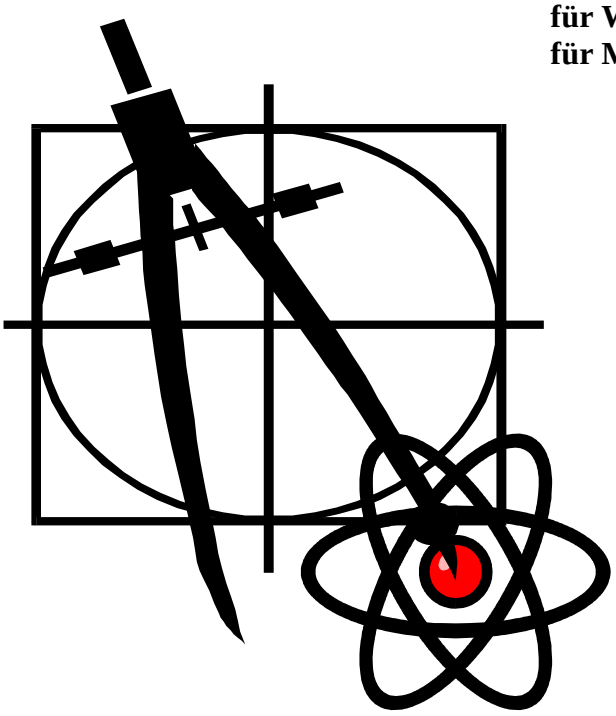
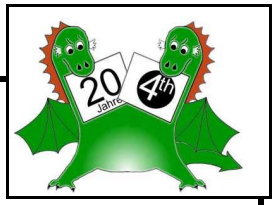
VIERTE DIMENSION

4 / 2004

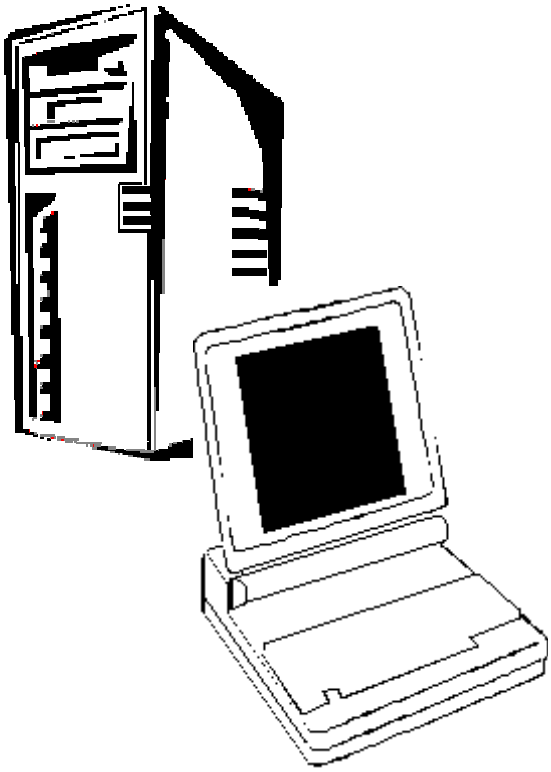
Das FORTH-Magazin

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





In dieser Ausgabe:



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

Organ der FORTH-Gesellschaft e.V.

Dienstleistungen und Produkte fördernder Mitglieder des Vereins

tematik GmbH
Technische Informatik

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

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

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

LEGO RCX-Verleih

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

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

Inhalt:

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

Anfragen bitte an

Martin.Bitter@t-online.de

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

Hier könnte IHRE Anzeige stehen!

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

Secratary@forth-ev.de

Forth Engineering
Dr. Wolf Wejgaard

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

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

KIMA Echtzeitsysteme GmbH

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

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

FORTecH Software
Entwicklungsbüro Dr.-Ing. Egmont Woitzel

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

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

Ingenieurbüro
Dipl.-Ing. Wolfgang Allinger

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

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

Ingenieurbüro
Klaus Kohl

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

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

Inhalt

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

http://www.forth-ev.de

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

In der nächsten Ausgabe finden Sie voraussichtlich:

- Themenschwerpunkt: FPGA, b16, µCore ...
- Einladung zur Tagung und Mitgliederversammlung 2005
- Was immer Sie uns bis dahin schreiben oder mailen

Editorial

IMPRESSUM

Name der Zeitschrift

Vierte Dimension

Herausgeberin

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

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

Redaktion & Layout

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

Anzeigenverwaltung

Büro der Herausgeberin

Redaktionsschluß

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

Erscheinungsweise

1 Ausgabe / Quartal

Einzelpreis

4,00 € + Porto u. Verpackung

Liebe Leser,

die letzte Ausgabe der Vierte Dimension in diesem Jahr hat Sie nun erreicht. Das Jubeljahr der Forthgesellschaft geht seinem Ende entgegen. Eine in ihrem Äußeren leicht veränderte VD mit kleinen SWAPs, die Anstecknadeln, die Ihnen das Forthbüro zusammen mit der Ausgabe 03 geschickt hat und ein richtiges Fest anläßlich der Tagung auf Fehmarn waren wohl die sichtbaren und für die Mitglieder erlebbaren Zeichen unseres Jubiläums.

Hat es für Sie etwas Besonderes, „forthiges“ darüber hinaus gegeben? Hat es für Sie in Bezug auf Forth, auf die Forthgesellschaft oder ganz allgemein „mit dem Computer“ in diesem Jahr etwas gegeben, was Sie besonders fasziniert und/oder beschäftigt hat? Bitte berichten Sie uns darüber.

Mich selbst beschäftigt die Frage nach den Bordcomputern in modernen Automobilen immer mehr. In meinem zwölf Jahre alten Auto war ein solcher Bordcomputer eingebaut. Der hat Außentemperaturen angezeigt, den aktuellen und mittleren Treibstoffverbrauch ausgewiesen, Wegstrecken gemessen und noch weitere Informationen geliefert. Wirklich nützlich war für mich persönlich hauptsächlich der Hinweis auf die Außentemperaturen und eine damit verbundene Warnung vor möglicher Eisglätte. In der Folge eines Unfalls mußte ich mir ein neues Auto zulegen. Das hat natürlich auch einen Bordcomputer. Und dieser Bordcomputer kann noch wesentlich mehr Informationen liefern, als sich interaktiv durch den Benutzer/Fahrer abfragen läßt. Zum Beispiel kann er die Meldung „Bitte den Ölstand kontrollieren“ auf das Display ausgeben. Eine Kontrolle des Ölstandes zeigt auf dem guten, alten Peilstab an, daß mehr als ausreichend Öl vorhanden ist. Die ebenfalls vorhandene Ölkontrolleuchte hatte ohnehin nicht aufgeleuchtet. Die um Rat angerufene Werkstatt konnte mir nur den Rat geben, den Bordcomputer zu ignorieren, solange ausreichend Öl mit dem Peilstab nachweisbar ist.

Das ist nicht wirklich lustig. Als Techniker bin ich daran gewöhnt, Fehlermeldungen eines Systems sehr ernst zu nehmen. Fehlermeldungen, die unnütz und zur Unzeit produziert aber ernst genommen werden, kosten viel Zeit, Kraft und letztlich auch Geld.

Da stellt sich mir die Frage, ob wir nicht in vielen Bereichen längst „übercomputerisiert“ sind. Wenn Sie mögen, können wir diese Frage gerne in der VD diskutieren.

Ihr

Friederich Prinz

Quelltext-Service

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

Die Forthgesellschaft wird durch ihr Direktorium vertreten:

Prof. Dr. Fred Behringer
Dr. Ulrich Hoffmann
Dipl.Inf. Bernd Paysan

Kontakte: Direktorium@Forth-ev.de

...aus dem Netz gefischt...

...\comp\lang\forth\de

Betreff: Wozu Forth ?
Von: Rafael Deliano <Rafael_Deliano@t-online.de>

> Frequentierung auf dieser Newsgroup

FORTH hat aber eben eine deutsche Newsgroup. Manche Sprachen haben keine mehr. Andere hätten gerne eine, stemmen aber halt keine Einführung.
Und Antworten kriegt man auf hier gestellte sachdienliche Fragen auch. Muß man nur mehr Fragen stellen, dann würde auch Volumen steigen.

> Ich meine, wenn ich systemnah programmiere, nehme ich
> C/C++,Delphi übergreifend zB. Java.

Auf einem 8 Bit Controller ?

> Gibt es ein Gebiet, für das Forth prädestiniert ist ?

s.o.

> Mit welchen Rechnern/Betriebssystemen arbeitet ihr ?

s.o.
Es gibt formal sogar 4 Bit CPUs, die mit FORTH programmiert werden und dann in Braun-Rasierapparaten oder RFIDs für Automobilen verbaut werden.

> Seht ihr für Forth eine Zukunft ?

FORTH bildet das grundlegende Konzept der 0-Adressmaschine ab. Wer einen Stackprozessor baut, landet recht direkt bei FORTH.
Deshalb gibt es auch keine "Weiterentwicklung" à la C -> C++ -> Java -> ?

Was schon das tut, was es soll, muß nicht verändert werden.

MfG JRD

Die Auseinandersetzung geht weiter; immer. Wozu braucht man Forth? Warum entwickelt Forth sich nicht entsprechend dem Mainstream? Wo sehen die Forth-Nutzer außerhalb des Mainstreams die Zukunft für Forth? ...

Rafael Deliano hat das in seiner kompetenten Antwort vor allem mit seinem letzten Satz m.E. bemerkenswert auf den Punkt gebracht. Forth ist entwicklungs– und veränderungsfähig, hat aber kaum Veränderungen nötig. Für das, was es leisten soll, ist es völlig ausreichend. fep

Von: Rafael_Deliano@t-online.de (Rafael Deliano)
An: Friederich.Prinz@t-online.de

Im www Gefundenes wäre Kurzmeldung wert:

Der Actel (www.actel.com) ist hinter Xilinx, Altera und Lattice ca. der 4.-grösste FPGA-Hersteller. Seine Spezialität sind ICs die durch "fusible links" konfiguriert werden, was sich gut für Raumfahrt eignet. Als unterstützende Entwicklungsfirma auf der Webseite ist u.a. aufgeführt:

ASIC Design RU
Moskovsky pr. 212
St. Petersburg 196066
www.asicdesign.ru

Prozessor-Cores, die diese Firma anbietet, sind: 186er, 51er, MIL-STD-1750 "and stack-based MCUs". Der russische Novix könnte also wohl nochmal auftauchen.

Zwar nicht FORTH, aber zum MIL-STD-1750 und zu ARINC 429 bzw. MIL-STD-1553 wären Artikel möglich.

...die VD ist immer interessiert und aufnahmebereit! fep

Der MIL-STD-1750 ist ein etwa 1980 entwickelter 16 Bit Prozessor, für den es angeblich keinen C und vermutlich auch keinen FORTH-Compiler gibt; dafür aber diverse Ada-Compiler. Der Prozessor gilt als etwas lahm. Ein Problem ist, daß er Float in Software mitmachen muß, weil Ada ohne Float unüblich ist. Er wird typisch in CMOS/SOS (silicon on saphire) gefertigt, was gut für Strahlungsfestigkeit und mil Temperaturbereich ist. Gilt als teuer, angeblich $5000/Stück.

Es gab sogar einen europäischen Hersteller: Plessey/ Marconi/Dynex. Anwendung: Militärflugzeuge, US-Panzer, zivile Raumfahrt.

ARINC-429 und MIL-STD-1553 sind serielle Bussysteme für Flugzeuge, die in den 70ern eingeführt wurden. ARINC-429 Zivilflugzeuge, 1553 Militärflugzeuge und Trägerraketen a la Ariane, rein von der Hardware her wesentlich robuster als CAN.

MfG JRD

Betreff: HolonR2k fuer Rabbits
Von: Wolf Wejgaard <wolf@see.organization.below>

Es ist eine Freude, hier eine neue Holon-Variante vorzustellen, für Entwicklungen an Rabbit Prozessoren.

HolonR2k ist aus Holon11 entstanden und bietet alle guten Holon-Eigenschaften wie u.a.:

- Quelltextverwaltung in einem Browser, Hypertext ohne zu-

saetzliche Linkfiles, direkten Zugriff zu jedem Wort im gesamten Programm, auch bevor die Worte geladen sind, Buchstruktur mit Kapiteln=Modulen, Abschnitten=Gruppen und Paragraphen=Worte.

- Der Code wird parallel gehalten zum Text. Wird der Text im bestehenden Programm korrigiert, wird auch der Code punktuell im bestehenden Programmcode ersetzt, dies ist auch im laufenden Programm möglich. Es wird (wahlweise) nur der Code geladen, der tatsächlich gebraucht wird, trotz umfangreicher Systemmodule ist also der Code schlank.

- Der Debugger ist mit dem Editor integriert, beim schrittweisen Ausführen des Programms werden die Schritte im Quelltext im Editor gezeigt zusammen mit Daten- und Returnstack. Holon führt auch Assemblercode schrittweise aus, mit Darstellung des Schrittes im Quelltext und Anzeige der CPU Register anstelle des Returnstacks.

- Ein Projekt wird in nur zwei Files gemanagt, je eines für Struktur und Text. Kein Verwalten von Filemengen, kein make und so, uvm.

Für weitere Infos bitte im Web nachschauen, siehe unten.

Kurz, Holon ermöglicht ein anderes sehr direktes Entwickeln.

HolonR2k ist entstanden dank der Initiative und Energie von Rafael Gonzalez Fuentetaja, der sich für seine Arbeiten ein Holonsystem für Rabbits wünschte. Rafael hat die Rabbit-bezogenen Teile gebaut -- Assembler, Monitor und alle Target Programm Module -- und er ist und bleibt auch der Rabbit Experte für HolonR2k.

HolonR2k kann frei verwendet werden. Zugang über

http://holonforth.com/tools/holonr2k.htm.

Möge es nützen, *Wolf Wejgaard*

Betreff: Neue VD
Von: Klaus.Schleisiek@send.de

Hallo Fritz,
herzlichen Glückwunsch zur neuen VD, ich habe Sie mit Interesse gelesen und muß leider gleich meckern, aber sonst hättest Du diese Mail gar nicht bekommen:

Auf Seite 33 "OVER heißt der kleine "Computer aided Programmer" ... ist natürlich Unsinn! Kennst Du einen "non-computer" aided Programmer? Ich nicht. Es handelt sich nämlich um einen "µCore aided Programmer" und das ist wirklich was Trendiges ;-)

Ich entwickele oft Programme, ohne mich dabei von einem Computer unterstützen zu lassen. Das sind dann natürlich keine Programme für einen Computer. Und wenn ich Forth programmiere, dann unterstützen mich Holon oder das ZF, aber nicht der Computer. ;-) *fep*

Ansonsten stellt sich in der konkreten Nutzung des uCores (siehe: www.microcore.org) in einem Produkt heraus, dass diese Herangehensweise an Embedded Systems Engineering noch wesentlich besser ist, als ich erhofft hatte. Wir sind notorisch mit der Softwareentwicklung schneller fertig als geplant - weil wir 1. uns die Hardware so bauen, wie wir sie von der Software her einfach bedienen können und weil wir 2. keine Zeit verlieren, um um unveränderliche Hardwarefehler herumzuprogrammieren bzw. erst mal dahinter zu kommen, dass es sich um so etwas handelt und das Ganze braucht 3. auch noch weniger Platz und Platinenlayoutprobleme, weil wir den Prozessorchip nicht mehr einbauen müssen.

Insbesondere das "Exception"-Konzept beinhaltet ungeahnte Möglichkeiten.

So haben wir inzwischen den üblichen 10ms-Counter, der Interrupts erzeugt, durch einen Delaytimer ersetzt, der auf 1 ms genau auflöst, keinen Interrupt mehr belegt und mit dem Exception Mechanismus "Warten" in der Software so erzeugt:

```
: ahead  ( time -- time' )   Delaytimer @ + ;
: sleep  ( time -- )         Delaytimer ! ;
```

... &200 ms ahead sleep ...

erzeugt eine Wartezeit von 200 ms (an der Stelle "Delaytimer !") und schaltet in eine andere Task um, solange die Zeit noch nicht erreicht ist (oder tritt auf der Stelle, wenn es keinen Multitasker gibt).

Keine Magie: Exception.

Oder die Semaphore, die wir gebaut haben, um unsere AD-Wandler zu konfigurieren (ekelhaft, über SPI-Interface!) oder die IDE-Festplatte zu bedienen. Krönung des Ganzen ist dann eine spezielle Instruktion, die ich EVENT genannt habe, die sowohl einen Semaphor abfragt, ob ein bestimmtes Ereignis stattgefunden hat, als auch einen Timeout überwacht, mit dem längstens auf das Ereignis gewartet wird. So wirds in der Software benutzt:

... &200 ms ahead <eventmask> EVENT ...

Solange weder das Ereignis, noch der Timeout erreicht ist, kommt das Programm an EVENT nicht weiter und macht statt dessen einen Taskwechsel. Wenn jedoch das Ereignis gemäß <eventmask> eingetreten ist, geht es weiter mit rückgesetztem Carrybit, wenn der Timeout zuschlägt, geht’s weiter mit gesetztem Carrybit, was ich dann bequem abfragen kann, weil uCore eine "carry IF" Instruktion hat.

Es gibt übrigens noch Prototypingboards auszuleihen und zu kaufen.

Interessenten bitte bei kschleisiek@send.de melden.

Mehr dazu auf der nächsten euroForth Konferenz in Dagstuhl vom 19.-22.11.2004 :-)

Klaus Schleisiek

Mit dem festen Vorsatz, diese Ausgabe noch so rechtzeitig den Lesern in die Hände zu geben, daß die Informationen über die euroForth nicht erst nach dieser Veranstaltung erscheinen, enthält diese VD die Einladung zur EuroForth (letzte Seite)

fep

Betreff: VD 3/2004
Von: Rafael Deliano

Ausgabe 3/04 dankend erhalten, dazu kann ich noch einen Leserbrief beisteuern:

Die Altersstruktur der Münchner Gruppe ist zwar auch nicht günstiger als die des Vereins im allgemeinen, wie man am Foto auf der Rückseite des Heftes sieht. Aber hat sich wenigstens den jugendlichen Elan erhalten, wie man beiliegender Orginalaufnahme entnehmen kann (vgl. MCHN.gif).

Zur üblichen Klage der Redaktion wegen quantitativen Mangels an Artikeln sei anzumerken, daß spätestens der Leser auch qualitative Ansprüche stellt. Positives Beispiel ist der Artikel von Eckes, der ein praktisch relevantes Thema lesbar, d.h. durch Illustrationen anschaulich darstellt. Auch Artikel wie die Biographie von C. Moore, die zwar weniger technisch, aber um so mehr unterhaltend sind, werden gern konsumiert.

Für Quantität gilt: Autoren rekrutieren sich aus Lesern. Wem die Leser schwinden, dem schwinden die Autoren. Selbst eine inhaltliche Besserung der VD würde an der Welt jedoch unbemerkt vorbeigehen. Die Androhung in der VD 2/04, "Diese Ausgabe der VD wird vermutlich ca. 4 bis 6 Wochen nach dem Erscheinen der Druckausgabe im Internet ... [als] PDF-Version ... zur Verfügung stehen" ist anscheinend folgenlos geblieben. Wo nichts ist, wird der Googlebot nichts finden.

Die Migration der VD weg von Papier hin zur Download-Version im WWW scheint ohnehin nicht als nötige Veränderung begriffen worden zu sein. Mini-Zeitschriften / Fanzines auf Papier hatten ehedem ihren Charme und ihre Berechtigung, weil die Leute sonst in diesen Nischen keine geeigneten Informationsquellen und Kommunikationsmedien hatten. Das hat sich durchs Internet drastisch geändert. Wer sich den Veränderungen nicht anpasst, wird nicht überleben. Der Verein ist aber bekanntlich reformunfähiger und untätiger als jede deutsche Behörde. Im Gegensatz zu denen aber ohne eingebaute Bestandsgarantie.

Wenn ich mir das Foto aus Fehmarn ansehe, habe ich allerdings die Vermutung, daß es nur noch um Jugenderinnerungen, Konservierung der verklärten Vergangenheit geht. Man hat sich innerlich mit dem langsamen, würdevollen Aushauchen des Vereins-Lebens abgefunden.

MfG JRD

Betreff: Golden Swap
Von: Volvovid@aol.com (Henry Vinerts)

Seit gestern bin ich überraschter und stolzer Besitzer eines Goldenen Swaps, der völlig unerwartet hier eingetroffen ist, zusammen mit einem eindrucksvollen Ordner, der Eure freundlichen Worte an mich in zwei Sprachen enthält.

Meinen Dank an alle meine Forth-Freunde in Deutschland!

Ebenso danke ich sehr für die beigelegte Photographie, in der ich letztlich einige Menschen identifizieren kann, die mir bisher nur durch meine Korrespondenz bekannt sind. Es freut mich zu sehen, daß Klaus Schleisiek dieses Mal nicht durch ein Gebäudeteil verdeckt wird, wie zuletzt in 2003 in Lambrecht, und es freut mich zu sehen, daß Friederich dieses Mal keine Zigarette raucht, und daß Martin Bitter seinen Sohn Malte mitgebracht hat.

Henry, ich habe vor mehr als drei Jahren mit dem Rauchen aufgehört. Das hat hier in Deutschland längst noch nicht jedermann mitbekommen. ;-) *fep*

Ist meine Annahme richtig, daß Claus-Werner Vogt derselbe Claus ist, mit dem ich korrespondiert habe, als er vor Jahren die Vierte Dimension produziert hat?

Da sind noch andere auf dem Photo, mit denen ich über Jahre hinweg E-Mails ausgetauscht habe. Es tut mir Leid, daß ich Wolf Wejgaard nicht auf dem Photo sehe, aber von ihm habe ich ein anderes Bild, auf dem er zusammen mit dem Super-Übersetzer, dem freundlichen Professor Beierlein aus Mittweida zu sehen ist.

Unglücklicherweise hat mich ein Schach-Turnier für Kinder von dem Juli-Treffen der SV-FIG abgehalten. Es gibt aber eine ausführliche Zusammenfassung über dieses Treffen bei:

http://www.forth.org/svfig/kk/07-2004.html

Das nächste Treffen ist für den 21. August vorgesehen. Wenn alles glatt läuft, werde ich sehen, welche Lebenszeichen ich dort vom Forth im Silicon Valley entdecken kann. Natürlich werde ich dabei den gerade erst erhaltenen Goldenen Swap tragen und vorzeigen.
Mit bestem Dank *Henry*

...Anmerkung des Festausschusses...

Henry bezieht sich in seinem Dankesbrief auf eine Ehrung, die ihm, Charles Moore und Elizabeth Rather zuteil wurde. Ein goldener SWAP in einer mit blauem Samt ausgeschlagenen Klip-Klip-Dose, ein auf Bütten gedruckter Text, aus dem hervorgeht, für welche Leistungen die Mitglieder der Forthgesellschaft ihn besonders ehren wollen, und das Photo, das auch auf der Rückseite der VD 03/2004 veröffentlicht wurde, zusammen mit der Auflistung der auf dem Photo abgelichteten Personen – das ist es, was „unseren Mann im Silicon Valley“ zu seinem Brief veranlaßt hat.

Bütten und Photo sind in einem schmalen Hefter sicher aufbewahrt. Davon wird in 2005 in Sachsen zumindest ein Ansichtsexemplar bereit liegen.

Weitere goldene Swaps sind samt Texten, Ordnern, Photos usw. vorbereitet, konnten aber erst kürzlich verschickt werden, weil dem Festausschuß die Adressen von Tom Zimmer und Leo Brodie nicht bekannt waren.

Zusätzlich sind fünf goldene SWAPs zum Versand vorbereitet. Damit sollen Menschen beschenkt werden, die aus den Reihen der Mitgliedschaft noch zu benennen sind. Hier gilt, daß die Benennung gleich den zu begleitenden Text enthalten soll (Warum soll ein goldener Swap an ... ?).

Von Elizabeth Rather liegt uns bisher keine Reaktion auf die Ehrung vor. Sobald diese eintrifft, werden wir sie in der jeweils folgenden VD veröffentlichen. Charles Moore hat sich bereits in einer Mail bedankt (siehe Seite 25).

Herny Vinerts hat vom Festausschuß einen Beutel mit rund 100 silbernen SWAPs bekommen, verbunden mit der Bitte, diese nach eigenem Ermessen auf den Treffen der SV-FIG zu verteilen. Henry hat uns zugesagt, der VD über das Verteilen und die darauf folgenden Reaktionen zu berichten. Den ersten silbernen SWAP hat er auf besondere Bitte des Festausschusses an seine Frau gegeben, die ihm seit vielen Jahren den Raum für Forth und seine Berichte aus der SV-FIG läßt, wofür ihr der Festausschuß ausdrücklich dankt.
Für den Festausschuß

Friederich Prinz

Der Festausschuß hat den goldenen SWAPs folgende Texte beigegeben:

...im April 2004 sind anläßlich der 20. Jahrestagung der Forthgesellschaft eine Reihe von Menschen besonders ausgezeichnet worden. Die Auszeichnung ist das sichtbare Zeichen der Anerkennung für unterschiedliche und vielfältige Arbeiten, die von diesen Menschen für die Forthgesellschaft und für Forth bisher geleistet wurden.

Elizabeth Rather: Besondere Leistungen in diesem Sinne wurden auch von Dir erbracht. Hierzulande ist dein Name verbunden mit professionellem Engagement für Forth durch breiten Einsatz und Weiterentwicklungen. Du hast dafür gearbeitet, Forth bekannt zu machen mit Publikationen über die Grenzen der USA hinaus. Und schließlich hat dein Einsatz im American National Standardisation Komitee, in dem du für ein gut handhabbares ANS-Forth als breite Basis für viele Forthsysteme gerungen hast, dich hier allen bekannt gemacht, die je mit Forth in Berührung kamen.

Charles Moore: Du hast 1968 Forth erfunden und darum möchten wir Dir heute unser aller herzlichsten Dank aussprechen für dieses wunderbare Geschenk, das du uns damit gemacht hast. Alle Forthsysteme, die bisher darauf aufbauten, und nun die Forthchips sind ein lebendiges Zeugnis der Bedeutung Deiner damaligen grundlegenden Arbeit.

Henry Vinerts: Besondere Leistungen in diesem Sinne wurden auch von Dir erbracht. Deine Berichte aus der Forth-Szene der USA, insbesondere die „aktuellen Lebenszeichen aus der FIG Silicon Valley“, schlagen eine Brücke zwischen dem alten Europa und der immer noch dynamischen Forth-Gemeinde in den USA. Die Leser der „Vierte Dimension“ erfahren durch Deine lebendigen Berichte seit vielen Jahren, was unsere amerikani-

schen Freunde umtreibt. Meinungen, Stimmungen und die Trends aus dem Mutterland des Forth erfahren wir durch Dich.

Tom Zimmer: Besondere Leistungen in diesem Sinne wurden auch von Dir erbracht. ZF, F-PC und Win32For sind für sich selbst sprechende Geschenke, die Du der Forthgemeinde gemacht hast.

Leo Brodie: Besondere Leistungen in diesem Sinne wurden auch von Dir erbracht. Besonders deine beiden Bücher, Starting Forth und Thinking Forth, haben die Programmiersprache Forth hierzulande vielen Menschen näher gebracht. Eine kleine Zeichnung aus deiner Feder hat dabei besondere Berühmtheit erlangt: Der "swap" Drache mit den beiden Köpfen. Er wurde zum Sinnbild für den Stack und schließlich zum Symbol für Forth selbst. Deine Zeichnung gab die Anregung für eine Bronzefigur, den "Swapdrachen der Forthgesellschaft," der seit 1986 jedes Jahr als besondere Auszeichnung an eine um Forth verdiente Person vergeben wird. Und es gibt ihn nun auch als Anstecknadel in Emaille, in Silberfassung für alle Mitglieder der Forthgesellschaft, und in goldener Fassung für Menschen, die sich besonders um Forth verdient gemacht haben.

Die Gemeinschaft der Forth-Freunde in Deutschland ist der einhelligen Auffassung, daß Dein Name in der Reihe der Ausgezeichneten nicht fehlen darf. Darum ist es uns eine Freude und eine Ehre, Dir im Auftrag der Mitglieder der Forthgesellschaft für Deine bisherige Arbeit für Forth zu danken, und Dir auf diesem Wege den "goldenen Swap" zu überreichen.

Alle Briefe wurden für das Direktorium von Fred Behringer unterschrieben.

Bericht aus der FIG Silicon Valley

Hallo,

wie das Schicksal es wollte, und wegen des Terminplanes des Cogswell Collegs, fand unser SV-FIG Treffen in diesem Monat eine Woche früher statt als gewöhnlich, so daß die SWAPs, die mich gestern erreicht haben, das Treffen um gerade einmal zwei Tage verpassen mußten. Und jetzt weiß ich nicht einmal sicher, zu welchem Termin das Treffen im September angesetzt ist. Aber ich werde versuchen, dort zu sein.

Die Teilnehmerzahl hat am letzten Samstag 15 betragen. Mir scheint, daß Dr. Tings Vortrag, der eine Fortführung seines aktuellen Projektes darstellte, eine handvoll nicht-regulärer Teilnehmer angezogen hat.

Wie immer gab es am Vormittag keine Pause, als wir den Vortrag über Details in der Portierung von e-forth auf Analog Devices' ARM7 Microprozessoren hörten. Aktuell besteht Dr. Tings Projekt aus zwei Phasen.

Die erste Phase befaßt sich mit etwas, was er "Firmware Engineering Workshop" nennt - ein Kurs mit einem Textbuch für chinesische Studenten der Computerwissenschaften. Dafür hat er erfolgreich das e-forth auf den ARM7 in den Game Boy Advance portiert.

Die zweite Phase hat mit der Entwicklung eines preiswerten digitalen Speicheroszilliskops zu tun, kurz: DSO. Diese Arbeit ist noch unvollendet, weil der UART auf dem ADuC7024 Microcontroller den Ladeprozeß des e-forth auf den ARM7 behindert. (Ich weiß nicht, ob ich dies alles korrekt wiedergebe. Ihr müßt einem Auslandskorrespondenten vergeben, der versucht, eine ihm fremde Kultur zu beschreiben). Der eigentliche Punkt hierbei ist, daß die 260 Dollar für das Entwicklungskit von Analog Devices preiswert sind, zumindest im Vergleich zu den tausenden von Dollar, die für kommerzielle DSOs bezahlt werden müssen.

Wenn ich die Mittagspause und die wie immer mehr als ausreichend bemessene und hoch willkommene Zeit für den "informal information interchange (III)" nicht mitrechne, dann haben wir den Rest des Tages damit verbracht, einem interessanten historischen Abriß des Nike-Raketensystems zu folgen, vorgetragen von Ed Thelen. Vielleicht erinnert Ihr euch an meinen Bericht über das Treffen im Januar. Eds Sohn Randy ist der "Vater" eines selbstgebauten TTL-basierten Forthcomputers mit dem Namen MIPPY. Wir sind froh, den "Großvater" jetzt ebenfalls in der SV-FIG zu haben.

Es ist ein großes Privileg für die SV-FIG, daß wir uns im Kolleg treffen können. Dieses Mal bekamen wir einen Raum zugewiesen, der mehr als ein Dutzend Computer online hatte. Wir konnten Ed Thelens Vortrag darum auf jeweils "eigenen" Bildschirmen folgen und mußten die alten Augen nicht mit Durchlichtprojektionen belasten.

Ich denke, Ihr werdet

http://ed-thelen.org/

besuchen wollen, und Euch gerne selbst ein Bild machen.

Es übersteigt mein Vorstellungsvermögen, welches Ausmaß die Luftabwehrsysteme hatten, die in der Zeit der Vakuum-Röhren vor einem halben Jahrhundert real existierten.
Und die Menge des Geldes, die in diesen Tagen von den Menschen ausgegeben wurde, um sich zu verteidigen (oder andere Menschen anzugreifen) ist mir schlicht unbegreiflich.

An dieser Stelle hoffe ich, daß ich Euch demnächst mehr über das "forthige Leben" in der SV-FIG senden kann.

Henry

Übersetzung: Friederich Prinz

Gehaltvolles

zusammengestellt und übertragen
von Fred Behringer

FORTHWRITE der FIG UK, Großbritannien

Nr. 126, Juli 2004

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

Chris Jakeman wird das Amt des Redakteurs nicht wieder aufnehmen. Sein neuer Job als Dozent und seine Ambitionen, in die Forschung zu gehen, lassen das nicht zu. Graeme war ursprünglich nur "eingesprungen". Er hat ebenfalls nicht genügend Zeit und möchte im Übrigen sein eigentliches Amt, das des Verwalters des Literatur-Verleihs, wieder voll aufnehmen. Es wird nach Lösungen gesucht.

3 Forth News
Graeme Dunbar

euroFORTH 2004 auf Schloss Dagstuhl, organisiert von Anton Ertl - 25 Jahre FIG-UK im November 2004 - Die FIG-UK-Forth-CD ist beinahe fertig - Ein neuer Forthwrite-Editor wird gesucht: Chris Jakeman ist endgültig zurückgetreten. Graeme wird auf jeden Fall auf der nächsten Jahresversammlung (im November) seinen Rücktritt erklären. Sein Forth-Wissen und seine Möglichkeiten sind beim Literatur-Verleih besser aufgehoben als bei der Herausgabe der Forthwrite. Graeme betrachtet die Forthwrite als ein zentrales Organ der FIG UK. "Man könnte und sollte sich die Arbeit teilen! Ein Team muss her!"

4 Debugging Tools
Steven Pelc <mpe@mpeltd.demon.co.uk>

Steven Pelc von MPE Ltd. (MicroProcessor Engineering) gibt die in der Datei Common\DebugTools.fth enthaltenen Debug-Tools wieder, die für MPE-eingebettete-Systeme gedacht sind, welche von Forth-6-Cross-Compilers erzeugt wurden. Hauptaugenmerk ist auf 32-Bit-Systeme und interaktives Austesten gerichtet. Die Programmstücke können leicht auf andere Systeme übertragen werden. Für nichtgewerbliche Zwecke bestehen keine Einschränkungen der Verwendung. Copyright bleibt bei MPE.

12 Certifying your Code
Paul E. Bennett

Das vollständige Austesten und Verifizieren eines Programms, gleich in welcher Programmiersprache, wird häufig vernachlässigt. "Wie vollständig aber ist vollständig?" Der Autor zeigt, dass Forth für diesen Zweck nicht ungeeignet ist. Paul E. Bennett ist ein unabhängiger EDV-Berater, der sich in sicherheitskritischen Industriezweigen (Atomenergie, Petrochemie, Schifffahrt, Eisenbahn, Medizintechnik, Transportwesen) auskennt.

17 Letters

Ein Brief von James Boyd: In Win32Forth gibt es keine Compiler-Sicherheit für Locals. Das Folgende reicht:

```
: locals| ?csp postpone locals| ; immedia-
te
:{ ?csp postpone { ; immediate
```

James gibt weitere Sicherheits-Worte, die auch den Returnstack einbeziehen.

18 A VNM in Forth: update
James A. Boyd

Im Anschluss an seine beiden Artikel über (V)irtuelle (N)ichtdeterministische (M)aschinen in Forth liefert James eine kleine Modifikation, die auch Locals berücksichtigt.

20 Book Review
Boris Fennema

Boris Fennema bespricht das Buch "Design of embedded systems using 68HC12/11 microcontrollers" von Richard E. Haskell. Es geht im Buch hauptsächlich um den HC12. Der HC12 hat Fuzzy-Logik-Hardware und ist für die englischen Forth-Freunde wegen ihres F12UK-Projektes interessant.

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

Henrys Berichte vom April/Mai/Juni 2004 über SVFIG-Aktivitäten in Originalfassung. Wir kennen sie in der Übersetzung von Thomas Beierlein. Henry hatte sich inzwischen für die Verleihung des FIG-UK-Forth-Errungenschafts-Preises bedankt und gesagt, er fühle sich so, als wenn die Königin ihn zum Ritter geschlagen hätte. Graeme Dunbar, der Redakteur, rät Henry, sich den Preis nicht zu Kopf steigen zu lassen und nach dem Ritterschlag das gemeine Volk nicht zu vergessen: "Henry, wir freuen uns schon auf deinen nächsten Bericht!"

24 What Languages Fix
Graeme Dunbar

Ein paar gesammelte Antworten aus dem Internet zur Frage, was Forth als Sprache festlegt. Unter anderem eine Forth-Charakterisierung von Bernd Paysan: FORTRAN ist nicht interaktiv und Chucks Prä-Forth-Interpreter war nicht erweiterbar.

25 Dutch Forth Users Group

Die Anzeige der holländischen Forth-Freunde zur Anwerbung von Mitgliedern für die HCC-Forth-gebruikersgroep.

26 Vierte Dimension 1/2004
Joe Anderson <jia@jus.abel.co.uk>

Joe bespricht das Heft 1/2004 unserer Vierten Dimension.

28 FIG UK Contacts and Information and Services to Members

Die Mitgliedschaft kostet 12 englische Pfund pro Jahr. Sie berechtigt zu sechs (in Ausnahmefällen, wie dieses Jahr, nur fünf) Forthwrite-Heften. Heft 127 wird bei Erscheinen der vorliegenden VD schon erschienen sein. Redaktionsschluss für Heft 128 ist der 27. Oktober 2004.

VIJGEBLAADJE der HCC Forth-gebruikersgroep, Niederlande

Nr. 45, August 2004

Forth vanaf de grond 3
Ron Minke

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

"Nach den drei Festsetzungen im zweiten Teil dieser Artikelserie interessieren wir uns nun dafür, wie wir Daten auf den Returnstack legen können. Für den Umgang mit dem Returnstack gibt es im AVR-Prozessor zwei spezielle Maschinencode-Befehle: PUSH und POP."

Tournee: Ushi en de Tingeltangels
Albert Nijhof

Albert berichtet für die Mitglieder der HCC-Forth-gebruikersgroep über den Besuch von Willem Ouwerkerk und ihm bei den "collega's van de Duitse Forthgroep" zum zwanzigsten Jahrestag der Gründung der Forth-Gesellschaft. Beide ihrer Demonstrationen auf der Tagung, die unverwüstlichen geräuschvollen Tingel-Tangels und Ushi, die (oder der?) ihren (seinen) Namen schrieb, sagt Albert, wurden "zeer gewaardeerd".

Zelf een Forth maken
Albert Nijhof

Unsere holländischen Forth-Freunde hatten eine Anzahl von 6809-Prozessoren aus Restbeständen bekommen und wollten (wollen) sich dafür ein Forth bauen. Sie bildeten eine Projektgruppe und fanden im Internet für den 6809 CamelForth. Der Camel-Metacompiler (zum Aufbau des 6809-Forths) basiert auf F83 mit dessen für Veränderungen unhandlicher Blockstruktur. Man wollte für die Meta-(Cross-)Compilation ein moderneres (32-Bit-)-PC-Forthsystem verwenden können. Anpassung schien zu schwierig. Albert schlug vor, einen ganz neuen Metacompiler zu schreiben. Es kamen Zweifel an der Durchführbarkeit auf. Albert zerstreute sie, indem er einfach ganz schnell einen solchen Metacompiler schrieb. Das Problem mit Worten gleichen Namens löste er dadurch, dass er dem Metacompiler einen eigenen Interpreter (im Artikel in sechs Zeilen niedergeschrieben) spendierte. Frans van der Markt passte den Cross-Assembler (siehe gleich folgenden August-Vortrag) an.

Tagesordnung des Treffens vom 14. August 2004:

10.30 Einlass in den Saal und Kaffee für Frühankömmlinge
11.00 "Forth selber machen?" (Albert Nijhof & Frans van der Markt)
12.00 Pause, mit Gelegenheit zu informellen Kontakten.
13.00 "I quadrat c" (Willem Ouwerkerk)
14.30 Schluss

Die Treffen finden jeweils statt in: KVG, Bisonspoor 1204, Maarssenbrook. Die Routenbeschreibung findet man unter:

http://www.forth.hccnet.nl/nieuws .

ANS-Forth-Buch für Ein- und Umsteiger

Das ANS-Forth-Kursbegleitbuch "Die Programmiersprache Forth" von Albert Nijhof (Übers.: Fred Behringer) befindet sich in der Druckvorbereitung und wird in den nächsten vier Wochen unter dem eben genannten Titel und der ISBN 3-937312-79-X bei allen dem Internet angeschlossenen Buchhandlungen aufgeführt werden. Es kommt beim mv-Verlag in Münster heraus und kann in den Buchhandlungen im "Books-on-Demand"-Verfahren bestellt werden. Ladenpreis Euro 21,50. Etwa 150 Seiten. Das Buch ist fürs Selbststudium am Computer gedacht und erscheint der leichteren Handhabung wegen in Drahtkammbindung (bleibt seitenweise aufgeschlagen liegen) im Format DIN-A4.

beh

Hashing für RFIDs

Rafael Deliano

RFIDs sind über Funk abfragbare Seriennummer-ICs. Hashing ist ein Verfahren für effizientes Suchen in Listen.

Die preiswertesten 125kHz Transponder (RFID) haben eine feste Nummer, die vom Hersteller z.B. per Laser eingeschrieben wird. Damit dieser trotz kontinuierlicher Fertigung eine einzigartige Seriennummer garantieren kann, ist diese ziemlich lang; typisch 64 Bit. Da darin auch Prüfsummen enthalten sind, ist die effektive Länge kürzer. Um die Portierung zwischen verschiedenen Anbietern zu vereinfachen, ist es für ein Lesegerät trotzdem günstiger, mit der vollen Länge von 64 Bit zu arbeiten.

Bei Anwendung in größeren Gebäuden und Hotels kann es erforderlich sein, daß das Lesegerät hunderte von Schlüsseln erkennen soll. Wenn diese in einem externen seriellen EEPROM gespeichert sind, ist der Zugriff langsam (Bild 1). Um die Reaktionszeit des Lesegeräts kurz zu halten, muß die Suche optimiert werden.

Hashing

Das Suchen in ungeordneten, langen Listen ist ein altes Problem der Datenverarbeitung. Eine bekannte Lösung heißt Hashing. Durch eine geeignete Hash-Funktion wird aus dem 64 Bit Datensatz des Transponders ein 10 Bit Wert berechnet, der als Zeiger in die Liste mit den 1024 x 64 Bit Worten dient (Bild 2). Dort sucht man nun linear aufwärts weiter. Wenn oberhalb 3FF ein Moduloüberlauf eintritt, geht die Suche ab Adresse 000 weiter (Bild 3). Der Endpunkt ist der ursprüngliche Einsprungpunkt auf den der Hash-Key zeigt. Normalerweise muß man aber nur ein kurzes Stück linear suchen. Beim Einprogrammieren eines Transponders sucht man die nächste leere Speicherzelle; also alle 64 Bits gesetzt. Dort programmiert man die RFID-ID ein. Beim Suchen vergleicht man die RFID-ID des Transponders mit dem Inhalt der Speicherzelle auf Identität, bzw. prüft, ob die Speicherzelle leer ist. In letzterem Fall ist die ID nicht im EEPROM.

Das Verfahren reduziert die Suche in einer langen Liste auf die Suche in vielen kurzen Listen. Diese entstehen zwangsläufig, da Hash-Kollisionen (mehrere IDs zeigen auf eine Teilliste) nicht vermeidbar sind.

Weiter entwickelte Hash-Verfahren führen statt linearer Suche noch einen 2. Hash-Schritt durch, aber das ist hier zu kompliziert; besonders weil man aus dem seriellen EEPROM nach der Startadresse die folgenden Bytes kontinuierlich lesen kann, während man für Sprünge noch mal Kopf und Adresse schreiben muß.

Effizienz

Die Wirksamkeit des Verfahrens hängt von der Statistik der Eingangsdaten und der Anpassung der Hash-Funktion an diese ab, ferner davon, wie voll der Speicher ist. Wegen des statistischen Einflusses kann man die Reaktionszeit des Systems nur per Monte Carlo Simulation verifizieren. Das hängt stark davon ab, wie gut man die RFID-IDs nachbilden kann.

ID-Statistik

Für die Produktion von IDs gibt es zwei Varianten: simple Binärzähler oder Pseudozufallszahlen. Wenn letztere z.B. durch LFSR mit m-Sequenz erzeugt werden, ergeben sich auch einzigartige IDs. Die Hersteller von Transpondern machen keine Angaben darüber, wie sie ihre IDs erzeugen. Es scheint sich aber um Zufallszahlen zu handeln. Von Microchip ist für PICs bekannt, dass sie LFSR mit wenig Taps verwenden [1]. Insofern liegt man mit dem simplen 64 Bit Polynom von Stahnke [1] für Simulation wohl durchaus richtig.

Ein simpler Verteilungstest ist, für alle 64 Bits die Wahrscheinlichkeit zu ermitteln, für die das Bit den Wert 1 hat. Dazu lässt man den Generator z.B. 1024 Samples erzeugen und zählt dabei in 64 Zählern mit, ob das jeweilige Bit 1 war. Für einen echten, guten Zufallszahlengenerator wäre der Wert überall 50%. Beim 64 Bit LFSR à la Stahnke ist die Verteilung etwas vom Startwert abhängig (Bild 4). Das liegt auch daran, daß die Stichprobe zu klein ist: nur 1024 von (2^64)-1 Worten. Beim Binärzähler ist die Schlagseite offensichtlich. Nur die untersten Bits ändern sich.

Hash-Funktionen

Simpelste Variante ist, einfach nur bestimmte Bits zu kopieren (Bild 6). Geeignet sind Bits, die möglichst mit 50% Wahrscheinlichkeit den Wert 1 haben. Beim Binärzähler sind das die untersten Bits. In der Hash-Adresse haben die oberen Bits die grösste Wirkung. Also wird man sie dorthin legen. Entsprechend würde man höherwertige Bits der ID geringer bewerten. Das Verfahren funktioniert also nur sicher, wenn die Statistik des Eingangssignals bekannt und verlässlich ist.

In der Praxis ist das selten der Fall. Robustere Verfahren sind nötig, z.B. solche zur Prüfsummenberechnung. In [2] schneiden CRCs, Flechter und simples XOR über alle Bytes recht gut ab. Bezüglich Fehlersicherung ist die stark abfallende Qualität dieser Verfahren bekannt. Für Hashing sind die Qualitätsunterschiede aber nicht so deutlich. Ein Problem bei der Implementierung ist, daß für die Hash-Adresse der krumme Wert 10 Bit benötigt wird. Eine einfache Lösung ist, aus jeweils 32 Bit zwei 8 Bit Teilworte zu bilden und diese durch XOR zu verknüpfen (Bild 5).

Test

Man läßt den Generator wieder 1024 Muster erzeugen, die dann über die Hash-Funktion auf 10 Bit Adressen komprimiert werden. Anhand dieser inkrementiert man jeweils einen von 1024 16 Bit Zählern, die die Speicherzellen des EEPROMs darstellen. Im Idealfall enthält jeder Zähler nach Ende des Tests den Wert 1. Jede RFID-ID wäre dann einer anderen Adresse zugeordnet worden. Praktisch kommt es aber zu Kollisionen. Also enthalten manche Zähler Werte grösser 1. Dementsprechend werden einige Adressen überhaupt nicht angesprochen und enthalten den Wert 0.

Tabelle 1 zeigt die Ergebnisse für 8 Bit CRC, Fletcher und XOR. Mit Testdaten, erzeugt vom Binärzähler und LFSR, letzterer mit 3 verschiedenen Startwerten (vgl. auch Bild 4). Die Spalte „1" enthält die guten Fälle, rechts davon sind die Kollisionsfehler. Um den Vergleich zu vereinfachen, ist am Ende noch eine gewichtete Fehlersumme aufgeführt, die Mehrfachkollisionen verschärft bewertet. Dreifachkollisionen mal 2, Vierfachkollisionen mal 4, Fünffach mal 8 usw..

Die CRC erreicht zwar hier nahezu ideale Werte für den Zähler. Aber das Ergebnis von Fletcher ist ausgeglichener. XOR ist sehr schlecht, man beachte die Vielfach-Kollisionen weit rechts in der Tabelle.

[1] „PN-Sequenzen" embedded (5) S. 6
[2] Jain „A Comparison of Hashing Schemes for Adress Lookup in Computer Networks" IEEE Trans.on Com 1992/10 S. 1570 - 1573

...weitere Abbildungen auf der folgenden Seite...

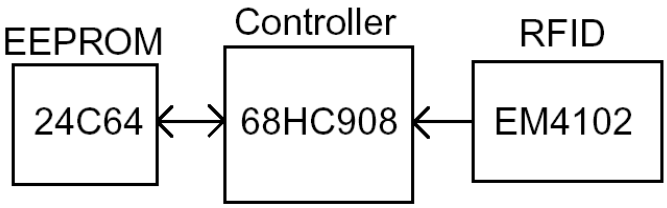
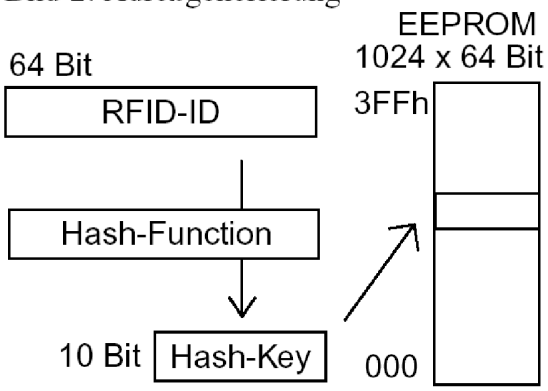


Bild 1: Lesegerät

Bild 2: Adreßgenerierung



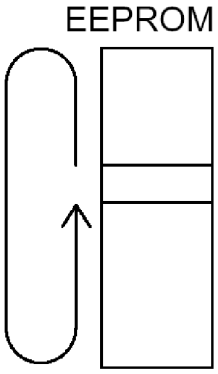


Bild 3: Suche

Bild 6:

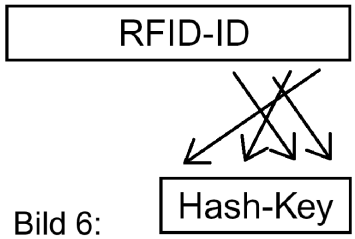


Bild 4: ID-Generatoren

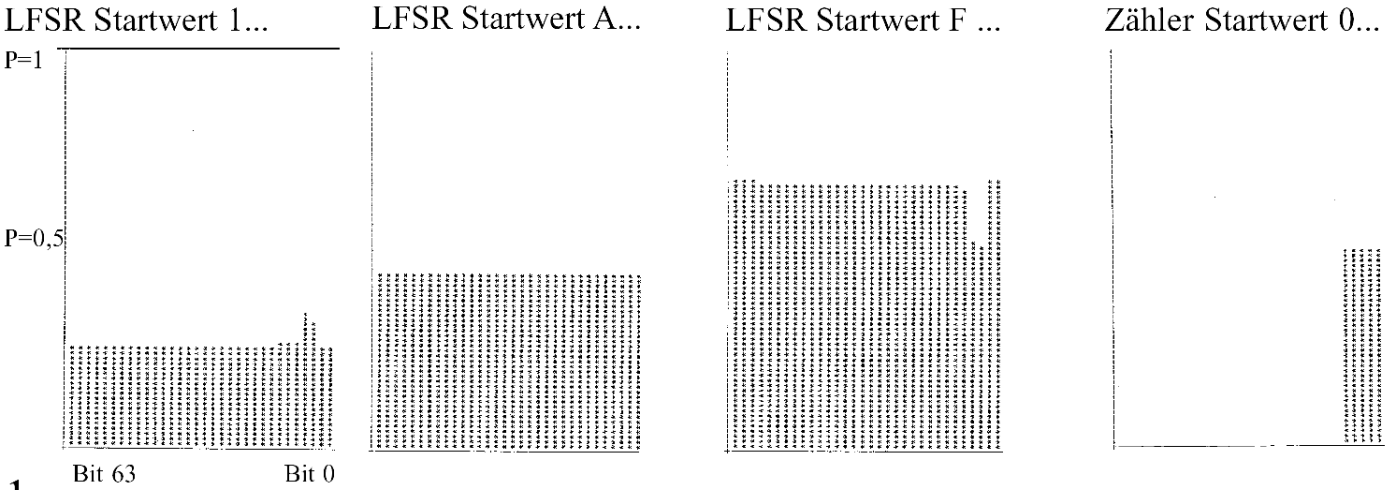


Tabelle 1: Ergebnisse Simulation

```
a = LFSR Startwert: 1000 0000 0000 0000 0000 0000 0000 0000
b = LFSR            FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF
c = LFSR            AAAA AAAA AAAA AAAA AAAA AAAA AAAA AAAA
d = Zähler          0000 0000 0000 0000 0000 0000 0000 0000

CRC
   0    1    2    3    4    5    6    7    8    9    A    B    C    D    SUM
a 0243 0088 009D 004D 002A 0018 0008 0001 0000 0000 0000 0000 0000 0000 033F
b 0245 0089 0091 005B 0026 0016 0008 0002 0000 0000 0000 0000 0000 0000 034F
c 0246 0086 008F 0060 0027 0017 0004 0003 0000 0000 0000 0000 0000 0000 0343
d 0001 03FE 0001 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0001

FLETCHER
a 0183 0162 00C8 0042 000D 0004 0000 0000 0000 0000 0000 0000 0000 0000 01A0
b 0172 0189 00B3 003D 0011 0002 0002 0000 0000 0000 0000 0000 0000 0000 01A1
c 0171 0178 00CF 003A 000A 0004 0000 0000 0000 0000 0000 0000 0000 0000 018B
d 00C8 027B 00B2 000B 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 00C8

XOR
a 0286 0097 0067 002B 0022 0005 000F 0009 0006 0003 0001 0002 0000 0000 0B7D
b 0312 0026 0033 002A 0018 0019 0011 000D 0007 0008 0002 0001 0004 0002 2E1F
c 0317 0035 0030 001C 0018 0013 000C 000B 000B 0008 0004 0005 0003 0001 2C40
d 0300 0000 0000 0001 00FE 0001 0000 0000 0000 0000 0000 0000 0000 0000 0402
```

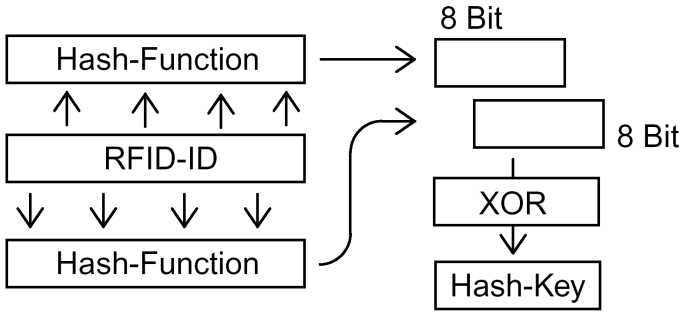


Bild 5:

Windows-Programmierung mit SwiftForth und MySQL

Stefan Schmiedl

<s@xss.de>

Die Vorgeschichte

Ein Konservenhersteller wollte kleine und schnelle Applikationen, die auch auf seinen "alten" Windows-Rechnern zügig laufen sollten. Die vorhandenen Daten aus der in Cobol programmierten Warenwirtschaft sollten wieder verwendet werden. Die Datenbestände selbst sollten zwischen verschiedenen, durch eine langsame Standleitung verbundenen Gebäuden der Firma repliziert werden können, da Eingabe und Auswertung an verschiedenen Stellen erfolgen sollten. Der Technische Direktor hätte die Daten auch gerne unter Access zur Verfügung gehabt, um ad-hoc Auswertungen zu erstellen.

Klingt spannend, nicht? Daten aus einem Fremdformat auslesen, Windows-Programme, Datenbanken ... alles drin, was Freude macht!

Als Datenbankserver setze ich gerne MySQL ein, weil es schön übersichtlich ist und dabei auch wartungsfreundlich und stabil. Die Programmierung der Applikationen wollte ich mit SwiftForth erledigen, denn ich hatte schon gemerkt, dass ich den Bogen vom Bit-Jonglieren bis zur hohen Abstraktion damit sehr schön spannen konnte.

Ein Überblick

Im ersten Teil dieser Artikelserie beschreibe ich die Anbindung von MySQL an SwiftForth vom Importieren der DLL-Funktionen bis zu einem komfortablen Satz an Worten für den täglichen Einsatz.

Der zweite Teil zeigt deren Anwendung in einer Batch-Applikation, die Daten aus den bestehenden Cobol-Dateien in die MySQL-Datenbank übernimmt. Außerdem gehe ich hier auf grundlegende Schritte bei der Entwicklung von Windows-Applikationen mit SwiftForth ein.

Einen dritten Teil habe ich für eine minimale GUI-Applikation (ein Eingabefeld, einige Schaltflächen) vorgesehen, die in dieser Form wirklich eingesetzt wird. Hier ist dann im Kleinen zu sehen, wie die Behandlung der unter Windows verschickten Botschaften vor sich geht.

Um Unschuldige zu schützen, wurden einige Namen im Code geändert, aber die Definitionen sind (bzw. waren zum Zeitpunkt des Schreibens) in der dargestellten Form erfolgreich im Einsatz (SwiftForth 2.2.2.9 und MySQL 4.0.18).

MySQL-Anbindung

Zunächst geht es darum, einen MySQL-Datenbankserver dazu zu bringen, die anfallenden Daten abzulegen. Ich behandle die einzelnen Teile in etwa in der Reihenfolge, in der sie auch im Quellcode vorkommen.

MySQL API

MySQL bietet ein sehr einfaches API für den Einsatz der Client-Bibliothek *libmysql.dll* an. Von diesem ohnehin schon schlanken API werden wiederum nur wenige Funktionen wirklich benötigt, zumindest für so einfache Applikationen, wie ich sie hier behandle. Besonders erfreulich ist auch, dass die vom Server zum Client übermittelten Daten alle als Null-terminierte C-Strings ankommen, wodurch der API-nahe Zugang noch einmal einfacher wird.

Verbindung zum Server

Der typische Ablauf einer Verbindung zu einem MySQL-Server sieht folgendermaßen aus: Zunächst reserviert man mit `mysql_init` einen Speicherblock für die Verbindungsdaten. Dieser Block wird bei vielen API-Funktionen als erster Parameter benötigt. Die Verbindung zum Datenbankserver wird mit der Funktion `mysql_real_connect` hergestellt, welche unter anderem die "üblichen" Verbindungsparameter verlangt: Hostname oder IP-Adresse des Datenbank-Servers, Benutzername, Kennwort sowie die Datenbank, mit der gearbeitet werden soll. Sind alle Anfragen bearbeitet, wird die Verbindung mit `mysql_close` wieder geschlossen.

Die Funktionsdeklarationen sehen im Detail so aus:

```
MYSQL *mysql_init(MYSQL *mysql)

MYSQL *mysql_real_connect(
  MYSQL *mysql, const char *host,
  const char *user,
```

```
  const char *passwd, const char *db,
  unsigned int port,
  const char *unix_socket,
  unsigned int client_flag)

void mysql_close(MYSQL *mysql)
```

Ruft man `mysql_init(0)` auf, wird der benötigte Speicherplatz von der Client-Bibliothek reserviert und mit `mysql_close(...)` auch wieder freigegeben. Der Einsatz von `mysql_real_connect` verliert auch viel von seinem Schrecken, wenn man beim Durchlesen der Beschreibung feststellt, dass „im Normalfall“ die letzten drei Parameter auf 0 gesetzt werden können.

Anfragen und Ergebnisse

Das Arbeiten mit SQL-Anfragen auf API-Niveau ist erfreulich einfach. Die Funktion `mysql_query` führt Anfragen aus, wobei hier noch kein Unterschied gemacht wird, ob die Abfrage eine Ergebnismenge zurückgibt oder nicht.

```
int mysql_query(MYSQL *mysql,
  const char *query)
```

Ein von Null verschiedener Rückgabewert zeigt einen Fehler bei der Ausführung der Anfrage an. Mit der Funktion `mysql_field_count` könnte man die Anzahl der Spalten ermitteln, die in der Ergebnismenge einer Anfrage zu finden ist, und so auf die Art der Anfrage zurück schließen, aber diese Allgemeinheit ist für unser Beispiel nicht notwendig. Wir wissen, was unsere Anfragen tun.

Um auf die Ergebnisse von `SELECT`-Abfragen zuzugreifen, gibt es zwei prinzipiell verschiedene Möglichkeiten. Holt man die Datensätze einzeln vom Server, kommt man in der Client-Applikation mit relativ wenig Speicher zurecht, blockiert allerdings den Server bis die Abarbeitung der Ergebnismenge abgeschlossen ist. Bei dem von mir bevorzugten zweiten Weg wird die komplette Ergebnismenge auf einmal zum Client übertragen.

```
MYSQL_RES *mysql_store_result(
  MYSQL *mysql)

my_ulonglong mysql_num_rows(
  MYSQL_RES *result)

void mysql_free_result(
  MYSQL_RES *result)
```

Die Funktion `mysql_store_result` *muss* aufgerufen werden, wenn eine Anfrage eine Ergebnismenge liefert, um die am Server belegten Ressourcen wieder freizugeben. Der clientseitig benötigte Speicher wird von der *libmysql*-Bibliothek verwaltet und mit der Funktion `mysql_free_result` wieder frei gegeben. Die Funktion `mysql_num_rows` schließlich gibt die Anzahl der Datensätze in der Ergebnismenge als ganze Zahl (einzellig, auch wenn my_ulonglong etwas anderes suggeriert) zurück.

Wie kommen wir an die Daten der Ergebnismenge? Für unsere Zwecke können wir uns hier auf eine der vorhandenen API-Funktionen beschränken:

```
MYSQL_ROW mysql_fetch_row(
  MYSQL_RES *result)
```

Das Ergebnis von `mysql_fetch_row` ist, wie schon angekündigt, ein Array von Zeigern auf Null-terminierte Zeichenketten mit Textrepräsentationen der Feldinhalte.
Zusammenfassung: Wir benötigen nur die folgenden acht `mysql`-Funktionen

```
_init, _real_connect, _query, _close,
_store_result, _num_rows, _fetch_row,
_free_result
```

SwiftForth und DLLs

Um in SwiftForth Funktionen aus einer DLL-Datei zu verwenden, muss diese zunächst einmal (im `PATH` oder aktuellen Verzeichnis) auffindbar sein. Wenn es sich nicht um Windows-eigene Bibliotheken handelt, ist meine Präferenz der aktuelle Applikationsordner, da dann keine systemweiten Einflüsse auftreten und eine einfache und vollständige Deinstallation durch Löschen dieses Ordners möglich ist.

Die Verbindung zur Client-Bibliothek wird mit

```
\ -- mysql.f   sws 04-09-03
library libmysql
```

hergestellt, anschließend können die benötigten Funktionen namentlich importiert werden.
SwiftForth stellt hierfür zwei Möglichkeiten zur Verfügung. Die ausführliche Variante

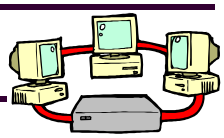
```
function: name ( arg1... -- ret )
```

verwendet den Stapelkommentar, um die Zahl der Parameter zu ermitteln, die Kurzversion `import:` erwartet die Zahl der Funktionsparameter auf dem Stack:

```
1 import: mysql_init
8 import: mysq_real_connect
1 import: mysql_close
2 import: mysql_query
1 import: mysql_store_result
1 import: mysql_num_rows
1 import: mysql_free_result
1 import: mysql_fetch_row
```

Anzumerken ist vielleicht noch, dass SwiftForth beim Aufruf die Parameter in der Reihenfolge erwartet, in der sie in der API-Deklaration geschrieben werden:

```
mysql_query(MYSQL *connection,
  char *query)
```



wird also zum Beispiel mit

```
conn z" show tables" mysql_query
```

aufgerufen, wenn `conn` auf eine aufgebaute Verbindung verweist.

Die erste Abstraktion

Das low-level API ist zwar schon verwendbar, aber für die meisten Anwendungsfälle unnötig wortreich und – zumindest für mich – anstrengend zu merken. Daher lege ich über die API-Importe eine Schicht aus "brauchbaren" Worten, über die im folgenden alle notwendigen Zugriffe auf den Datenbankserver abgewickelt werden. Zunächst jedoch definiere ich einige knappe Fehlercodes:

```
throw#
  s" Verbindungsfehler"
    >throw enum IOR-DBCONNECTION
  s" Anfragefehler"
    >throw enum IOR-QUERY
to throw#
```

Ein *Verbindungsfehler* soll ausgelöst werden, wenn beim Anmelden am Server etwas schief läuft, ein *Anfragefehler* bei fehlerhaften Anfragen.

Damit sieht die Verwaltung der Verbindung folgendermaßen aus:

```
: DB-INIT ( -- conn ) 0 mysql_init ;

: DB-CONNECT
  ( conn host user pw db -- )
  0 0 0 mysql_real_connect
  0= ior-dbconnection ?throw ;

: DB-CLOSE ( conn -- ) mysql_close ;
```

Für Anfragen, die keine Ergebnismengen zurückliefern, definiere ich ein einzelnes Wort:

```
: QRY-EXECUTE ( conn sql -- )
  mysql_query ior-query ?throw ;
```

Für SELECT-Abfragen und ähnliche Operationen, die eine Ergebnismenge liefern, stelle ich zunächst einmal das folgende Wort zur Verfügung, das neben der Ergebnismenge auch die Zahl der übertragenen Datensätze ermittelt.

```
: QRY-OPEN  ( conn sql -- res #rows )
  over swap
  mysql_query ior-query ?throw
  mysql_store_result
  dup mysql_num_rows ;
```

Der nächste Datensatz wird mit

```
: QRY-NEXT  ( res -- z[] )
  mysql_fetch_row ;
```

aus der Ergebnismenge gezogen, bevor der reservierte Speicherblock mit

```
: QRY-FREE  ( res -- )
  mysql_free_result ;
```

wieder freigegeben wird.

Ich verwende oft Abfragen, bei denen genau ein Datensatz zurückkommt, der unter Umständen auch nur aus einem einzelnen Feld besteht. Für diese Fälle definiere ich bequeme Abkürzungen:

```
: QRY-1ROW  ( conn sql -- res z[] )
  qry-open 1 - ior-query ?throw
  dup qry-next ;
: QRY-1FIELD  ( conn sql -- z )
  qry-1row @ zcount pad dup>r
  zplace qry-free r> ;
\ mysql.f --
```

Mit diesen wenigen Worten lässt sich schon ganz gut mit der Datenbank arbeiten, so richtig bequem ist es allerdings noch nicht. Denn in den meisten Fällen ist der Abfragetext bis auf einige wenige Parameter bekannt und konstant, so dass wir dringend eine Methode benötigen, mit der wir solche parametrisierten Strings bearbeiten können.

Strings mit Platzhaltern

Beim Ausfüllen von Stringvorlagen gilt es, verschiedene Anforderungen zu beachten. Zunächst einmal soll der „Arbeitsbereich“, in dem der ausgefüllte String aufgebaut wird, vom Aufrufer zur Verfügung gestellt werden. Die Lückenfüller erfordern gelegentlich noch die eine oder andere Manipulation, daher sollte während des Ausfüllens der Stapel „frei“ sein.

Analog zu anderen, ähnlich „geklammerten“ Konstruktionen in SwiftForth könnte die Benutzung (mit einem Fragezeichen als Platzhalter) also so aussehen:

```
z" ?, ?" buffer size [pstr
  z" hallo" >>
  z" welt"  >>       pstr] zcount type
```

Das Ausfüllen von String-Vorlagen verspricht, allgemein nützlich zu sein, weshalb ich diese Funktionalität in ein eigenes Package stelle. Packages werden in SwiftForth zur Organisation des Namensraums verwendet. Definitionen innerhalb eines Package sind standardmäßig „private“, das heißt, nur innerhalb der Package-Defintion sichtbar.

```
\ -- pstr.f   sws 04-09-03
package PARAM-STRING
variable  (PSTR)
2variable (POINTERS)
variable  (PARAM)  char ? (param) c!
```

Die Variable `(pstr)` zeigt auf den Speicherblock, in dem der ausgefüllte String aufgebaut wird, in den beiden Zellen von `(pointers)` sind während der Abarbeitung die aktuellen Positionen in Vorlage und Ergebnis zu finden, `(param)` schließ-

lich enthält das Symbol, das in der Anfrage als Platzhalter verwendet wird.

Die Hauptarbeit wird vom langen Wort `(copy)` erledigt, in dem Zeichen aus der Vorlage in das Ziel kopiert werden, bis der Platzhalter auftritt oder die Vorlage zu Ende ist.

```
: (COPY)  ( src dest -- src' dest' )
  (param) c@ >r
  begin
    over c@  dup 0<>
    over r@  <>  and
  while
    over c!  1+ swap 1+ swap
  repeat  r> 2drop ;
```

Dieses Arbeitspferd lebt in einem Stall, der es mit dem Notwendigsten versorgt:

```
: (NEXT-BLOCK)  ( -- )
  (pointers) 2@ (copy) (pointers) 2! ;
```

Schließlich muss noch der Ersatztext für den Platzhalter kopiert werden:

```
: (SPLICE)  ( dest text – dest' )
  zcount >r  over r@ cmove  r> + ;
```

Damit sind die internen Definitionen beendet und ich komme zum für den Anwender des Packages gedachten Teil. Zunächst schaffe ich die Möglichkeit, ein beliebiges Zeichen als Platzhalter festzulegen:

```
public

: !PCHAR  ( n -- )  (param) c! ;
: [PSTR  ( src dest n -- )
  drop  dup (pstr) ! (pointers) 2!
  (next-block) ;
: >>  ( prm -- )
  (pointers) 2@
    rot (splice)
    swap 1+ swap (copy)
  (pointers) 2! ;
: PSTR]  ( -- z )
  0 (pointers) 2@ nip c! (pstr) @ ;
end-package
\ pstr.f --
```

`[pstr` bestückt die Variablen mit den übergebenen Werten und kopiert den Textbereich bis zum ersten Platzhalter. `>>` fügt an dieser Stelle den Parameter-Text ein und hängt den Text bis zum nächsten Platzhalter an. Zu guter Letzt terminiert `pstr]` noch den aufgebauten String und gibt dessen Startadresse an das aufrufende Wort zurück.

Bequeme SQL-Konstruktion

Nun kann ich nach dem gleichen Muster einige Worte definieren, die das Arbeiten mit SQL-Anfragen schön bequem machen.

```
\ -- mysql.f

variable (CONN)

: [SQL  ( conn src dest n -- )
  [pstr  (conn) ! ;
: SQL]  ( -- conn sql )
  (conn) @ pstr] ;
: SQL]EXEC  ( -- )
  sql] qry-execute ;
: SQL]OPEN  ( -- res #rows )
  sql] qry-open ;
: SQL]1ROW  ( -- res z[] )
  sql] qry-1row ;
: SQL]1FIELD  ( -- z )
  sql] qry-1field ;

\ mysql.f --
```

Im folgenden Mitschnitt aus einer SwiftForth-Sitzung sind die Benutzereingaben fett gedruckt, die Antworten des Systems erscheinen in normaler Schrift. Es wird gezeigt, wie man mit einer (unnötigerweise parametrisierten) Abfrage die Benutzer herausfindet, die sich vom lokalen Rechner aus am Datenbankserver anmelden können.

```
SwiftForth 2.2.2.9 07May2001

include pstr  ok
include mysql  ok
db-init value conn  ok
1024 allocate drop value buffer  ok

conn z" select user from user where
  host = '?'" buffer 1024  ok
[sql z" localhost" >> sql]open  ok

. 3  ok

dup qry-next @ zcount type  ok
dup qry-next @ zcount type 4dtest ok
dup qry-next @ zcount type root ok

qry-free  ok
```

Der „fehlende“ Benutzername ist in der Tat ein leerer String, der von MySQL für anonyme Anmeldungen verwendet wird.

Wie geht’s weiter?

Wir haben jetzt das Werkzeug vorbereitet, mit dem wir in der zweiten Folge den Datentransfer von Cobol nach MySQL vollziehen werden. Bis dann!

Ach ja, fast vergessen ...

Hausaufgabe: Wo ist der schlimmste Fehler im gezeigten Code?

Protokoll der öffentlichen Mitgliederversammlung der

Forth-Gesellschaft e.V.

vom 18. April 2004

Eine Tagesordnung lag vor (s. Anlage). Sie wurde mit der letzten „Vierte Dimension“ an alle Mitglieder des Vereins versandt.

Ort:	Hotel Schützenhof 23769 Burg (Fehmarn)
Beginn:	08:36 Uhr
Ende:	11:55 Uhr

Tagesordnung:

Top 1: Begrüßung
Top 2: Wahl des Schriftführers
Top 3: Wahl des Versammlungsleiters
Top 4: Ergänzungen zur Tagesordnung
Top 4a: Laudatio auf den diesjährigen Drachenträger
Top 5: Bericht des Direktoriums
Top 6: Entlastung des Direktoriums
Top 7: Wahl des Direktoriums
Top 8: Satzungsänderung
Top 9: Verschiedenes
Top 10: Ende der Versammlung

Top 1: Begrüßung

Das Direktoriumsmitglied Ulrich Hoffmann, der auch Organisator der diesjährigen Tagung und Versammlung ist, begrüßt die Anwesenden.

Top 2: Wahl des Schriftführers

Zum Schriftführer wird per Akklamation Martin Bitter gewählt.

Top 3: Wahl des Versammlungsleiters

Zum Versammlungsleiter wird Heinz Schnitter ebenfalls per Akklamation gewählt. Er stellt fest, dass mit 28 anwesenden von 129 stimmberechtigten Mitgliedern die Beschlussfähigkeit gegeben ist.

Top 4: Ergänzungen zur Tagesordnung

Aus dem Publikum werden folgende Ergänzungen zur Tagesordnung beantragt und nach mehr oder weniger langen Diskussionen angenommen:

Forth an die Universitäten und Fachhochschulen

Stand des Scan-Projektes (Vierte Dimension)

b16 und µCore

Neuauflage von „Starting Forth“

Notebook für das Forthbüro

Knoppix/Forth

Top 4a: Laudatio auf den diesjährigen Drachenträger

Thomas Beierlein hält die Laudatio auf den diesjährigen Drachenträger. Er würdigt die Verdienste des Laureaten um Forth, dessen unermüdlichen Einsatz, die gelungene Organisation der letzten Tagung der Gesellschaft und die Beachtung seines Triceps-Konzeptes auf verschiedenen (Linux-) Veranstaltungen. Es ist Ewald Rieger! Herzlichen Glückwunsch!

Top 5: Bericht des Direktoriums

Rund um die VD (F. Prinz)

Fritz Prinz bittet um zahlreiche Beiträge. „Die einzige Sorge, die mich als Editor bedrückt, ist die um eine ausreichende Anzahl von Artikeln, Leserbriefen u.ä.“. Der Stichtag für die nächste Ausgabe ist Mitte Juni.

Bei dieser Gelegenheit bittet Ulrich Hoffman darum, die Verschriftlichungen der Tagungsbeiträge bis zum 15. Juni einzureichen.

Mitgliederentwicklung und Kassenstand (R. Schöne)

Rolf Schöne berichtet über die Entwicklung der Mitgliederzahlen und legt einen Kassenbericht vor.

Im Jahre 2003 haben 20 Mitglieder die Forth-Gesellschaft e. V. verlassen, davon 10 mit ordentlicher Kündigung. Eingetreten sind in diesem Zeitraum 5 Mitglieder, davon sind 4! (80%) reaktivierte Mitglieder.

Im Jahre 2004 haben 7 Mitglieder die Forth-Gesellschaft e. V. verlassen, davon 2 mit ordentlicher Kündigung. Eingetreten sind in diesem Zeitraum 2 Mitglieder.

Der Mitgliederstand beträgt zum April 2004 129 Mitglieder.

(Der Kassenbericht kann in der Anlage eingesehen werden.)

Der Verein „Forth-Gesellschaft e.V.“ verfügt zum Stichtag (31.12.2003) über ein Vermögen von 24.667,02 €. Die Einnahmen betrugen 29.609,25 €, die Ausgaben beliefen sich auf 27.407,09 €. Das bedeutet, dass der Verein eine Ausgabereserve von einem Rechnungsjahr hat.

Die größten Ausgabenposten waren die Druck- und Versandkosten für die „Vierte Dimension“. An dieser Stelle wurde Friederich Prinz großer Dank für seine unermüdliche, kostenlose Arbeit bei der Edition der „Vierte Dimension“ ausgesprochen.

Für die Jubiläumsfeier der Forth-Gesellschaft e.V. wurde eine Musikgruppe engagiert, es wurden Gedenktassen ge-

brannt und Anstecknadeln geordert. Dies schlug in 2004 mit 2.325,68 € zu Buche.

Die Abrechnung ist durch die Kassenprüferin E. Rohrmayer überprüft worden.

Die Versammlung entlastet Herrn Schöne mit 27 Stimmen von 28. Es gab eine Enthaltung.

Internetpräsenz (U. Hoffmann)

Ulrich Hoffmann berichtet, dass er nicht in der Lage ist, die WebSite der Gesellschaft hinreichend redaktionell zu pflegen und zu betreuen. Er benötigt dringend Mithilfe. Sein Vorschlag: Ähnlich wie unter www.honkbude.de sollten WebLogs eingerichtet werden, die von jeweils ausgewählten Mitgliedern betreut werden. Er (Hoffmann) werde sich um das 'Technische' kümmern, Inhalte und Gestaltung müssten von anderen bearbeitet werden.

Dazu bemerkt u.a. Klaus Schleisiek, dass man mit dem Forth-WiKi-Server schlechte Erfahrungen durch Surf-Vandalen gemacht habe. Dies sei bei der angestrebten Lösung nicht möglich, entgegnet U. Hoffmann, da dort nur autorisierte User redaktionellen Zugriff auf die Inhalte haben.

Aus Zeitgründen wird die Diskussion hier abgebrochen und auf (vielleicht) später (Verschiedenes) verschoben.

Auslandsarbeit (F. Behringer)

Fred Behringer berichtet über die Zusammenarbeit mit niederländischen Forthfreunden. Unter anderem ist seine Übersetzung eines Forth-Kurses von Albert Nijhof fertig: „Die Programmiersprache Forth“. Der Kurs wird im mv-Verlag in Münster im Verfahren „books on demand“ erscheinen.

Es besteht eine regelmäßige Zusammenarbeit mit den Niederländern und den Briten. Beide FGs veröffentlichen regelmäßig eine Anzeige der Forth-Gesellschaft e.V.

Kontakt zu den USA (hier Silicon Valley) wird über Henry Vinerts gehalten, der regelmäßig von den Treffen der Forthbegeisterten um Herrn Dr. C.H. Ting berichtet.

Die Forthgesellschaft der USA ist ohne weitere Angabe von Gründen eingeschlafen. Hier wird kurz diskutiert. Ein Beitrag besagt, dass es an den Finanzen läge. Die Gesellschaft (USA) habe ein bezahltes Direktorium gehabt, als erst die Mitglieder und dann das Geld ausblieb, brach dort die Organisation zusammen. Jemand anders weiß von einem amtierenden „Übergangspräsidenten“ (John Hall) zu berichten. Aktiv ist weiterhin die Gruppe um Dr. C.H. Ting.

Vorhaben des Direktoriums (B. Paysan)

Bernd Paysan berichtet von den Aktivitäten des letzten Jahres:

Es ist ein Entwicklungsboard für den µCore vorhanden. Es beruht auf einer Xilinxbasis, eine zweite Version für einen Altera Chip ist angedacht.

Bernd Paysan hat sich bemüht, für eine Neuauflage und Aktualisierung (ANS-Forth) die Rechte von Leo Brodies „Starting Forth“ zu bekommen. Dazu traf er sich unter anderem mit Elizabeth Rather, die als Inhaberin von Forth Inc. einen Teil der Rechte hält. Sie zeigte sich angetan von dem Projekt und will es tatkräftig unterstützen. Die Lage ist ein wenig verworren, da ein Teil der Rechte bei Leo Brodie liegt. Frau Rather will 'Chief Editor' werden und Leo Brodie einspannen. Im Gespräch mit Bernd wurden eine Reihe von Änderungswünschen gesammelt, um die Neuauflage zu modernisieren.

Kurze Diskussion: Fr. Rather bestimmt?

Ulrich Hoffmann hatte Kontakte zu Leo Brodie geknüpft (Rechte an den Comics auf der WebSite der Forth-Gesellschaft e.V.). Er möchte diese für das Übersetzungsprojekt intensivieren.

Irgendjemand erwähnt, dass die Rechte für eine evtl. Übersetzung weder bei Rather/Forth Inc. noch bei Brodie lägen, sondern beim damaligen Übersetzer. Bernd entgegnet, das träfe nur zu, wenn dieser das Werk in den letzten Jahren noch vertrieben hätte. Dies sei nicht geschehen, also seien diese Rechte wieder frei.

Rolf Kretzschmar bietet sich an, evtl. neue Comics bzw. das Titelbild zu entwerfen.

Top 6: Entlastung des Direktoriums

Das Direktorium wird mit 26 Stimmen entlastet. Es gibt 2 Enthaltungen.

Top 7: Wahl des Direktoriums

Es wird vorgeschlagen, die bisherigen Direktoriumsmitglieder in ihrem Amt zu bestätigen. Dazu werden keine Gegenvorschläge geäußert. In öffentlicher Wahl wird in einem einzigen Wahlgang das gesamte Direktorium wiedergewählt. Die Wahl ist einstimmig!

Direktoren sind:

Fred Behringer

Ulrich Hoffmann

Bernd Paysan

Top 8: Satzungsänderung

Aufgrund einer Änderung im Vereinsrecht (Gemeinnützigkeit) muss ein Passus über die Verwendung der Vereinsmittel in die Satzung aufgenommen werden. Der Vorschlag zur Änderung des Paragrafen 2 der Satzung ist fünf Wochen vor der Versammlung zusammen mit der Einladung in Form einer Beilage zur „Vierte Dimension“ an alle Mitglieder des Vereins verschickt worden. Es hat keine schriftlichen Reaktionen darauf gegeben.

> „Mittel des Vereins dürfen nur für die satzungsgemäßen Zwecke verwendet werden. Die Mitglieder erhalten kei-

ne Zuwendungen aus Mitteln des Vereins.

Es darf keine Person durch Ausgaben, die dem Zweck der Körperschaft fremd sind, oder durch unverhältnismäßig hohe Vergütungen begünstigt werden.

Die Förderung des gewerbsmäßigen Vertriebs von Forth-bezogenen Produkten ist nicht Aufgabe des Vereins.“

Nach der Verlesung der geplanten Veränderung ergibt sich eine anhaltende Diskussion über den dritten Abschnitt. Hier geht es um die Klärung des Begriffs 'gewerbsmäßig'. Kann, darf die Forth-Gesellschaft e.V. bei der Entwicklung des µCore und des b16 (finanzielle) Hilfe leisten?

Die Satzungsänderung wird mit 25 Stimmen angenommen. Es gibt eine Gegenstimme und zwei Enthaltungen.

Top 9: Verschiedenes

Die unter Top 4 eingebrachten Ergänzungen wurden nach kurzer Diskussion zum Teil gestrichen (meist weil sie unter anderen Tops subsumiert wurden) oder ergänzt. Folgende Punkte wurden behandelt:

LinuxTag

Carsten Strotmann macht auf den nächsten LinuxTag vom 23. bis 26.06.2004 in Karlsruhe aufmerksam. Bei den vergangenen Veranstaltungen war die Reaktion auf den Stand der Forth-Gesellschaft e.V. positiv. Wegen der Länge der Veranstaltung werden noch Helfer benötigt. Ebenso sollten möglichst viele Projekte etc. eingeschickt werden.

Carsten Strotmann zeigt einen Flyer, der mit geringen Änderungen neu gedruckt und auf dem LinuxTag verteilt werden kann.

Jemand aus dem Publikum erinnert an den LinuxTag in Chemnitz im März.

Knoppix

Carsten Strotmann berichtet über eine Knoppix-CD mit Forthsoftware, die bei Anlässen wie dem LinuxTag gut verteilt werden kann. Die dazugehörigen ISO-Dateien stehen bei www.strotmann.de und (hoffentlich bald) bei www.knopper.net zum Download bereit. In diesem Zusammenhang wird die Diskussion um den Flyer aufgegriffen. Carsten wird gebeten, ihn für das Forth-Büro zur Verfügung zu stellen. Wer den Flyer bei besonderen Anlässen verteilen kann, soll ihn dann über das Forthbüro anfordern können. Ein weiterer Vorschlag besagt, den Flyer als PDF-Datei über die WebSite der Forth-Gesellschaft e.V. zur Verfügung zu stellen.

Notebook

Heftig wird über die Anschaffung eines Notebooks für das Forthbüro diskutiert. Rolf Schöne hätte gerne eines. Es ist auch schon angeschafft. Einige Mitglieder erachten ein Notebook für das Büro als nicht unbedingt notwendig. Eine Desktoplösung täte es auch. Bernd Paysan schlägt vor, das Notebook nicht nur für das Büro, sondern auch für Veranstaltungen (z.B. LinuxTag) zu verwenden, dort sei ein 'offizielles' Notebook oft hilfreich.

E. Rohrmayer stellt eine Spende von 500 € in Aussicht.

Die Abstimmung ergibt 17 Stimmen für eine Anschaffung des Notebooks, 3 Stimmen dagegen und 8 Enthaltungen.

Laufende Projekte

Forth an die Universitäten

Rolf Schöne bittet um Vorschläge und Erlaubnis, ausgewählte Funktionsträger an Universitäten und Hochschulen anzuschreiben und Werbung für Forth und die Forth-Gesellschaft e.V. zu betreiben.

Klaus Schleisiek regt an, gezielt Diplomanden zu suchen, die mit und an µCore und b16 arbeiten.

Johannes Reilhofer stimmt dem zu: Dies mache Forth attraktiv, und Attraktivität brauche Forth unbedingt!

Karsten Roederer hält Win32For mit seinem 'endlosen' wieder verwertbaren Editorfenster für einen solchen attraktiven Beitrag.

Fred Behringer schlägt vor, Diplomanden (s.o.) einige µCore-Boards durch die Forth-Gesellschaft e.V. kostenfrei zur Verfügung zu stellen.

Egmont Woitzel rät, zu diesem Zweck gezielt Menschen aus den Lehrkörpern anzusprechen.

Bernd Paysan hält dies (Forthboard und Diplomarbeit) ebenfalls für eine gute Idee. Er kennt Leute, die schon darüber arbeiten.

Ulrich Hoffmann will Lehrende im Bereich „computer science“ ansprechen und sie für µCore gewinnen.

Egmont Woitzel will gezielt E-Techniker an der Uni Rostock ansprechen und ihnen im Namen der Forth-Gesellschaft e.V. das Board für Forschungszwecke kostenlos zur Verfügung stellen.

Bernd Paysan stimmt dem zu und stellt einen entsprechenden Antrag, dessen Formulierung daraufhin diskutiert wird. Der Antrag wird zurückgestellt. Es folgt:

Stand µCore und b16

Klaus Schleisiek berichtet über den Stand des µCore Projektes. In seiner Firma SEND wurden 13 Boards entworfen und angefertigt. Davon mussten drei in Handarbeit angepasst werden (verdrehter IC). Er kann Hilfe bei der Anpassung des Layouts für den Altera-Chip gebrauchen.

Die Entwicklungskosten hat die Firma SEND vorgestreckt. Klaus Schleisiek beklagt sich darüber, dass seiner Bitte, eine Anschubfinanzierung zu leisten, nicht entsprochen wurde. Ein Direktoriumsmitglied erwidert, das sei einem gemeinnützigen Verein nicht möglich. Wohl könne man fertige Produkte kaufen und an potentielle Multiplikatoren weitergeben.

Klaus Schleisiek teilt mit, dass die Anfertigung weiterer Boards, darunter auch Alteraversionen, mit ca. 8000 € zu Buche schlage.

Bisher gibt es drei Interessenten, die ein Board kaufen wollen.

Egmont Woitzel stellt folgenden Antrag:

„Das Direktorium wird ermächtigt, bis zu 3500 € für die Entwicklung und Fertigung von µCore-Boards auszugeben."

Dieser Antrag wird mit 24 Stimmen angenommen. Es gibt 4 Enthaltungen.

(Der Versammlungsraum soll geräumt werden. Es entsteht einige Unruhe.)

Ein Ergänzungsantrag wird gestellt und angenommen:

„Die Entwicklung von µCore-Boards in zwei Versionen (Altera, Xilinx) soll durch die Forth-Gesellschaft e.V. gefördert werden. Das Direktorium hat volle Entscheidungsfreiheit über die Finanzen des Vereins."

Dieser Antrag wird mit 27 Stimmen angenommen, es gibt 1 Enthaltung.

(11:30 Uhr. Der Versammlungsraum wird geräumt. Die Versammlung tagt im Restaurantbereich weiter.)

Termin für die **nächste Tagung/Mitgliederversammlung**

Einige Mitglieder verlassen die Versammlung, daher wird jetzt der Ort für die nächste Tagung festgelegt.

Ort: Großraum Dresden

Organisator: Thomas Beierlein

(weiter: Laufende Projekte)

VD Scan-Projekt

Ulrich Hoffmann berichtet über das Scan-Projekt: Der größte Teil der Arbeit sei geschafft. Es stünden noch 5 bis 6 Ausgaben der „Vierte Dimension" aus. Wegen anderer großer Belastung sei mit dem Erscheinen der Scan-CD erst im nächsten Jahr zu rechnen.

EuroForth-Tagung in Dagstuhl

Die diesjährige EuroForth-Tagung wird (wieder) in Dagstuhl stattfinden. Dagstuhl ist ein sehr guter Ort für Tagungen, da sowohl die Organisation, wie auch die Ausrüstung durch ein erfahrenes Haus durchgeführt wird. Termin: voraussichtlich 19. - 21.11.2004. Es wird überlegt, dass die Forth-Gesellschaft e.V. für diese Tagung Sicherungsleistungen übernimmt. (Kein Antrag. Keine Abstimmung, da freie Entscheidung des Direktoriums.)

Übersetzung Starting Forth (Fortsetzung von Top 5)

Es müssen noch einige rechtliche Dinge geklärt werden.

Organisation und Koordination übernimmt Bernd Paysan. Leider gibt es keine elektronische Vorlage. Es werden noch Helfer gesucht, die entweder kapitelweise „Starting Forth" abtippen, bzw. einscannen und „OCRen". Dazu wird ein Exemplar „Starting Forth" zum Zerschneiden und Einscannen benötigt; Michael Kalus bietet sich an.

Fünf Kapitel werden nahe am Original bleiben können. Alle übrigen sind entweder obsolet oder tiefgehend bearbeitungsbedürftig. Es werden Autoren für neue Kapitel gesucht z.B. zum Thema „floating point" und „internals".

Bernd Paysan wird bei http://sourceforge.net eine Abteilung „Starting Forth" einrichten. Damit steht dann die notwendige Technik zum verteilten Editieren (CVS u.Ä.) zur Verfügung (vgl.http://sourceforge.net/users/paysan/). Geplantes Format ist LaTex, über einen WiKi-Server wird LaTex-ungewohnten Menschen eine einfache Eingabe möglich gemacht.

Über die Einrichtungen bei http://sourceforge.net und des WiKi-Servers wird bei comp.lang.forth, comp.lang.forth.de und auf der WebSite der Gesellschaft berichtet werden.

Top 10: Ende der Versammlung

Heinz Schnitter stellt das Ende der Versammlung fest. Er dankt dem Organisator Ulrich Hoffmann und vielen anderen.

Mehrhoog, September 2004 München, September 2004

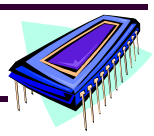
Für das Protokoll: Für das Direktorium:

gez.: Martin Bitter gez.: Fred Behringer

Liebe Mitglieder,

wie Sie der Niederschrift zu unserer Versammlung 2004 entnehmen können, ist in unserem Verein „ganz schön was los". Kommen Sie zur nächsten Mitgliederversammlung. Gestalten Sie unseren Verein mit.

fep



FORTH auf FLASHControllern

Rafael Deliano

Inzwischen kann ein IC im DIL40 Gehäuse das, wofür man früher einen Einplatinencomputer brauchte: ein interaktives FORTH, das Sourcecode direkt ins FLASH compiliert.

Zum Test wird nur die übliche Minimalbeschaltung benötigt (Bild 1). Der 68HC08 ist eine simple 8 Bit Akkumulatormaschine mit einem Index-Register X. Er ist eine aufgebohrte Version des 68HC05 und hat Ähnlichkeit mit 6502 und 68HC11.

Die Variante GP32 enthält als I/O die üblichen Ports, 8 Bit A/D-Wandler und Timer, die auch PWM können. Eine Besonderheit ist die Takterzeugung mittels PLL aus einem 32kHz Quarz. Man kann damit die Busfrequenz in einem weiten Bereich frei wählen und damit den Stromverbrauch optimieren. Es sind aber hier nur Taktfrequenzen möglich, die sinnvolle Baudraten ergeben.

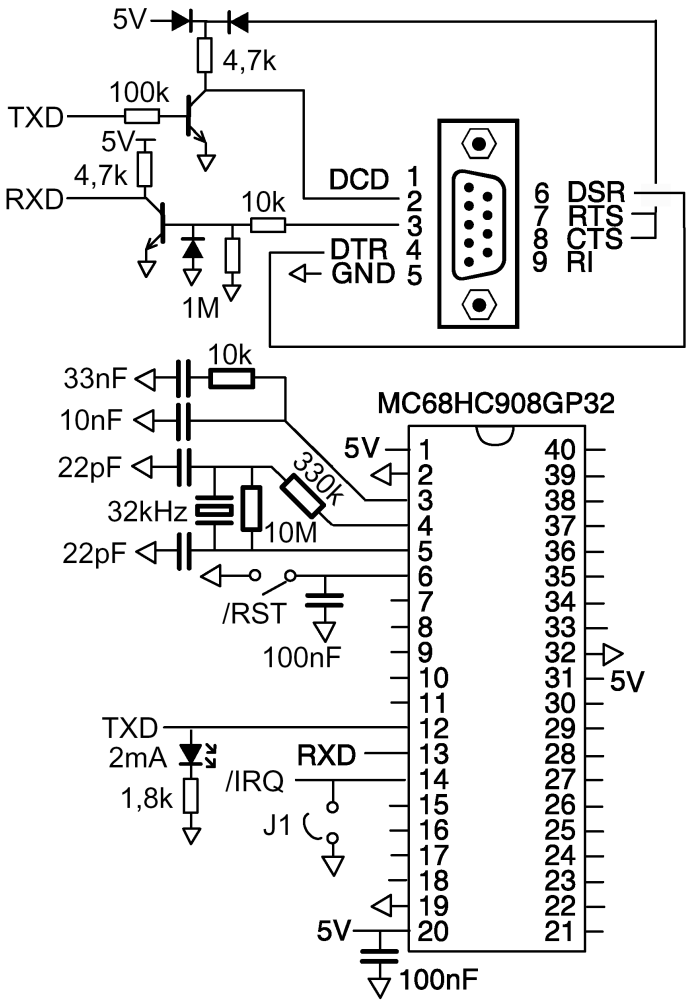


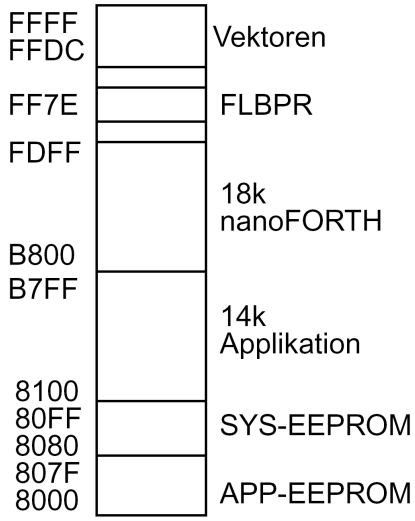
Bild1: GP32 mit einfachem V24-Interface

Die Programmierung erfolgt von einem PC aus, über V24 mit üblichem Terminalprogramm. Die Einstellung ist 9600 Baud 8N1, Handshake XON/XOFF, es sollte Source als ASCII in Klartext von Diskette laden können.

Speicher

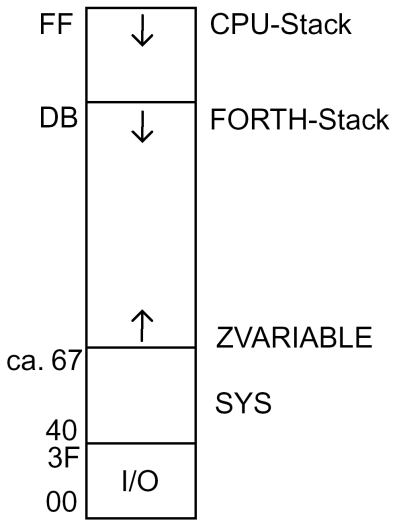
FLASH-Speicher ist mit 32k Byte reichlich vorhanden (Bild 2). Davon belegt das FORTH-System 18k. Für die Applikationen stehen ca. 14k zur Verfügung. Sowohl für FORTH, als auch für die Applikation wird in zwei 128 Byte Segmenten im FLASH ein EEPROM simuliert. Das ist für Konstanten, die man ab und zu ändern will, brauchbar. Da das Schreiben von FLASH aber umständlich ist, sollte man in Applikationen besser externes serielles EEPROM verwenden, wenn man im Betrieb Daten ändern muß.

Bild 2: FLASH-Speicher



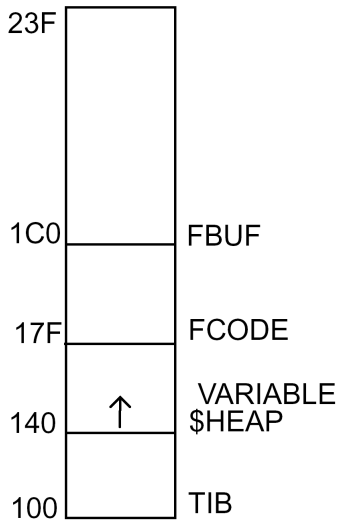
Während man FLASH genug hat, ist das RAM mit 512 Byte recht knapp. In der ZeroPage (Bild 3) befinden sich etwa 40d Byte System-Variab-len. Von dort wachsen auch Variablen, die man in der Applikation definiert, aufwärts. Ihnen entgegen wächst der FORTH-Stack abwärts. Er wird durch das X-Register simuliert. Darüber befindet sich ein Speichersegment fester Läge für den Stack der CPU.

Bild 3: ZeroPage RAM & I/O



Oberhalb der ZeroPage (Bild 4) wirds im RAM besonders eng. Der Terminal Input Buffer TIB ist 64 Byte lang. Das reicht im Interpretermodus von FORTH nur deshalb für die 80 Zeichen/Zeile eines Terminals, weil die Stackdarstellung von nanoFORTH am Zeilenanfang 17 Bytes belegt.

Bild 4: RAM



Das Segement $HEAP speichert während der Compilierung die Labels des Assemblers. Der Bereich ist doppelt belegt: später kann man hier Variablen für die Applikation anlegen.

Viel RAM frißt das Schreiben von FLASH. FBUF ist eine 128 Byte Seite, die als Zwischenspeicher für Daten benötigt wird, wenn man das EEPROM schreibt. FCODE sind 65 Byte, wo Programme gespeichert werden, die FLASH schreiben oder löschen. Soweit man das während einer Applikation nicht tut, kann man beide Bereiche als freies RAM verwenden.

FLASH

Motorolas Variante ist in Seiten von 128 Byte löschbar. Will man also z.B. im EEPROM-Bereich nur ein Byte ändern, muß man das komplette Segment ins RAM nach FBUF kopieren, FLASH löschen, die Änderung im RAM vornehmen, die Seite wieder ins FLASH programmieren. Während des Schreibens und Löschens von FLASH muß das steuernde Programm sich im RAM, also in FCODE befinden.

Erfreuliche Eigenschaften von Motorolas FLASH sind: Es läßt sich byteweise schreiben. Es ist schnell. Ohne Overhead dauert das Löschen einer Seite etwa 4 msec, das Schreiben eines Bytes 40 usec. Und es ist für 10k Schreib/Lesezyklen spezifiziert.

Source

Um komplette Textfiles direkt ins FLASH zu compilieren, sind damit die Rahmenbedingungen abgesteckt.

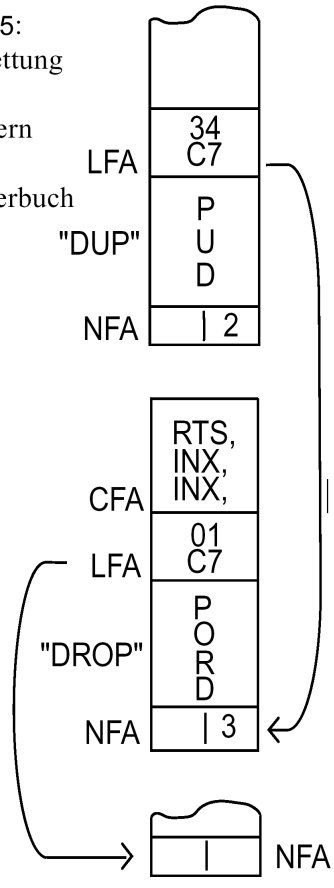
In nanoFORTH werden solche Files durch die Klammern <| ... |> gekennzeichnet, so daß FORTH vom Terminalmodus in den Filemodus umschalten kann. Um das wenige RAM in TIB optimal zu nutzen, eliminiert der Parser die Kommentare. Alle Strings werden also nur durch ein einzelnes Leerzeichen getrennt in TIB gespeichert. Strings sind hier Namen und Zahlen. Sie sind in nanoFORTH auf 16 Byte Länge beschränkt. Sobald ein kompletter String empfangen wurde, schickt der Parser ans Terminal-Programm ein XOFF, um die Übertragung abzuwürgen und startet den Compiler, um den String zu verarbeiten. Typisch führt das zum Schreiben von einigen Bytes FLASH. Deshalb kann die UART auch nicht als Interruptprogramm parallel im Hintergrund zum Compiler laufen. Ein Interrupt kann während des Schreibens des FLASH ja nicht bearbeitet werden.

Wenn der Compiler den String geschluckt hat, wirft er den Parser wieder an, der ein XON ans Terminal schickt, um mehr Daten anzufordern. Dieses häppchenweise einige Buchstaben einlesen und einige Bytes programmieren bremst natürlich die Compilierung, so daß man sich um Optimierung bemühen muß.

Liste

In der Grundstruktur unterscheidet sich erstmal nichts vom alten fig-FORTH. Damit das FORTH interaktiv verwendbar ist, sind die Befehle mit Header, die durch eine Liste verbunden sind, abgespeichert (Bild 5). Der Header hat an der „Name Field Adress“ ein Byte, das die Länge des Namens, sowie einige Angaben für den Compiler enthält. Ihm folgt der bis zu 16 Bytes lange Name. Dann kommt an der „Link Field Adress“ LFA die NFA-Adresse des letzten Befehls. Ab der „Code Field Adress“ CFA liegt der Befehlsteil, da hier JSR-threaded Code verwendet wird, also Assembler. Auf den Kopf der Liste zeigt der Inhalt der Variable LATEST. Im letzten Befehl befindet sich in der LFA die ungültige Adresse FFFF als Stopper (Bild 6).

Bild 5: Verkettung von Wörtern im Wörterbuch



Der Compiler verbraucht viel Zeit damit, anhand eines Namens im TIB die Liste zu durchsuchen, um den Befehl mit diesem Namen zu finden. Die Suchfunktion ist zwar in Assembler codiert, aber trotzdem nicht beliebig schnell. Besonders wenn das System in natürlicher Reihenfolge compiliert wurde (Bild 7). Diese ergibt sich dadurch, daß ein Befehl, bevor er verwendet werden kann, bereits definiert sein muß. Die Primitives sind die grundlegendsten und häufigsten Befehle, aber genau deshalb in der Liste ganz unten.

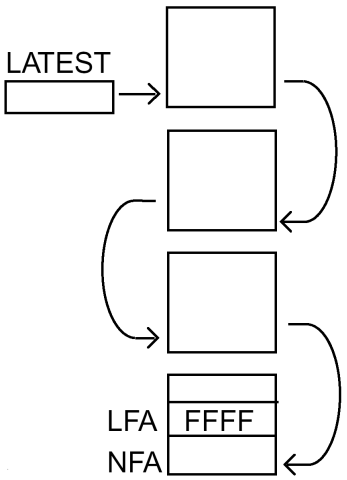
Hashing würde die Suche zwar beschleunigen, hat aber wegen des ungünstigen Speicherverbrauchs andere Nachteile.

Schnellere Liste

Stattdessen kann man versuchen, die konventionelle verkettete Liste zumindest für das FORTH-System zu optimieren. Dieser Teil wird von einem Crosscompiler erzeugt und ist damit leicht manipulierbar.

Erste Maßnahme ist headerless Code, also Köpfe von Befehlen

Bild 6: Anfang und Ende des Wörterbuchs



zu entfernen, die der Anwender im Zielsystem später nie interaktiv benutzen wird. Ein Kopf, der nicht mehr da ist, bremst die Suche nicht.

Zweitens ist das Ausblenden von Teillisten nützlich. Das macht hier besonders für den Assembler Sinn. Er wird also nur noch innerhalb von : CODE ... CODE; zugeschaltet (Bild 8).

Drittens kann man Teillisten intern optimieren, d.h., häufig benutzte Befehle am oberen Ende der Liste anordnen, selten benutze am unteren Ende. Das ist besonders bei den Primitives einfach, da es sich hier fast durchwegs um Befehle handelt, die nicht aufeinander aufbauen. Man kann sie damit im Sourcefile in beliebiger Reihenfolge anordnen.

Viertens kann man die natürliche Reihenfolge der Teillisten nachträglich verändern. Insbesondere bei einem Crosscompiler ist das abschliessende Patchen einiger Sprungziele in den LFAs leicht möglich. Während die Anordnung im Speicher wie in Bild 7 bleibt, verhält sich die logische Suchliste wie in Bild 9.

Die Optimierung der Befehlsliste ist nur ein Beispiel für eine Vielzahl von Möglichkeiten, mit denen man derart kleine Systeme verbessern kann.

Rafael Deliano

Betreff: **embedded 9**
An: Friederich.Prinz@t-online.de

Inhalt:
* Flashprogrammierung 68HC908
* OCR-Eingabegerät
* Einfacher Logarithmus
* CompactFlash: Hardware
* Fletcher Prüfsumme
* Hashing für RFIDs
* Lineare Interpolation in Tabellen

Zum Download bereit auf:
http://www.embeddedFORTH.de

MfG JRD

Betreff: **swap pin**
Datum: Sun, 26 Sep 2004 19:39:28 +0200
Von: "**Chuck Moore**" <chipchuck@colorforth.com>

I received your Golden Swap Pin. Thank you very much for the honor it represents. Let's look forward to 20 more years of Forth.

My congratulations to whoever designed the pin. It's a very clever and appropriate symbol.
—

Im Namen des Festausschusses gebe ich an dieser Stelle das Lob von Charles Moore an Bernd Paysan und Martin Bitter weiter, die uns die schönen SWAPs gemacht haben.

fep

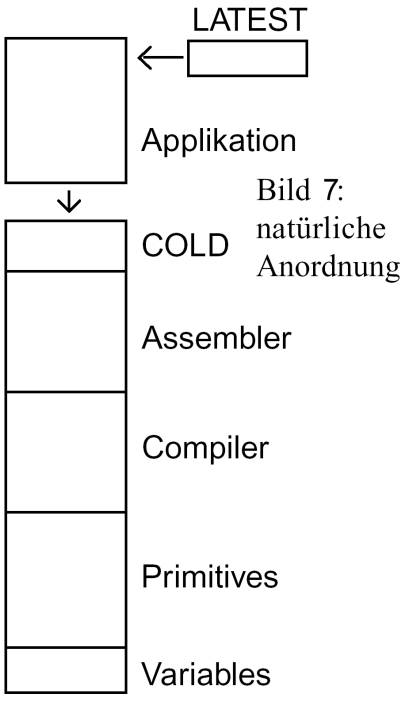


Bild 7: natürliche Anordnung

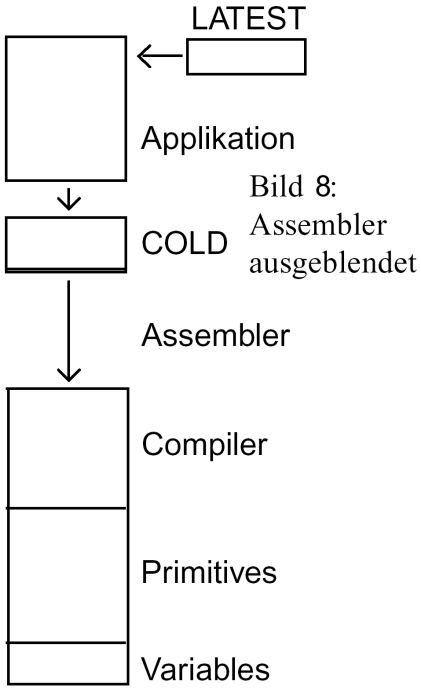


Bild 8: Assembler ausgeblendet

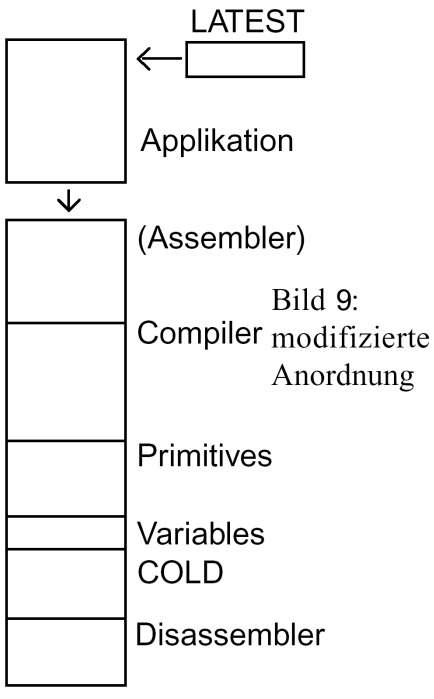


Bild 9: modifizierte Anordnung

Mitgepuzzelt

Forth löst c't-Puzzle

Ewald Rieger

eMail: ewald.rieger@t-online.de
Littersheimer Weg 10
67240 Bobenheim-Roxheim

28.Mai 2004

Abbildung 1: c‘t-Quader

1. Einleitung

Am Samstag, den 26.03.2003, lag das neue c't-Magazin 7/2003 in meinem Briefkasten. Neugierig überflog ich sogleich das Magazin und fand auf Seite 230 den sehr interessanten Artikel "Zusammengewürfelt". Dieser Artikel beschreibt in anschaulicher und ausführlicher Weise am Beispiel eines C++-Programms, wie man Würfel-Puzzles zusammenbaut und die Anzahl seiner Bauvarianten berechnet. Am Ende des Artikels schrieb der Autor Harald Bögeholz einen Programmierwettbewerb aus [2, 4]. Die Aufgabe bestand darin, einen Quader der Größe 3*4*5 gemessen in Einheitswürfeln aus seinen 12 Einzelteilen zusammenzubauen. Jeder Teilnehmer durfte mit seinem Lieblings-Compiler programmieren. Das Programm sollte unter Windows XP oder Red Hat Linux auf einem PC mit Pentium 4 und 512 MB RAM lauffähig sein. Gesucht wurde das schnellste Programm, das die Anzahl aller Zusammenbauvarianten ohne Symmetrien berechnen kann. Als Preis sollte der Gewinner das langsamste Puzzle aus hölzernen Bauteilen erhalten. Als Zusatzaufgabe wurde die graphische Darstellung des Quaders, insbesondere die Lösung mit der richtigen Darstellung des c't-Logos gesucht. Angespornt von meiner Programmierung des Pflock-Solitaire-Spiels war ich entschlossen an diesem Wettbewerb teilzunehmen. Die gerade laufende Vorbereitung zur Forth-Tagung 2003 in Lambrecht zwang mich allerdings, den Start auf die Tage nach der Tagung aufzuschieben. So verblieben gerade mal zwei Wochen bis zum Einsendeschluss am 30.04.2003.

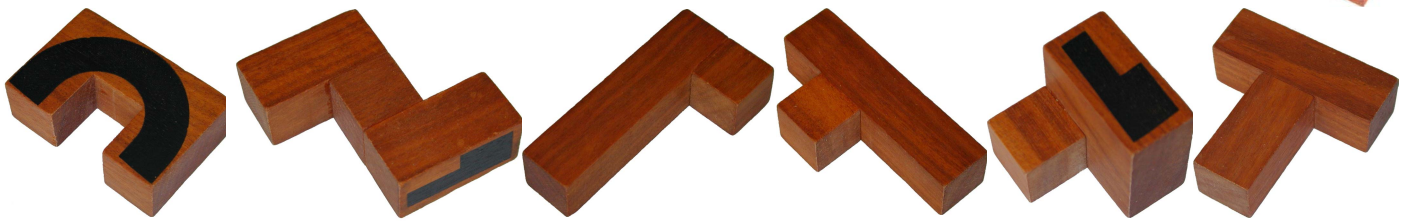
2. Lösungsstrategie

Bereits in "Zusammengewürfelt" schlug Harald Bögeholz vor, den Quader in einem Bitvektor darzustellen. Jedes Bit repräsentiert einen Einheitswürfel des Quaders. Im ersten Schritt werden nun alle passenden Teilevarianten, die man durch Drehen und Wenden innerhalb des Quaders erhalten kann, als Bitvektoren erzeugt und im Speicher abgelegt. Erst jetzt beginnt der eigentliche Zusammenbau des Quaders. Nun belegen wir den Quader mit einer Variante von Teil 1 vor. Dann entnehmen wir eine Variante des nächsten Bauteils und prüfen auf Überlappung mit den bereits verbauten Teilen durch eine logische-Und-Operation der beiden Bitvektoren. Überlappen die Teile nicht, baut ein logisches Oder das Teil in den Quader ein. Zur Lösung der Aufgabe müssen nun alle Teilevarianten in einem typischen Back-tracking-Algorithmus verglichen werden. Die Teilevielfalt lässt aber eine extrem lange Rechenzeit erwarten, so dass wir um eine Optimierung nicht herum kommen. Entgegen dem Vorschlag, der langen Rechenzeit durch systematisches Auffüllen des Quader entgegen zu wirken [4], entschied ich mich für zufälliges Einfügen der Teile und Entfernen der überlappenden Teile aus der Liste der noch zu verbauenden Teile nach dem Einbau eines jeden Teils. Zusätzlich müssen die übrig gebliebenen Teile gewährleisten, daß kein Loch im Quader entsteht. Dieser Lochtest wird durch logische-Oder-Operationen mit dem gerade aktuellen Baustadium des Quaders und den noch zur Verfügung stehenden Teilen erreicht. Zur Beseitigung der Symmetrien entschloss ich mich, von dem asymmetrischen Teil 12 mit dem Aufdruck "t" die Teilevarianten entsprechend zu reduzieren. Dieser einfache Lösungsansatz erschien für die kurze Zeitspanne bis zum Einsendeschluss erfolgversprechend zu sein.

3. Die Teile

Zur Definition der Teile legen wir die Koordinatendifferenz der Einheitswürfel relativ zum Vorgängerwürfel eines Teils auf den Datenstack und speichern die Teiledefinitionen im Dictionary ab. Bei der späteren Berechnung erhalten wir ausgehend von der momentanen Position durch Vektoraddition die jeweils nächste Position eines jeden Einheitswürfels zum Bauteil. Für einen indizierten Zugriff und wählbare Reihenfolge adressieren wir die Teile über das Array *Teile*.

```
: v, ( x y z -- )
    rot , swap , , ;
```



(a) T1 (b) T2 (c) T3 (d) T4 (e) T5 (f) T6

(g) T7 (h) T8 (i) T9 (j) T10 (k) T11 (l) T12

Abbildung 2: Einzelteile des c‘t-Quaders

```
: teil: ( x1 y1 z1 .. xn yn zn n -- )
    Create dup , 0 ?DO v, LOOP ;

 1  0  0   0  1  0   0  1  0
-1  0  0   0  0  0            5 teil: t1
 1  0  0   0  1  0   0  1  0
 1  0  0   0  0  0            5 teil: t2
 0  1  0   0  1  0   0  1  0
 1  0  0   0  0  0            5 teil: t3
-1 -1  0   0  1  0   0  1  0
 0  1  0   0  0  0            5 teil: t4
 0 -1  0   0  0  1  -1  1  0
 1  0  0   0  0  0            5 teil: t5
-2  0  0   1  0  0   0  1  0
 0  1  0   0  0  0            5 teil: t6
-1 -1  0   0  1  0   0  1  0
 1  0  0   0  0  0            5 teil: t7
 0  1  0   1  0  0   0  1  0
 1  0  0   0  0  0            5 teil: t8
 0 -1  0   1  0 -1   0  0  1
 0  0  0                      4 teil: t9
 1 -1  0   0  1  0   0  1  0
 1  0  0   0  0  0            5 teil: t10
 1  1  0  -2  0  0   1  0  0
 0  1  0   0  0  0            5 teil: t11
 1 -1  0   0  1  0   0  1  0
-1  1  0   1  0  0   0  0  0  6 teil: t12

Table: teile t11 t1 t10 t12 t2 t6 t7 t5 t9 t3 t8 t4 ]
DOES> swap cells + perform ;
```

4. Drehen und Wenden

Ein asymmetrisches Teil können wir auf seine sechs Seiten legen und in jeder Lage nochmals in 4 verschiedene Richtungen drehen. Daraus ergeben sich insgesamt 24 verschiedene Lagen für jedes Teil. Durch Vertauschen und Negieren der x-, y-, und z-Werte berechnen wir - auch ohne Matrixmultiplikation - die unterschiedlichen Bauteillagen. Das Forthwort *Rotation1* fasst die 24 möglichen Lageberechnungen in einem Array zusammen. Während *Rotation2* nur die Lagen ohne Symmetrien für das Bauteil t12 erzeugt.

```
: r1 ;
: r2 negate swap ;
: r3 negate swap negate swap ;
: r4 swap negate ;
: r5 -rot ;
: r6 -rot negate swap ;
: r7 -rot negate swap negate swap ;
: r8 -rot swap negate ;
: r9 rot ;
: r10 rot negate swap ;
: r11 rot negate swap negate swap ;
: r12 rot swap negate ;
: r13 rot negate rot negate rot ;
: r14 rot negate rot negate rot negate swap ;
: r15 rot negate -rot negate ;
: r16 rot negate -rot swap ;
: r17 negate -rot swap negate swap ;
: r18 negate -rot negate swap negate ;
: r19 negate -rot negate swap swap ;
: r20 negate -rot swap ;
: r21 swap negate -rot ;
: r22 swap negate -rot negate swap ;
: r23 swap negate -rot negate swap negate swap ;
: r24 swap negate -rot swap negate ;
Table: rotation1 r1 r2 r3 r4 r5 r6 r7 r8 r9 r10 r11 r12
r13 r14 r15 r16 r17 r18 r19 r20 r21 r22 r23 r24 ]
DOES> swap cells + perform ;
Table: rotation2 r1 r18 r21 r24 ]
DOES> swap cells + perform ;
```

5. Teile erzeugen

Bevor wir die Teile berechnen, legen wir die Größe des Quaders und die Länge des Bitvektors als Konstanten fest.

```
3 Constant #x
4 Constant #y
5 Constant #z

#x #y * #z * Constant #block
```

Die Variable w enthält den Bitvektor während wir ein Bauteil berechnen. Ragt das Teil aus dem Quader, wird dies durch ein true-Flag in der Variable Fehler angezeigt. Die Variable teil adressiert das gerade zu bearbeitende Bauteil und Richtung legt die momentane Lage des Teils fest.

```
2Variable w
Variable fehler
Variable teil           \ Adresse von Teil
Variable richtung

: w> ( x y z -- nr )
  #y * + #x * +
  ( dup 0 #block within 0= abort" fehler" ) ;
: chk ( x y z -- x y z false / x y z true )
  vdup 0 #z within
  >r 0 #y within
  >r 0 #x within r> and r> and ;
: +element ( adr x y z -- )
  chk IF w> +bit ELSE fehler on 2drop 2drop THEN ;
```

Das Wort *+element* erwartet die Adresse eines Bitvektors auf dem Datenstack, gefolgt von den Koordinaten des einzufügenden Würfels. Zuerst prüfen wir, ob die Koordinaten innerhalb des Quaders liegen, berechnen im positiven Falle die Bitnummer und setzen dieses Bit im Quader. Liegt ein Fehler vor, setzen wir das Fehlerflag und entfernen die Argumente vom Datenstack.

```
: fehler> ( -- flg )
  fehler @ ;

Variable #teil -1 #teil !

: t> ( n adr -- x y z ) >r 3 * cells r> + cell+ v@ ;
: +teil ( adr x y z teil-adr -- )
  fehler off dup teil !
  @ 0 DO I teil @ t>
      teil @ [ t12 ] Literal =
      IF richtung @ 3 min rotation2
      ELSE richtung @ rotation1 THEN
      v+ ( vdup) 3 pick 3 pick 3 pick 3 pick
      +element LOOP 2drop 2drop ;
```

Das Wort *+teil* erwartet gegenüber *+element* noch zusätzlich die Adresse des einzufügenden Teils auf dem Datenstack. Seine Aufgabe besteht darin, ein Teil in einer bestimmten Lage vollständig in den Quader einzubauen.

6. Teile verwalten

Zum Speichern der Bauteilvarianten allokieren wir auf dem Heap für 12 Teile in 24 Lagen à 60 Positionen mit je 2 Zellen und benötigen diesen Stapel insgesamt 12 Mal, entsprechend der maximalen Iterationstiefe.

```
60 24 * 12 * 12 * 2cells * allocate 0=
[IF] Value teilesatz
[ELSE] cr .( neu starten mit Option : -m 50M ) [THEN]
: teilesatz-empty
  teilesatz 60 24 * 12 * 12 * 2cells * erase ;
```

Die Varianten eines Teils verwalten wir in einer gecounteten Liste. Wobei der Count eine Zelle breit ist. Alle Teilelisten reihen wir aneinander und markieren das Ende mit -1. Das Wort *wuerfel-enthalten?* überprüft, ob das in *w* gespeicherte Teil bereits in der Liste enthalten ist. Nur neue Teile lassen wir von *neue-einfuegen* an die Liste anhängen. Schließlich rechnen wir mit dem Forthwort *teile-berechnen* für alle Teile die passenden Varianten aus.

```
: value- ( wadr -- flg )
  2@ w 2@  d- or ;   macro
: (wuerfel-enthalten? ( adr len ( adr len -- wadr / 0 )
  0 -rot bounds
  ?DO I value- 0= IF drop I leave THEN 2cells +LOOP ;
: wuerfel-enthalten? ( adr -- wadr / 0 )
  dcount (wuerfel-enthalten? ;
: einfuegen ( wadr adr -- )
  >r 2@ r@ dcount + 2! 2cells r> +! ; macro
: neue-einfuegen ( adr -- )
  dup wuerfel-enthalten? 0=
  IF w swap einfuegen ELSE drop THEN ;
Variable #teil 0 #teil !
Variable einfuegeptr
: mark-ende
  -1 einfuegeptr @ ! ;
: naechster
  einfuegeptr @ dcount + dup einfuegeptr !  off ; macro
: teile-berechnen
  teilestack-empty
  teilesatz-empty teilesatz einfuegeptr ! einfuegeptr @ off
  einfuegeptr @ teilestack dcount + ! cell teilestack +!
  60 24 * 12 * 0
  DO I 5 /mod 4 /mod 3 /mod 24 /mod 12 mod
  dup #teil @ -
  IF naechster dup #teil ! THEN
  teile swap 0. w 2! richtung ! w 4 -roll +teil
  fehler> 0= IF einfuegeptr @ neue-einfuegen THEN
  LOOP naechster mark-ende naechster ;
teile-berechnen
```

7. Diagnose-Hilfsmittel

Zum Überprüfen der korrekten Berechnung der Teile brauchen wir noch ein paar Hilfsmittel. Das Wort *print* erwartet auf dem Datenstack die Adresse eines Quaders und druckt die Anordnung der Quaderbelegung auf dem Terminal aus. *Print-set* er-

wartet die Adresse einer gecounteten Quaderliste und druckt sie aus. Oft genügt es, die Anzahl Varianten eines Teils zu kennen, dazu dient das Wort *.lens*, das gleichfalls die Adresse einer Quaderliste auf dem Datenstack erwartet. Schließlich können wir mit *printteilestack*, wie später noch gezeigt wird, uns die Entwicklung der Variantenanzahl während den Quaderberechnungen zu jedem Zeitpunkt betrachten.

```
: print ( adr -- ) cr
  #block
  0 DO I #x mod 0= IF cr THEN dup I bit@
  IF 1 ELSE 0 THEN . LOOP drop ;
: print-set ( adr -- )
  dcount bounds ?DO I print 2cells +LOOP ;
: .lens ( adr -- )
  BEGIN dup @ -1 -
  WHILE dup @ 2cells / space 5 u.r dcount + REPEAT drop ;
: printteilestack
  0 teilestack dcount bounds
  ?DO cr dup 6 * spaces 1+ I @ .lens cell +LOOP drop cr ;
```

Nach dem Berechnen der Teile rufen wir *printteilestack* auf und erhalten als Ergebnis die Anzahl Varianten zu jedem Teil. Bei Teil 12 erhalten wir nach Berechnung ohne Symmetrien nur noch 56 von 224 möglichen Varianten. Insgesamt erhalten wir 3132 Varianten

Teile-Nr	Varianten
1	252
2	160
3	224
4	224
5	576
6	160
7	504
8	160
9	288
10	320
11	40
12	(56) 224

Tabelle 1: Anzahl der Teilevarianten

8. Zusammenbau des Quaders

Nach dem Berechnen der Teile können wir nun endlich puzzeln. Dem Programm liegt der nachfolgend beschriebene Algorithmus zu Grunde:

* Entnehme erste Variante des ersten Teils aus der obersten Liste des Teilestapel.

* Füge Teil in den Quaderteilestapel ein.

* Enthält der Quaderteilestapel 12 Teile, liegt eine neue Quadervariante vor. Dann Quader in Quadervariantenliste speichern.

* Erzeuge neue Teileliste aller noch nicht verbauten Teile und ihrer nicht überlappenden Varianten dieser Teile und lege sie auf den Teilestapel. Erkennt der Lochtest ein Loch im Quader, dann Liste vom Stapel entfernen.

* Wiederhole solange, bis oberste Liste leer ist, kehre dann zur vorhergehenden Liste zurück.

* Wiederhole solange, bis Teilestapel leer ist.

Das Forthwort *overlap* erwartet auf dem Datenstack die Adresse eines Teils und vergleicht mit dem in der Variable *#overlap* liegenden Quader. Liegt keine Überlappung vor, gibt das Wort ein False-Flag, sonst einen Wert ungleich Null zurück. Ein neues Teil fügen wir mit *add* zum Bitvektor *#ueberdeckung* hinzu, damit testen wir später auf Löcher im Quader. Beide Worte sind als Makros definiert, dies bewirkt den direkten Einbau des Codes in ein neues Forthwort, ohne den zeitverschwenderischen Aufruf über JSR oder gar indirekt-threaded-Code[5]. Aber erst durch Einsatz des Assemblers erreichen wir bei 64bit-Operationen auf einem 32bit-Forth einen deutlichen Geschwindigkeitsgewinn.

```
: overlap ( adr -- flg )
  2@ #overlap 2@ rot and >r and r> or  ; macro
\ : add ( adr -- )
\  2@ #ueberdeckung 2@ rot or >r or r> #ueberdeckung
   2! ; macro
Code add
    #ueberdeckung # DX mov
        AX  ) CX mov  CX      DX  ) or
    cell AX D) CX mov  CX cell DX D) or
    AX pop Next end-code macro
```

Das Wort *((neue-suchen* hat die Aufgabe, die Varianten eines Bauteils mit dem gerade verbauten Teil zu vergleichen und bei Nichtüberlappung in eine neue Liste zu schreiben. Zusätzlich wird das Teil noch zur Überdeckung addiert. Bei seinem Aufruf nimmt das Wort die Adresse einer Teileliste vom Datenstack und gibt die Länge der neuen Liste zurück.

```
: ((neue-suchen ( adr --  flg )
    dcount bounds
    ?DO I overlap 0= IF I dup add einfuegeptr @
       einfuegen THEN
    2cells +LOOP einfuegeptr @ @ ;
```

Im nächsten Schritt bearbeiten wir den kompletten Teilesatz. Entsteht bei einem Teil eine neue leere Liste, weil ihre Teile

überlappen, dann setzen wir die Variable *produktiv?* auf den Wert "*off*" und beenden sofort das Erstellen des Teilesatzes.

```
: (neue-suche ( adr -- )
  dcount + einfuegeptr @ off naechster
  BEGIN dup @ 0< 0=
  WHILE dup ((neue-suchen 0=
  IF drop produktiv? off EXIT THEN
  dcount + naechster REPEAT drop mark-ende ;
```

Neue-Suche setzt die Variablen *#ueberdeckung* und *produktiv?* zurück, errechnet dann die Überdeckung der bereits verbauten Teile (Versuche ergaben, dass das wiederholte Berechnen die schnellste Methode ist.) und übergibt den obersten Teilesatz an *(neue-suche* zur Bearbeitung weiter.

```
: neue-suche ( -- )
  0. #ueberdeckung 2! produktiv? on
  einfuege-stack  dcount bounds ?DO I add 2cells +LOOP
  einfuegeptr @ teilestack dcount + ! cell teilestack +!
  teilestack dup @ + cell- @ dcount + (neue-suche ;
```

Durch die Verwaltung der Teile mit gecounteten Listen ist die Entnahme eines Teiles komplex geraten. Zunächst überprüfen wir, ob überhaupt noch ein Teilesatz auf dem Stapel vorhanden ist. Dann adressieren wir den obersten Teilesatz und überspringen mit *dcount* + die erste Liste (Hilfskonstruktion zur Adressierung der Liste nach Entnahme von Teilen), und gelangen nun zur ersten Teileliste. Nun prüfen wir, ob noch Teile vorhanden sind, und schreiben im positiven Fall das Teil in die Variable *#overlap* und aktualisieren die Counts der beiden Listen.

```
: ein-teil-entnehmen ( -- adr true/0 )
  teilestack @ 0= IF 0 EXIT THEN
  teilestack dup @ + @
  dup dcount + dcount dup 0<
  IF 2drop drop 0 EXIT THEN dup >r dup
  IF over 2@ #overlap 2!
  cell /string cell- swap ! 2cells  swap +!
  ELSE drop 2drop THEN r> ;
```

Ein neuer Teilesatz ist nur dann *produktiv*, wenn dieser wenigstens noch eine Variante eines Teils enthält und kein Abbruch bei der Listenerzeugung auftrat. Das heißt, von jedem nicht verbauten Teil muss noch wenigstens ein Vertreter vorhanden sein.

```
: produktiv ( -- flg )
  teilestack dup @ + @
  dcount + @ 0< 0= produktiv? @ and ;
Variable #loesungen
```

Ist ein Quader vollständig zusammengebaut, zählen wir die Variable *#loesungen* um 1 hoch und merken uns die zwölf Teile dieser Lösung zur späteren graphischen Darstellung auf dem Heap.

```
: loesung-merken  ( -- )
  einfuege-stack cell+ loesungen dcount + 96 move
  96 loesungen +! 1 #loesungen +! ;
  loesungen-empty
```

Inzwischen sind wir so weit fortgeschritten, dass wir einen kompletten Programmschritt zu einer Teilevariante ausführen können. Das Forthwort *zerlege* entnimmt ein Teil aus dem obersten Teilesatz, fügt es dem Einfügestack hinzu und erzeugt einen neuen Teilesatz der auf die vorhandenen oben aufgelegt wird. Sind zwölf Teile auf dem Einfügestack, liegt eine neue Lösungsvariante vor. Konnte jedoch kein Teil entnommen werden oder ist der neue Teilesatz nicht produktiv, dann entfernen wir den obersten Teilesatz und arbeiten mit dem darunter liegenden weiter.

```
: zerlege
  ein-teil-entnehmen
  IF  #overlap einfuege-stack einfuegen neue-suche
  ELSE produktiv? off THEN
  einfuege-stack @ 96 = IF  loesung-merken THEN
  #ueberdeckung ueberdecked 0= produktiv 0= or
  IF [ 2cells negate ] Literal einfuege-stack +!
    [  cell  negate ] Literal teilestack +!
    teilestack dcount + @ einfuegeptr ! THEN ;
```

Das Wort *wuerfeln* baut uns nun in der Tat die Quader zusammen und beendet seine Arbeit auf Tastendruck oder arbeitet solange weiter, bis sämtliche Teile aufgebraucht sind.

```
\needs durchlaeufe Variable durchlaeufe
: wuerfeln durchlaeufe off
  BEGIN  1 durchlaeufe +! zerlege
  teilestack @ 0= [IFDEF] break?  key? or  [THEN]
UNTIL ;
```

Nach einer Unterbrechung durch einen Tastendruck können wir uns mit dem Aufruf von *printteilestack* vom Fortschritt des Programms überzeugen und anschliessend mit erneutem Aufruf von *wuerfeln* bis zum Ende weiter rechnen.

```
printteilestack
39 56 160 160 160 224 224 252 288 320 504 576
   39 108 108 109 165 170 176 220 216 359 424
        49  70  73 102 102 116 166 143 244 301
            33  36  62  71  64 111  72 143 193
                 1  31  37  41  79  46  94 126
                    6  32  34  69  40  79 113
                       3  15  41  20  40  59
                          5  22  11  19  24
                             4   8  13   5
                                 2   5   4
                                     2   4

OK
```

9. Der Quader mit dem c't-Logo

Ehrlich gesagt räumte ich mir von Anfang an keine große Chance auf einen der ersten Plätze gegen die Übermacht von C-Programmierern und ihren optimierenden Compilern ein. Trotz der kurzen Zeit blieb noch der Gedanke, eine graphische Darstellung des Quaders mit Hilfe von Minos unter Bigforth zu er-

stellen. Zumal schon das grobe Gerüst aus anderen Anwendungen vorhanden war. Buchstäblich in letzter Sekunde vor dem Einsendeschluss konnte ich noch eine graphische Darstellung herbeizaubern. Zur Anzeige des Quaders mit c't-Logo müssen wir deshalb noch unter den vielen Quadervarianten die richtige Variante mit dem c't-Logo suchen. Dazu definieren wir die 3 markanten Teile mit den Buchstaben des Logos und speichern diese Merkmale ab. Das Forthwort ctwuerfel-suchen durchsucht die gefundenen Lösungen und gibt die Nummer des Quaders mit c't-Logo zurück.

```
: >w ( adr -- )
  [ 5 4 * 3 * ] Literal
  0 DO swap
     IF    dup I' I - 1-  +bit
     ELSE  dup I' I - 1-  -bit THEN LOOP drop ;
Create merkmale 3 2 * cells allot
0 0 0 1 1
0 0 0 1 0
0 0 0 1 1
0 0 0 1 0
0 0 0 0 0
0 0 0 0 0
0 0 0 0 0
0 0 0 0 0
0 0 0 0 0
0 0 0 0 0
0 0 0 0 0
0 0 0 0 0
merkmale >w
1 1 0 0 0
1 0 0 0 0
1 1 0 0 0
0 0 0 0 0
0 0 0 0 0
0 0 0 0 0
0 0 0 0 0
0 0 0 0 0
0 0 0 0 0
0 0 0 0 0
0 0 0 0 0
0 0 0 0 0
merkmale 2 cells + >w
0 0 0 0 0
0 0 0 0 0
0 0 1 0 0
0 0 1 0 0
0 0 0 0 0
0 0 0 0 0
0 0 1 1 0
0 0 1 0 0
0 0 0 0 0
0 0 0 0 0
0 0 0 0 0
0 0 0 0 0
merkmale 4 cells + >w
```

```
: ctwuerfel-suchen ( -- nr )
   0 loesungen dcount bounds ?DO
   merkmale          2@ w 2! I 96 (wuerfel-enthalten?
   merkmale 2 cells + 2@ w 2! I 96 (wuerfel-enthalten? and
   merkmale 4 cells + 2@ w 2! I 96 (wuerfel-enthalten? and
   IF drop I loesungen cell+ - 96 / leave THEN
   96 +LOOP ;
```

Ja, nun nähern wir uns dem Ende. Wir merken uns mit *!time* den Startzeitpunkt, *wuerfeln* die Quader zusammen und nach längerem Warten erhalten wir die Zahl 409963 für gefundene Lösungen und sehen ein neues Fenster mit der Abbildung des c't-Quaders aufgehen. Dazu musste das Programm die beachtliche Leistung von 2.515.525.163 Bauteilvergleichen durchführen und jeweils eine neue Teileliste schreiben.

```
!time
[IFUNDEF] break? singletask [THEN]  wuerfeln
[IFUNDEF] break? multitask  [THEN]
cr .( Rechenzeit : ) .time
cr .( ausgeführte Vergleichszyclen : ) durchlaeufe @ u.
cr .( gefundene Lösungen : ) #loesungen @ .
ctwuerfel-suchen ?dup
[IF] cr .( Variante mit CT-Logo : ) .
[ELSE] cr .( keine Variante mit CT-Logo gefunden )
[THEN] cr
include ctwgraph.m
ctwuerfel open 100 wait actualized? On
```

Das untere Fenster (Abbildung 3 (a)) zeigt den Quader mit c't-Logo, wie er von Bigforth dargestellt wird. Mit Hilfe der Schieber kann der Quader in allen Richtungen gedreht und in seiner Größe verändert werden. Jede der 409963 Varianten kann das Programm darstellen und die Teile einzeln durch Klicken auf ihre Knöpfe ein- und ausschalten.

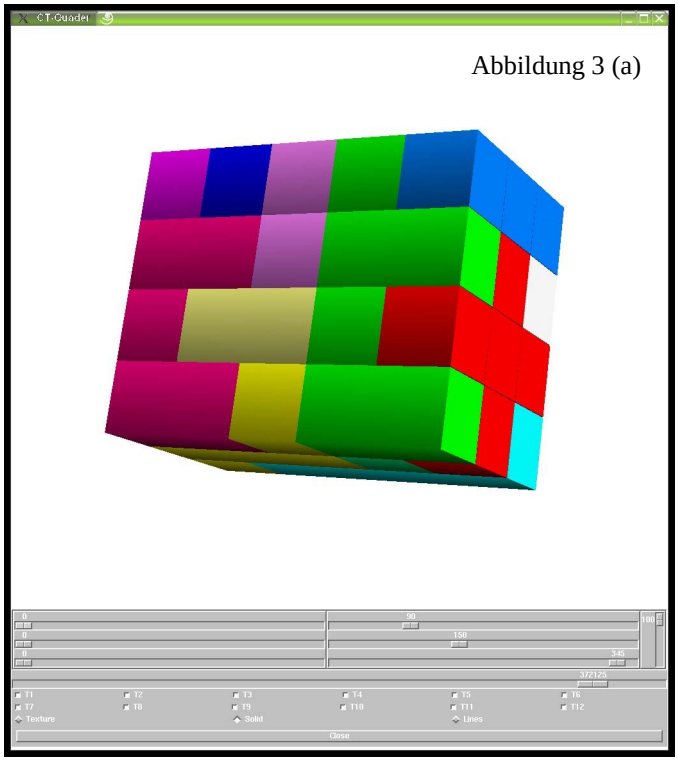


Abbildung 3 (a)

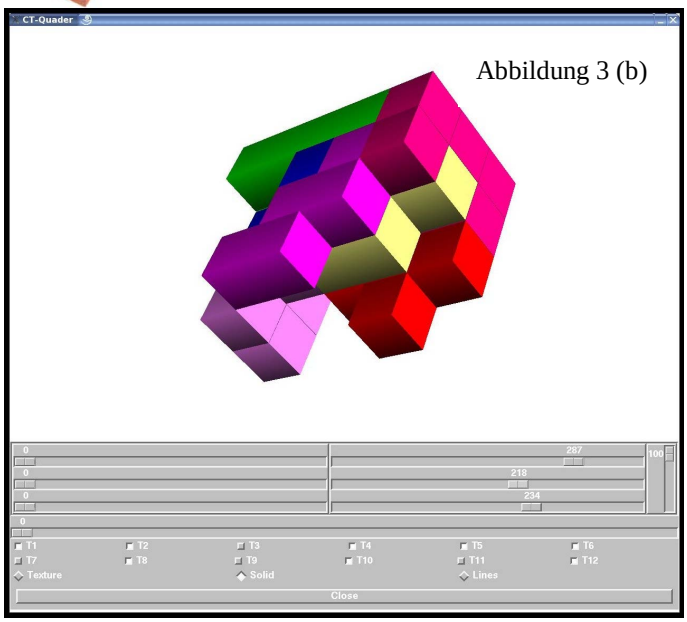


Abbildung 3 (b)

10. Etappen der Programmierung

Schnell war mit Hilfe der Vorarbeiten aus Pflock-Solitaire ein Programm entstanden, das die Teile erzeugt und sie komfortabel in Objektlisten verwaltet. Und bald entstand in der interaktiven Programmierumgebung von Forth ein Programm, das Quader zusammenbauen konnte. Doch nach einigen Stunden Rechenzeit ergab eine Hochrechnung, dass das Programm ca. 1500 Stunden (Dell Inspiron 8100 866MHz 512Mb) zur Verarbeitung aller Teile benötigen würde. So mußte die komfortable objektorientierte Listenverwaltung einem gewöhnlichen Forthprogramm weichen. Der Lohn war laut Hochrechnung nun ein Programm mit ca. 150 Stunden Rechenzeit. Zu diesem Zeitpunkt gab es allerdings noch keinen Lochtest und das Programm berechnete auch noch die Symmetrien mit. Nach Entfernung der Symmetrien von Teil 12 (Teil mit Aufdruck "t"), kam ich nun auf eine Rechenzeit von 40 Stunden und zum erstem Mal zeigte sich die Zahl 409963 als Ergebnis. Aber erst der Lochtest brachte eine kurze Rechenzeit von knapp 4 Stunden. Mit diesem Stand musste ich mein Programm bei der c't-Redaktion einreichen. Auf dem Testrechner der c't-Redaktion mit Pentium 4 3,2 GHz erwartete ich nochmals eine erheblich kürzere Rechenzeit. Aber eigene Tests ergaben inzwischen gerade mal eine um 25% kürzere Rechenzeit auf einem Pentium 4 mit 2,6 GHz. Weitere Versuche zeigten eine starke Abhängigkeit der Rechenzeit vom Zugriff auf das RAM. Da ich gerade so richtig in Schwung war, optimierte ich in Assembler, auch unter Einsatz von MMX-Registern, weiter. Am Ende der Optimierung war das Wort (neue-suche rein in Assembler formuliert. Damit erreichte ich eine Rechenzeit von 1 Std. 21 Min. auf Pentium 3 mit 866 MHz und 61 Minuten auf Pentium 4 mit 2,6 GHz

```
Code (neue-suche ( adr -- )
R: BX push BP push S:
#ueberdeckung #) MM0 PLDQ
#overlap #) BP mov einfuegeptr #) OP mov
BEGIN AX ) CX mov CX CX test 0>=
WHILE
0 # OP ) mov 0 # DX mov
BEGIN
-cell CX AX DI) BX mov BP BX and
0= IF
CX AX I) BX mov cell #overlap + #) BX and
0= IF
-cell CX AX DI) MM1 PLDQ MM1 MM0 por
MM1 cell DX OP DI) PSTQ
2cells # DX add
THEN THEN 2cells # CX sub 0= UNTIL
DX OP ) mov
DX DX test 0=
IF AX pop
R: BP pop BX pop S: Next THEN
tp-len # AX add tp-len # OP add
REPEAT
-1 # OP ) mov tp-len # OP add OP einfuegeptr #) mov
MM0 #ueberdeckung #) PSTQ
-1 # AX mov AX produktiv? #) mov AX pop
R: BP pop BX pop S: NEXT end-code
```

11. Die Stunde der Wahrheit

Nach einigen Wochen bekam ich eine eMail von c't-Redakteur Harald Bögeholz mit der Bitte, ich möge die Größe des T-Shirt, das es als Trost-Preis geben solle, mitteilen und dem Hinweis auf die erste vorläufige Auswertung, die im Internet zu sehen war [1][3]. Immerhin war jetzt klar, daß mein Programm richtig gerechnet hatte. Aber nachdem ich mir die Gewinnerliste im Internet betrachtet hatte, war ich doch sichtlich schockiert über den großen Abstand zu den ersten Plätzen. So wurden die ersten 36 Plätze, das ist genau die Hälfte der richtigen Einsendungen, ausschliesslich von C und C++ Programmierern belegt. 52 Programmierer kamen in die Zeitwertung (Programme mit einer Rechenzeit größer 1 Stunde wurden abgebrochen), darunter gerade mal ein Java und 3 Delphi Programme. 20 weitere waren ausserhalb der Zeitwertung.

12. Eine neue Lösung für das c't-Puzzle

Nun - diese Schlappe konnte ich nicht auf mir sitzen lassen! Mich quälte die Frage: Lag es an meinen Programmierkünsten oder gar an Forth selbst? Deshalb sah ich mir einige der schnellsten C-Programme an und entschied mich, zum Vergleich das kurze und leicht verständliche C-Programm von Uli Schuhmacher in Forth nachzuprogrammieren [1]. Der wesentliche Unterschied bestand in einer anderen Lösungsstrategie.

12.1 Der X-Trick

Der X-Trick wurde bereits 3 Wochen vor Einsendeschluss im Heise-Diskussionsforum bekannt. Er beruht darauf, dass das wie ein + oder auch X aussehende Teil 11 auf Grund seiner Symmetrieeigenschaften mit 12 Varianten auskommt. Allerdings entstehen daraus Probleme mit den Symmetrien an 4 Po-

Platz	Name	Zeit	Graphik	Sprache	Betriebsystem
1	Ronald Nölte	83,0	-	C++	Windows
2	Jochen Hoenicke	86,5	-	C	Linux
3	Oliver Ehli	87,5	-	C	Linux
4	Uli Schuhmacher	87,7	-	C++	Windows
5	Thomas Althaus	89,3	-	C++	Windows
6	Johannes Overmann	94,4	-	C++	Linux
7	Georgios Papoutsis	94,7	-	C++	Linux
8	Daniel Gutekunst	95,7	-	C++	Linux
9	Tobias Glasmachers	102	-	C++	Windows
10	Bernd Lehle	113	-	C++	Linux
..	..	..	..	..	..
37	Oliver Faulhaber	1181	schönstes	Java	
..	..	..	..	..	..
47	Matthias Haferkorn	1878	-	Delphi	Windows
..	..	..	..	..	..
49	Reinhard Franzkeit	2251	-	Delphi	Windows
50	Ingo Warnke	2795	-	Delphi	Windows
–	Wolfgang Zeidler	timeout	-	Pascal/JavaScript	DOS
–	Arne Binder	timeout	-	Delphi	Windows
–	*Ewald Rieger*	*timeout*	-	*Forth*	*Windows*
–	Hannes Kuchler	timeout	Textmode	C	Windows
–	Günter Jantzen	timeout	-	Python	Linux
–	Josef Dühnen	timeout	-	Delphi 3	Windows
–	Hendrik Payer	timeout	DirectX	Delphi	Windows
–	Reinhard Buchholz	timeout	-	Delphi	Windows
–	Wolfgang Kais	timeout	-	VB 5.0	Windows
–	Armin Braun	timeout		Lisp	Windows

Tabelle 2: Auszug aus der Gewinnerliste

sitionen, die über ein weiteres asymmetrisches Teil entfernt werden müßen. Zu diesem Zweck nehmen wir das asymmetrische Teil 5. Dieses Teil passt in maximal 576 Varianten in den Quader, das heißt wir beseitigen damit auch die größte Zahl an Varianten. Dazu flippen wir die Einheitswürfel entlang zweier Hauptachsen und erhalten dadurch eine neue Spiegelvariante um eine der Hauptachsen.

```
Variable fliporder
: (flipw
  >r >r fliporder @ %001 and IF #x 1- swap - THEN
    r> fliporder @ %010 and IF #y 1- swap - THEN
    r> fliporder @ %100 and IF #z 1- swap - THEN ;

: flipw ( w-adr order - w )
  fliporder ! 0. rot
  #block 0
  DO dup I bit@
   IF sp@ cell+ I #x /mod #y /mod
   (flipw w> +bit
   THEN LOOP drop ;
2Variable vorbelegung
```

Das Forthwort *minmuster?* erwartet eine Teileadresse auf dem Datenstack und vergleicht mit dem gerade erzeugten Teil in der Variable *w*. Ist das Teil in *w* zu dem Teil an der Adresse *adr* um eine der Hauptachsen gespiegelt, wird ein true-Flag übergeben. Dieses Wort wird auf die Varianten des X-Teils angewendet.

```
: minmuster? ( adr -- flg )
  >r w %011 flipw r@ 2@ d=
    IF rdrop true EXIT THEN
    w %101 flipw r@ 2@ d=
    IF rdrop true EXIT THEN
    w %110 flipw r@ 2@ d=
  rdrop ;
```

Das Forthwort *maxmuster?* vergleicht das neue Teil mit maximaler Variantenanzahl in der Weise, dass Symmetrien nur dann entfernt werden, wenn eine Symmetrie um die gleiche Hauptachse wie beim gerade verwendeten X-Teil vorliegt. Deshalb müßen wir den Teilesatz für jedes X-Teil neu berechnen.

```
: maxmuster? ( adr -- flg ) >r
  w %011 flipw r@ 2@ >r $0FFFFFFF
and r> d=
  vorbelegung %011 flipw vorbelegung
2@ d= and
  IF rdrop true EXIT THEN
  w %101 flipw r@ 2@ >r $0FFFFFFF
and r> d=
  vorbelegung %101 flipw vorbelegung
2@ d= and
  IF rdrop true EXIT THEN
  w %110 flipw r@ 2@ >r $0FFFFFFF and r> d=
  vorbelegung %110 flipw vorbelegung 2@ d= and
  rdrop ;
Variable minteil t11 minteil !
Variable maxteil t5 maxteil !
```

Pos? gibt uns das erste freie Bit in einem 32bit-Vektor zurück.

```
Code pos? ( adr -- n )
  AX ) DX mov DX DX test 0<>
  IF DX DX bsf DX AX mov NEXT THEN
  cell AX D) AX mov AX AX bsf 32 # AX add
  NEXT end-code
```

Das im ersten Teil dieses Artikels beschriebene Forth-Wort *neue-einfuegen* wird nun so modifiziert, daß es nur noch X-Teile ohne Symmetrien und im Bitvektor möglichst weit vorne liegende Teile erzeugt. Dies bewirkt eine frühe Kollision mit anderen Teilen und bringt uns den größten Geschwindigkeitsgewinn überhaupt. *Maxmuster?* entfernt (wie oben beschrieben) einige Symmetrien und verwendet möglichst spät kollidierende Teile (bringt eine geringe Geschwindigkeitszunahme).

```
: neue-einfuegen ( adr -- )
  dup dcount bounds
```

```
?DO I 2@ w 2@ d- or 0=
  IF drop unloop EXIT THEN
  teil @ minteil @ =
  IF I minmuster?
  IF w pos? I pos? <
  IF w 2@ I 2! THEN drop unloop EXIT THEN THEN
  teil @ maxteil @ =
  IF I maxmuster?
  IF w pos? I pos? >
  IF w 2@ I 2! THEN drop unloop EXIT THEN THEN
  2cells +LOOP w swap einfuegen ;
```

Abgesehen von zwei Ausnahmen bleibt die Teileerzeugung gleich. Die X-Teile berechnen wir zuerst und speichern sie in einer separaten Liste ab. Danach belegen wir den Quader mit einem X-Teil vor und erzeugen für die restlichen Teile nur die mit dem X-Teil nicht überlappenden Varianten.

12.2 Systematisches Auffüllen des Quaders

Systematisch bedeutet, dass wir den Bitvektor des Quaders von einer Seite beginnend auffüllen. Das nächste freie Bit wird immer zuerst belegt. Daraus ergeben sich eine ganze Reihe von Vorteilen.

* Nach dem Auffüllen der unteren 32 Bits (entspricht einer Zelle in einem 32bit-Forth) genügt es, nur noch mit den restlichen 28 Bits weiterzurechnen.

* Die Reihenfolge der Bits im Bitvektor wählen wir so, dass die Bit-Nummerierung aufsteigend entlang der x-Achse, dann der y-Achse und zuletzt an der z-Achse erfolgt. Wenn wir den Quader auf die kleinste Fläche stellen (x=3 y=4 z=5) und von unten nach oben auffüllen, haben die Teile weniger Freiheitsgrade als bei jeder anderen Anordnung. Stellen wir den Quader auf seine größte Fläche (x=5 y=4 z=3), benötigt das Programm die 5-fache Rechenzeit. Uli Schuhmacher hat sogar eine spiralförmige Anordnung gewählt und damit nochmals die Rechenzeit um 6% verkürzt.

* Zum Besetzen einer bestimmten Bitposition benötigen wir eine komplette Liste aller noch nicht eingebauten Teile, die genau dieses Bit belegen können. Da schon alle vorausgehenden Bits bereits im Bitvektor besetzt sind, können wir mit diesem Bereich überlappende Teilevarianten streichen. Diese Teilelisten verwalten wir in einer Matrix, die wir über die zu belegende Bitposition und Teilekombination indizieren. Diese Matrix brauchen wir allerdings zweimal, eine für die gesamte Bitvektorbreite und eine für die Berechnungen mit den restlichen 28 Bits.

12.3 Zugriff auf das RAM minimieren

In den gerade beschriebenen Teilelisten muss zu jedem Bitvektor seine Teilenummer bekannt sein. Dazu nutzen wir die oberen vier freien Bits die den Bitvektor zu 64 Bit ergänzen. Damit holen wir Bitvektor und Teile-Nr. mit einem bzw. zwei Speicherzugriffen in die CPU.

12.4 Das Programm

Der nachfolgende Quelltext beschreibt die wesentlichen Teile des in Forth nachcodierten Programms von Uli Schuhmacher. Das Wort freiesbit erwartet auf dem Datenstack eine Zelle eines Quaders, gefolgt von der Teilekombination der schon verwendeten Teile, die nur dupliziert wird. In dem Argument w wird mit dem Assemblerbefehl bsf das erste freie Bit bestimmt und dessen Position auf den Datenstack gelegt.

```
Code freiesbit ( w tb -- w tb tb bit )
  AX push AX push 2 cells SP D) AX mov
  AX not AX AX bsf NEXT end-code :ax 0 T&P macro
```

Nun prüfen wir mit dem Forthwort *inner-not-overlap-teil?* auf Überlappung des Teilquaders *wh1* mit dem Quader, auf den die Adresse *adr* zeigt. Zuvor müssen wir aber die Teilenummer ausmaskieren. Liegt kein passendes Teil vor, wird ein False-Flag auf den Datenstack gelegt. Andernfalls wird der Stack für eine neue Rekursion vorbereitet, indem die Kopien der Argumente Quader und Teilekombination auf den Stack gelegt werden und das Teil an der Adresse *adr* zum Quader hinzugefügt wird. Weiter markieren wir mit einem True-Flag.

```
Code inner-not-overlap-teil?
  ( wh1 tb adr -- wh1 tb adr wh2 tb t / wh1 tb adr f )
  AX push AX ) OP mov
  OP CX mov $0FFFFFFF # OP and
  2 cells SP D) DX mov OP DX test 0<>
  IF 0 # AX mov
  ELSE OP DX or DX push
  28 # CX shr 1 # AX mov AX shl
  2 cells SP D) AX or AX push
  -1 # AX mov THEN NEXT end-code :ax 0 T&P macro
```

Zum Messen des Rechenaufwands für unterschiedliche Lösungsansätze zählen wir in der Variable *tries* die rekursiven Aufrufe von *(innerresolve* und *(resolve*.

```
Variable tries
```

Das Forthwort *(innerresolve* bearbeitet nur die obere Quaderzelle. Zuerst prüfen wir die Teilekombination auf einen vollständigen Quader. Liegt ein vollständiger Quader vor, zählen wir die Variable *loesungen* um 1 hoch und brechen die Rekursion ab. Sonst suchen wir das erste freie Bit im Quader und holen uns mit dem Wort *positssm* die Adresse der Teileliste für die Bitposition und Teilekombination auf den Datenstack. Diese Liste wird nun solange iteriert bis wir auf eine leere Quaderzelle als Endmarke stoßen. Wird dabei ein passendes Teil gefunden, rufen wir *(innerresolve* durch *recurse* in sich selbst auf.

```
: (innerresolve ( wh teilcomb -- )
  dup %111111111111 =
  IF 2drop 1 loesungen +! EXIT THEN
```

```
    1 tries +!
    freiesbit positssm
    BEGIN dup @
    WHILE inner-not-overlap-teil?
    IF recurse THEN cell+ REPEAT
    2drop drop ;
: unterenbits-besetzt ( wl tb -- wl tb flg )
    over -1 = ; macro
Code posits-ende? ( adr -- adr flg )
    AX push AX ) DX mov
    cell AX D) AX mov DX AX or
    NEXT end-code :ax :or T&P macro
```

Der Code von *not-overlap-teil?* erledigt die gleiche Aufgabe wie *inner-not-overlap-teil?* für einen ganzen Quader.

```
Code not-overlap-teil?
  ( wh wl tb adr -- wh wl tb adr wh wl tb t / wh wl tb
    adr f )
  AX push cell AX D) OP mov
  OP CX mov $0FFFFFFF # OP and
  3 cells SP D) DX mov OP DX test 0<>
  IF 0 # AX mov
  ELSE OP DX or DX push
  AX ) OP mov
  3 cells SP D) DX mov OP DX test 0<>
  IF AX pop 0 # AX mov
  ELSE OP DX or DX push
  28 # CX shr 1 # AX mov AX shl
  3 cells SP D) AX or AX push
  -1 # AX mov THEN THEN NEXT end-code
  :ax 0 T&P macro
```

Nun kommen wir zur Rekursion über den gesamten Quader. Das Forthwort *(resolve* sieht nach, ob die *unterenbits-besetzt* sind und ruft dann *(innerresolve* auf, um den oberen Teil des Quaders aufzufüllen. Der restliche Code arbeitet sinngemäß wie bei *(innerresolve* beschrieben.

```
: (resolve ( wh wl teilcomb -- )
  unterenbits-besetzt
  IF nip (innerresolve EXIT THEN
  1 tries +!
  freiesbit posits
  BEGIN posits-ende?
  WHILE not-overlap-teil?
  IF recurse THEN 2cells + REPEAT
  2drop 2drop ;
```

Nun kommen wir zur letzten Definition des Wortes *resolve*. Seine Aufgabe besteht darin, die Berechnungen für die 12 Varianten des X-Teils durchzuführen. Vor dem Eintritt in die DO-LOOP-Schleife setzen wir den Zähler für die Lösungen auf null und merken uns die Startzeit. In der Schleife beginnen wir mit dem Löschen der Zeiger auf Teilelisten in den Tabellen *posits* und *positssm* und setzen noch den *Einfügeptr* für die Teilelisten auf den Anfang. Nun entnehmen wir ein X-Teil und schreiben ein Duplikat in die Variable *Vorbelegung* (siehe Entfernen von Symmetrien unter 12.1 der X-Trick) und legen die Teilenummer auf den Datenstack. Damit starten wir *(resolve* zur Berechnung der Lösungen für ein X-Teil. Vor der nächsten Berechnung drucken wir die aktuelle Variantenanzahl und danach die Zwischenzeit aus.

```
: resolve
  loesungen off !time
  12 0 DO
    posits-empty positssm-empty
    #einfuegeptr dup off einfuegeptr !
    #x-teilesatz cell+ 2cells I * + 2@
    2dup vorbelegung 2! teile-berechnen
    cr #teilesatz .lens 10 wait
    1 (resolve cr loesungen @ 6 u.r space .time 10 wait
    LOOP ;
hex
#x-teilesatz dcount bounds [do] [i] 2@ swap cr ud. 8 [+loop]
decimal
resolve
cr tries @ u.
```

12.5 Zusammenfassung

Während den Berechnungen schreibt das Programm bei jedem X-Teil zwischen 10.000 und 20.000 Teilelisten in die Tabellen *posits* und *positssm*, aus denen Bitvektor und Teilenummer gemeinsam gelesen werden. Sind die unteren 32bit aufgefüllt, genügt ein einziger 32bit Zugriff pro Teil. Das Programm legt die Teilekombination und den Teilquader auf dem Datenstack ab und startet damit eine neue Rekursion. Geschickt programmierte Primitives erledigen ihre Arbeit weitgehend in den Registern der CPU. Damit erreichen wir auf einem Nativ-Code-Forth zu C vergleichbare Geschwindigkeiten. Das Programm benötigt zur Berechnung aller Varianten 1.459.803.256 Rekursionen und meldet sich auf einem Pentium 4 mit 2,6 GHz nach 194 Sekunden mit dem richtigen Ergebnis von 409963 gefundenen Lösungen zurück. Das ist wenige Sekunden schneller als das nicht optimierte C-Programm von Uli Schuhmacher. Compiliert man das Programm in der höchsten Geschwindigkeitsoptimierungsstufe braucht das C-Programm nur noch 120 Sekunden. Bei Forth ohne Macrodefinitionen [5] compiliert, steigt die Rechenzeit auf 300 Sekunden an. Kurios war auch zunächst die Tatsache, dass das Zählen der Rekursionen in der Variablen *tries* die Rechenzeit von 194 Sekunden auf 7 Minuten ansteigen ließ. Erst nachdem die Variable am Anfang des Programms deklariert wurde, schlug das Zählen nur noch mit 4 Sekunden zu Buche. Offensichtlich wurde der Programm-Cache ungültig und musste ständig neu aus dem RAM geladen werden. Auch wenn der pfiffige Algorithmus von Uli Schuhmacher für meine Teilnahme am Wettbewerb zu spät kam, um sich auf den ersten Plätzen einzureihen, so konnte ich wenigstens zeigen, dass eine zeitgemäße Forthimplementierung, sowohl bei der graphischen Darstellung, als auch bei der Performance nicht zum alten Eisen gehört.

Literatur

[1] Harald Bögeholz. Das c't Puzzle Lösungsprogramme. http://www.heise.de/ct/puzzle/programme

[2] Harald Bögeholz. Programmierwettbewerb. http://www.heise.de/ct/03/07/230/#wettbewerb

[3] Harald Bögeholz. Die Quader sind gefallen, volume 14. c't-Magazin, 2003.

[4] Harald Bögeholz. Zusamengewürfelt. c't-Magazin, 7, 2003.

[5] Bernd Paysan. bigFORTH+MINOS Dokumentation Kap. 19. Der optimierte Compiler. http://www.jwdt.com/~paysan/.

Betreff: swap pin
Von: Michael Kalus <michael.kalus@onlinehome.de>

Moin Friederich,

habe ich Dir eigentlich schon gesagt, daß Dein Design der VD in PDF, also die Online-Ausgabe, klasse ist? Die Bilder sind in Farbe und klar und deutlich, die Trennlinien in den dezenten Grüntönen, das kleine, grüne Swaplogo und die farbige Flagge der USA bei den "Lebenszeichen" - das kommt sehr gut, um mal einige Beispiele zu nennen. Demgegenüber ist die gedruckte Fassung, in der alles grau in grau wird, ja eher trist.
Besonders die geringe Auflösung der Bilder ist nicht so gut im Print.
Also Leute - immer die Onlineversion lesen.

Michael

Es freut mich natürlich, wenn ich höre/lese, daß das Design der VD gefällt. Tatsächlich sind die PDF-Versionen und die Online-Ausgaben deutlich „bunter" . Im s/w-Druck geht eben doch manches verloren. Neben den Farben, z.B. in den Photographien, leidet vor allem auch die „Schärfe" in Abbildungen unter den Rasterungen der Druckausgabe. Die Vervielfältigung der VD im Kopiergeschäft kostet weitere „Tiefenschärfe". Das ließe sich nur mit erheblichem wirtschaftlichen Aufwand verhindern (Herstellung der VD durch einen Druckbetrieb). Was den wirtschaftlichen Aufwand anbelangt, sind wir aber allesamt, meine ich, sehr froh darüber, daß die VD heute nur noch 4,- € pro Ausgabe kostet. In 2002 waren das noch 5,11 €, bzw. 10,- DM. Die „gedruckte" Ausgabe der VD ist heute, um es sinngemäß mit Wolf Wejgaards Worten zu sagen, der Anzahl der Mitglieder und dem Verein angemessen.

Die Online-Ausgaben der VD liegen auf dem Server der Forthgesellschaft bereit und können bei

http://www.forth-ev.de/vd.htm#online

eingesehen werden. Allerdings sind dort nur diejenigen Ausgaben verfügbar, die hier in Moers entstanden sind. Das sind also die Ausgaben 01/1998 bis 04/2003; abzüglich der Ausgaben 01/2001 bis 01/2002, die Martin Bitter in Hamminkeln gemacht hat. Die vier Ausgaben des aktuellen Jahres werden vermutlich im Herbst 2004 ebenfalls dort eingestellt werden.

Darüber hinaus arbeitet die Forthgesellschaft daran, soweit dies möglich ist, alle bisherigen Ausgaben der VD im PDF auf ihrem Server verfügbar zu machen. Die Vollständigkeit dieser Arbeit hängt allein davon ab, inwieweit sich noch Mitarbeiter finden, die Ulrich Hoffmann dabei helfen, vorhandene Ausgaben zu scannen.

Die in Moers hergestellten VD liegen bereits ausnahmslos als PDF vor. Einige davon sind wegen des darin enthaltenen, umfangreichen Bildmaterials bis zu 10 Mbyte groß. Auch hierzu wird der kommenden Herbst ausreichend Gelegenheiten bieten, diese Dateien weiter zu komprimieren, ohne Qualitätsabstriche zu akzeptieren.

Also erlaube ich mir, die Aufforderung von Michael Kalus wie folgt zu ändern – immer wieder mal auf dem Server der Forthgesellschaft nachsehen, wie weit die Arbeit gediehen ist.

fep

Betreff: Meldungen für die Vierte Dimension ?
Von: Friederich.Prinz@t-online.de

Hallo Friederich,

anbei ein kleiner Artikel über die Neuerungen von Win32Forth für die nächste Ausgabe der VD.

MfG Dirk Busch

Neues von Win32Forth

Win32Forth lebt. Vor rund 2 Jahren hat sich eine Gruppe von Forth-Programmieren zusammengefunden, die sich zur Aufgabe gemacht hat, Tom Zimmer's Win32Forth zu pflegen und weiter zu entwickeln.

Inzwischen wurde unter www.win32forth.org die Version 6.09.12 veröffentlicht. Einige der wesentlichen Neuerungen seit Toms letzter „offiziellen" Version 4.2 Build: 0671 sind:

- Erzeugung von nativen WIN32 Applikationen
- Trennung von Code und Daten zur Steigerung der Geschwindigkeit
- Absolute Adressierung, d.h. es ist keine umständliche Konvertierung von Zeigern mittels ABS>REL und REL>ABS mehr notwendig, wenn Windows Funktionen angesprochen werden sollen.
- Neue Tool's wie:
  'ForthForm' zur Erstellung von Dialogboxen und Toolbars für Win32Forth Applikationen, dem 'Projektmanager' zur Verwaltung von Forth-Projekten und 'SciEdit', dem neuen MDI-Editor mit Syntax coloring, integrierter Onlinehilfe, einem Vokabular und Klassenbrowser, und integriertem Debugger als Ersatz für Toms WinViev

Weitere Info's zu Win32Forth gibt's unter:

www.win32forth.org oder **dirk@win32forth.org** oder **http://www.win32forth.de.vu/**

Erinnern Sie sich an die Ausgabe 02/2000 der VD? Erinnern Sie sich an den Bericht über das damals gerade sehr gut angelaufene Projekt SETI der Universität Berkeley? Können Sie sich vorstellen, daß die Suche nach Signalen von „ET“ mehr als 4,5 Millionen Menschen weltweit dazu bewegen konnte, überwiegend private, ungenutzte Rechnerkapazitäten für einen längeren Zeitraum kostenlos der Wissenschaft zur Verfügung zu stellen?

Die aktuell für das SETI-Projekt arbeitenden rund 1 Million Computer leisten ca. 60 TeraFLOPS (Billionen Gleitkommaoperationen pro Sekunde). Das ist das Fünffache von dem, was der zur Zeit leistungsstärkste konventionelle Zentralrechner (IBM ASCI White) erreichen kann. Und das Potential der privaten Rechnerkapazitäten ist bereits heute ein Vielfaches größer als das, was für SETI aktuell bereitgestellt wird.

Public Computing
Menschen wieder mit der Wissenschaft verbinden

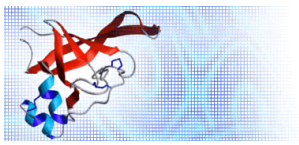
Das ist der Titel eines Aufsatzes von Dr. David P. Anderson, vom Space Science Laboratory der University of California - Berkeley (siehe: *http://boinc.de/madrid_de.htm*).

Und das ist eines der wesentlichen Anliegen des Projektes BOINC (**B**erkeley **O**pen **I**nfrastructure for **N**etwork **C**omputing). Vor dem Hintergrund des unglaublichen Erfolges den SETI nach wie vor hat (ET ist noch nicht gefunden, aber die “Beteiligung der Menschen an Wissenschaft“ hat in diesem Ausmaß niemand für möglich gehalten), wuchsen nach und nach Begehrlichkeiten anderer Forschungseinrichtungen, die Bedarf an ähnlichen Rechenkapazitäten haben. Um die Bedarfe und deren Befriedigung zu kanalisieren, bzw. um Projekte und private Rechnerkapazitäten zusammenzubringen, hat Berkeley BOINC (siehe: *http://boinc.berkeley.edu/*) entwickelt. Im Rahmen der von BOINC vorgegebenen Strukturen können dort Projekte für sich “werben“. Interessierte Menschen können sich aussuchen, welchen Projekten sie welche Kapazitäten zur Verfügung stellen möchten. Und eine ständig wachsende Gemeinde, im positiven Sinne des Wortes, müht sich in - mal mehr und mal weniger ernst genommenen - “Competitions“ um gute (leistungsstarke) Ränge in den Credit-Listen der Projekte.

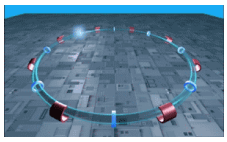
Die Projekte stellen ihren “Nutzern“ Foren zur Verfügung, deren Threads von den Nutzern weitgehend selbst betreut werden. In den Foren werden Informationen über die Hard– und Software der privaten Rechner ausgetauscht, Treffen verabredet, eigene Programmierprojekte aufgelegt und durchgeführt (z. B. zur Erstellung und Auswertung der “Ranglisten“) und manchmal auch die wissenschaftlichen Hintergründe der Projekte mit den beteiligten Wissenschaftlern selbst diskutiert. Das Vereinsleben blüht, um so mehr, weil die Zahl der Rechnerplattformen und Betriebssysteme, mit denen eine Beteiligung möglich ist, vor allem aufgrund der Initiative der Gemeinde ständig wächst.

Aktuell bieten sich in BOINC vier Projekte zur “privaten Beteiligung“ an. Allen voran steht selbstverständlich SETI, das über die bereits angesprochene, unglaubliche Fangemeinde verfügt und per Definition noch für viele Jahre die Rechenleistung dieser Gemeinde benötigen wird.

Climateprediction.net überprüft in vielen Simulationen die möglichen Genauigkeiten, die sich in Klimavoraussagen über unser Jahrhundert erreichen lassen. Das Projekt hat einen starken Zulauf. Es trifft ganz offensichtlich einen Nerv unserer Zeit.



Predictor@home sucht nach Antworten auf Fragen zu Krankheiten, die in verschiedenen Zusammenhängen mit Proteinen stehen. Predictor ist aktuell inaktiv. Im Internet einsehbare Statistiken zeigen, daß auch dieses Projekt während seiner aktiven Zeit reichlich “Mitarbeiter“ gewinnen konnte.



Rechtzeitig zum 50. Geburtstag des CERN konnte das Projekt LHC (Large Hadron Collider) “in Produktion“ gehen. Im 27 km langen Ring des großen Beschleunigers, der 2007 seine Arbeit aufnehmen wird, sollen Protonen in zwei Partikelstrahlen nahezu auf Lichtgeschwindigkeit beschleunigt werden und dann frontal aufeinander prallen. Bereits heute beschäftigen sich die Physiker des CERN mit Fragen nach der Stabilität der Partikelstrahlen. Auch hier sollen Simulationen Voraussagen machen oder Schätzungen absichern. Berechnet werden dabei üblicherweise 60 Partikel im Einfluß wechselnder Magnetfelder. Simuliert werden dabei 100.000 Durchgänge der Partikelwolke durch das größte wissenschaftliche Werkzeug der Welt.

LHC hatte bereits in der Testphase so viele Interessenten, daß die Projektleitung gezwungen war, die Anzahl der Teilnehmer zu begrenzen, weil die von CERN zur Verfügung gestellte Hardware gar nicht in der Lage war, die Menge der Anfragen nach Arbeit und der zurückgelieferten Ergebnisse pro Zeiteinheit aufzunehmen und weiter zu verarbeiten. Mit dem Start der “Produktion“ wurde die mögliche Teilnehmerzahl auf 5.000 erhöht. Sofort nach der Veröffentlichung dieser Zahl und der Bekanntgabe, daß offizielle Anmeldungen für das Projekt LHC nun möglich seien, setzte ein regelrechter Sturmlauf auf die verfügbaren „Accounts“ ein.

Die Menschen wieder mit der Wissenschaft zu verbinden, scheint keine allzu schwierige Aufgabe zu sein. Die Bereitschaft, privat finanzierte Rechnerkapazitäten, Telephongebühren oder DSL-Leitungen, Zeit, eigene Fertigkeiten und Fähigkeiten (zum Beispiel für notwendige Programmierarbeiten) zur Verfügung zu stellen, und vor allem die Bereitschaft, sich zu interessieren und sich mit den Zielen wissenschaftlicher Projekte auseinander zu setzen, diese Bereitschaft ist potentiell vorhanden. Um die potentielle Bereitschaft in konkrete Mitarbeit umzuwandeln, haben SETIs Wissenschaftler im wesentlichen nur zwei Dinge tun müssen.

Das Erste war, den Schritt aus dem Elfenbeinturm zu wagen und sich dem Interesse technisch, wissenschaftlich gebildeter Menschen zu stellen.

Der zweite Schritt war vielleicht viel schwieriger. Die Wissenschaftler mußten zulassen (und sogar fördern), daß um ihr Projekt herum eine Gemeinde entstand, die zu helfen bereit war, aber in einer universellen Währung bezahlt werden wollte. Diese Währung heißt Spaß. Spaß an einem ausgefallenen Bildschirmschoner, Spaß an einem Leistungsvergleich mit anderen Teilnehmern, Spaß am Tüfteln und Basteln an der eigenen Hard– und Software und natürlich auch Spaß an der Kommunikation auch und gerade mit den Wissenschaftlern – das waren die Katalysatoren für den Erfolg von SETI.

fep

Eine "meiner" Partikelwolken; 65 Protonen auf der 44.833-ten Runde im LHC: Ich bin sehr sicher, daß diese Wolke auch in 2007 stabil bleibt. ;=) *fep*

Nijhof, Albert
Die Programmiersprache Forth

mv-Wissenschaft

Drahtkammbindung, Großformat, 130 Seiten ISBN: 3-937312-79-X
Preis: 21,50 EUR

Monsenstein & Vannerdat 2003

Besprechung: Martin Bitter

Vor langer Zeit sang Hannes Wader: „ ... denn, was neu ist, wird alt und was gestern noch galt, stimmt schon heut' oder morgen nicht mehr." Lohnt es sich dann, ein Buch in mühsamer Arbeit zu übersetzen, das fast 11 Jahre alt ist? Zumal, wenn es sich um ein Buch über eine Programmiersprache handelt? Ist nicht der Wandel in der IT-Welt unglaublich schnell?
Die Antwort vorweg genommen: Ja, es lohnt sich!
Fred Behringer hat Albert Nijhofs „De Programeertaal FORTH" (Erstauflage 1992) ins Deutsche übersetzt – und das hat er einfach gut gemacht. Gut gemacht – nicht nur in der kongenialen Art und Weise, wie er Alberts Stil trifft und selbst kleine Wortspiele überträgt, nein: auch gut gemacht – weil er hier einem großen (?) Kreis die Möglichkeit gibt, Alberts vortrefflichen Forth-Kurs kennenzulernen.
Die Qualität einer Arbeit ist abhängig davon, wie sie die Bedürfnisse des Publikums trifft. Deshalb hier der Versuch, den Kreis derer, die einen Nutzen oder (auch nur) Spaß an „Die Programmiersprache Forth" haben könnten, einzukreisen:

Sie kennen bereits einen oder mehrere nicht-ANSI-Forthdialekte und wollen sich vertraut mit dem ANSI-Standard machen.
Sie kennen sich in Ansätzen mit Programmieren aus und möchten einen Kursus durchlaufen, der Ihnen kompakt und unterhaltsam die Sprache „FORTH" näherbringt.
Sie bezeichnen sich selbst als Forthkenner, und haben Spaß daran, zu sehen, auf welche Art andere Forthkenner versuchen, die Grundzüge von Forth zu vermitteln.
Sie möchten einen Forth-Kurs gestalten und brauchen einen Leitfaden oder ein Lehrwerk dazu.

Dann ist dieses Buch genau richtig für Sie.

Wenn ein Buch, ein Vortrag oder ein Kursus dem Publikum nicht gefällt, so kann dies an falsch geweckten Erwartungen liegen. Alberts Buch will nicht zeigen:

wie man in FORTH grafische Benutzeroberflächen bedient.
wie man Windows-, Mac- oder linuxspezifische (X-Server) Anwendungen schreibt.

Dies sind Aufgaben, die dem Handbuch des jeweils erworbenen Forthsystems obliegen.

„Die Programmiersprache Forth" ist aus einem Kursus hervorgegangen. Und so ist das Buch zu Anfang auch aufgebaut; als Kursus. Systematisch kann der Leser sich den Stoff in kleinen Häppchen erarbeiten. Albert dazu: „Der auf den Kurs bezogene Teil des Buches ist didaktisch geordnet, der Rest nicht. Es war nicht meine Absicht, den Leser das Buch Seite für Seite von Anfang bis Ende durcharbeiten zu lassen. Im Gegenteil rate ich ihm, wenn er etwa bei Kapitel 15 angekommen ist und beginnt sich in Forth zu Hause zu fühlen, daß er sich dann im systematischen Teil und in den losen Artikeln umschaut. Je früher man lernt, selbständig in Forth zu denken, desto besser." Dem kann ich nichts hinzufügen!

Methodisch verwendet Albert einen geschickten Wechsel von Vorträgen, Dialogen, ein wenig knappen Humor und kleinen sehr, sehr gut durchdachten Programmbeispielen und Aufgaben. Manchmal wird der Leser geduzt – das ist die nette niederländische Art.
Für Leute die sich auskennen: „Die Programmiersprache Forth" von Albert Nijhof ist ernsthafter als Leo Brodies Werke und nicht ganz so theoretisch (trotzdem schon recht in die Tiefe gehend) wie „Die Programmiersprache Forth" von Zech (In Alberts Werk kommt das Wort 'trivial' nicht vor!).

Alberts Werk ist bei "mv-Wissenschaft" erschienen und kann über den Buchhandel oder bequem via Internet (http://www.mv-buchhandel.de/mv-buchhandel.htm : in das Suche-Feld 'Forth' eingeben) bestellt werden.
Sie sollten - aus welchem Grund auch immer – schon ein Forth auf ihrem Rechner installiert haben. Albert gibt keine Anweisungen, wie dies zu machen ist! Bei diesem Forth sollte es sich um ein 'normiertes' ANSI-Forth handeln. Sind diese Voraussetzungen ein Hindernis für Sie? Keine Sorge, im Internet werden Sie genug Hilfe dazu finden, aber welcher Leser der Vierten Dimension benötigt diese noch?

Martin Bitter

Forth-Gruppen regional

Moers

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

Mannheim

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

München

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

Hamburg

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

Gruppengründungen, Kontakte

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

µP-Controller Verleih

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

Forth-Hilfe für Ratsuchende

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

Spezielle Fachgebiete

FORTHchips
(FRP 1600, RTX, Novix)

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

KI, Object Oriented Forth,
Sicherheitskritische
Systeme

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

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

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

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

Schreiben Sie einfach der VD - oder rufen Sie an - oder schicken Sie uns eine E-Mail!

Hinweise zu den Angaben nach den Telefonnummern:

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

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

20th euroForth conference

on the Forth Programming Language and Forth Processors invites you to participate in the annual get-together of the European Forth Community from Friday, Nov. 19th 2004 until Sunday, Nov. 21st at Dagstuhl Castle, Germany with the traditional 4th day as an option until Monday.

This years motto:

Stack Based Micro- and Nano-Kernels - Forth's Role in Systems-On-A-Chip

We will be inviting a keynote speaker from the silicon industry.

As always, the conference is open to all Forth related topics:
Forth compilation techniques, advances in platform independence, stack based architectures, compilation techniques for stack based architectures, real-time and embedded systems solutions ...

You can contribute by presenting a Paper (20 min), by presenting a Poster, or by hosting a Workshop. If you mail (euro4th@send.de) your paper (< 15 pages) by Nov. 11th in PDF format it will be included in the conference handouts, this is also the deadline for workshop requests. An abstract and your registration is needed by Nov 1st.

And, as always, spontaneous contributions will be given ample space and time.

A limited number of students may participate at a reduced rate.

The venue: http://www.dagstuhl.de/index.en.html

Conference site: http://dec.bournemouth.ac.uk/forth/euro/index.html

I hope to see your there :)

Klaus Schleisiek

This years euroForth office:
Gudrun Zaretzke, c/o SEND GmbH,
Rostocker Str. 20, D-20099 Hamburg
Fon: +49 40 37500803
Fax: +49 40 37500893
Mail: euro4th@send.de