoffmanns Artikel hat mich dazu verleitet, das Ganze mal komplementär durchzuspielen: Ist **W** die Gesamtmenge von Worten des gegebenen Forth-Systems und **H** die Hoffmannsche Teilmenge, dann schließe ich **W\H** über **FORBID** aus und lasse damit **H** übrig.

Selbstverständlich bleiben die ausgeschlossenen Worte für das Gesamtsystem noch in Funktion. Durch **FORBID** werden sie nur überbrückt und für den weiteren Gebrauch verboten. Bei Ulrich Hoffmann bleiben die nicht **ALIAS**-erzeugten Worte für das System auch erhalten. Sie werden dem Benutzer durch Vokabular-Umschaltung entzogen.

Nachteil: W\H ist im allgemeinen wesentlich größer als **H** .

Vorteil: FORBID vergrößert das System um kein einziges Byte.

Getestet habe ich das Ganze mit einer frühen 16-Bit-Version von Turbo-Forth, die ich 1989 eingedeutscht habe. **FORBID** taucht in der Literatur überall immer wieder auf. Ich verwende die Version, die im Vorläufer [9] zum Turbo-Forth-Handbuch als Anwendungsbeispiel zu **HASH** angegeben wird. Ich lasse **FORBID** hier auf **CONTEXT** wirken, nicht auf **CURRENT** .

In meinem Transputer-Forth-System F-TP 1.00 (liegt auf ftp://ftp.taygeta.com/pub/forth/compilers/native/dos/transputer/) habe ich **FORBID** ohne die Klammern bei **2DUP** (siehe unten) verwendet. Damit kann man dann ein (zeitweilig) per **FORBID XXX** ausgeklinktes Wort durch einen einfachen Aufruf von **ALLOW** (**: ALLOW ! ;**) wieder einklinken.

Zunächst ein Vorschlag, der nahezu gar keinen Programmieraufwand erfordert, der aber durch eine im allgemeinen viel zu lange Ausschluß-Liste erkauft werden muß. Anschließend mache ich dann einen Vorschlag, der mit einer genauso kurzen Liste wie bei Ulrich Hoffmann [6] auskommt, der aber noch ein paar zusätzliche Programmier-Anleihen bei **LEARN** aus dem **HELP**-Mechanismus von Turbo-Forth macht.

Verkürzungsprogramm erster Art

```
ONLY FORTH DEFINITIONS  DECIMAL

: FORBID ( <name> -- )      \ Überbrücken unerwünschter
                            \ Worte
  BL WORD CAPS @ IF DUP COUNT UPPER THEN
  CONTEXT @ OVER SWAP HASH TUCK @ (FIND)
  0= ?MISSING >LINK SWAP
  BEGIN 2DUP @ = NOT WHILE @ REPEAT ( 2DUP )
  >R @ R> ! ;

: MEMBERS ( n -- ) 0 ?DO FORBID LOOP ;
                    \ Automatisierung der Überbrückung

SEAL FORTH    \ FORTH ab jetzt einziges Vokabular
```

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

HCC-Forth-gebruikersgroep .

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

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

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

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

```
5 MEMBERS HIDDEN BUG ROOT
  USER ASSEMBLER
8 MEMBERS EMPTY MARK HELLO MENU
  OPTIONS BYE? VOC<DIR> NEW-EDITOR

7 MEMBERS ASSIGN$ FUNCTION OTHER-KEY
  FUNCTAB EDIT EDIT$ PROGRAM$
```

Ende des Verkürzungsprogramms erster Art

Mit **SEAL FORTH** wird hier dem Benutzer nur noch das Vokabular **FORTH** als einziges überlassen. Trotzdem könnte man dann zum Beispiel noch das Vokabular **ASSEMBLER**, das ja im Vokabular **FORTH** definiert ist, aufrufen - und ins Leere stolpern. Daher habe ich oben als erstes alle in **FORTH** definierten weiteren Vokabulare über **MEMBERS** per **FORBID** ausgeschlossen. Die Zahl vor **MEMBERS** muß mit der Anzahl der auf **MEMBERS** folgenden Worte übereinstimmen. Die auf **MEMBERS** folgenden Worte gehören zum Eingangsstrom für die **FORBID**-Schleife und müssen hintereinanderweg ohne Wagenrücklauf eingegeben werden. Notfalls mache man, so wie hier, mehrere **MEMBERS**-Aufrufe.

Die oben in den letzten beiden Zeilen per **FORBID** ausgeschlossenen Worte sind die letzten 15 Worte des **FORTH**-Vokabulars in Turbo-Forth. Weitere unerwünschte Worte können nach demselben Schema beseitigt werden.

Unerwünschte Worte. Speichert man das verbleibende System per **SAVE-SYSTEM** ab, so hat man ein System vom selben Umfang wie vorher (Datei derselben Größe), bei dem aber dem Benutzer die für ihn weniger interessanten Worte verheimlicht werden (die **WORDS**-Liste wird kürzer). Für den Anfänger uninteressant sind beispielsweise alle Worte, die lediglich zum Aufbau des Systems benötigt werden, alle Worte, die den Discompiler **SEE** aufbauen, alles, was zu **DEBUG** gehört, die Umgebung von **WORDS**, die Umgebung von **LIST** usw. Es könnte ein Benutzer den Wunsch haben, sein System vom Gleitkommateil, der bei ihm natürlich mitten im Vokabular **FORTH** steht, zu befreien. Das wäre dann eine vernünftig kurze Aufgabe für meine eben vorgeschlagene Negativ-Auswahl über **FORBID** .

Nur erwünschte Worte. Das unter "Unerwünschte Worte" Gesagte stellt eine negative Auswahl dar. Will man so vorgehen wie bei Ulrich Hoffmann [6], der ein Minimal-Forth aufbaut, das dem Benutzer (dort für Lehrzwecke) nur eine kleinstmögliche Anzahl von Worten (zum gefälligen Selbststudium) überläßt, dann ist die eben erwähnte Vorgehensweise doch recht umständlich. Man muß sich dabei stark überlegen, was man alles weglassen will. Die anzugebende Negativ-Liste wird viel zu lang. Einfacher wäre es, wie in [6] in dieser Liste nur die erwünschten Worte angeben zu können, also eine positive Auswahl zu treffen. Hierzu mache ich den folgenden Vorschlag.

Verfahren wie bei LEARN. Turbo-Forth enthält das Wort **LEARN** . Für jedes Vokabular, ich beschränke mich hier auf **FORTH**, ist eine DOS-Datei **XXX.VOC**, hier **FORTH.VOC**, vorgesehen. **LEARN** geht sämtliche Worte des Vokabulars durch und versieht das Viewfeld eines jeden Wortes mit dem Byte-Offset der Stelle in **FORTH.VOC**, in welcher die Erklärung zu dem betreffenden Wort steht. Ruft man **HELP** <wort> auf, so wird die Erklärung eingelesen und am Bildschirm angezeigt.

Dieses **LEARN** habe ich in meinem Transputer-Forth-System F-TP 1.00 nicht nur einfach übernommen, sondern auch noch mit "statistischen" Zählfunktionen ausgestattet. Im Viewfeld hat F-TP 1.00 vier Bytes zur Verfügung, mehr als genug, um Byte-Offsets einer jeden **FORTH.VOC**-Datei vernünftiger Länge festzuhalten. Es bleibt noch Raum für Kennzeichnungen anderer Art. So habe ich beispielsweise zu jedem Wort die Anzahl der Doppelt- oder Mehrfachdefinitionen (können ja vorkommen) festgehalten.

Negativkennzeichnung auch für 16-Bit-Systeme. In Turbo-Forth und anderen 16-Bit-Systemen würde eine Kennzeichnung der eben angedeuteten Art nicht gehen: Es bleibt kein Platz in den zwei Bytes. Eine einigermaßen vernünftige **FORTH.VOC**-Datei benötigt alle 16 Bits für den Byte-Offset. Außerdem wäre es programmiertechnisch sowieso viel zu umständlich, wenn man versuchen wollte, bei der "Verkürzung" des Systems den **HELP**-Mechanismus nicht anzutasten. Ich verwende einfach eine einleuchtende Hauruckmethode und kümmere mich (zunächst einmal) um den **HELP**-Mechanismus überhaupt nicht:

Durch Aufruf von **VIEWSOFF** werden **alle** Worte im Viewfeld mit 0 gekennzeichnet. Dann werden über das "Listen-Wort" **MEMBERS** (Vorsicht, anders als im Verkürzungsprogramm erster Art !) alle **zugelassenen** Worte im Viewfeld **ON**-geschaltet. Schließlich werden durch Aufruf von **ADMIT** (einer Mischung aus **FORBID** und **LEARN**) alle Worte mit 0 im Viewfeld link-überbrückt. Der **HELP**-Mechanismus wird, so erwünscht, durch ein abschließendes Aufrufen von **LEARN** nachträglich wieder eingebunden. Natürlich müssen dazu in der positiven Auswahlliste **HELP** und **LEARN** mit aufgenommen worden sein.

Verkürzungsprogramm zweiter Art

```
ONLY FORTH DEFINITIONS  DECIMAL

: CELLS    ( -- ) 2 * ;   \ ANS-Zugeständnis,
                          \ in Turbo-Forth unbekannt

: VIEWSOFF ( -- ) CONTEXT @ #THREADS 0
                  \ Viewfelder im Vokabular CONTEXT auf 0
  DO DUP I CELLS + BEGIN @ ?DUP WHILE DUP
  LINK> >VIEW OFF REPEAT LOOP DROP ;
```

```
: MEMBERS ( n -- ) 0 ?DO ' >VIEW ON LOOP ;
                \ n Worte zur Zulassung kennzeichnen


: ADMIT ( -- )          \ Link-Überbrückung der nicht
                        \ zugelassenen Worte
   #THREADS 0       \ In Turbo-Forth 4 HASH-Stränge
  DO CONTEXT @ I CELLS + DUP @
                              \ lfa des letzten Strang-Wortes
     IF DUP @ BEGIN DUP LINK> >VIEW @
     0= WHILE   @ REPEAT OVER ! THEN DROP
  LOOP              \ Schleife zur Anpassung der lfas der
                    \ letzten Strangworte in CONTEXT
  #THREADS 0    \ In Turbo-Forth 4 HASH-Stränge
  DO CONTEXT @ I CELLS + @     \ lfa des letzten
                                 \ Strang-Wortes
     BEGIN ?DUP WHILE DUP @ LINK> >VIEW @
     0= IF DUP @ @ OVER ! ELSE @ THEN REPEAT
  LOOP ;            \ Schleife zum Ausklinken der Worte
mit                   \ 0 im Viewfeld

VIEWSOFF SEAL FORTH       \ Viewfelder auf 0 und
                          \ FORTH ab jetzt einziges Vokabular

26 MEMBERS ! * */ */MOD + +! - / /MOD 0< 0= 0>
1+ 1- 2+ 2- 2/ < = > >R ?DUP @ ABS AND C!

14 MEMBERS C@ CMOVE CMOVE> COUNT D+
DEPTH DNEGATE DROP DUP EXECUTE EXIT
FILL I J

17 MEMBERS MAX MIN MOD NEGATE NOT OR
OVER PICK R> R@ ROLL ROT SWAP U< UM* UM/
MOD XOR

 8 MEMBERS BUFFER CR EMIT EXPECT KEY
SPACE SPACES TYPE

16 MEMBERS # #> #S #TIB ' ( -TRAILING . .( <#
>BODY >IN ABORT BASE CONVERT DECIMAL

14 MEMBERS DEFINITIONS FIND FORGET FORTH
FORTH-83 HERE HOLD PAD QUIT SIGN SPAN TIB
U. WORD

15 MEMBERS +LOOP , ." : ; ABORT" ALLOT BEGIN
COMPILE CONSTANT CREATE DO DOES> ELSE IF

11 MEMBERS IMMEDIATE LEAVE LITERAL LOOP
REPEAT STATE THEN UNTIL VARIABLE VOCABU-
LARY WHILE

12 MEMBERS [ ['] [COMPILE] ] SAVE-SYSTEM BYE
LIST .S WORDS DUMP HELP LEARN

ADMIT LEARN
```

Ende des Verkürzungsprogramms zweiter Art

Die hier angegebene Positivliste ist im wesentlichen die von Ulrich Hoffmann in [6] angegebene Wunschliste. Turbo-Forth ist nicht blockorientiert. Weggelassen wurden also die in Turbo-Forth nicht mehr enthaltenen Worte **BLK BLOCK FLUSH UPDATE SAVE-BUFFERS** . Zusätzlich aufgenommen wurden **DUMP HELP LEARN** .

Kleinste erzeugende Wortmenge. Ray Allwright schrieb kürzlich (1998) in Forthwrite [1]: "Ein Punkt, den ich interessant finde, ist die Suche nach der kleinsten Zahl von **CODE**-Definitionen, mit denen sich ein lebensfähiges Forth aufbauen läßt." Das ist genau die Frage, die ich in meinem Leserbrief 1995 in der Vierten Dimension [2] gestellt hatte. Rafael Deliano [4] versuchte eine Antwort, ich präzisierte meine Frage [3]. Die Frage selbst liegt auf der Hand. Auf was man wartet, sind Begriffsbestimmungen und zufriedenstellende Antworten. Al Moreira hat in der allerersten Forthwrite (1981) sein System AlsForth [7] beschrieben, in dem er eben diese Frage (zwar nicht schlüssig beantwortet, aber) konstruktiv angeht: Er verwendet 26 (55) Worte (deren Herkunft nicht weiter erläutert wird) als Grundstock zum Aufbau seines Systems, das zum erklärten Nebenziel hat, ohne Vorwärtsreferenzen auszukommen. 25 (oder so um die Gegend herum) scheint die magische Zahl für erzeugende Minimalsysteme zu sein. Sie tritt auch im Titel einer Arbeit von Harris in der BYTE von 1980 [5] auf, die sich mit demselben Fragenkomplex beschäftigt. Ray Allwright [1] weist darauf hin, daß eForth 1.00 mit 31 **CODE**-Definitionen auskommt. Außerdem erwähnt er in [1] einen Forum-Beitrag von Bernd Paysan [8] aus dem Jahre 1997, der mit 16 Worten auskommt.

Die Vorschläge von Ulrich Hoffmann [6] oder die im vorliegenden Artikel gemachten Vorschläge können als Werkzeug zur experimentellen Erforschung des Themenkreises "Minimales Forth-Erzeugendensystem" betrachtet werden.

[1] Allwright, R.: From the ´Net - Minimal Word Sets, Forthwrite 95 (1998), S.28-32.

[2] Behringer, F.: Kleinstes Forth?, Vierte Dimension 11 (1995), Heft 3, S.6.

[3] Behringer, F.: Nochmal ´kleinstes Forth´, Vierte Dimension 12 (1096), Heft 1, S.7.

[4] Deliano, R.: Leserbrief zu ´kleinstes Forth´, Vierte Dimension 11 (1995), Heft 4, S.6.

[5] Harris, K.: FORTH Extensibility or How to Write a Compiler in 25 Words or Less, BYTE, August 1980, S.169-181.

Von: hansoft@bigfoot.com (Hans 'the Beez' Bezemer)
Firma:HanSoft & Partners

Hi!

I've made a page in my website presenting all the Integrated Developer Environments available for 4tH. Including one with syntax coloring.

Hans

Turbo-Forth, Conways "Life" und die DNF als IF-Bedingung

von Fred Behringer
Planegger Str. 24, 81241 München

IF-Bedingungen, die von mehreren Booleschen Variablen abhängen, welche über **AND**, **OR** und unter Einbeziehung von **NOT** verknüpft werden, sind meist recht umständlich zu programmieren und ungerechtfertigt hoch im speichermäßigen und zeitlichen Aufwand. Verschachtelte **IF-THEN**-Konstruktionen tragen auch nicht gerade zur Verschönerung oder Erhellung bei. Ich gehe von bis zu acht **ON-OFF**-Variablen, zum Vektor x zusammengefaßt, aus, betrachte alle möglichen **ON-OFF**-Funktionen s(x) dieser Variablen in Disjunktiver Normalform und mache Vorschläge zur Forth-Implementation, um s(x) schnell zu charakterisieren und in **IF-THEN**-Konstruktionen einbauen zu können. Erweiterungen auf beliebig (endlich) viele Eingangsvariablen sind vorgezeichnet.

Stichworte: Turbo-Forth, Life, Boolesche Funktion, DNF, IF ... THEN

Einstimmung. In Conways Life-Spiel wird ein Individuum geboren, wenn genau drei Nachbarn vorhanden sind. Es bleibt am Leben, wenn zwei oder drei Nachbarn existieren. In allen anderen Fällen stirbt es. **Nachbarn** heißt hier und im folgenden unmittelbare Nachbarn. Es gibt 8 Nachbarn. Die Verhältnisse sind in arithmetisierter Form leicht überschaubar. Man braucht nur abzuzählen. Schwieriger wird es, wenn die Nachbarn namentlich unterschieden werden und die Bedingungen nicht nur von deren Gesamtzahl, sondern in unterschiedlicher Weise von einzelnen Gruppierungen abhängen.

Der Fragenkomplex "Life" (Zyklen, Pulsare, Oszillatoren, Gleiter, stabile Figuren, Schlucker, Segler, Tetrominos, Kanonen) gehört in die Theorie der **Zellulären Automaten** (8 Nachbarn = **Mooresche Nachbarschaften,** 4 Nachbarn = **Von-Neumannsche Nachbarschaften**). Life scheint von John Horton Conway zu stammen [15] und von Martin Gardner [6] zum ersten Mal publik gemacht worden zu sein [15]. Interessant ist auch, was in [15] zum historischen Hintergrund gesagt wird.

Life in Forth und woanders. Life-Implementationen in Forth findet man bei Thomas Almy [1], Dave Boulton [4], Coos Haak [9], Friederich Prinz [16], C.H. Ting [21] und Rick VanNorman [22]. Mathematisch, im Zusammenhang mit der Chaos-Theorie, ist mir Life in [14] aufgefallen. Dort werden alle möglichen Abwandlungen im Rahmen der Theorie der Zellulären Automaten besprochen und viele Literaturhinweise gegeben. Lesenswert sind auch die 8 Seiten "Life" im Buch [5] des Nobelpreisträgers Manfred Eigen und seiner Mitarbeiterin Ruthild Winkler, in welchem dieser sich aus der Sicht des Biochemikers über Selbstorganisation, Überlebenskampf, Selbstreproduktion und Selbstreparatur (ohne große Mathematik, sehr anschaulich) Gedanken macht. Eine kurze Erwähnung von Life und ein Turbo-Pascal-Programm für einen Modulo-2-Automaten, der in gewisser Verwandtschaft zu Life steht, findet man in [10]. Life-Programme in BASIC stehen in [11],[15], in BASIC-aufrufbarer 6502-Maschinensprache für den VC20 in [20], für eine CRAY-MP8/832-Hochleistungs-Maschine, aber auch für den PC, ganz allgemein (zelluläre Automaten) in ANSI-C in [8], für Intel-Assembler in [7], für Pascal in [13].

Boolesche Darstellung. Die herkömmliche Aussagenlogik [3], Shannons Schaltalgebra [18],[23],[24] und die auf Moore and Shannon zurückgehende Zuverlässigkeitstheorie [12] bauen mathematisch gesehen auf ein und denselben Formalismus auf. Ich halte mich an die in der Systemtheorie der Zuverlässigkeit (intakt / nicht intakt) übliche Darstellungsweise [2]. Ich numeriere die Nachbarn durch und bezeichne den i-ten Nachbarn mit x_i. Jeder Nachbar kann zwei Zustände (**Daseinswerte)** haben: existent (1) oder nicht existent (0): $x_i \in B := \{0,1\}$. Das Individuum selbst kann ebenfalls die beiden Daseinswerte 0 oder 1 (aber nur die) annehmen. Diese hängen von den Daseinswerten der Nachbarn ab. Die Abhängigkeit beschreibe ich durch die **Daseinsfunktion** $s : B^n \rightarrow B$. Hier stellt B^n die Menge aller **Daseinsvektoren** $x := (x_1 ,..., x_n)$ dar. Als Verknüpfungen werden in B Konjunktion und Disjunktion zugelassen, ergänzt durch die Negation. Ich erkläre diese (auf die "Wahrheitswerte" 0 und 1 wirkenden) logischen Operationen arithmetisch: Ich fasse B als Teilmenge von Z (Menge der Ganzzahlen) auf (0 und 1 als ganze Zahlen interpretiert), lasse die übliche Arithmetik in Z zu und definiere die **Konjunktion** durch $x_1 \wedge x_2 : \Leftrightarrow x_1 * x_2$, die **Disjunktion** durch $x_1 \vee x_2 :\Leftrightarrow x_1+x_2 - x_1*x_2$ und die **Negation** durch $\neg x_1 := 1-x_1$, vorsichtshalber (um bei Bedarf logische und arithmetische Operationen mischen zu können) jeweils immer gleich für alle $x_1,x_2 \in Z$. Es ist klar, daß die so definierten logischen Operationen bei Beschränkung auf B als Argumentmenge nicht aus B hinausführen.

Disjunktive Normalform. Es gibt bei n Komponenten $x := (x_1 ,..., x_n)$ die stattliche Zahl von 2^n Ja-Nein-Entscheidungskombinationen. Bei 8 Komponenten x_i sind das 256 Möglichkeiten. Jede Belegungskombination für x (ich sage **Daseinsbelegung**) kann zu s(x) = 1 oder s(x) = 0 führen, je nachdem, welche Daseinsfunktion s man gerade betrachtet.

Eine **Vollkonjunktion** ist die konjunktive Verknüpfung aller n Daseinskomponenten, wobei jede einzelne Komponente entweder affirmativ (x_i) oder negiert ($\neg x_i$) auftreten kann. Die **Disjunktive Normalform** (**DNF**) einer Daseinsfunktion $s : B^n \rightarrow B$ ist ihre Darstellung als Disjunktion von Vollkonjunktionen derart, daß für jede Belegung von x, die zu s(x) = 1 führt, genau eine Vollkonjunktion den Wert 1 annimmt. Man erhält alle Vollkonjunktionen der DNF von s, wenn man zu jedem x mit s(x) = 1 alle Indizes i durchgeht und die i-te Komponente in der Form x_i in die Vollkonjunktion aufnimmt, falls bei der speziellen Belegung $x_i = 1$, und in der Form $\neg x_i$, falls $x_i = 0$. Man kann die möglichen Vollkonjunktionen durchnumerieren, indem man zu einer jeden von ihnen die zugeordnete Daseinsbelegung als Index in Binärdarstellung interpretiert. Es gibt 2^n nichtnegative n-ziffrige Binärzahlen. Dementsprechend gibt es 2^n mögliche Vollkonjunktionen. Es gibt also $2^{(2\wedge n)}$ verschiedene Daseinsfunktionen s bei Abhängigkeit von n Komponten. Bei n = 8 sind das 2^{256} Möglichkeiten, eine unüberschaubar große Zahl.

Life-Regeln. Wie man sich leicht überlegt, kann man die oben angegebenen Regeln des Life-Spiels auch wie folgt festlegen: Der Daseinsvektor x hat 9 Komponenten. Eine davon ist das Individuum, die anderen sind seine 8 Nachbarn. Damit werden die Daseinswerte von Individuum und Nachbarn vor dem Übergang von einer Generation zur nächsten beschrieben. Der Wert s(x) der Daseinsfunktion s stellt den Daseinswert des Individuums nach dem Übergang zur nächsten Generation dar. s (x) = 1, wenn für genau 3 Indizes $x_i = 1$ gilt, und zwar gleichermaßen für die Fälle, daß dabei das Individuum mitgerechnet wird oder auch nicht. In allen anderen Fällen gilt s(x) = 0. Die Darstellung von s als DNF überlasse ich den Lesern.

Ziel. Soviel zur Einleitung. Ich möchte jetzt ganz allgemein solche s(x) bei sagen wir n = 8 Daseinskomponenten als Eingangsbedingung für eine **IF-THEN**-Abfrage verwenden und eine nicht zu aufwendige Implementation angeben. Denkbar sind weitere Implementationsmöglichkeiten, die herauszufinden und vorzuschlagen ich (ähnlich wie das damals beim **CASE**-Wettbewerb in der Forth Dimensions gemacht wurde) hiermit zur Aufgabe stellen möchte.

Konventioneller Aufwand. Man könnte leicht versucht sein, den Aufwand bei konventioneller **AND-OR-NOT**-Programmierung zu unterschätzen. Manche s (die meisten, die vorkommen?) sind so symmetrisch, daß einige wenige **AND**s, **OR**s und **NOT**s genügen. Andere s sind das ganz bestimmt nicht. Man finde zu jedem s die Minimalzahl der benötigten drei genannten Operationen heraus oder/und gebe dasjenige (diejenigen) s an, bei denen die Minimalzahl am größten ist. Ist das letztere s eindeutig bestimmt? Natürlich nicht (?) Aber vielleicht bis auf Vertauschungen der Daseinskomponenten x_i? Natürlich braucht man bei **AND-OR-NOT**-Behandlung nicht unbedingt über die Disjunktive Normalform (siehe unten) zu gehen. Dennoch: Nach einer Abschätzung von Shannon [19] kann es allgemeine (nicht notwendig reihen-parallele) Zusammenschaltungen von Schaltern (mit den beiden Zuständen "offen" oder "geschlossen") geben, die zu ihrer Verwirklichung mindestens $2^{n-1}+18$ Schaltelemente benötigen. Bei n = 8 sind das immerhin 146 Elemente. Entsprechend groß ist der Aufwand an **AND**s, **OR**s und **NOT**s, zumal man ja im allgemeinen erst noch eine Umwandlung (der Ersatzschaltung als unmittelbar gegebener Ausgangsform) in eine brauchbare Reihen-Parallel-Form vornehmen muß.

Einfacheres Beispiel. Bevor ich zu meinen Implementationsvorschlägen komme, möchte ich gern Life und die Daseinsfunktion noch einmal anhand eines Beispiels beleuchten, das nur 5 Daseinskomponenten hat, weitgehend symmetrisch aufgebaut ist, trotzdem sich für **AND-OR-NOT**-Behandlung noch recht komplex ausnimmt, einen anderen als den Mooreschen Nachbarschaftsbegriff zu betrachten gestattet, mir erlaubt, auf mein Lieblingsthema Transputer zurückzukommen, und sich in der "Ersatzschaltung" als "Brücke" entpuppt (das einfachste Schalternetzwerk mit genau einem Eingang und genau einem Ausgang, das keine "Reihenparallelschaltung" ist).

Transputertopologie. Ein Transputer (z.B. T800) hat vier "Links" als Verbindung zur "Außenwelt". Es liegt also nahe, Trans-puter-Verbände zu betrachten, bei denen jedes "Individuum" (jeder einzelne Transputer) vier Nachbarn hat.

(1)

(4) (3) (5)

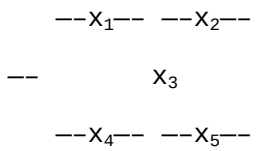
(2)

Transputer (3) mit vier Nachbarn. Von-Neumannsches Nachbarschaftsmodell

Überlebensregeln. Für das Überleben (Erhalt der Funktionstüchtigkeit des Transputer-Programms) gelte: (a) Wenn eine ganze Diagonale (oben-Mitte-unten oder/und links-Mitte-rechts) funktioniert, dann bleibt das Programm des Individuums (Mitte) funktionstüchtig. (b) Wenn das Programm des Individuums (Mitte) nicht mehr funktioniert, aber die Diagonalnachbarn (oben-unten oder/und links-rechts) funktionieren, dann korrigiert sich der Fehler und das Individuum (Mitte) wird wieder funktionstüchtig (lebt wieder auf). (c) Wenn das Individuum funktioniert und die Nachbarn oben-rechts oder/und unten-links funktionieren, dann funktioniert das Individuum weiterhin (bleibt am Leben). (d) In allen sonstigen Fällen bleibt das

Programm des Individuums (Mitte) funktionsuntüchtig, wenn es auch vorher funktionsuntüchtig war, und stirbt ab, wenn es vorher funktioniert hat.
Vernünftig? Abwegig sind diese Regeln (Selbstreparatur von Transputer-Programmen) sicher nicht. Die Natur macht etwas Ähnliches mit der DNS, die ja ständig der Höhenstrahlung vom Weltraum ausgesetzt ist. Es "wird die Doppelstrangstruktur der DNS-Helix zur ´Dunkelreparatur´" (eine besondere Form der Selbstreparatur) "von Mutationsschäden ausgenutzt" ([17]). Was stört, ist die Unsymmetrie unter (c). Sie führt zu Interpretationsschwierigkeiten. Aber was soll´s, Conways Regeln sind auch nicht unbedingt zwingend vorgegeben.

Ersatzschaltung. Stellt man sich das Intakt- oder Nichtintaktsein der Programme der beteiligten Transputer (Individuum und vier Nachbarn) in einer elektronischen Ersatzschaltung als Schalter (intakt = geschlossen, nicht intakt = offen) vor, dann gelangt man zu einer Brückenschaltung mit dem Individuum (ich nenne es aus Symmetriegründen x_3) im Brückenweg und dem "Daseinswert" der Gesamtschaltung als Daseinswert des Individuums nach dem Generationswechsel.



Ersatzschaltung (Strom durchlassend = 1, Strom nicht durchlassend = 0). Schalter offen = 0, Schalter geschlossen = 1.

Reihen-parallel-Schaltung. Diese Brückenschaltung kann man mit Hilfe des Entwicklungssatzes (siehe Lehrbücher der Logik oder Zuverlässigkeitsthorie) in eine Reihenparallelschaltung verwandeln. Mit den üblichen Umformungsregeln (distributive Trivi-alerweiterungen) oder durch wiederholte Anwendung des Entwicklungssatzes gelangt man dann schließlich auch zur DNF. Die DNF kann man natürlich auch über die Wahrheitstafel (Funktionstafel) direkt ermitteln.

Die DNF. Die Wahrheitstafel ist ein systematisches Instrument, mit dem man jede auch noch so unübersichtliche Daseinsfunktion charakterisieren und aufbereiten kann. Jeder möglichen Belegung des Daseinsvektors $x := (x_1, ..., x_n)$ mit Kombinationen von 0-1-Werten wird eindeutig derjenige 0-1-Wert s(x) zugeordnet, den s bei eben dieser Belegung annehmen soll. In die DNF von s werden genau diejenigen Vollkonjunktionen aufgenommen, bei denen die betreffende Belegung x zu s(x) = 1 führt, und zwar mit "x_i" in der Vollkonjunktion, wenn bei der betreffenden Belegung $x_i = 1$, und mit "$\neg x_i$" in der Vollkonjunktion, wenn $x_i = 0$. Man kann die zu einem Vektor zusammengefaßten zugeordneten s(x)-Werte als Binärzahl auffassen und diese (Binär-) Zahlen zur Indizierung der Menge aller möglichen s-Funktionen verwenden. Ich nenne eine Daseinsfunktion s **n-komponentig**, wenn sie von einem Daseinsvektor mit n Komponenten abhängt. Es gibt, auch das läßt sich an der Wahrheitstafel erkennen, 2^{32} 5-komponentige Daseinsfunktionen. Stellt man den "Daseinsindex" hexadezimal dar, dann handelt es sich bei der hier diskutierten Brückenfunktion um $\mathbf{s_{FFA8C888}}$. Ihre Wahrheitstafel lautet:

x_1	**1**	**0**	**1**	**0**	**1**	**0**	**1**	**0**	**1**	**0**	**1**	**0**	**1**	**0**	**1**	**0**	**1**	**0**	**1**	**0**	**1**	**0**	**1**	**0**	**1**	**0**	**1**	**0**	**1**	**0**	**1**	**0**
x_2	**1**	**1**	**0**	**0**	**1**	**1**	**0**	**0**	**1**	**1**	**0**	**0**	**1**	**1**	**0**	**0**	**1**	**1**	**0**	**0**	**1**	**1**	**0**	**0**	**1**	**1**	**0**	**0**	**1**	**1**	**0**	**0**
x_3	**1**	**1**	**1**	**1**	**0**	**0**	**0**	**0**	**1**	**1**	**1**	**1**	**0**	**0**	**0**	**0**	**1**	**1**	**1**	**1**	**0**	**0**	**0**	**0**	**1**	**1**	**1**	**1**	**0**	**0**	**0**	**0**
x_4	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
x_5	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
s(x)	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**0**	**1**	**0**	**1**	**0**	**0**	**0**	**1**	**1**	**0**	**0**	**1**	**0**	**1**	**0**	**0**	**0**	**1**	**0**	**0**	**0**	**1**	**0**

Um die Indizierungsmöglichkeit für den Daseinsvektor $x := (x_1, ..., x_5, ..., x_n)$ nach oben (n) hin offen halten zu können, lese ich die die einzelne Vollkonjunktion als Index charakterisierende Zusammenfassung von 0en und 1en zu einer Binärzahl von unten nach oben (Ziffer der niedrigsten Stelle oben). Ähnlich bei der Zusammenfassung von 0en und 1en zu einer Binärzahl, die den Index abgeben soll, der das gesamte s charakterisiert: Der kleinste Vollkonjunktions-Index steht rechts, so daß sich die Binärzahl, die den Index für s abgibt, in natürlicher Weise lesen läßt, die Ziffer der niedrigsten Stelle rechts. Mit anderen Worten, bei Vergrößerung der Tabelle (Erhöhung von n) wird links und unten angefügt.

Die DNF dieses s hat, wie man aus der Tafel erkennt, 16 Vollkonjunktionen. Der besseren Übersicht wegen schreibe ich y_i für $\neg x_i$. Jeder Spalte mit s(x) = 1 ist eine Vollkonjunktion zugeordnet. Die DNF der "Daseinsfunktion" der "Brücke" lautet:

s(x) =
$x_1 \dot{\wedge} x_2 \dot{\wedge} y_3 \dot{\wedge} y_4 \dot{\wedge} y_5$ Ú $x_1 \dot{\wedge} x_2 \dot{\wedge} x_3 \dot{\wedge} y_4 \dot{\wedge} y_5$ Ú $x_1 \dot{\wedge} x_2 \dot{\wedge} y_3 \dot{\wedge} x_4 \dot{\wedge} y_5$ Ú $y_1 \dot{\wedge} x_2 \dot{\wedge} x_3 \dot{\wedge} x_4 \dot{\wedge} y_5$ Ú
$x_1 \dot{\wedge} x_2 \dot{\wedge} x_3 \dot{\wedge} x_4 \dot{\wedge} y_5$ Ú $x_1 \dot{\wedge} x_2 \dot{\wedge} y_3 \dot{\wedge} y_4 \dot{\wedge} x_5$ Ú $x_1 \dot{\wedge} y_2 \dot{\wedge} x_3 \dot{\wedge} y_4 \dot{\wedge} x_5$ Ú $x_1 \dot{\wedge} x_2 \dot{\wedge} x_3 \dot{\wedge} y_4 \dot{\wedge} x_5$ Ú
$y_1 \dot{\wedge} y_2 \dot{\wedge} y_3 \dot{\wedge} x_4 \dot{\wedge} x_5$ Ú $x_1 \dot{\wedge} y_2 \dot{\wedge} y_3 \dot{\wedge} x_4 \dot{\wedge} x_5$ Ú $y_1 \dot{\wedge} x_2 \dot{\wedge} y_3 \dot{\wedge} x_4 \dot{\wedge} x_5$ Ú $x_1 \dot{\wedge} x_2 \dot{\wedge} y_3 \dot{\wedge} x_4 \dot{\wedge} x_5$ Ú
$y_1 \dot{\wedge} y_2 \dot{\wedge} x_3 \dot{\wedge} x_4 \dot{\wedge} x_5$ Ú $x_1 \dot{\wedge} y_2 \dot{\wedge} x_3 \dot{\wedge} x_4 \dot{\wedge} x_5$ Ú $y_1 \dot{\wedge} x_2 \dot{\wedge} x_3 \dot{\wedge} x_4 \dot{\wedge} x_5$ Ú $x_1 \dot{\wedge} x_2 \dot{\wedge} x_3 \dot{\wedge} x_4 \dot{\wedge} x_5$ für alle $x \in B^n$

DNF-Indizes einfacher Funktionen. Bevor ich zur Besprechung der Programmbestandteile komme, zur Verdeutlichung noch die DNF-Indizes einiger weiterer einfacher Boolescher Funktionen (Zahlenangaben hexadezimal):

$\neg(x_1 Ú x_2)$	**X1 @**	**X2 @**	**OR NOT**	**1**	**$(x_1 Ù \neg x_2) Ú (\neg x_1 Ù x_2)$**	**X1 @ X2 @ XOR**			**6**
$x_1 Ù \neg x_2$	**X1 @**	**X2 @**	**NOT AND**	**2**	**$\neg x_1 Ù (x_2 Ú x_3)$**	**X1 @ NOT X2 @ X3 @ OR AND**			**54**

Bitvektoren. Zur Bearbeitung von DNFs führe ich Bitvektoren ein. Das sind Boolesche Vektoren, die aus einer Aneinanderreihung von Bytes bestehen, deren Bits die Komponenten des Vektors ausmachen. Das definierende Wort **BIT-VECTOR**, angeführt von der gewünschten Zahl an Bitkomponenten als Parameter auf dem Stack, erzeugt, ähnlich wie bei **STRING**, einen Container (eine "Vektorvariable"), sagen wir **XXX (-- ad)**, für eine Bitvektorbelegung. Bei Aufruf von **XXX** wird ad = pfa+2 auf den Datenstack gelegt. In ad-2 ist die in **XXX** maximal verfügbare Zahl von Bytes enthalten, in ad-1 die für die jeweilige Belegung von **XXX** maßgebende Zahl von Komponenten minus 1 (gezählt in Bits). Die Komponentenzählung beginnt bei 0. Bit 0 des Bytes in ad ist Komponente 0 (Bit 0) des gesamten Bitvektors. Die im Byte von ad-1 enthaltene Zahl ist also die Nummer der höchsten momentan vorgesehenen Bitvektorkomponente, die Komponentenzahl-1. Sie läßt sich über **VLEN! (n ad --)** nachträglich verändern (solange dabei die in ad-2 enthaltene maximale Bytezahl nicht überschritten wird) und über **VLEN@ (ad -- n)** auslesen. Die maximal verfügbare Bytezahl läßt sich über **VMAX@ (ad -- n)** auslesen. Das Ganze ist hier auf ein Operieren mit Bitvektoren von bis zu höchstens 8 Bytes ausgelegt. Es sollte aber auch nur gezeigt werden, wie sowas grundsätzlich gemacht werden kann. "Leere" Bitvektoren (0 Bits) sind nicht vorgesehen.

Bearbeitung. Die Bearbeitung von Bitvektoren teilt sich auf in eine Aussonderung eines bestimmten Bytes aus der Folge von Bytes, aus welcher der Vektor besteht, und ein Aufsuchen des gewünschten Bits im betreffenden Byte. Die Bitbearbeitung im Byte habe ich, ähnlich wie bei **CSET** , **CRESET** und **CTOGGLE** , in Intel-Maschinensprache angelegt. Aus den angegebenen Kommentaren heraus müßte es schnell möglich sein, in andere Maschinensprachen oder nach High-Level-Forth hin zu übertragen. Ich darf die Worte der Bitbearbeitung kurz durchgehen.

Bitselektion. BSET (i ad --) setzt Bit i (0...7) im Byte an der Adresse ad (auf 1), **BRESET (i ad --)** setzt es auf 0 und **BTOGGLE (i ad --)** kehrt den Zustand um ($0 \rightarrow 1$, $1 \rightarrow 0$). **B@ (i ad -- fl)** fragt den Zustand von Bit i im Byte an der Adresse ad ab und legt das Ergebnis fl (**TRUE** oder **FALSE**) auf den Stack. **B! (fl i ad --)** holt fl vom Stack (=0 oder <>0) und legt dementsprechend 0 oder 1 auf Bit i des Bytes an der Adresse ad . **[I]B@ (i ad -- fl)** holt das Bit Nr. i (Zählung beginnt bei 0) eines mit **BIT-VECTOR (<name> n --)** erzeugten Bitvektors, der bei Aufruf seine Anfangsadresse ad auf den Stack legt, und liefert auf dem Datenstack **fl = FALSE**, falls das eingeholte Bit = 0 ist, und sonst **fl = TRUE**. **[I]B! (fl i ad --)** setzt Bit i des Bitvektors bei ad auf 0, falls **fl = FALSE**, und auf 1 sonst.

DNF interaktiv erstellen. Über **n BIT-VECTOR XXX** sei der Bitvektor **XXX** der gewünschten Bitlänge n erzeugt worden. Man fasse den Belegungswert s(x) bei (vektorieller) Belegung x einer bestimmten Daseinsfunktion s (siehe Abschnitt "Die DNF") ins Auge. Die DNF zu s beschafft man sich über **XXX MAKE-DNF**. Am Bildschirm werden nacheinander die Belegungskombinationen x_n ... x_1 (in dieser Reihenfolge) ausgegeben. In der Funktionstafel ("Wahrheitstafel") im Abschnitt "Die DNF" würde sich im 20. Interaktionsschritt die Kombination 10011 (gelesen von unten nach oben und in der Spaltenabfolge von rechts nach links) ergeben. Erwartet wird vom Benutzer an dieser Stelle des Beispiels die Eingabe einer 1. Der Bitvektor **XXX** ist nach Beendigung der interaktiven Prozedur mit dem Index der zu s gehörenden DNF belegt. Mit **XXX .IDX-DNF** kann man sich diesen Index byteweise in Blöcken von je zwei Hexziffern jederzeit auf dem Bildschirm ausgeben lassen. Geht es nur darum, in einem schon belegten Bitvektor (nachträglich) nur eine einzige Vollkonjunktion anders als bisher zu berücksichtigen, so kann man das über **DNF! (fl fln ... fl1 n ad --)** erledigen. fl läßt die betreffende Vollkonjunktion aufnehmen oder nicht (1 oder 0 in Abhängigkeit von **fl = TRUE** oder **FALSE**), über fln bis fl1 (in dieser Reihenfolge) wird die zur betreffenden Vollkonjunktion gehörende Belegungskombination (von **FALSE**- und **TRUE**-Werten) eingegeben (**FALSE** für $x_i = 0$ und **TRUE** für $x_i = 1$), n ist die Zahl der (Booleschen) Eingangsvariablen und ad ist die Anfangsadresse des betreffenden Bitvektors. Mit **DNF@ (fln ... fl1 n ad -- fl)** kann man nachprüfen, ob die zur Belegung (x_1, .. x_n) gehörende Vollkonjunktion in der DNF enthalten ist (**fl = TRUE**) oder nicht (**fl = FALSE**). **DNF@** ist also dasjenige Forthwort, über das in dieser Arbeit gesprochen werden sollte und das in **IF-THEN**-Konstruktionen und ähnlichen Abfragen verwendet werden kann.

DNF-Index von und nach DOS-Datei. Der die DNF einer Daseinsfunktion charakterisierende Index nützt wenig, wenn man ihn nicht weiterreichen kann, von Bitvektor zu Bitvektor, von Forth-Programm zu Forth-Programm, vom Quelltext zum Forth-System. Mit **IDX-DNF@ (ad -- c1 ... cs n m)** kann man den Index (einschließlich der Bitvektorlänge-1 und der maximalen Bytezahl) Byte für Byte auf den Stack legen und mit **IDX-DNF! (c1 ... cs n m ad --)** kann man das Ganze wieder vom Stack in denselben oder einen anderen Bitvektor einlesen. Man beachte die Reihenfolge, niedrigstes Byte zuerst. Mit **c1 ... cs n m SAVE-IDX YYY.FTH** gelingt es schließlich, aus einem schon belegten Bitvektor die betreffende Information herauszuziehen und in eine DOS-Datei (hier **YYY.FTH**) abzuspeichern, von wo sie dann später per **INCLUDE YYY.FTH** wieder eingelesen

und auf den Stack gelegt werden kann. (Will man die Endung bei **YYY.FTH** weglassen, muß man **YYY.** schreiben. Andernfalls wird die Datei **YYY.COM** erzeugt, was beim Wiedereinlesen berücksichtigt werden muß.)

Programm-Listing

```
HEX

: BIT-VECTOR ( <name> n -- )          \ Bitvektor mit n Komponenten erzeugen
     CREATE DUP NEGATE 8 / NEGATE     \ Im 1. Byte wird die nächstliegende Bytezahl
     TUCK C, DUP 1- C,                \ >= n/8 abgespeichert, im 2. Byte die Bitzahl-1
     1 100 BETWEEN NOT ABORT" Es muß 0 < Komponentenzahl < 101h sein" 0
     ?DO 0 C, LOOP                    \ Bitvektor wird mit Nullen vorbelegt
     DOES> 2 + ;                      \ Bei Aufruf von <name> pfa+2 auf Stack legen

: VLEN@ ( ad -- n ) 1- C@ 1+ ;        \ n = Komponentenzahl des Bitvektors bei ad

: VLEN! ( n ad -- )                   \ n = Komponentenzahl des Bitvektors bei ad
     SWAP DUP 1- -ROT NEGATE 8 / NEGATE OVER 2 -
     C@ 1 SWAP BETWEEN NOT ABORT" Eingabefehler" 1- C! ;

: VMAX@ ( ad -- n ) 2 - C@ ;          \ n = verfügbare Bytezahl des Bitvektors bei ad

CODE BSET ( i ad -- )                 \ Bit i (0...7) im Byte bei ad setzen
     BX POP  1 # AL MOV               \ ad nach BX holen und Bit 0 (nur Bit 0) in AL setzen.
     CX POP  CH CH XOR  AL CL SHL     \ 1 in Bit 0 um i Stellen (nur 0...7) links-versch.
     AL 0 [BX] OR   NEXT END-CODE     \ Bit i in Adresse ad setzen

CODE BRESET ( i ad -- )               \ Bit i im Byte bei ad zurücksetzen
     BX POP  1 # AL MOV               \ ad nach BX holen und Bit 0 (nur Bit 0) in AL setzen.
     CX POP  CH CH XOR                \ i (welches Bit?) holen(nur 0 bis 7).
     AL CL SHL  AL NOT                \ 1 in Bit 0 um i Stellen nach links und invertieren
     AL 0 [BX] AND  NEXT END-CODE     \ Bit i in Adresse ad zurücksetzen

CODE BTOGGLE ( i ad -- )              \ Bit i im Byte bei ad invertieren
     BX POP  1 # AL MOV               \ ad nach BX holen und Bit 0 (nur Bit 0) in AL setzen.
     CX POP  CH CH XOR  AL CL SHL     \ 1 in Bit 0 um i Stellen (nur 0...7) links-versch.
     AL 0 [BX] XOR  NEXT END-CODE     \ Bit i in Adresse ad invertieren

CODE B@ ( i ad -- fl )                \ Bit i im Byte bei ad holen
     BX POP  1 # AL MOV               \ ad nach BX holen und Bit 0 (nur Bit 0) in AL setzen.
     CX POP  CH CH XOR  AL CL SHL     \ 1 in Bit 0 um i Stellen (nur 0...7) links-versch.
     0 [BX] AL AND  0=                \ Bit aus ad übernehmen. Zero-Flag = 0 ?
     IF 0 # AX MOV ELSE -1 # AX MOV   \ Ja, dann fl = FALSE; nein, dann fl = TRUE
     THEN            1PUSH END-CODE   \ fl auf Stack

CODE B! ( fl i ad -- )                \ fl (0/1) in Bit i im Byte bei ad plazieren
     BX POP  1 # AL MOV               \ ad nach BX holen und Bit 0 (nur Bit 0) in AL setzen.
     CX POP  CH CH XOR  AL CL SHL     \ 1 in Bit 0 um i Stellen (nur 0...7) links-versch.
     CX POP  CL INC  CL DEC  0=       \ fl holen. Zero-Flag = 0 ?
     IF   AL NOT  AL 0 [BX] AND       \ Invertieren und Bit i im Byte bei ad zurücksetzen
     ELSE AL 0 [BX] OR                \ Bit i im Byte bei ad setzen
     THEN         NEXT END-CODE

: [I]B@ ( i ad -- fl )                \ Hole Bit i (gezählt ab 0) vom Bitvektor bei
     DUP 1- C@ >R SWAP                \ ad ; fl = FALSE, falls 0; sonst fl = TRUE .
     DUP 0 R> BETWEEN NOT ABORT" Eingabefehler"  8 /MOD ROT + B@ ;

: [I]B! ( fl i ad -- )                \ Setze Bit i (gezählt ab 0) vom Bitvektor
     DUP 1- C@ >R SWAP                \ bei ad auf 0, falls fl = FALSE, sonst auf 1
     DUP 0 R> BETWEEN NOT ABORT" Eingabefehler"  8 /MOD ROT + B! ;

: DNF@ ( fln ... fl1 n ad -- fl )     \ 0 < n < 9 !!! ad = Bitvektor-Anfang
     >R PAD OFF 0 ?DO I PAD [I]B! LOOP \ fl1 bis fln werden als Zwischenergebnis im Byte
     PAD @ R> [I]B@ ;                 \ auf PAD gesammelt. fl1 = Bit 0, fln = Bit n-1.
```

```
: DNF! ( fl fln ... fl1 n ad -- )      \ 0 < n < 9 !!! ad = Bitvektor-Anfang
     >R PAD OFF 0 ?DO I PAD [I]B! LOOP
     PAD @ R> [I]B! ;                  \ fl wie in DNF@

: MAKE-DNF ( ad -- )                   \ ad = Anfang des DNF-Bitvektors
     BASE @ 2 BASE ! SWAP DUP 1- C@ 1+ \ Basis aufbewahren und Komponentenzahl-1 aus ad-1
     CR ." x[7]...x[0]  RET = Aussprung" 0
     ?DO CR 2 SPACES I S>D
         <# # # # # # # # # #> TYPE    \ Vollkonjunktion anzeigen
         2 SPACES KEY                  \ 0/1-Belegung eingeben
         DUP 0D = IF DROP LEAVE THEN   \ RET = Aussprung
         ASCII 0 <>                    \ "0" = 0, "<>0" = 1
         DUP IF 1 . ELSE 0 . THEN      \ Zur Kontrolle ausgeben
         OVER I SWAP [I]B!             \ In DNF-Bitvektor eintragen
     LOOP   DROP BASE ! ;              \ Basis wiederherstellen

: IDX-DNF@ ( ad -- c1 ... cs n m )     \ DNF-Index byteweise auf Datenstack legen
     DUP 1- C@ 1+                      \ Vektorlänge-1 n in Bits
     NEGATE 8 / NEGATE 0               \ Nur die benötigten s von den
     ?DO DUP I + C@ SWAP LOOP          \ verfügbaren m Bytes auf Stack legen.
     DUP 1- C@ SWAP 2 - C@ ;           \ (ad-1) als n und (ad-2) als m auf Stack

: IDX-DNF! ( c1 ... cs n m ad -- )     \ DNF-Index vom Stack her rekonstruieren
     TUCK 2 - C! TUCK 1- C!            \ m nach ad-2, n nach ad-1
     DUP 1- C@ NEGATE 8 / NEGATE       \ Nach oben aufgerundet
     TUCK + 1- SWAP 0
     ?DO TUCK I - C! LOOP              \ ad+s-1-I (I = 0...s-1)
     DROP ;

: .IDX-DNF ( ad -- )                   \ DNF-Index auf Bildschirm ausgeben
     IDX-DNF@ OVER 1+ >R               \ Komponentenzahl zwischenspeichern
     CR ." Verfügbare Bytezahl:"    4 .R
     CR ." Bit-Komponentenzahl:" 1+ 4 .R
     CR ." DNF-Index, Hexbytes:"
     CR R> NEGATE 8 / NEGATE 0         \ Zahl belegter Bytes, aufgerundet
     BASE @ >R HEX                     \ Basis aufbewahren. Dann in Hex
     ?DO S>D <# # # #> TYPE SPACE LOOP \ mit führenden Nullen ausgeben.
     R> BASE ! ;                       \ Basis wiederherstellen

: SAVE-IDX ( <name> c1 ... cs n m -- ) \ Daten von IDX-DNF@ in Datei speichern
     OVER NEGATE 8 / NEGATE 2 + >R     \ n/8 auf Byte aufgerundet + 2
     HERE >R 200 ALLOT R> R@ OVER >R 0 \ Zwischenspeicher
     ?DO TUCK !      2 + LOOP  DROP    \ Stack auslagern und in
     R> R@ OVER >R 0                   \ umgekehrter Reihenfolge
     ?DO DUP  @ SWAP 2 + LOOP  DROP    \ wieder hereinholen.
     R> R> OVER >R 0
     ?DO SWAP S>D <# #S #>             \ Stackeintrag für Stackeintrag nach ASCII
       >R OVER R@ MOVE                 \ umwandeln und in Zwischenspeicher legen.
       R> + BL OVER C! 1+              \ Zwischenraum (Space) hinzufügen. In Datei
     LOOP   R> SWAP SAVE -200 ALLOT ;  \ speichern und Zwischenspeicher wieder beseitigen.
```

Anmerkung. Gegen Eingabefehler habe ich nur das Gröbste abgesichert. Ich wollte nur das Grundsätzliche aufzeigen.

[1] Almy, T.: LIFE.4TH für ForthCMP (1985).
[2] Barlow, R.E., and Proschan, F.: Statistische Theorie der Zuverlässigkeit, Frankfurt/M. 1978.
[3] Böhme, G.: Einstieg in die Mathematische Logik, Hanser-Verlag München 1981.
[4] Boulton, D.: Life, Forth Dimensions III/5 (1981), S.24
[5] Eigen, M. und Winkler, R.: Das Spiel: Naturgesetze steuern den Zufall, Piper-Verlag München 1975, S.217-226.
[6] Gardner, M.: On Cellular Automata, Self-Reproduction, the Garden of Eden, and the Game of Life, Scientific American 224 (1971), Februar S.112, März S.106, April S.114.
[7] Garfinkel, S.L.: LIFE.ASM in Dr. Dobb's Journal 80 (1983), Juni-Heft.
[8] Gerling, R.W.: Zellulare Automaten auf dem PC, MNU 43/8, Dezember 1990, S.451-456.
[9] Haak, C.: LIFE.FRT für CHForth 1.25 (1994).
[10] Kern, U.: Zelluläre Automaten, CHIP-SPECIAL 22 (1993), S.58-61.
[11] Lee, J.D., Beech, G., and Lee, T.D.: Computer Programs that Work: Fully Tested Mathematics, Science and Games Programs in BASIC,

Sigma Technical Press, Wolverhampton (19??), S.27-32.
[12] Moore, E.F., and Shannon, C.E.: Reliable Circuits Using Less Reliable Relays, Journal of the Franklin Institute 262 (1956) S.191-208 und S.281-297.
[13] Norton, R.: PC_LIFE.PAS (1988) im Verzeichnis \PASCAL\LIBRARY\DOS\GAMES\ auf der Tewi-CD 1 von "Power-Programmierung".
[14] Peitgen, H.-O., Jürgens, H. und Saupe, D.: Chaos: Bausteine der Ordnung, Springer-Verlag 1994.
[15] Perry, K.E.: Abstract Mathematical Art, BYTE, Dezember 1986, S.181-192.
[16] Prinz, F.: LIFE.SEQ für ZF und TCLIFE.SEQ für TCOM (1993).
[17] Schöneburg, E., Heinzmann, F. und Feddersen, S.: Genetische Algorithmen und Evolutionsstrategien, Addison-Wesley, New York 1994, S.83.
[18] Shannon, C.E.: A Symbolic Analysis of Relay and Switching Circuits, Trans. A.I.E.E. 57 (1938), S.713-722.
[19] Shannon, C.E.: The Synthesis of Two-Terminal Switching Circuits, Bell System Technical Journal 28 (1949), S. 73, Theorem 4.
[20] Stolte, J.: Game of Life, Computer Persönlich 19 (1983), S.106-114.
[21] Ting, C.H.: Life, Forth Dimensions 15 (1993), Mai/Juni, S.30-34.
[22] VanNorman, R.: LIFE.4 in seinem "32-bit Protected-Mode Subroutine Threaded Forth" für DOS und OS/2, 1993.
[23] Weyh, U.: Elemente der Schaltungsalgebra, München 1960.
[24] Zemanek, H.: Schaltalgebra, NTF 3 (1956), S.93-113.

Den Gesamtartikel, aus dem man das Programm leicht herausziehen kann, schicke ich auf Anfrage gern elektronisch zu. Turbo-Forth in deutscher Version könnte ich ebenfalls beilegen.

Fred Behringer <behringe@mathematik.tu-muenchen.de>

Werkstatt ... MISC

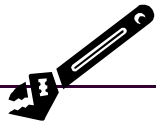
MISC
Minimal Instruction Set Computer

Soeren Tiedemann
Freiburg im Breisgau
tiedema@mail.uni-freiburg.de

Erinnern wir uns kurz: Mit dem NC4000 von Novix gelang erstmalig die Implementation eines Forth-Modells in Hardware im Gate-Array Design mit nur 4000 Gates. Er war einer der schnellsten Chips seiner Zeit.
Darauf aufbauend entwickelte Harris die bekannte RTX-2000 Reihe. Auch der FRP1600 der Fraunhofer-Gesellschaft gehört in diese Familie.
Chuck Moore schritt mit der Entwicklung von 32-bit Prozessoren fort. Der Sh-Boom, wieder in Gate-Array-Technology konnte bis zu 4 8-bit Forth Instruktionen in ein 32-bit Wort packen. Chuck Moore konnte zeigen, daß ein 50 MIPS Forth Prozessor mit nur wenigen Gates aufgebaut werden konnte.
Patriot Scientific entwickelte noch einen Nachfolger des original Sh-Booms, den PSC1000, der auch für neue Sprachen, wie JAVA, hervorragend geeignet ist. Laut Patriot Scientific und der Ineltek (Vertreiber des PSC1000 in Deutschland) kann der PSC1000 38%-40% des JAVA-Byte-Codes hardwaremäßig ausführen. Wir werden noch sehen, daß das Prinzip der Verarbeitung von Opcodes dem des MuP21/F21 sehr ähnlich ist.

Mehrere Generationen von 32-bit FORTH-Chips wurden an der John Hopkins Universitaet von John Hayes, Marty Fraeman, Robert Williams, Sue Lee, Tom Zaremba, Bob Henshaw and Jay Dettmer entwickelt. Einer der letzten, der SC32 wurde im Jahr 1986 produziert. Er entstand aus dem AMD2901 aus den frühen 80ern, wo er für das Hopkins Ultraviolet Telescope zum Einsatz kam. Der SC32 kam für die NASA im Topex Satellite, im Freja Magnetic Field Experiment Magnetometer und im Flare Genesis Balloon Borne Solar Observatory zum Einsatz.

Für Chuck Moore schien das Gate-Array Design ausgeschöpft und er wandte sich der VLSI-Technologie zu und begann eine eigene VLSI-CAD Software zu schreiben. Immer im Kampf gegen unnötige Komplexitaet und für einfache und flexible Systeme versuchte Chuck Moore, sogar seine ganz persönliche Codierungserfindung FORTH zu verlassen, da ihm selbst diese zu komplex erschien. So entstand eine wesentlich simplifizierte Systemform, wobei das simplifiziert den Kern nicht trifft: es war das minimalste mögliche Betriebssystem und diente als Basis der VLSI-CAD Software. OK, so der Name dieses Minimalstsystems war in der Philosophie von FORTH codiert, benutzte aber nicht ein einziges der vorhandenen FORTH-Werkzeuge. OK war nur eine kleine Ansammlung von Maschinensprachedefinitionen, wobei die ersten 4 Versionen nur durch das direkte Hacken des Kernels mit einem Debugger erschaffen wurden. OK interpretierte nicht und compilierte nicht. Letzten Endes, da schloß sich der Kreis wieder, erschuf Chuck Moore ein neues, minimales, hochgradig portables weil so einfaches FORTH-System. Chuck Moore entdeckte auf diesem Weg FORTH für sich neu. Er erkannte, daß er letztendlich ein neues, minimaleres FORTH-System erschaffen hatte. So kehrte er reumütig, wie er selbst sagte, zurück und meinte, daß er FORTH verlassen hatte und FORTH neu schuf und er wahrscheinlich noch in 30 Jahren bei erneuten Simplifizierungsversuchen wieder zu FORTH zurückkehren würde.
Das zugrundeliegende Modell sowohl für OKAD (gespro-



chen Oh CAD) als auch für die damit designten Prozessoren ist das MISC-Modell. Jeff Fox (Ultra Technology) bezeichnete diese ungewöhnliche, aber konsequente Vorgehensweise mit 'Hardware-Metacompiling'. Die Software, mit der die Prozessoren designt wurden, war schon in der Sprache der neuen Prozessoren geschrieben und nur in Form kleiner Maschinenroutinen auf gängigen Prozessoren simuliert worden, bis der erste funktionsfähige MISC-Prozessor, der MuP21 (21-bit MultiProzessor) zur Verfügung stand.

Dem minimalsten Prozessor kommen die neuen MISC-Chips wahrscheinlich sehr nahe: MuP21 und der Nachfolger F21 (im Prototypenstadium) sind Stack-Engines.

Das Prozessormodell:

```
Register             MuP21      F21

Adressregister        A          A

Returnstack          3 Zellen   16 Zellen
Top of Returnstack   R          R
                     --------   ---------
gesamt               4 Zellen   17 Zellen

Top of Datastack     T          T
Next of Datastack    N          N
Datastack            4 Zellen   16 Zellen
                      --------   ---------
gesamt               6 Zellen   18 Zellen

( ProgramCounter     PC         PC)-->intern

Konfigurationsreg.   ---        C
```

Alle Register und Stackplätze umfassen 21-bit. Der Datenbus ist 20-bit breit, der Adressbus 21-bit. Das 21.bit der Register und Stackplätze dient dementsprechend als Carry- oder als Adressbit, je nach Verwendung. Daten werden auf den Top of Stack gebracht (Register T), der vorherige Inhalt von T wird zum Next of Stack (Register N) übertragen und der Wert von N landet auf dem Datenstack. ALU Instruktionen arbeiten mit T und N, Ergebnisse landen in T, Datenstack wird gepop-t nach N.

Maschinenbefehle sind einfach durchnumeriert von 0 bis 31, wobei aber viele noch ungenutzt sind. Es reichen also zur Codierung 5-bit aus. Damit können 4 Maschinenbefehle in ein 20-bit Speicherwort gepackt werden.

Branch- und Jump- Instruktionen

```
CODE NAME DESCRIPTION            AS FORTH
                                 WITH
                                 VARIABLE A

00   jump unconditional Jump      ELSE, RE-
PEAT
01   T=0  Jump if T0-19 zero      DUP IF
02   call push PC+1 to R, jump    :
03   C=0  Jump if T20
          (Carry) zero           CARRY? IF
04
05
06   ;    pop PC from R           ;
07
```

Memory-Access Instruktionen

```
CODE NAME DESCRIPTION     AS FORTH WITH
                          VARAIBLE A

08   @r+  get contents    R@ @  R> 1+ >R
          of address      --> nur F21 !!
          in R to T,
          increment R
09   @a+  get contents     A @ @  1 A +!
          of address
          in A to T,
          increment A
0A   #    get 20-bit      LIT
          inline literal
0B   @a   get contents     A @ @
          of address
          in A to T.
0C   !r+  store T to       R@ !  R> 1+ >R
          address in R,   --> nur F21 !!
          increment R
0D   !a+  store T to        A @ !  1 A +!
          address in A,
          increment A
0E
0F   !a   store T to       A @ !
          address in A.
```

ALU Instruktionen

```
10   com  complement T    -1 XOR
11   2*   shift T,        2*
          0 T0
12   2/   shift T,        2/
          T20 to T19
13   +*   add N to T      DUP 1 AND IF OVER +
          if T0 one        THEN
          preserve N
14   -or  exclusive-or    XOR
          N to T
15   and  and N to T      AND
16
17   +    add N to T      +
```

Stack Instruktionen

```
18   pop  pop R,          R>
          push into T
19    a@    push T,         A @   --> alter-
nativ
          copy A to T     auch nur: a
1A    dup   push T into T   DUP
1B    over  push N into T   OVER
1C    push  pop T,          >R
          push into R
1D    a!    pop T into A     A !   --> muss
bei
                            MuP21 mit nop a!
                            codiert werden
1E    nop   no operation    NOP
1F    drop  pop T           DROP
```

20-bit Speicherworte:

```
bit  20  ....15 ....10 ....5 ....0
       slot0  slot1  slot2 slot3
```

Wie schon gesagt, werden 5-bit Instruktionen in ein Speicherwort gepackt:

z.B.: dup dup -or com

Ausnahmen bilden die Branch- und Jump- Instruktionen (nicht ;). Wird eine dieser Instruktionen in slot0 gefunden, so werden die Bits 0-14 als Sprungadresse interpretiert.
Eine Instruktion dieser Klasse kann auch noch in slot1 erscheinen, dann werden die Bits 0-10 als Adresse verwendet. slot0 steht dann noch für eine beliebige andere Instruktion zur Verfügung. Für einen 21 bit Sprung muss die Befehlsfolge:

push ;

verwendet werden. (push T into R and Ret)

Durch den Befehl # (Literal) wird wird ein volles 20-bit Literal aus dem näechsten Speicherwort (PC+1) nach T gepush-t und der PC dementsprechend justiert:

```
x          #      #      #      #
x+1        1000
x+2        2000
x+3        3000
x+4        4000
x+5        drop   drop   drop   drop
```

Die 4 # aus dem Speicherwort x würden nacheinander die Inhalte von x+1, x+2, x+3 und x+4 auf den Stack bringen, jedes # wird den PC adjustieren und letztendlich wird bei x+5 fortgefahren und in diesem Beispiel werden alle Werte wieder weggeschmissen.

Addition:
F21 benutzt einen ripple carry Mechanismus. Eine Addition muß 'nop +' codiert werden, damit sich das Carry-bit ueber 4 Stellen bewegen kann. ' nop nop +' ist für eine Bewegung von 6 Stellen nötig, 'nop nop nop +' wird für 8 Stellen gebraucht. Diese Besonderheit entfällt, wenn die Additionen im ersten slot stehen (slot0). Denn es geht in diesem Fall ja der Instruction prefetch voraus, der dem Carry-bit genügend Zeit gibt, durch alle 20 Stellen zu laufen. Daher muss die Addition im ersten slot stehen, wenn das Carry auf jeden Fall gebraucht wird! Siehe Timing.

Timing:
Jede Instruktion wird in 2 Nanosekunden ausgefuehrt. Nun muss noch, je nach Speicherart verschieden, eine Memory Access Zeitspanne dazuaddiert werden. (Für den nächsten Instruction fetch). Memory Access Time gliedert sich in die Memory Setup Time (8ns) und die dann tatsächliche Zugriffszeit SRAM 12-40ns, DRAM 35-60ns onpage (112-200 ns offpage) etc.
F21 beginnt mit dem nächsten Instruction Fetch, wenn kein anderer Speicherzugriff 'hängt'. Dies bedeutet:

```
x          dup    dup    dup    dup
x+1        ...    ...    ...    ...
```

2ns werden für jedes dup benötigt. Instruction prefetch von x+1 beginnt gleichzeitig mit dem ersten dup. Nach 8ns ist die Befehlsfolge abgearbeitet, gleichzeitig hier auch schon die Memory Setup Time abgeschlossen, es vergehen also nach dem letzten dup noch zB.: 12ns (SRAM) um den Prefetch von x+1
zu beenden. --> 20ns

```
x          a!     @a+    dup    dup
x+1        ...    ...    ...    ...
```

In diesem Beispiel: 2ns für das a!, 2ns für das @a+. Da @a+ selbst ein Speicherzugriff ist, benötigen wir jetzt 8ns Memory Setup Time, 12ns Zugriffszeit (bei SRAM). Wenn @a+ beendet ist, also nicht mehr 'hängt', dann beginnt gleichzeitig mit dem ersten dup die Memory Setup Time für den nächsten Instruction prefetch. 2ns für das erste dup, 2ns für das letzte dup, 4ns Rest der Memory Setup Time und 12ns (wieder SRAM angenommen) für die Beendigung des Instruction prefetch bei x+1.

--> 2+2+8+12+2+2+4+12 : 44ns.

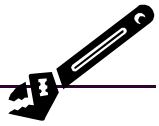
Beispiele und Stackkommentare:
Der Standard für die Stackkommentare sieht momentan folgendermaßen aus:

Register: A RS: ... ... R DS: T N ... ...
Als erstes kommt der Inhalt des Adressregisters A. Als nächstes folgt von hinten der Returnstack, der mit dem Top of Returnstack R abschließt. Nun kommt der Top of Stack T, der Next of Stack N und der Rest des Datenstacks.

```
...  ...  ...  ...  A: --  RS: --  DS: x a b
dup  nop  nop  nop  A: --  RS: --  DS: x x a b
push nop  nop  nop  A: --  RS: x   DS: x a b
a!   nop  nop  nop  A: x   RS: x   DS: a b
!a+  nop  nop  nop  A: x+1 RS: x   DS: b
!a+  nop  nop  nop  A: x+2 RS: x   DS: --
a@   nop  nop  nop  A: x+2 RS: x   DS: x+2
pop  nop  nop  nop  A: x+2 RS: --  DS: x x+2
```

Dieses Beispiel soll nur für einige ausgesuchte Befehle die Stackoperationen verdeutlichen.
Das Wertepaar a,b wird an x und x+1 gespeichert, und x und x+2 wird auf dem Stack hinterlassen. Übliche FORTH-Notation für dieses Beispiel waere:

```
: ab! ( b a x -- x+2 x )
   ...
;
```



Natürlich schreibt man ernstzunehmender :

a)

```
...  ...  ...  ...  A:  --  RS: --  DS: x a b
Dup  push a!  nop  A:  x   RS: x   DS: a b
!a+  !a+  a@  pop  A:  x+2 RS: --  DS: x x+2
```

Es ergeben sich:

```
                    2+2+2+2+12=20 ns
   2+8+12+2+8+12+2+2+4+12=64 ns
   ──────────────────────────────
                                84 ns
```

Zum Vergleich:

b)

```
...  ...  ...  ...  A: --  RS: --  DS: x a b
dup  push a!  !a+  A: x+1 RS: x   DS: b
!a+  a@  pop  nop  A: x+2 RS: --  DS: x x+2
```

Es ergeben sich:

```
              2+2+2+2+8+12+8+12=48 ns
              2+8+12+2+2+2+2+12=42 ns
          ──────────────────────────
                                90 ns
```

Variante a) ist durch das Einfügen des nop vom Kommentar aus gesehen wesentlich besser lesbar und um 8ns schneller. Der Grund liegt in dem Speicherzugriff im letzten slot bei b) (!a+). Erst wenn dieser beendet ist, kann mit dem prefetch begonnen werden.

Es ist ganz klar ersichtlich, wie der Speicherzugriff den Prozessor ausbremst. Achtet man jetzt darauf, möglichst viele Speicherzugriffe zu vermeiden und durch sequentielle Stackoperationen zu ersetzten, führt dies zu einigen Besonderheiten:

```
dup    dup    -or    ...  Wert in T: 0

dup    dup    -or    com  Wert in T: -1

dup    dup    -or    com  Wert in T: -2
2*     ...    ...    ...

dup    dup    -or    com  Wert in T: 1
2*     com    ...    ...
```

Ein Wert, der mit sich selbst ge-XOR-t wird, ergibt selbstverständlich 0. Das Einerkomplement von 0 ergibt -1. Durch einen bit-shift nach links (2*) erhalte ich aus -1 die -2, die nach Einerkomplementierung +1 ergibt. Diese Beispiele sind auf jedenfall schneller als der Literalbefehl #, da hier ja ein Speicherzugriff (wenn auch im selben Speicherbereich) ausgeführt werden muß. (Gilt hauptsächlich fuer 0 und -1)

Besondere Befehlsfolgen (Makros) für häufige FORTH-Operationen:

```
a!     @a     ...   ...   @
a!     !a     ...   ...   !
dup    dup    -or   ...   0
dup    dup    -or   com   -1
over   com    and   -or   OR
a!     push   a@    pop   SWAP
#      (com)  push  ;     long_jump
```

Zu erläutern sind jetzt noch die Makros für OR und SWAP. F21 verfügt über kein OR. -or oder XOR löscht aber 2 gesetzte Bits, die das OR erhalten würde.

```
       DS: a b
over       b a b
com        NOT(b) a b
and        (NOT(b) AND a) b
-or        (a OR b)
```

Vor dem -or muß sichergestellt werden, daß keine Bits an gleicher Stelle in a und b gesetzt sind. Invertiere ich jetzt eine Kopie von b, und AND-e sie mit a, so stelle ich sicher, daß in a alle die Bits, die im Original-b gesetzt sind auf jedenfall gelöscht werden. Nun kann ich doch das -or verwenden.

Das zweifellos schnellste SWAP ist einfach ein over:

```
       DS: a b
over       b a (b)
```

Allerdings muß ich hierbei beachten, daß ich keine weiteren Werte auf dem Stack benötige. Das einfache Vergessen des zweiten b im obigen Beispiel führt natürlich dazu, daß ich an tiefere Werte nicht mehr herankomme, es sei denn, die ersten Operanden werden von einer Funktion vernichtet, dann muß ich nach Beendigung noch das (b) vernichten. Dies kann aber zu sehr unleserlichem Code führen. Die sauberste Lösung ist also allemal:

```
a! push a@ pop
```

Nun kann es aber sein, daß das Adressregister A nicht frei ist, also für diese Tauschaktion nicht benutzt werden kann. Dann:

```
... ...  ... ...     A: x   RS: --   DS: a b
a@  push a!  push    A: a   RS: x b  DS: --
a@  pop  pop a!      A: x   RS: --   DS: b a
```

ist länger als:

```
...  ...  ...  ...   A: x   RS: --   DS: a b
over push push drop  A: x   RS: b a  DS: --
pop  pop  ...  ...   A: x   RS: --   DS: b a
```

Die Erzeugung von Flags und Vergleichen:
Die bedingten Sprungbefehle C=0 und T=0 vernichten den Top of Stack nicht. Möchte ich 2 Werte auf exakte Gleichheit prüfen, so kommt die Befehlsfolge

```
-or T=0          entspricht: <> IF
```

zum Einsatz. -or produziert bei gleichen Werten eine 0, und der bedingte Sprung wird bei 0 ausgeführt. Bei Vergleichen (größer, kleiner) wird es etwas schwieriger:

```
... ... ... ...  DS: a b
com nop nop nop       Not(a) b
+ C=0                 Not(a)+b
                      entspricht: a b >= IF
```

Nehme ich das Einerkomplement der Zahl n (Not(n)) und addiere n so erhalte ich nicht 0, sondern -1. Im obigen Beispiel: Nur wenn b größer als a ist, dann ist das Carry 0, ansonsten immer 1. Der Sprung erfolgt also bei a<b. Solange a>=b ist fahre ich normal fort. Natürlich kann ich dies auch logisch durch ein erneutes com umdrehen:

```
com nop nop nop
 +   com nop nop
 C=0
```

Hier springe ich, wenn a nicht kleiner als b ist. (Sprung bei a>=b) und fahre normal fort, solange a nicht größer oder gleich b ist (a<b). Raucht der Kopf?

Der Assembler von Ultra Technology bietet etwas augenfreundlichere Strukturen:

```
if/-if   begin          begin      begin
   ...      ...            ...        ...
else     while/-while  repeat          until/-
until
   ...      ...
then     repeat
```

else und repeat sind unbedingte jumps und bei allen bedingten Sprüngen (if, while, until) wird noch zwischen T=0 (normal) und C=0 (vorangestelltes -) unterschieden. Flags müssen natürlich von Hand erzeugt werden.

Für Literale ist folgende Schreibweise vorgesehen:

```
12345 # 67890 # nop nop
```

Alle Informationen zu den MISC-Chips sind im Internet unter

www.ultratechnology.com

erhältlich. Hier kann man nach Lust und Laune durch die MISC-Welt surfen. Ultra Technology bietet eine Übersicht über alle Produkte, auch die noch in Arbeit befindlichen und bietet darüberhinaus hervorragenden Service bei Anregungen und Anfragen jeglicher Art.
Jeff Fox hat sich entschlossen, auch den F21-Simulator frei zur Verfügung zu stellen. Dieser simuliert die F21 CPU und den Video-Koprozessor. Sämtliche Source-Codes und Kurzanleitung wird von ihm zum Herunterladen aus dem Netz zur Verfügung gestellt. Diese Systeme sind in FPC geschrieben.

Da ich keinen Intel-kompatiblen Rechner besitze und gleichermaßen der 16-bit Welt überdrüssig geworden bin, habe ich den F21-Assembler für meine 32-bit Umgebungen angepasst, einen komplett neuen, eigenen Disassembler und Simulator geschrieben und das ganze auf das Win32for-System übertragen. Auch für die Intel-Welt habe ich meinen Simulator aus Geschwindigkeitsgründen in reinem Assembler geschrieben und praktisch mit den MISC Befehlen ein neues, einfaches und schnelles FORTH-Modell implementiert.

Da ich diese Informationen aus einer Flut von Internetveröffentlichungen zusammengesucht und kompakt zu gliedern bemüht war, mit dem Versuch, zu den meisten Details Beispiele zu geben, kann ich natürlich keine Garantien jeglicher Art geben. Bei Fragen stehe ich natürlich zur Verfügung, weise aber daraufhin, dass Jeff Fox bei Ultra Technology der kompetente Ansprechpartner ist.

MuP21 hat einen Transitorcount von nur 7000.
Intern arbeitet der Prozessor mit 100 MIPS, aber Speicherzugriffe begrenzen die aktuelle Leistung auf 30-80 MIPS.
Der Nachfolger F21 mit einem Transistorcount von 15000 mit integriertem Video I/O Koprozessor, integriertem Net-

Internet Relay Chat

Ungewohnt, anstrengend, aufregend: meine ersten Schritte auf einer englischen IRC-Forth-Sitzung

Fred Behringer
Planegger Str. 24, 81241 München
behringe@mathematik.tu-muenchen.de

IRC (Internet Relay Chatting), elektronische Konferenzschaltung, gleich über den Ärmelkanal hinweg, neu und faszinierend

Stichworte: IRC, mIRC, IRCnet, Forth allgemein, Gespräche in Echtzeit

Vorgeschichte. Am 25.10.98 bekam ich das Heft 98 der Forthwrite (Zeitschrift der Forth Interest Group UK). Chris Jakeman (der Redakteur) läßt sich auf Seite 23 darüber aus, daß über die Zeiten hinweg immer weniger Mitglieder zu den Versammlungen gekommen sind, daß jetzt aber ein neues Zeitalter angebrochen sei: das Zeitalter der Versammlungen per Internet. Elektronische Versammlungen. Round-Table-Gespräche (genannt "chat" - also "Schwatz") per IRC (Internet Relay Chatting).

Konkreter Aufruf. Software, sagt Chris, besorge man sich über www.irc.net . Er, so sagt er, verwende mIRC für Windows 95 (Shareware - 10 Pfund nach 30 Tagen). Er (Chris) werde jeden ersten Samstag des Monats (als Channel-op, also Kanalmoderator, Kanalleiter) auf Kanal #FIGUK im Netz IRCnet sein, und zwar um 10:00pm . (Fast hätte ich mich vertan und wäre schon um 21.00 im Kanal gewesen. Also 23.00 heißt das für uns.) Gleich nach Erhalt der Forthwrite kam eine E-Mail von Chris Jakeman an mich, ob ich mit von der Partie sein könne und werde. Natürlich. Klare Sache. Und dann ging's für mich los. Eine geschlagene Woche (fast) nichts weiter als IRC: Wie katapultiere ich mich vom Ahnungslosen zum Eingeweihten?

Meine Vorbereitungen. Ein paar E-Mails mit Friederich

Prinz ausgetauscht. Er wollte ursprünglich ebenfalls schon am 7.11.98 "mit von der Partie" sein, war aber dann auf Direktorensitzung - Thomas Beierlein und Egmont Woitzel zu Gast bei ihm am Rhein. Ein paar E-Mails mit Chris Jakeman ausgetauscht. mirc541t.exe von http://www.irc.net heruntergeladen, Dokumentationen und Hilfe (auch deutsch) zusammengesucht: http://www.irc.net , http://www.irchelp.org , http://irc.fu-berlin.de , mirc541t.exe (natürlich off-line und in Ruhe) aufgerufen und damit installiert, d.h., die Setup-Fragen fleißig beantwortet, Konfigurationsparameter zusammengesucht (E-Mail-Anfragen bei und Hinweise von Chris Jakeman), IRC-Befehle zusammengesucht (/JOIN und /LEAVE reichen - sind aber eigentlich auch gar nicht nötig - bei mirc geht alles über Pop-up-Menüs) und am 6.11.98, einen Tag vor dem großen Ereignis, war ich soweit: Modemverbindung mit Provider (Leibniz-Rechenzentrum München) per PPP in Windows 95 hergestellt, an IRCnet-Server (für mich war irc.informatik.tu-muenchen.de naheliegend) angekoppelt und Kanal #FIGUK aufgesucht (/JOIN #FIGUK). Dort erhielt ich natürlich channel-op- (Kanalleiter-) Befugnisse (um diese Zeit war ja keiner im Kanal - der Kanal existierte also noch gar nicht und wurde mit meinem Eintritt erst ins Leben gerufen - und mit meinem sofortigen Austritt wieder exterminiert): erhebendes Gefühl, schnell wieder raus (/LEAVE #FIGUK) und ausgekundschaftet, wie ich das Protokoll der Sitzung, die noch gar keine richtige war, (für die Ewigkeit) festhalten kann: Mausklick auf logging in irgendeinem Menü, von irgendeinem Knopf (Button) auf der oberen Leiste zum Aufblühen gebracht. E-Mail an Chris Jakeman: "Ich bin bereit".

Und dann der vorgesehene Samstag. Ich werde weiter unten den Ablauf dieser "Inauguralsitzung" andeuten. Es war ganz schön nervenaufreibend. Alles flutscht auf dem Bildschirm an einem vorbei. Kaum, daß man in der Aufregung daran denkt, die Textfläche großzuschalten. Während man in bestem Sonntagsenglisch auf eine gestellte Frage eine Antwort austüftelt (intelligent soll sie ja obendrein auch noch sein), prasseln Antworten auf eigene Fragen auf einen ein, die man viel weiter "oben" (auf dem Bildschirm) gestellt hat und die schon längst herausgescrollt sind. Zudem fällt die Zuordnung der "Namen", die ja keine sind, nicht immer leicht. Den im IRC-Verkehr verlangten "Nickname" oder "Nick" (Spitzname, Pseudonym, Künstlername) sollte man sorgfältig auswählen (9 Buchstaben sind zugelassen). Phantasienamen sind nichtssagend und wohl auch verpönt. Jener Teil der E-Mail-Adresse vor dem @ wäre vielleicht gar nicht so schlecht. Wer natürlich in der Gruppe so bekannt ist wie beispielsweise CJ, ist gut beraten, der Kürze halber eben diese seine Initialien zu verwenden. Kurz ist die Wurz - wollte sagen in der Kürze liegt die Würze - wenn sich "Gespräche" auf dem Tasten-Hackbrett entwickeln sollen.

Fazit. DAS MÜSSEN WIR AUCH MACHEN! Unsere englischen Forth-Freunde haben es uns vorgemacht. - Am jeweils zweiten Samstag (Sonnabend) im Monat im Kanal #FORTH-EV.DE im Netz IRCnet, wäre das nichts? Wir natürlich in unserer eigenen Sprache. Aber gelegentliche oder fest vereinbarte gemeinsame Sitzungen (FIGUK/Forth-Gesellschaft) wären ja auch möglich.

Und hier ein paar Worte über den Ablauf

Session Start: Sat Nov 07 23:05:45 1998
*** Now talking in #FIGUK
<dja> Probably the first and ..... (ich, "FredBeh", lasse den Rest dieses Satzes weg, da er nichts zur Sache tut)
<CJ> Hi Fred, we were just talking about you.

..... So fing die Sitzung für mich an. Ich hatte noch ein paar Minuten damit vertrödelt, die richtigen Knöpfe und Fenster zu finden, die zu bedienen waren, bevor man mit von der Partie war. Wahrscheinlich haben die anderen sich in der Zwischenzeit gewundert, wo denn der Fred Behringer aus Germany bleibt.

<CJ> Hi Ray,
<dja> Iv'e got ...
<CJ> Ray, I saw ...
<dja> I'm trying to ...
<rayma> CJ, ok, no hurry

... und so weiter und so fort ...

<FredBeh> Hi Chris, hi everybody! Did my message reach you two minutes ago?
.....
<JeremyFow> Fred, could you repeat the message ?
<CJ> Hi Fred, I downloaded an e-mail from Friday night (You stay up late don't you)
.....
<FredBeh> Chris, Friederich Prinz from Deutsche Forth-Gesellschaft cannot participate to-day but he will be on this c hannel next month, he said in an e-mail to me.

..... Und so ging es dann weiter. Ich war 40 Minuten dabei. Während dieser Zeit habe ich acht Teilnehmer gezählt. Ich gebe mit Absicht und nach Absprache mit Chris Jakeman und Friederich Prinz nicht das volle Protokoll wieder. Die Sitzungen sind zwar "öffentlich" (theoretisch könnte jederzeit jedermann aus aller Welt, ob Forthler oder Nicht-Forthler, dazustoßen), es kommen aber in der Eile leicht Formulierungen vor, die man für eine Veröffentlichung stark redigieren müßte. Ich lasse es bei diesen den tatsächlichen Ablauf nur andeutenden Bruchstücken und hoffe, damit das Interesse unserer Leser für dieses neue, faszinierende Kommunikationserlebnis zu wecken.

Die Sitzung ging für mich zu Ende mit

<FredBeh> To all participants: I have six more minutes I think. If my provider (local server) disconnects me next time, I will leave and stay off. Bye then.
.....
<CJ> Before you go, Fred, will you have your ANS file words ready soon?
.....
<FredBeh> Well folks, I'll leave now. Bye.
<CJ> Bye Fred.
Session Close: Sat Nov 07 23:41:13 1998
... und das war's. Vielleicht bleibt noch anzumerken, daß

work I/O Koprozessor und integriertem Analog I/O Koprozessor arbeitet bei einer internen Leistung von 500 MIPS gebremst durch Speicherzugriffe aktuell bei 333 MIPS ROM, 222 MIPS SRAM und 111 MIPS DRAM. F21 soll in einer Evaluation-Board-Version 1MWort DRAM (2.5MB), 16KWorte SRAM sowie Boot-(EP)Rom bieten.
F21 befindet sich im Prototypenstadium. Eine Version von F21 der i21 (21-bit Internetprozessor) wird für den Einsatz in Zusatzboxen für Fernsehgeräte für den Internetzugang vorbereitet. i21 wird von der iTV-Corporation produziert. F21 ist das Flaggschiff der Ultra Technology unter Jeff Fox.

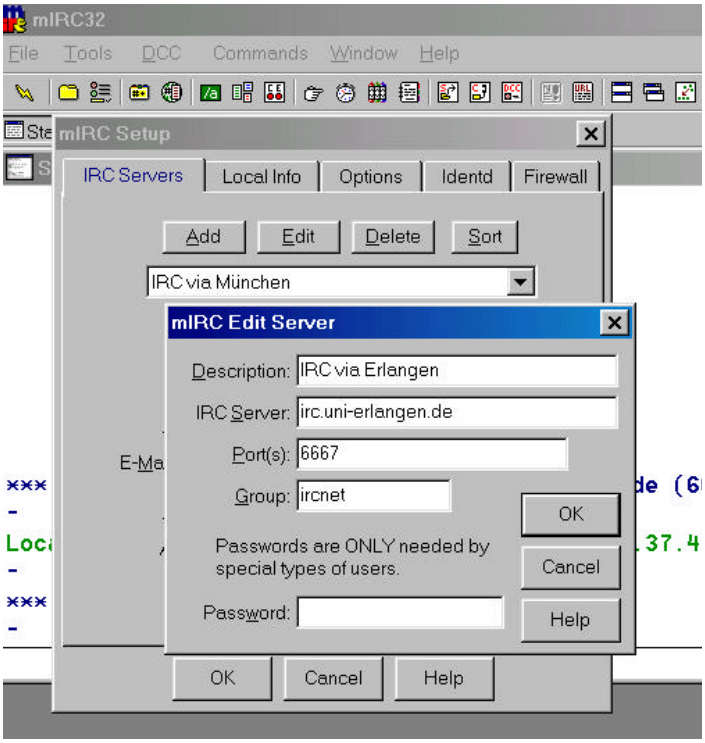
Soeren Tiedemann

IRC – Sitzungen der **Forth-EV.DE**

Ich glaube mich zu erinnern, daß Claus Vogt genau dies bereits vor zwei oder drei Jahren angeregt hat, nämlich daß die Forther der FG sich zu festen Terminen in einem IRC-Channel treffen und sich dort online austauschen oder einfach nur miteinander raatschen können. Claus erinnert sich gar nicht mehr daran. Ich wundere mich, daß aus dieser Idee bisher nichts geworden ist...

Wie auch immer, Fred Behringers Anstoß kam genau zum ‘richtigen‘ Zeitpunkt – als ich eigentlich gar keine Zeit hatte, mir Software im Netz zu suchen, herunter zu laden, zu installieren und herum zu probieren, bis endlich – hoffentlich – ein erster Connect mit ihm und Anderen zustande käme... Ich war angenehm überrascht, als sich einfach alles viel einfacher gestaltete.

IRC-Clients liegen im Netz ‚wie Sand am Meer‘ herum. Ich



habe mich für den mIRC entschieden, einen Klassiker unter den IRC-Clients, den man als Shareware am besten direkt beim Hersteller abholt (http://www.mirc.co.uk). Mit dieser Software geht alles (fast) wie von selbst.

Ich starte die Software meines Providers (T-ONLINE), um eine Verbindung in das Internet zu bekommen. Damit ich mich weder um TCP/IP Einstellungen noch um PPP, SLIP oder Ähnliches kümmern muß, lasse ich zu, daß T-ONLINE mir sofort ‚meinen‘ NetScape Browser hochfährt. Nur den mIRC muß ich noch per Mausklick ‚nachziehen‘.

Die allererste ‚Sitzung‘ hat einen erstaunlich geringen Konfigurationsaufwand verlangt.
Im Menü <File>.<Setup>.<Edit> mußte ich den von mir bevorzugten IRC-Server, den Kommunikationsport und die Netzgruppe angeben. Wie in dem Bild – hoffentlich – zu erkennen ist, habe ich die Einstellungen ‚gewählt‘, zu denen man mir auf diversen Webpages geraten hat: IRC.UNI-ERLANGEN.DE; Port 6667; IRCNET.

Der Server der Uni-Erlangen behauptet von sich selbst in seiner Einschaltmeldung, der erste IRC-Server in der BRD gewesen zu sein. Es gibt natürlich eine Vielzahl anderer IRC-Server in der BRD. Hinweise darauf finden sich im WWW tausendfach.

Ich hatte in comp.lang.forth.de angekündigt, mit der Moerser Gruppe Samstags ab 14:00 Uhr den Kanal #forth-ev.de öffnen zu wollen. Das hat bis Heute nicht funktioniert, was aber an der Netz-Hardware liegt, die wir im MALZ zur Verfügung haben. Dafür haben Egmont Woitzel und Thomas Beierlein den ersten Termin aufgegriffen – und gleich verabredet, daß der Kanal #forth-ev.de zukünftig Freitags ab 21:00 Uhr geöffnet sein soll.

MEINE erste IRC-Sitzung fand darum am darauffolgenden Freitag (27/11.98) statt. Egmont Woitzel, Thomas Beierlein, Fred Behringer, Martin Bitter, Winfried Clemens und zeitweise Chris Jakeman haben sich auf diesem Kanal getroffen. Da ging es kunterbunt ‚zur Sache‘. Probleme bei der Inbetriebnahme von 16,8 Gbyte-Festplatten, mehrfach geschachtelte CREATE ... DOES> Konstrukte, das Wetter, die Familien und jede Menge ‚Frozzeleien‘ wurden mit einem Tempo hin und her geworfen, daß man schon gelegentlich die Übersicht verlieren konnte. Immerhin hatte ich mindestens zwei ‘Hintergrundtasks‘ zu bedienen, die mich mindestens genauso forderten wie der IRC. Michael Major war bei mir, um mit mir ‚Objekte in klassischem Forth‘ zu diskutieren. Und meine Frau mußte ich gelegentlich überreden, uns mit frischem Cappuccino zu versorgen – was um so schwieriger wird, je mehr der Stundenzeiger vorrückt.

Die nächste, freitägliche Sitzung fand dann am 04/12.98 statt. Wieder wurden Informationen über die großen Festplatten ausgetauscht (Egmont Woitzel hat seine Platte übrigens , ans Laufen‘ bekommen), sowie ‚Forthiges‘ und ‚Familiäres‘. Man kennt sich halt. Freunde treffen Freunde. Die langsamen Finger zwingen dazu, das „Sehr geehrter Herr...“ fallen zu lassen und zu dem viel schnelleren „Du“ zu wechseln. Forth verbindet uns – und läßt mehr als genug Raum für das Zwi-

Gehaltvolles

zusammengestellt und übertragen
von Fred Behringer

FORTHWRITE der FIG UK, Großbritannien

Nr. 99, November 1998

1 Editorial
Chris Jakeman

Begrüßt werden neue Mitglieder aus Spanien, Schweden, Frankreich und aus England selbst. Chris überlegt, ob man nicht die Berichte von Alan Wenham über unsere Vierte Dimension einstellen soll, wenn weiterhin keine Reaktionen kommen. (Mir selbst machen meine Rezensionen der Forthwrite für unsere FG-Mitglieder großen Spaß, aber hier und da eine Reaktion wäre natürlich auch wünschenswert; nur um ganz sicher zu sein, daß ich mir die Arbeit nicht umsonst mache.)

2 Forth News
Chris Jakeman

FreeBSD hat einen Bootloader, der in einer Forth-Variante, Ficl, programmiert ist; Peter Knaggs und ANS-Forth-Überarbeitung; Marcel Hendrix und ein Buch über Forth für Windows; SwiftForth 1.5 von FORTH, Inc.; Quartus Forth 0.81B; Aztec Forth 2; Ficl 2.02 (ANS-Forth in C), ByteForth-Cross-Compiler (Holland); MINOS von Bernd Paysan für Win 95 und Linux; Forth für dBase IV; 2 CDs für 5 Pfund von der Universität Brighton mit Compiler in mehr als 20 Sprachen - auch Forth. Vorschlag von Chris: 3 alphanumerische Zeichen in Basis 36 in 1 Zelle auf den 16-Bit-Stack laden.

5 Object-Oriented Forth - A Minimal Approach
Chris Jakeman

Eine gute Beschreibung eines "Mini-OOF"s, das nur 12 Zeilen umfaßt ('7,5' Colon-Definitionen) und in den Grundzügen von Bernd Paysan stammt. Chris arbeitet das aus und gibt eine anschauliche Darstellung der Wirkungsweise. ***(Anmerkung der Redaktion: Chris Jakemans Artikel wird in der nächsten Ausgabe der VD in einer Übersetzung abgedruckt)***

11 The EuroForth98 Conference
Paul Bennett

Elf (!) Seiten gute Besprechung der diesmaligen europäischen Forth-Tagung. Auf Schloß Dagstuhl im Saarland. Paul fuhr mit seinem siebenjährigen Sohn zum Tagungsort. Ein paar Eindrücke von unterwegs. Arbeitsgruppe "Internationalisierung". (Es geht um unsere Umlaute, die französischen diakritischen Zeichen, das Esszett usw. Wir hatten 1989 in JEDI Diskussionen darüber. Jetzt wird es endlich ernst.) C@ , C! usw. sollten nicht verändert werden. Vorgeschlagen wurde an das ANS-Komitee: Einführung eines neuen Word-Sets "Internationalisierung". E-Mail-Diskussionen werden über Anton Ertl laufen. Es folgen ausführliche Berichte über die Tagung selbst, die einzelnen Beiträge, die anstehenden Projekte und schließlich ein paar Zeilen (3 Seiten mit Programmen) über den diesjährigen Ulk-Wettbewerb (Preisträger: Anton Ertl, Paul Bennett, Malcolm Bugler, Jason Bennett). Eindruck des Rezensenten: Guter Bericht eines Aktiven über eine gelungene, straff geführte Tagung, die auch wissenschaftlichen Ansprüchen genügt, von und für Profis. Anfänger oder solche, die Forth neben ihrem eigentlichen Beruf betreiben, können da wohl nicht (mehr) mithalten.

22 jeForth Project
Chris Jakeman

Es geht um eForth für Java von Mike Losh (USA), das bei Aufruf seiner Webseite sofort (auch im "local cache") zur Verfügung steht: http://www.amsystech.com/mlosh/ (mit "click here for Deutsch bei Fred Behringer"). Das wird jetzt DAS Projekt der FIG UK mit ANS-Anbindung, Beispielen, bewegten Bildern und sonstigen "Attraktoren" und mit vielen und vorwiegend auch für den Anfänger sofort verständlichen "Tutorials" in "möglichst vielen Sprachen dieser Erde" (Chris Jakeman). Deutsch gehört dazu, der Rezensent ist gebeten worden und hat sich eingeklinkt.

24 From the 'Net - Newbies and Gurus
Ray Allwright

Matthias Warkus, ein "Newbie" (wie er sich selbst bezeichnet) und sechzehnjähriges "Wunderkind" (von Jerry Avins im Englischen verwendetes Lehnwort), bemerkt, daß Elisabeth D. Rather doch eigentlich recht viel Reklame für Forth Inc. macht, und fragt, warum es in Forth keinen anständigen Editor wie Emacs gibt. "Elisabeth Rather hat sehr viel für Forth geleistet, ihre Antworten und Vorschläge sind ausgezeichnet und für viele brauchbar, das bißchen Reklame kann man dulden", war der eine Antwortfaden. "Schreib selbst einen; in Forth eine Kleinigkeit; auch für Neuhinzukommende", war der andere Antwortfaden. "Vorsicht, eine Riesenaufgabe", warnte Klaus Schleisiek.

28 Getting Together
Chris Jakeman

Bericht über die erste IRC-Sitzung auf Kanal #FIGUK im IRCnet am ersten Novembersamstag 1998. Recht hat er, der Chris: Ich (der Rezensent) hatte nicht nur "die Ehre", der erste "Übersee"-Teilnehmer gewesen zu sein; mir hat es auch riesigen Spaß gemacht. Aber bei uns grassiert ja das IRC-Fieber inzwischen auch. (Und bei Drucklegung wird es überwältigende Ausmaße angenommen haben...) - Beim zweiten

#FIGUK-Treffen war Friederich Prinz dabei und Chris hat uns auf #FORTH-EV.DE die Ehre gegeben.

29 Multiple Language Programs Made Easy
Howerd Oakford

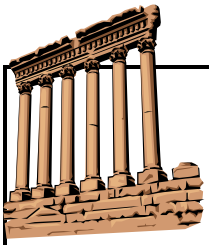
Der Autor beschäftigt sich seit 1979 und auch schon davor mit Apparaturen zur Messung und Weiterverarbeitung des Feuchtigkeitsgehaltes in Getreide und Ähnlichem. Seine Apparatur (Embedded Devices) basiert neuerdings auf einem 8031 und wird in chipForth programmiert. Die Menüsteuerung seiner Gerätschaften sollte ans Spanische angepaßt werden. Der Autor richtet sein Programm (leichte Erweiterung von chipForth) so ein, daß man an sein nur ein einziges Mal zu schreibendes Programm Übersetzungsdateien anhängen kann, die das Anpassen an die gewünschte andere Sprache (hier zunächst nur Spanisch) per "Knopfdruck" bewirkt. Die dazu nötige Datei, welche mit einem beliebigen ASCII-Editor hergestellt werden kann, enthält jede Menüzeile einmal in englischer und einmal in der jeweils anderen Sprache. 4 Seiten chipForth. (Der Rezensent: Wirklich prima Idee. TurboForth macht etwas Ähnliches schon auf Metacompilationsebene gleich mit dem eigentlichen Forth-System.)

37 AGM Report
Chris Jakeman

Jahresversammlung der FIG UK mit Kassenbericht. Alle sind immer jederzeit auf der Versammlung willkommen. Der Jahresbeitrag bleibt weiterhin bei 10 engl. Pfund. Austritte und Eintritte halten sich zur Zeit die Waage. Neue Mitglieder kommen in zunehmender Zahl angelockt durch das Internet herein. 1998 war das Internet-Jahr. 1999 soll das Projekt-Jahr werden. jeForth von Mike Losh wird als Gemeinschaftsprojekt zur Weiterbearbeitung, ANSisierung, Internationalisierung und Vervollkommnung anvisiert. (Wie steht es denn mit dem Interesse bei uns?)

39 Letters
Chris Jakeman

Der eine Brief wurde von meiner Wenigkeit eingeschickt. Ich war halt über die erste IRC-Sitzung der FIGUK über alle Maßen begeistert (vergl. S. 28). Der andere Brief kommt von Graham Telfer von Japan her über die Frage, wie leicht es ist, Forth zu lernen, und wie schwierig es ist, es wirklich zu beherrschen.



Auf den Trümmern von DELPHI...

...wollten die Direktoren der **Forth**gesellschaft sich Ihnen zeigen. Reale Kulisse hierfür war der archäologische Park in Xanten am Niederrhein, wo sich das Direktorium zum Herbsttreffen versammelt hatte. Das dort ‚geschossene' Photo ist leider nicht zum Abdruck geeignet, weil darauf selbst die Ruinen eines alten (römischen) Tempels die Direktoren noch winzig und verloren erscheinen lassen .

;-) *fep*

The Forth Source

Eine forthige Quelle bietet Glen B. Haydons **Mountain View Press** dem Suchenden und dem Interessierten. Der Redaktion liegt eine 11-seitige Zusammenstellung aller „Features – Forth Levels – Catalog – History – Hindsight" vor, die MVP aktuell anbietet. Darunter finden sich das MVP – Forth, Publikationen, ein Katalog mit Public Domain Implementierungen auf 5 Ebenen (von 63 Worten von C.H. Moore bis zum MVP-F32 von Rick VanNorman) und vieles mehr. Der Abschnitt , Reference Books' zeigt 10 hoch interessante Titel, denen die Angebote zu Hardware-Implementierungen (WISC CPU/16 und WISC CPU/32) sicher nicht nachstehen.

Glen B. Haydon weiß um die Probleme, die deutsche Konsumenten haben, wenn sie in den USA ‚einkaufen' wollen, und akzeptiert es gerne, wenn Sie Ihre Kreditkartennummer in zwei Teilen in zwei getrennten E-Mails nennen möchten. Bei allen anderen Problemen bittet Mr. Haydon darum, ihn direkt anzusprechen.

Sie können das vorliegende Material von der Redaktion der VD anfordern, oder sich direkt wenden an:

Mountain View Press
Route 2, Box 429,
La Honda CA 94020
(650) 747 0760

Einfacher und schneller geht es mit einem Internetanschluß:

http://theforthsource.com,
bzw.
ghaydon@theforthsource.com

fep

Ein LOGO für die FG...

...wird gesucht !

Die Forthgesellschaft sollte ein ‚LOGO' haben, das sich in der VD, in unserer WEB-Site und auf Briefbögen mit hohem Wiedererkennungswert verwenden läßt.

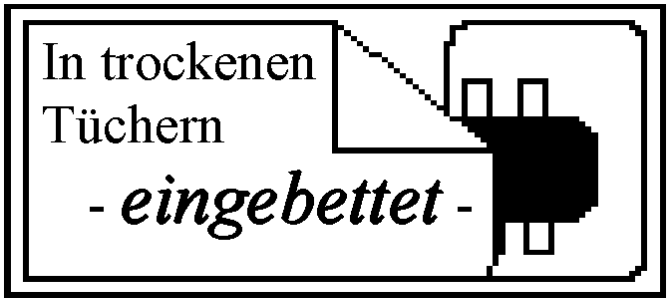
Natürlich sollte das LOGO einen deutlich sichtbaren Bezug zu Forth und zur Forthgesellschaft haben.

SIE sind aufgerufen, ein solches LOGO zu entwerfen – ganz gleich wie und womit – und Ihren Entwurf an die Redaktion der VD zu senden. Die Mitgliederversammlung im April 1999 soll, sofern Ihre Vorschläge rechtzeitig hier eingehen, DAS LOGO der FG auswählen.

Dem Sieger dieses Wettberwerbs winkt als Preis die kostenlose Teilnahme an der Jahresversammlung 2000 !

Für das Direktorium

Friederich Prinz



Rafael Delianos zweite “embedded“ ist erschienen.

Vorgestellt von Martin Bitter

Rafael Deliano ist in der Forth-Szene durch verschiedene Aktivitäten bekannt geworden. Unter anderem durch das Projekt Creeping Forth aus der Zeit um 1993 und seine im Namen des Forth e.V. (*nicht Forthgesellschaft e.V. – die Redaktion*) herausgegebene Reihe MARC4, die in der letzten Vierten Dimension bereits lobend erwähnt wurde.

Nun liegt bereits die zweite Ausgabe der **embedded** vor, die Rafael Deliano im Selbstverlag herausgibt. Wie schon die erste Ausgabe, umfaßt diese 28 Seiten voller geballter Information über die Programmierung diverser Mikros.
Den einzelnen Artikeln ist die praktische Erfahrung des Autors anzumerken, die vorgestellten Lösungen und Vorschläge gehen in die Tiefe, lassen aber auch Raum für eine eigene Gestaltung. Ausschnitte aus fertigen Listings helfen letztendlich Implementationshürden und Tippfehler (z.B. Flußdiagramm Booth) zu überwinden.
Komplette Listings sind bei Deliano auf Diskette erhältlich - eine Konvertierungsanleitung des verwendeten Forthdialektes „nanoFORTH“ befindet sich (nur) in der ersten Ausgabe - bei Bedarf anfordern.

Im Einzelnen findet sich ein kleiner historische Rückblick über die Entwicklung einzelner **8-bit CPUs**.

Ausführlich wird die Möglichkeit erörtert, digitale, d.h. **softwaregenerierte PIDs** (proportionale Regler mit einem Integrations- und einem Differnzierglied (Filter)) in einem Mikro zu programmieren. Zahlreiche Diagramme, einige Formeln und ein komplettes, vierteiliges Flußdiagramm strukturieren die Darstellung. Es wird auf den verbrauchten Speicherplatz, sowie die Ausführungszeiten eingegangen. Die Grenzen solcher Softwareregler werden dargestellt, und im Anschluß zeigt Rafael Deliano wie man **digitale Filter testen** kann.

Beim **Booth-Multiplizierer** wird gezeigt, wie vorzeichenbehaftete Zahlen (Zweierkomplement) direkt miteinander multipliziert werden können. Ich selbst stehe wieder einmal staunend vor der „Ingeniösität“ eines Mannes wie Booth, der den grundlegenden Algorithmus entwickelte. Durch Beachtung des Bit-Wechsels einer Binärzahl wird das Produkt zweier Zahlen gebildet: „ Bei einem Übergang von 1 nach 0 wird der Multiplikand zum Produkt addiert. Bei einem Wechsel von 0 nach 1 wird er vom Produkt subtrahiert. In beiden anderen Fällen ist keine Operation nötig.“ Das ist natürlich nicht alles, aber der Kern des Algorithmus. Mir (M.B.) macht es einfach ungeheuren Spaß so etwas zu sehen und nachvollziehend zu verstehen (vielleicht). Wer jetzt mehr wissen will, sollte sich die **embedded 2** besorgen.

Am Beispiel eines Mitsubishi-6502 kann man lernen, wie ein Busprotokoll für den I2C-Bus programmiert werden kann. Erläutert werden Implementierungen bei bidirektionalen Portpins. Das Hauptaugenmerk liegt aber auf Workarounds? bei unidirektionalen Pins mit Hilfe von Schottkydioden. (Es gibt sogar ein Diagramm zur Abschätzung der kapazitiven Last bei mehreren ICs am Bus.) Schlußendlich wird gezeigt, wie ein **EEPROM am IC2-Bus** angesprochen wird und wofür man das nützlicherweise verwendet.

Viele haben schon Schrittmotorentreiber programmiert. Hier zeigt Rafael Deliano nicht nur, wie das geht, sondern sogar, wie man „ekligen“ Verdrahtungswirrwarr elegant umgehen kann, indem der Schrittmotor als Generator betrieben wird und die dann auslesbaren Strom-Spannungs-Muster zur Definition der Ansteuerung benutzt werden. Genial!

Ein **simpler Zeitscheibenmultitasker,** der für eine Reihe von kleinen (Assembler)Task bei kleinen Applikationen gedacht ist, wird in seiner Arbeitsweise - wie immer mit Listing - vorgestellt. Eingegangen wird auf Watchdog-Schaltungen und Ähnliches. Grundlegender Mechanismus ist die „gerade“ und „krumme“ Frequenzteilung eines Softwaretimers. Gezeigt wird das Anstoßen von zyklischen Vorgängen über einen weiten Zeitbereich vom Sekundenbereich bis hin zu tausendstel Sekunden.

In der **embedded 1** wurde in sehr knapper Form die Ansteuerung eines **CCD-Zeilensensor**s gezeigt. Auf Wunsch zahlreicher (?) Leser erläutert Rafael Deliano in der **embedded 2** die zugrundeliegende Hardware und ihre Verschaltung. Zeit- und Taktdiagramme inbegriffen. Fast könnte man meinen, sich seinen Scanner selbst bauen zu können (2048 Pixel pro Zeile).

Alles in allem ist es erstaunlich über welche Produktivität Rafael Deliano verfügt. Ich habe meine Freude dran. Wer sich selbst die gleiche Freude machen will, der wende sich an:

Rafael Deliano
Steinbergstr. 37
82110 Germering
Tel. 089/8418317

Einladung

zur ordentlichen Mitgliederversammlung
der Forth-Gesellschaft e. V.

am
Sonntag, den 25. April 1999 um $9^{\underline{00}}$ Uhr,

im
Parkhotel Sonnenhof
in Oberammergau

Tagesordnung

1. Rechenschaftsbericht des Direktoriums
2. Kassenbericht und Bilanz 1998
3. Aussprache und Entlastung des Direktoriums
4. Vorhaben und Ziele der Gesellschaft
5. Verschiedenes

Ergänzende Tagesordnungspunkte sind bis zum 18. April 1999 über das Forth-Büro beim Direktorium einzureichen.

Die Teilnahme an der Mitgliederversammlung ist kostenlos. Sie findet wie jedes Jahr direkt nach Ende der Jahrestagung der Forth-Gesellschaft an deren Veranstaltungsort statt.

Wir bitten um zahlreiches Erscheinen, ausreichend Zeit und viele gute Ideen für das nächste Jahr der Forth-Gesellschaft.

Die Mitgliederversammlung wird voraussichtlich etwa 3 Stunden dauern.

Das Direktorium

Forth-Gruppen regional

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

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

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

µP-Controller Verleih

Thomas Prinz
Tel.: 06271-2830 (p)

Gruppengründungen, Kontakte

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

Forth-Hilfe für Ratsuchende

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

Jörg Staben
Tel.: 02103-24 06 09 (p)

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

Spezielle Fachgebiete

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

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

Schreiben Sie einfach der VD - oder rufen Sie an -

Hinweise zu den Angaben nach den Telefonnummern:

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

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

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

Der Termin für die FORTH-Tagung 1999 steht fest !

Vom 23. bis zum 25. April 1999

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

Anmeldeformulare und weitere Informationen finden Sie in dieser VD.

Ulrike und Heinz Schnitter freuen sich auf Ihr Kommen !