Vierte Dimension

3/2008

Das Forth–Magazin

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

In dieser Ausgabe:

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 Forth–Gesellschaft 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“!

RetroForth
Linux · Windows · Native
Generic · L4Ka::Pistachio · Dex4u
Public Domain
`http://www.retroforth.org`
`http://retro.tunes.org`

Diese Anzeige wird gesponsort von:
EDV–Beratung Schmiedl, Am Bräuweiher 4, 93499 Zandt

Hier könnte Ihre Anzeige stehen!

Wenn Sie ein Förderer der Forth–Gesellschaft e.V. sind oder werden möchten, sprechen Sie mit dem Forth–Büro über die Konditionen einer festen Anzeige.

`Secretary@forth-ev.de`

KIMA Echtzeitsysteme GmbH

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

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

FORTecH Software GmbH
Entwicklungsbüro Dr.–Ing. Egmont Woitzel

Bergstraße 10	D–18057 Rostock
Tel.: +49 381 496800-0	Fax: +49 381 496800-29

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.

Hier könnte Ihre Anzeige stehen!

Wenn Sie ein Förderer der Forth–Gesellschaft e.V. sind oder werden möchten, sprechen Sie mit dem Forth–Büro über die Konditionen einer festen Anzeige.

`Secretary@forth-ev.de`

Ingenieurbüro
Klaus Kohl–Schöpe

Tel.: 07044/908789
Buchenweg 11
D–71299 Wimsheim

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

Impressum

Name der Zeitschrift

Vierte Dimension

Herausgeberin

Forth–Gesellschaft e. V.
Postfach 32 01 24
68273 Mannheim
Tel: ++49(0)6239 9201-85, Fax: -86
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

Bernd Paysan, Ulrich Hoffmann
E–Mail: `4d@forth-ev.de`

Anzeigenverwaltung

Büro der Herausgeberin

Redaktionsschluss

Januar, April, Juli, Oktober jeweils in der dritten Woche

Erscheinungsweise

1 Ausgabe / Quartal

Einzelpreis

4,00€ + Porto u. Verpackung

Manuskripte und Rechte

Berücksichtigt werden alle eingesandten Manuskripte. Leserbriefe können ohne Rücksprache wiedergegeben werden. Für die mit dem Namen des Verfassers gekennzeichneten Beiträge übernimmt die Redaktion lediglich die presserechtliche Verantwortung. Die in diesem Magazin veröffentlichten Beiträge sind urheberrechtlich geschützt. Übersetzung, Vervielfältigung, sowie Speicherung auf beliebigen Medien, ganz oder auszugsweise nur mit genauer Quellenangabe erlaubt. Die eingereichten Beiträge müssen frei von Ansprüchen Dritter sein. Veröffentlichte Programme gehen — soweit nichts anderes vermerkt ist — in die Public Domain über. Für Text, Schaltbilder oder Aufbauskizzen, die zum Nichtfunktionieren oder eventuellem Schadhaftwerden von Bauelementen führen, kann keine Haftung übernommen werden. Sämtliche Veröffentlichungen erfolgen ohne Berücksichtigung eines eventuellen Patentschutzes. Warennamen werden ohne Gewährleistung einer freien Verwendung benutzt.

Liebe Leser,

hier ist es schon, das dritte Heft dieses Jahres. Ich freue mich wiederum, eine schöne Mischung aus Hardware–nahen und high–level–Artikeln präsentieren zu können.

Manfred Mahlow zeigt uns in diesem Heft, wie er seine Forth–GTK+–Anbindung für ein Anzeigeprogramm der vordefinierten graphischen GTK–Objekte (*Stock–Objekte*) einsetzt. Das erleichtert den Umgang mit den Stock–Objekten ungemein.

Für seine CSP–Technik (Context Switching Prelude) benötigt Manfred einen leicht modifizierten Forth–Outer–Interpreter, der bei `prelude`–Worten eben auch ihre `prelude`–Aktion ausführt. Auch `create` (und jedes andere definierende Wort) muss in der Lage sein, die `prelude`–Aktion dem neu definierten Wort zuzuordnen. Um nicht jedes Forth–System, auf dem CSP zu Einsatz kommen soll, eigens anpassen zu müssen, macht es Sinn, über sogenannte *Hooks* nachzudenken. Das sind ganz bestimmte Stellen im Forth–System, die zunächst offengelassen sind, an denen Erweiterungen aber Spezial–Funktionalität einhängen können. Häufig werden solche Hooks durch `Defer`ed–Worte realisiert. Zum Beispiel versucht der Forth–Outer–Interpreter, nachdem er ein Token erfolglos im Dictionary gesucht hat, es in eine Zahl umzuwandeln. Ist auch dies erfolglos, so gibt er eine Fehlermeldung aus (*xlerb?*). An dieser Stelle könnte nun der `notfound`–Hook stehen, der im Standardfall die Meldung ausgibt (`complain`).

Eine Erweiterung, die beispielsweise hexadezimale Zahlen mit einem führenden `$` erkennt, würde nun zusätzlich analysieren, ob das Token die Form `$xx` besitzt, und die entsprechende Zahl erzeugen. Diese Funktionalität würde dann im `notfound`–Hook verankert werden und das Forth–System wäre auf elegante Weise erweitert worden.

Auch Target– und Meta–Compiler können von passenden Hooks profitieren.

Für die CSP–Technik benötigt man zwei Hooks, einen `prolog`–Hook, der im Outer–Interpreter nach dem erfolgreichen Suchen des Tokens im Dictionary aufgerufen wird, und eben einen weiteren Hook in `create` nach dem Lesen des neu zu definierenden Wortnamens und vor dem Anlegen des Headers.

Eine Diskussion über das Für und Wider von Hooks fände ich sehr spannend. Vielleicht hast Du ja dazu auch eine Meinung? Über einen Leserbrief würden wir uns sehr freuen.

Was gibt es noch in diesem Heft zu lesen? Nun, Fred Behringer zeigt uns, wie man mit Forth–Mitteln erfolgreich auf den Bootsektor und die Partitionstabelle von PCs zugreift. Wolfgang Allinger verrät uns das Geheimnis, wie Hardware–Seriennummern auf Basis des DS2401 die Firmware schützen können, und in gleich zwei Grundlagenartikeln erklärt uns Michael Kalus, wie in ANS–Forth `catch` und `throw` funktionieren und wie Gforth Zeitmessungen erlaubt.

Ich wünsche uns allen einen schönen Herbst, viel Spaß beim Lesen und viel Erfolg beim Experimentieren.

Ulrich Hoffmann

Die Quelltexte in der VD müssen Sie nicht abtippen. Sie können sie auch von der Web–Seite des Vereins herunterladen.
`http://www.forth-ev.de/filemgmt/index.php`

Die Forth–Gesellschaft e. V. wird durch ihr Direktorium vertreten:

Ulrich Hoffmann Kontakt: `Direktorium@Forth-ev.de`
Bernd Paysan
Ewald Rieger

Heft 2/2008

Die VD 2/2008 war wohl das interessanteste Forth–Magazin, das ich in den letzten Jahren gelesen habe. Ich habe sie sofort in einem Rutsch durchgelesen.

Besonders gefallen hat mir der Artikel von Carsten Strotmann *Forth ohne/als Betriebssystem* zu den diversen Varianten, einen nackten Rechner mit verschiedensten BIOS–Surrogaten in beliebigste Richtungen zu booten, und der Artikel von Manfred Mahlow *Widgets zum Anfassen* mit seiner Bibliothek zur Nutzung der GTK–Oberfläche in Forth. Beide Themen beschäftigen mich zur Zeit und eine kurze Einführung durch Kollegen mit fundierten Kenntnissen spart da viel Zeit beim Reinfummeln. Eine prima Idee übrigens, den Finanzbericht abzudrucken. Da kann jeder gleich sehen, wohin sein Geld geht.

Damit mich auch die nächste VD erreicht: Könntest Du der Abo–Vertriebsabteilung bitte meine neue Adresse weiterleiten (gerne auch als Leserbrief, dann wissen's auch die alten Kumpanen):

Claus Vogt (wieder in Berlin): Yorckstr. 75-L4, D-10965 Berlin, clv@arcor.de, 030-788 95 036.

Grüße, Claus

In erster Durchsicht: Eine gelungene Ausgabe. Danke an Ulrich und seine Helfer.

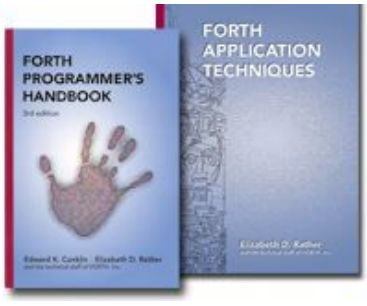
Besonderes Interesse hat bei mir Carsten Strotmanns Artikel *Forth ohne/als Betriebssystem* geweckt. Das hat erkennbar Arbeit gemacht. Vielen Dank für die Aufarbeitung! Was mir dennoch fehlt, ist eine *Entscheidungsmatrix*, die es mir bei meinem sehr begrenzten *Insiderwissen* ermöglicht, mein Interesse auf eine Option zu lenken, die meinen Neigungen (kein Linux) und Kenntnissen (DOS und ZF–Forth werden m.E. nur von HOLON getoppt) entspricht. GRUB kenne ich ja noch. Aber welche Voraussetzungen müsste ich schaffen, um auf meinen etwas älteren Intel–Geräten mit Smart Firmware, Open Firmware oder SLOF (oder andere Optionen) experimentieren zu können? Was muss ich mir selbst erarbeitet haben (oder nachholen) und welche Hardware benötige ich zusätzlich?

Wegen dieser offen gebliebenen Fragen greift mir das etwas *flapsige* Fazit deutlich zu kurz.

Vielleicht gibt es in einer der nächsten Ausgaben nicht nur die Matrix, sondern eine vertiefende Fortsetzung? Darauf würde ich schon heute sehr gespannt warten.

Mit einem freundlichen Glückauf Friederich Prinz

Bücher, um mit Forth zu beginnen



Gavino fragte am 21. August 2008 auf comp.lang.forth: „. . . Wenn ich die Bücher *Starting Forth* (SF) und *Thinking Forth* (TF) durchgearbeitet habe, bin ich dann in der Lage, den Code des Webservers von Bernd Paysan zu lesen? Oder was müsste ich dazu in Forth noch lernen?"

Elizabeth D. Rather antwortete: „Diese beiden Bücher sind schon ziemlich veraltet. Einige der übleren Tücken des SF wurden beseitigt und in der neueren Version des Buches auf `www.forth.com` frei zugänglich gemacht. Aber manch Anachronismus ist doch noch immer darin, besonders die Annahmen darüber, wie Forth–Implementierungen strukturiert sind.

Meine Bücher (1) stellen beide moderne Forth–Systeme und deren Gebrauch dar. Und gehen darüber hinaus auch mehr in die Details. Auch gibt es ein neues Buch, das auf der MPE–Website (2) angeboten wird. . . ."

Ich meine, Rathers Bücher sind gut geeignet, um mit Forth zu beginnen, und dann tiefer einzudringen, und über Amazon auch hier bei uns erhältlich. Vor allem die *Forth Application Techniques*, vom Team der Forth Inc. inzwischen ja schon über Jahrzehnte gepflegt und aktualisiert, und dort benutzt, um deren eigenen Nachwuchs ins Forth einzuweihen, nimmt für sich in Anspruch, kurz und bündig, prägnant, aber dennoch detailliert und Schritt für Schritt an vielen Beispielen, und begleitet von Diskussionen zu Problemen der typischen Techniken des Forth, ein Expertenwissen aufzubauen, arbeitet man es durch.

Das *Programming Forth* von Stephen Pelc behandelt ebenfalls den ANS–Standard und den Umgang mit modernen Forthsystemen, und wurde zuletzt 2004 neu bearbeitet. Es setzt zwar einige allgemeine und grundlegende Programmierkenntnisse voraus, führt in Forth aber von ersten Schritten bis hin zu fortgeschrittenen Techniken — „die in anderen Büchern so nicht behandelt werden", meinte der Autor damals, 2004. Möglicherweise wurde er inzwischen überholt von Frau E. D. Rather — eine kritische Rezension würde ich mir an dieser Stelle wünschen.

Das *Thinking Forth*, mehr eine Philosophie zum Forth als ein technisches Lehrbuch, ist ebenfalls online verfügbar, sogar als Buch–PDF. Kann man sicher auch mal gerne mit Gewinn durchschmökern.

Nach wie vor ist es aber wohl auch nicht so falsch, die allerersten Gehversuche mit dem *Starting Forth* zu machen. Leider gibt es bisher die neuere Version nicht in deutscher Sprache.

In deutscher Sprache verfügbar ist hingegen das Buch von Albert Nijhof (3) und wurde hier im Heft ja auch schon besprochen — Besprechung des Forthbuchs von Albert Nijhof von Martin Bitter; Heft 4d2004–04.

Lehrbücher zum ANS–Forth–Standard:

(1) Forth Application Techniques; Überarbeitete Neuauflage März 2008 - Elisabeth D. Rather Forth Programmer's Handbook; 3rd Edition 2007 - Elizabeth D. Rather, Edward K. Conklin

(2) Programming Forth; revised 2004 - Stephen Pelc et al.

(3) Die Programmiersprache Forth; 2003 - Nijhof, Albert

Quellen:

`http://www.forth.com/forth/forth-books.html`
`http://www.forth.com/starting-forth/`
`http://thinking-forth.sourceforge.net/`
`http://www.mpeforth.com/books.htm`

http://www.mpeforth.com/arena/ProgramForth.pdf
http://www.jwdt.com/~paysan/httpd-en.html
http://www.forth-ev.de/wiki/doku.php/projects:4dinhalt

Viele Grüße, Michael

Interview mit Chuck Moore

Guten Morgen lieber Editor.

Die Zeitschrift Computerworld machte neulich Interviews zu den Programmiersprachen. Und dabei sprachen sie mit keinem Geringeren als Charles H. Moore über Forth. Das Interview hat mir besonders deshalb gefallen, weil Charles Moore es darin verstanden hat, sein Forth so selbstverständlich wie nur was darzustellen. Besonders gefiel mir sein Ausspruch:

Die Beobachtung, dass Forth wie ein Verstärker wirkt, hat mir gefallen: Ein guter Programmierer schreibt damit großartige Programme, ein schlechter ganz scheußliche. Ich habe nicht das Bedürfnis, schlechten Programmierern zu dienen.

Quelle:

http://www.computerworld.com.au/index.php

Interviews: The A–Z of Programming Languages: Forth 27/06/2008 08:01:45.

Charles Moore: *I like the observation that Forth is an amplifier: a good programmer can write a great program; a bad programmer a terrible one. I feel no need to cater to bad programmers.*

Viele Grüße, Michael

Quersummen–Aufgabe: Kürzestes Forth–Programm

Wir kennen alle mindestens zwei literaturbekannte Aufgaben, bei deren Lösung ein genügend langes Durchrechnen zur Erhärtung einer Vermutung und damit zur Ideenbildung für den Beweis beitragen kann: (1) Gibt es unendlich viele Primzahlzwillinge $(p, p+2)$? (2) Gegeben eine natürliche Anfangszahl $m(1)$ und die Vorschrift „$m(n+1) = m(n)/2$ bei geradem $m(n)$ und $m(n+1) = 3 * m(n) + 1$ bei ungeradem m(n)“, endet der Prozess dann immer mit der Schleife $2, 1, 4, 2$, oder gibt es andere Schleifen oder gibt es auch den Fall, dass der Prozess überhaupt nicht endet?

Hier eine Aufgabe von ähnlichem Charakter: Wer schreibt das kürzeste (nicht notwendig schnellste) High–Level–Forth–Programm zur Berechnung der fortgesetzten Quersummen im gleich folgenden Problem oder/und wer liefert einen Beweis dafür, dass bei beliebig vorgegebener natürlicher (d. h. positiv–ganzzahliger) Zahl n von Dreiergruppen der Prozess immer mit einer 1 als *Rest* endet?

Alle Zahlen seien nicht–negative Ganzzahlen in Dezimaldarstellung (mit den Ziffern $0 \ldots 9$). Unter $QS(x)$ sei die Quersumme der Dezimalzahl x zu verstehen, also die Summe der in x auftretenden Dezimalziffern. Natürlich kann man von einer Quersumme (als Dezimalzahl dargestellt) wieder die Quersumme bilden, $QS(QS(x))$, usw.

Vorsicht: Der Begriff der Quersumme ist in dieser Aufgabe ganz stark von der Voraussetzung abhängig, dass die Zahl, von welcher die Quersumme gebildet werden soll, in Dezimaldarstellung vorliegt.

Man betrachte die Reihe:

$$1 \; + \; 2+3+4 \; + \; 5+6+7 \; + \; 8+9+10 \; + \; \ldots$$

Viele würden das *eleganter* mit einem Summenzeichen und einer liegenden Acht darüber schreiben — und sich vielleicht was dabei denken. In Anlehnung an Goethe: „Wo sich eine mathematische Formel hinschreiben lässt, müsste doch eigentlich auch ein mathematischer Begriff dahinterstecken?“ Ja, schon — aber:

Diese Reihe (ausgedrückt durch das ...) ist natürlich divergent (hat gar keine *Summe*). Das soll uns aber wenig kümmern. Wir interessieren uns nur für die wie folgt aufeinanderfolgenden Partialsummen:

$$\begin{aligned}
S0 &:= 1 \\
S1 &:= 1 \; + \; 2+3+4 \\
S2 &:= 1 \; + \; 2+3+4 \; + \; 5+6+7 \\
S3 &:= 1 \; + \; 2+3+4 \; + \; 5+6+7 \; + \; 8+9+10 \\
S4 &:= 1 \; + \; 2+3+4 \; + \; 5+6+7 \; + \; 8+9+10 \\
&\quad + \; 11+12+13 \\
S5 &:= 1 \; + \; 2+3+4 \; + \; 5+6+7 \; + \; 8+9+10 \\
&\quad + \; 11+12+13 \; + \; 14+15+16 \\
&\ldots \\
&\ldots \\
Sn &:= 1 \; + \; 2+3+4 \; + \; 5+6+7 \; + \; 8+9+10 \\
&\quad + \; 11+12+13 \; + \; 14+15+16 \; + \ldots \\
&\quad + \; 3*(n-1) + 3*(n) + 3*(n+1)
\end{aligned}$$

und so weiter, für alle natürlichen Zahlen n. Man bilde

$$\begin{aligned}
QS(S0) &= QS(1) &&= 1 \\
QS(S1) &= QS(10) &&= 1 \\
QS(S2) &= QS(28) &&= 10 \quad QS(10) = 1 \\
QS(S3) &= QS(55) &&= 10 \quad QS(10) = 1 \\
QS(S4) &= QS(91) &&= 10 \quad QS(10) = 1 \\
QS(S5) &= QS(136) &&= 10 \quad QS(10) = 1 \\
&&&\ldots \\
QS(S17) &= QS(1378) &&= 19 \quad QS(19) = 10 \quad QS(10) = 1 \\
&&&\ldots
\end{aligned}$$

Behauptung: Die fortgesetzten Quersummen enden bei beliebig vorgegebener natürlicher Zahl n immer mit $QS(10) = 1$.

(Dass der Prozess bei beliebig vorgegebenem n irgendwann mal abbrechen muss, halten wir alle noch für unmittelbar klar und selbstverständlich. Woher nehmen wir aber die Gewissheit, dass die vorletzte Quersumme bei genügend großem n nicht mal beispielsweise $QS(17) = QS(8) = 8$ ist? Wir können ja nicht bis in alle Ewigkeit — die Ewigkeit ist ein theologischer Begriff — abzählen und durchrechnen. Auch ein künftiger Quantencomputer nicht. Hier wird kein Computer, hier wird kein Rechnen, hier wird Überlegung verlangt. Der Computer kann uns aber dabei helfen, einen Weg zum Beweis zu finden. An welcher Stelle im gesuchten Forth–Programm zur Berechnung der fortgesetzten Quersummen erleichtert die obige Behauptung, wenn sie stimmt, das Verfahren?)

PS: Zum gesuchten High–Level–Forth–Programm gehört natürlich auch die Aufgabe, die Quersumme einer Dezimalzahl zu berechnen. Eine Vorstufe dazu wäre sicher, die Anzahl der Einsen in einer Binärzahl zu finden. Zu diesem Thema hat es in der VD (und in der Forthwrite) mal einen Artikel gegeben. Aber mit *High–Level* war da nicht durchzukommen.

Fred Behringer

Syntaktischer Zuckerguss im Beitrag Euler 9 aus Heft 2/2008

Ich wurde gefragt woher das Wort `?EXIT` stammt. Es ist dem gforth entnommen und compiliert ein bedingtes `EXIT`.

```
: ?EXIT  ( flag -- )
    postpone IF
    postpone EXIT
    postpone THEN ; immediate
```

Die Syntax ist am `?DO` oder `?LEAVE` angelehnt. Es hat sich so eingebürgert, Forthworten, die ein Flag verbrauchen, ein `?` voran zu stellen. Im Falle eines `TRUE` tun sie was, sonst nicht. Es ist syntaktischer Zuckerguß, sonst nichts. Im Unterschied dazu dienen nachgestellte `?` schon mal dazu, ein Flag zu liefern. Wie im Fall des `KEY?` das dann ein `TRUE` liefert, falls eine Taste gedrückt worden ist.

Vielleicht hätte ich in dem Beitrag auch erwähnen sollen, das gforth ja mit cell=4 läuft, also alle Integer–Arithmetik daher sowieso schon in 32 Bit ausgeführt wird. Mit einem 16-Bitter geht das Beispiel möglicherweise schief und benötigt eine Umstellung auf 32-Bit–Arithmetik, also z. B. das Double–Number–Wordset `d-number.fs` von Luca Masini. (siehe: `http://www.forth-ev.de/wiki/doku.php/` oder seinen Beitrag im VD Heft 4d2008-01.)

Michael

Googelt mal wieder nach *forth*

Googelt mal wieder nach *forth* - was man da so alles findet :-)

`http://www.forth.gr/`

Die Scheibe von Phaistos wurde 1908 bei den Ruinen des frühen Minoischen Palastes von Phaistos auf Kreta, Griechenland, gefunden. Sie stammt vermutlich aus dem 17. Jahrhundert vor Christus. Angefertigt aus Ton misst sie 16cm im Durchmesser und ist 2cm dick. Sie gilt als das älteste erhaltene Druckerzeugnis der Menschheit. Ihre Inschrift ist bisher nicht entziffert und daher eine besondere Herausforderung an die Wissenschaft. Darum hat das FORTH–Institut in Hellas (The Foundation for Research and Technology — Hellas) sie als Wappen angenommen. Leider ist deren Onassis–Foundation noch nicht auf uns, die Forth–Gesellschaft, aufmerksam geworden.

Michael

Eine ganz andere Lösung für das Problem Euler 9

Ihr müsst wissen, auf dem College besuchte ich den Mathematik–Hauptunterricht. Mathe interessierte mich immer schon. Und von dem einen der wenigen Mathematiker hier in der SVFIG (Silicon Valley Forth Interest Group) erhielt ich einen Tip für zwei Bücher von Roger B. Nelsen *Proof Without Words*, einem Professor am Lewis & Clark College, der die Bücher so um 1993 geschrieben hat — sie sind noch immer im Handel (Amazon). Es ist eine Sammlung verschiedener grafischer und optischer Beweise, von unterschiedlichsten, teil historischen, aber auch modernen Autoren wie Euklid, Dudeney, und natürlich Nelsen selbst. Ich hatte das Buch fast schon vergessen, und war nun glücklich, auf der Seite 141 des ersten Bandes den Titel *Pythagorean Triples via Double Angle Formulas* von David Houston vorzufinden. Dort sieht man zwei Zeichnungen rechtwinkliger Dreiecke. In der ersten ist die kurze Seite mit n bezeichnet, die lange mit m und der Winkel zwischen m und der Hypotenuse mit θ. m und n sind ganze Zahlen $m > n > 0$ und zwei Ausdrücke stehen daneben:

$$\sin\theta = \frac{n}{\sqrt{m^2+n^2}}$$

$$\cos\theta = \frac{m}{\sqrt{m^2+n^2}}$$

Das ist mir klar.

Das zweite Dreieck ist auch rechtwinklig, aber seine Seiten wurden so benannt: $2mn$, m^2-n^2, und m^2+n^2.

Die Seite $2mn$ liegt vertikal gegenüber dem Winkel 2θ, der von der anderen Seite und der Hypotenuse gebildet wird. Nun behauptet der Autor Folgendes:

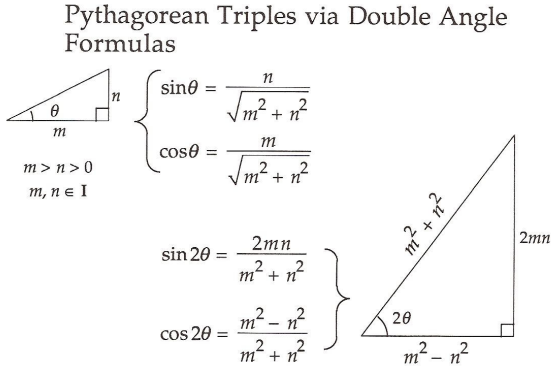
$$\sin 2\theta = \frac{2mn}{m^2+n^2}$$

$$\cos 2\theta = \frac{m^2-n^2}{m^2+n^2}$$

was aus den Bezeichnungen der Seiten offensichtlich ist.

Alles, was ich tun musste, war $n = 1$ zu setzen und probeweise $m = 2$, 3 und 4 in diesem zweiten Dreieck, um zum Tripel 8, 15, 17 zu kommen, um zu erkennen, dass $8+15+17 = 40$ ist. Und weil $40 = 1000/25$ ist, folgt, dass schon das dritte Tripel multipliziert mit 25 die Lösung für Euler 9 darstellt, also 200, 375, 425.

Viele Grüße, Henry (Übersetzung: M. Kalus)



GTK+ mit Forth (1) - GTK Stock Viewer

Manfred Mahlow

In der VD 02/2008 hatte ich eine GTK+-API für (csp)Forth vorgestellt. Hier folgt nun ein erstes Anwendungsbeispiel mit einem gewissen praktischen Nutzen, ein Viewer für den GTK Stock.

Der GTK Stock

Der GTK Stock ist eine Sammlung von vordefinierten graphischen Objekten, *GTK Stock Items* genannt. Die Stock Items sind Kombinationen von Minibildern (Icons), Beschriftungen, Abkürzungsbuchstaben und Tastenkombinationen, die bevorzugt für Menüeinträge, Dialoge und Knöpfe verwendet werden sollen, um GTK+-Benutzerschnittstellen ein einheitliches Aussehen zu geben. Auf die Stock Items kann man über eindeutige Bezeichner, die *Stock Ids*, zugreifen.

Will man ein bestimmtes GTK Stock Item verwenden, muß man seinen Bezeichner kennen. Um dazu nicht immer wieder die GTK+-Dokumentation wälzen zu müssen, bietet es sich an, ein kleines Tool zu schreiben, mit dem man den Gtk Stock durchstöbern kann.

Ein GTK Stock Viewer

In einer der GTK+-Bibliotheken, der *libgtk*, gibt es eine Funktion *gtk_ stock_ list_ ids*, die einen Zeiger auf eine Liste aller verfügbaren GTK Stock Items zurückgibt. Man könnte diese Liste einer *GtkComboBox* zuweisen. Mit dieser könnte man die Liste anzeigen und ein Listenelement auswählen, das dann als *GtkImage* und als *GtkButton* angezeigt werden könnte.

Das Ergebnis dieser Überlegungen ist der GTK Stock Viewer in Abbildung 1.

Der Code

Der cspForth-Code, der den Viewer auf den Bildschirm zaubert, ist im Listing 1 zu sehen.

Der Code in den Zeilen 3 bis 9 stellt sicher, dass alle benötigten GTK+-Widget-Klassen geladen werden (einmaliges Laden der Quelltextmodule). Außerdem wird die Klasse *GSList* benötigt, die einfach verknüpfte Listen der libglib zur Verfügung stellt (Zeile 11).

In Zeile 19 wird die bereits erwähnte Funktion *gtk_ stocklist_ ids* aus der *libgtk* in das Vokabular *gtk api* importiert.

Der Viewer ist ein Beispiel für die Verwendung der Klasse *GtkComboBoxText*. Für den Viewer legen wir deshalb ein privates Vokabular im Kontext (in der Wortliste) dieser Klasse an (Zeile 22 u. 24). Privat bedeutet hier, dass das Vokabular für Objekte der Klasse nicht sichtbar ist.

In den Zeilen 26 bis 34 werden die für den Viewer benötigten Objekte definiert:

GtkToplevel Ein GtkWindow des Typs GTK_WINDOW_TO

GtkTable Ein Container-Widget, in das andere Widgets in Zeilen und Spalten gepackt werden können.

GtkLabel Array Ein Array, dessen Elemente Objekte der Klasse *GtkLabel* (Beschriftungen) sind.

GtkAlignment Array Ein Array, dessen Elemente Objekte der Klasse *GtkAlignment* sind. Ein Objekt der Klasse GtkAlignment ist ein GtkContainer-Widget, in das man ein anderes Widget packen kann, das horizontal und vertikal ausgerichtet werden kann.

GtkCombBoxText Ein *GtkButton* (Knopf), der mit einer Liste von Textelementen verbunden ist, die sich auf Knopfdruck öffnet und aus der man genau ein Element auswählen kann. Das ausgewählte Element wird als Knopfbeschriftung angezeigt.

GtkImage Ein GtkWidget zur Anzeige eines Bildes, dem das Icon des ausgewählten GtkStockItems zugewiesen werden soll.

GtkButton Ein Knopf, der mit den Eigenschaften des ausgewählten GtkStockItems dargestellt werden soll.

GSList Eine einfach verknüpfte Liste für die Handhabung der GTK Stock Ids.

In Zeile 37 wird ein Signalhandler *cb.quit* definiert, der aufgerufen werden soll, wenn das Hauptfenster *window* geschlossen wird.

Abbildung 1: GTK Stock Viewer unter Ubuntu 6.06

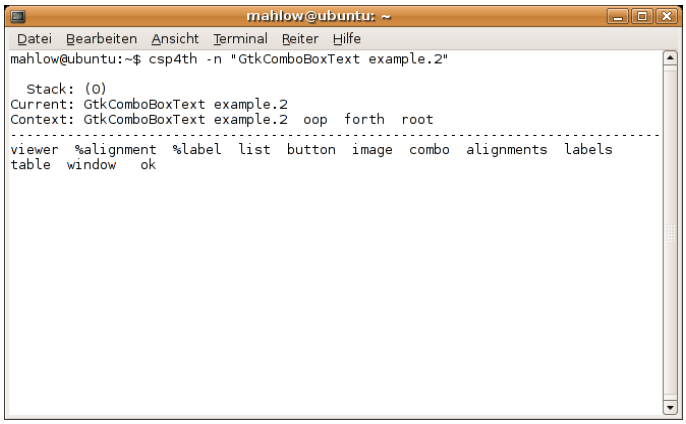


Abbildung 1: Status nach dem Laden von Listing 1

In den Zeilen 39 bis 42 wird ein Signalhandler *cb.changed* definiert, der ausgeführt werden soll, wenn mit der GtkComboBox ein Listenelement ausgewählt worden ist. Der Handler liest das ausgewählte Element aus der GtkComboBox *combo* (Zeile 40) aus und weist es dem GtkButton *button* und dem GtkLabel *label* zu.

Die Zeilen 44 u. 45 legen Optionen fest, mit denen die Elemente des GtkLabel-Arrays und des GtkAlignment-Arrays in das GtkTable-Widget gepackt werden sollen.

In Zeile 49 beginnt die Initialisierungsroutine des Viewers. Sie initialisiert die in den Zeilen 26 bis 34 definierten Objekte, erzeugt die Widgets, packt sie in ihre Container und zeigt dann das Hauptfenster an.

In Zeile 49 wird das Hauptfenster des Viewers initialisiert. Die *init*-Methode erzeugt ein GtkWindow vom Typ GTK_TOPLEVEL_WINDOW und weist diesem den String, der auf dem Stack übergeben wird, als Titel zu.

In Zeile 50 erhält das Fenster einen inneren Rand von 12 Pixel und es wird festgelegt, dass die Fenstergröße vom Anwender nicht geändert werden kann.

In Zeile 51 werden das *destroy*-Signal und der in Zeile 37 definierte Callback-Handler *cb.quit* an das Hauptfenster gebunden. Damit wird festgelegt, dass *cb.quit* ausgeführt wird, wenn das Fenster vom Fenstermanager ein *destroy*-Signal erhält.

In Zeile 53 wird das GtkTable-Widget *table* initialisiert und in das Hauptfenster *window* gepackt. Anschließend wird in Zeile 54 der Spalten- und Zeilenabstand auf 8 Pixel festgelegt.

Die Tabellengröße muss nicht angegeben werden. Sie wird automatisch angepasst, wenn Widgets in die Tabelle gepackt werden.

In den Zeilen 56 bis 60 werden die Beschriftungen (GtkLabels) initialisiert und in die erste Spalte des GtkTable-Widgets gepackt. Die drei Beschriftungen sind Elemente des Arrays *labels,* das in Zeile 28 definiert wurde.

Die Initialisierung des Arrays erfolgt in zwei Schritten. Zuerst wird seine Größe festgelegt (Zeile 56), danach werden dann seine Elemente initialisiert (Zeile 57-59).

Die *init*-Methode der Klasse GtkLabel erwartet den Text, der als Beschriftung angezeigt werden soll, als String auf dem Stack. Sie erzeugt ein GtkLabel-Widget und gibt dessen Adresse - in Forth *widget identifier (wid)* genannt - auf dem Stack zurück. Diese Adresse wird von der *attach*-Methode der Klasse GtkAlignment verwendet, um das GtkLabel-Widget in die erste Spalte des GtkTable-Widgets *table* zu packen.

In Zeile 60 werden dann die Beschriftungen in der ersten Spalte des GtkTable-Widgets *table* rechtsbündig ausgerichtet.

In den Zeilen 62 u. 63 wird das in Zeile 29 definierte GtkAlignment-Array initialisiert. Seine drei Elemente (GtkAlignment-Widgets) werden in die zweite Spalte des GtkTable-Widgets *table* gepackt. In die GtkAlignment-Widgets werden dann das GtkComoBox-Widget *combo*, das GtkImage-Widget *image* und das GtkButton-Widget *button* gepackt (Zeile 64-67) und linksbündig ausgerichtet (Zeile 68).

In den Zeilen 70 bis 74 wird die von der Funktion gtk_stock_list_ids übergebene Liste der GTK Stock Ids in die GtkComboBox eingetragen.

Alle Widgets sind nun erzeugt, aber noch nicht zu sehen. Sie werden durch den Code in Zeile 77 sichtbar gemacht.

In Zeile 84 wird der Viewer gestartet. Wurde der cspForth-Prozess, der den Code ausführt, in einem Terminal gestartet, zeigt das Wort *??* den aktuellen Zustand des Forth-Systems an (Abbildung 1). Mit cspForth kann dann interaktiv gearbeitet werden und man kann auf die Widgets des Viewers zugreifen. Die Verarbeitung der Gtk-Ereignisse erfolgt im Hintergrund, während auf die Eingabe einer Kommandozeile gewartet wird.

Ist der cspForth-Prozess nicht mit einem Terminal verbunden, gibt es keine Ereignisverarbeitung im Hintergrund. Es muss dann eine GTK+-Hauptschleife gestartet werden ([ELSE]-Zweig in Zeile 84), die auf Ereignisse wartet und dann den zugeordneten Code ausführt. Sie wird erst wieder verlassen, wenn der Callback-Handler cb.quit beim Schließen des Hauptfensters ausgeführt wird.

Literatur

[1] GTK+-Dokumentation. http://www.gtk.org

[2] Matthias Warkus: GNOME 2.0 - Das Entwickler-Handbuch. Galileo Press, 2003

[3] Manfred Mahlow: Widgets zum Anfassen - GUI-Skripting mit Forth und GTK+. Vierte Dimension 2/2008

Listing 1:

```
 1  \ GtkComboBoxText/example.2.4th
 2
 3    needs GtkToplevel
 4    needs GtkTable
 5    needs GtkLabel Array
 6    needs GtkAlignment Array
 7    needs GtkComboBoxText
 8    needs GtkImageFromStock
 9    needs GtkButton
10
11    needs GSList
12
13  \ ------------------------------------------------------------------------
14  \  csp4th :  GtkComboBox Example 2                              MM-080311
15  \ ------------------------------------------------------------------------
16
17    gtk api definitions
18
19    libgtk import gtk_stock_list_ids ( -- *list )
20
21
22    GtkComboBoxText definitions
23
24    private vocabulary example.2  self example.2 definitions  also gtk api
25
26    GtkToplevel        new window
27    GtkTable           new  table
28    GtkLabel Array     new   labels
29    GtkAlignment Array new   alignments
30    GtkComboBoxText    new    combo
31    GtkImage           new    image
32    GtkButton          new    button
33
34    GSList new list
35
36
37    :: ( wid data -- ) 2drop gtk quit ;  2 20 cb cb.quit
38
39    :: ( wid data -- )
40       2drop combo @ dup if
41         2dup image from-stock button-size !  button label !
42       then ;  2 20 cb cb.changed
43
44    : %label ( -- xopts yopts xpad ypad )     GTK_FILL 0  0 0 ;
45    : %alignment ( -- xopts yopts xpad ypad ) GTK_FILL 0 12 0 ;
46
47    : viewer ( -- )
48
49        " Gtk Stock Item Viewer" window init
```

```
50        12 window border-width !   window resizable off
51        " destroy" cb.quit 0 window signal connect drop
52
53        table init  window add
54        8 table column-spacing !  8 table row-spacing !
55
56        3 labels init
57        " Stock Item:" 0 labels of init  0 1 0 1 %label table attach
58        " Image:" 1 labels of init       0 1 1 2 %label table attach
59        " Button:" 2 labels of init      0 1 2 3 %label table attach
60        3 0 do i labels of xalign right loop
61
62        3 alignments init
63        3 0 do i alignments of init  1 2 i dup 1+ %alignment table attach loop
64        combo init  0 alignments of add
65        " changed" cb.changed 0 combo signal connect drop
66        " ?" image from-stock button-size init  1 alignments of add
67        " ?" button init  2 alignments of add
68        3 0 do i alignments of xalign left loop
69
70        gtk_stock_list_ids list init
71          list size dup 1- swap 0 do
72            dup i - list of @ dup zcount combo append  free
73          loop drop
74        list free
75        0 combo activate
76
77        window show all
78    ;
79
80    hide cb.quit  hide cb.changed
81
82    previous
83
84    viewer term? [IF] ?? [ELSE] gtk main bye [THEN]
85  \ ------------------------------------------------------------------------
86  \ Last revision: MM-080525
```

Salvador Dali: *Profil der Zeit*
Die Bronze ist im *Arkady Wrocławskie* Einkaufszentrum in Breslau ausgestellt.
(Photo: Wikimedia/Julo, 4/2008)

Lebenszeichen

Bericht aus der FIG Silicon Valley: *Henry Vinerts*

Dear Fred,

ich dachte, dass dir vielleicht ein paar Zeilen von mir gefallen würden, die die lange Stille seit meinem letzten Brief an dich nach dem SVFIG–Treffen im April dieses Jahres unterbrechen sollen. Aus der neuen VD–Ausgabe 2/2008 ersehe ich, dass du wieder als bereitwilliger und sachkundiger Übersetzer gewirkt hast. Ich möchte dir kurz erklären, warum ich über eine Woche gebraucht habe, um dir zu danken — zu danken auch allen, die dazu beigetragen haben, wieder eine schöne Ausgabe der meines Wissens einzigen Forth–Zeitschrift, die noch veröffentlicht wird, herauszubringen.

Zur Beruhigung meines schlechten Gewissens kann ich mindestens drei Leute nennen, deren Arbeiten mich gefesselt und damit zur Verzögerung beigetragen haben — nicht zu vergessen natürlich diejenigen, die das Heft 2/2008 zusammengestellt und es übers Meer hinweg in meine Hände gelegt haben. Speziell erwähnen möchte ich Bernd Paysan, den Schöpfer von *Das Forth–e.V.–Wiki* (siehe VD–Heft 3/2006, S.35). Sodann Michael Kalus, den Autor der Forth–Lösung zum Projekt Euler, Problem 9 (welches in der neuen VD–Ausgabe wie auch auf der zitierten Web–Site erscheint). Und erwähnen möchte ich schließlich dich, Herr Professor, der unendliche Geduld mit einem der schlechtesten Forth–Studenten aufbringt, die es je gegeben hat, nämlich mit mir.

Nach einem kurzen Blick auf die neue VD–Ausgabe war mir sofort klar, dass mich das Problem aus dem Projekt Euler noch eine Weile fesseln wird, solange, bis ich eine Lösung (soweit eine solche existiert?) finde. Als Erstes habe ich natürlich meine alte DOS–Maschine angeworfen und mit dem TURBO–Forth v.3.0, das du mir vor drei Jahren geschickt hast, versucht, Michaels Programm zum Laufen zu bringen. Dazu hatte ich mir eine ganze Menge Zeug ins Gedächtnis zu rufen, das in meiner Erinnerung schon längst eingerostet war. Es hat eine geraume Zeit gedauert, bevor ich ohne 32–Bit–Operationen das Pythagoreische Tripel 20, 21, 29 erreichte. Hierzu ersetzte ich in Michaels Programm in der Variablen xx den Wert 1000 durch 70 und strich die Zeile mit `?EXIT`, das es in Turbo–Forth nicht gibt, indem ich überall, wo nötig, `70 1 DO ... LOOP` einsetzte.

Ich werde wohl kaum dahinterkommen, wie ich mit den Mitteln, die mir zur Verfügung stehen, 32–Bit–Mathematik betreiben kann, und du wirst wohl erst im November wieder von mir hören, wenn SVFIG aller Voraussicht nach die jährliche Forth–Tagung abhält. Wenn man sich die Berichte über die SVFIG–Treffen unter `www.forth.org` (`http://www.forth.org`) ansieht, wird man erkennen, dass sich diesen Sommer in unserer Gruppe wenig getan hat. Cogswell College war während der Ferien geschlossen und wir hatten uns nach einem anderen Versammlungsort umzusehen. Wie man hört, gibt es nur ganz wenige Vortragende, die bereit wären, auf den Versammlungen zu sprechen (mit Ausnahme natürlich von Ting, wenn er gerade im Land ist). Und wenn es anderen genauso geht wie mir, dann ist man in der heutigen Zeit wegen der hohen Benzinkosten beim Fahren zu weiter Strecken sehr zurückhaltend.

Wie ich sehe, ist es nur noch einen Monat hin bis zur EuroForth–Konferenz in Wien und ihr seid sicher mit den Vorbereitungen beschäftigt. Weil ich gerade daran denke, möchte ich euch gern noch ein paar Fragen stellen. Ihr braucht sie natürlich nicht sofort zu beantworten! Zunächst einmal eine Kostenfrage: Ich weiß, dass es einen Haufen Geld kostet, mir jedesmal ein VD–Heft in Papierform zu schicken. Ich schätze das sehr, aber es würde mir auch genügen, die Hefte auf der Website `http://www.forth--ev.de` zu lesen — wenn sie dort eingesehen werden könnten. Wie ich sehe, sind dort nur ältere Ausgaben aufgelistet. Gibt es irgendwelche Pläne, auch die laufenden Ausgaben dort elektronisch zur Verfügung zu stellen?

Sodann wäre ich neugierig zu erfahren, ob meine Vermutung richtig ist, dass auf dem Foto auf Seite 24 in Heft 2/2008 Michael Kalus der Forth–Freund mit dem Bart ist, der in der zweiten Reihe von deiner rechten Schulter aus gesehen der Nächste ist? Ich würde mich jedenfalls dafür interessieren zu erfahren, welche Personen *aus Fleisch und Blut* hinter den Namen stecken, die in der VD erwähnt werden. Ich habe eine gewisse Zeit damit verbracht, den Autor von Turbo–Forth zu suchen, und ärgerte mich schließlich über mich selbst und über mein schlechter werdendes Gedächtnis. War es Marc Petremann? Oder war Marcel Hendrix auch Autor? Welches Forth hat Michael für seinen Artikel über das Euler–Projekt verwendet? Ist `?EXIT` ein ANS–Forth–Wort, und wenn nicht, würde `?LEAVE` aus Turbo–Forth dem `?EXIT` entsprechen?

Ist es sehr wahrscheinlich, dass die letzten beiden VD–Ausgaben als Doppelheft erscheinen? Wenn es dann für mich nicht zu spät ist, Anfang Dezember einen *Bericht* zu schicken, würde ich alle meine Anstrengungen dareinsetzen, zu unserer Forth–Tagung zu fahren und darüber zu berichten. Aber, Fred, mach dir nicht zu viele Mühe, alles zu übersetzen, was ich hier schreibe. Ich wollte dir und den anderen deutschen Forthfreunden eigentlich nur danken und euch wissen lassen, dass es im Silicon Valley auf jeden Fall mindestens einen gibt, der die europäischen Forth–Aktivitäten mit Interesse verfolgt.

Mit meinen allerbesten Grüßen euch allen,

Henry Übersetzt von Fred Behringer

Lieber Henry,

vielen Dank für deine freundlichen Worte. Deine Vermutung über den *Mann mit dem Bart* ist richtig und du hast

ja inzwischen mit Michael Kontakt aufgenommen. `?EXIT` ist (meines Wissens) kein ANS–Forth. Michael verwendet gForth. gForth bietet wesentlich mehr als *nur* ANS–Forth. Turbo–Forth enthält kein `?EXIT`. Aber man kann sich ja in jedem Forth–System die meisten der fehlenden Worte, ob ANS oder nicht, ganz schnell selbst machen. In Hinsicht auf ANS–Forth leistet für F83–Derivate, wie Turbo–Forth, der zweiteilige Artikel von Willem Ouwerkerk aus dem Vijgeblad 42 und 44 (1993/94) große Hilfe. In Turbo–Forth `?LEAVE` für `?EXIT` einzusetzen, wäre falsch. Die übliche Forth–Syntax mit dem vorangesetzten Fragezeichen lautet: `IF EXIT THEN` — und du würdest gut daran tun, `?EXIT` einfach zu `IF EXIT THEN` *auszuschreiben*. Die Frage wegen der elektronischen Weiterleitung der VD–Hefte an dich gebe ich hiermit an das Direktorium weiter. Und deine Frage in Bezug auf das Doppelheft hat sich, wenn du diese Zeilen liest, erledigt. Die Redaktion bemüht sich, wie ich als Außenstehender beobachten kann, ständig, aber ohne die Mithilfe der Mitglieder (und auch der VD–lesenden Nicht–Mitglieder!) als Autoren ist alle Mühe vergeblich.

Herzlichen Gruß bis zum nächsten Mal,

Fred

Hallo Henry und Fred,

ihr fragt nach den online–Versionen der aktuellen Ausgaben der Vierten Dimension? Nun, alle Jahrgänge von 1984 bis 2008, derzeit sind das insgesamt 102 Hefte, sind verfügbar unter `http://www.forth-ev.de/filemgmt/viewcat.php?cid=2`.

`?EXIT` geht übrigens auf Klaus Schleisiek zurück und lässt sich ganz einfach definieren, wie von Michael auf Seite 7 erklärt.

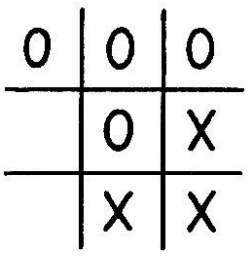
Viel Spaß beim Schmökern,

Ulli

Gesucht: Tic Tac Toe in Forth

Ulrich Hoffmann

Im Alltag trifft man auf immer mehr mobile Geräte — seien es iPods, Handys, Palm Pilots, Eee PCs, Nokia Internet Tablets, *One Laptop Per Child*–XO Laptops oder ähnliche —, die sich im Prinzip mit Forth programmieren lassen. Damit wir vorführen können, dass es in Forth ganz leicht ist, für diese Geräte Applikationen zu entwickeln, benötigen wir eine schöne Beispielanwendung. Welche könnte das sein? Auf der Suche nach der Antwort auf diese Frage kam mir wieder das Spiel Tic Tac Toe (`http://de.wikipedia.org/wiki/Tic_Tac_Toe`) in den Sinn.



Es wird also ein Forth–Programm gesucht, das Tic Tac Toe spielen kann. Das Programm soll das Spielfeld verwalten und auch einen Computer–Gegner realisieren. Von Vorteil wäre es, wenn Programmlogik und Ein- und Ausgabe gut voneinander getrennt wären, damit das Programm leicht an unterschiedliche Systeme angepasst werden kann. Eine Terminal–Ein-Ausgabe sollte als Beispiel realisiert sein. Portierungen auf mobile Geräte sind natürlich auch sehr willkommen.

Wer traut sich an dieses Problem? Schickt Eure Lösungen bitte an die Redaktion der Vierten Dimension (vd@forth-ev.de). Wir werden in den kommenden Ausgaben über die Einsendungen berichten.

Viel Erfolg!

Ulli

Reverse–Engineering–Preventer mit DS2401

Wolfgang Allinger

Ein *Reverse–Engineering–Preventer* mit DS2401 und Dallas–One–Wire–Worten für den 8031

Übersicht

Das Dallas–One–Wire–Protokoll (DOW) wird hier benutzt, um eine eindeutige Seriennummer aus dem DS2401 zu lesen. Es gibt weitere DOW–Bausteine, u. a. für Temperaturmessung, I/O–Signale, RAM, Uhr, ADC und DAC ...

Warum die Mühe?

In diesem Fall sollten Kopien der Applikation verhindert werden. Den DS2401 gibt es in einem TO92–Gehäuse, und er sieht dann wie ein normaler Transistor (EBC) aus. Aber nur die zwei Anschlüsse *Basis* und *Emitter* sind aktiv, der *Kollektor* ist n/c (so eine Art Holzbein :-).

Wenn man die *Basis* an einen bidirektionalen 8031–Port–Pin, den *Emitter* an GND und den *Kollektor* an irgendeinen anderen Pin verdrahtet, kann man die meisten Hacker schön zum Narren halten.

Entsprechende Abfragen beim Start mit viel Dummy `ROT PICK TUCK SWAP DROP TUCK` n `ROLL`, `M/MOD` (Plutimikation ;-) ... garniert, irgendwo dabei tiefer im Stack die Nummer über n `PICK` abfragen und dann nach Start der Anwendung und einer Zufallszeit gegen eine codierte Kopie im EPROM vergleichen, und dann lapidar mit „EEPROM–Datenfehler"–Meldung anhalten. Hat prima funktioniert, zumal ich die Beschriftung vom DS2401 abgeschliffen habe und der in der Stückliste und im Schaltplan als NPN–Transistor aufgeführt worden war.

Ich bin nicht böse, nur der Umgang mit *Geiz–ist–geil–Kunden* hat mich das erfolgreich gelehrt. Man musste allerdings zum 1. Start noch eine Routine ins RAM kompilieren, die dann die aktuelle Nummer im Target auslas und in das Image für das EPROM schrieb. Das geht heutzutage mit geflashten Programmen leichter.

DOW–Bus–Prinzip

Der DOW–Bus ist als Open–Collector–Leitung zu sehen, die von einem Pullup hochgezogen wird. Alle Teilnehmer sind an dieser einzigen (neben GND) Leitung angeschlossen. Die Teilnehmer werden durch High–Pulse und einen damit geladenen internen Kondensator gespeist.

Nach einem High–Puls vom DOW–Master zieht der aktivierte Slave für eine gewisse Zeit den Bus auf Masse oder auch nicht. Je nachdem wird die Antwort als 0 oder 1 interpretiert.

Ein 8031–Port im bidirektionalen Mode ist ideal als BUS–Master. Man braucht keinen externen Pullup–Widerstand, da der 8031 einen weak–pullup (ca. 20kΩ) hat, der beim Wechsel von L nach H für nur 2 Taktzyklen niederohmig kräftig nach High gezogen wird. Den dann folgenden schlappen High–Pegel können die Slaves dann leicht herunterziehen.

Der DS2401 schickt seine Seriennummer, die eindeutig (unique) ist, als 8 Byte einschließlich einer CRC8 zurück.

In dieser Applikation schicke ich immer entweder das `RD`–Kommando (33h) oder eine Dummy–Date (FFh) zum Bus, aus dem unteren Bit heraus, und kopiere dann die Antwort ins untere Bit zurück, und rotiere das solange, bis alle 8 Bit ausgetauscht sind. Also ein spezielles Software–UART.

Beim 8031 kann man sehr geschickt das CY-Bit (carry) auch komplementär mit einem Portpin verwursten, was ich auch ausnutze. Der 8031 hat neben der normalen 8–Bit–ALU (arithmetic logic unit = Recheneinheit) noch eine 1-Bit–ALU, das ist das CY-Bit.

Das genaue Zeitverhalten und die sonstigen Eigenheiten dieses DOW bitte im Dallas–Handbuch nachlesen. Das einzige Übel dabei ist, dass Dallas vor einiger Zeit von Maxim geschluckt wurde.

Anmerkung

Der nachfolgend wiedergegebene Code ist aus einer Anwendung von 1995 herauskopiert und editiert worden. Ich hab alles Spezielle dieser Anwendung weggelassen, habe es aber nicht getestet. So ist es einfach eine Info, aber ich hoffe, dass diese Info nützlich ist.

Die Applikation wurde ursprünglich für den LMI–Forth–Metacompiler geschrieben, aber ich habe die Quell–Screens bzw. Blocks umeditiert in eine Datei.

Der Bericht von Bernd Paysan: *Dallas 1–Wire mit Forth ansprechen* aus der VD 1/2008 hat mich angeregt, meine sehr hardwarenahe Sicht darzulegen. Ich brauchte etwa 40 Zeilen für den eigentlichen 1–Wire–Code. Der Rest ist Anwendung und Kommentar.

NOTAM: Die Timing–Schleifen laufen alle für einen alten 8031 mit 12MHz Clock. Mit modernen, optimierten und sehr schnellen 8031–Derivaten geht das natürlich nicht so einfach :-)

Villa Elisa, Paraguay all2001@gmx.de ALL20080416

123h ist hexadezimal, 456. ist dezimal # in Kommentaren

Quellen

DOW31d.txt editiert und übersetzt aus: `ALL20080416 DOW31.SCR`

Dallas One Wire words 8031 `ALL 10:08 12JAN95 Last changed screen # 010 ALL 21:21 14APR96`

DS2401 Silicon Serial Number `http://pdfserv.maxim-ic.com/en/ds/DS2401.pdf`

Listing

```
1
2   \ -------------------------------------------------------------
3   \ history:  last revision first                ALL 14:07 28AUG95
4
5   \ ALL950828 V0.06 first release
6   \ ....
7   \ ALL950112 V0.01 first try
8
9
10  \ -------------------------------------------------------------
11  \ TMEM?         DALLAS TOUCHMEMORY             ALL 11:53 28AUG95
12  DECIMAL
13
14  ASSEMBLER
15   P1.5 EQU   $PTM     \ define TOUCH-MEM DATA port pin
16  FORTH
17
18  HEX 33   EQU cTMRD  DECIMAL     \ 0F or 33 are read commands
19
20
21  \ -------------------------------------------------------------
22  \ TMEM?         DALLAS TOUCHMEMORY present?    ALL 21:13 14APR96
23
24  CODE TMEM?      ( -- t=PRESENT )    \ TM present?
25      A,# 4 MOV       $PTM CLR        \ start of RESET pulse LOW
26      B ,# 250 MOV    1$: B , 1$ DJNZ \ 480..960uSEC->500Tcycles
27      $PTM SETB                       \ end of RESET pulse   HIGH
28      B ,# 6 MOV      C CLR           \ clear CY = NO_presence
29  2$:     $PTM , 3$ JB                \ B: HIGH, leave loop
30          B    , 2$ DJNZ              \   LOOP for 3360uSEC
31          ACC  , 2$ DJNZ              \    if DATA stays LOW
32      9$ SJMP                         \ STUCK LOW
33  3$: B ,#  40 MOV                    \ 60..240uSEC-> 160Tcycles
34  4$: C,/ $PTM ORL    B , 4$ DJNZ \ delay CY=1=PRESENCE detect
35  9$: A CLR   A,# 0 SUBB   R1,A MOV
36      APUSH LJMP   END-CODE
37
38
39  \ -------------------------------------------------------------
40  \ TM><CY TM><bit0    exchange bits w/ DOW      ALL 21:14 14APR96
41  \ 15uS DS2401 window, then MASTER window, 60..120uS total slot
42
43  PROC TM><CY     \ exchange CY and TMdata bit
44      B PUSH
45      $PTM CLR      NOP NOP NOP NOP   \ start time slot 1..15uSEC
46      $PTM ,C MOV                     \ >=1uS ..6T, send CY
47      B ,# 05 MOV   1$: B , 1$ DJNZ   \ ..6+2+10=18Tcycles
48      C, $PTM MOV                     \ read DS_DATA ..20Tcycles
49      B ,# 18 MOV   1$: B , 1$ DJNZ   \ 20+2+36=78Tcycles
50      B POP                           \ =60..120uSEC =tSLOT
51      $PTM SETB                       \ terminate time slot
52      RET   END-PROC
```

```
53
54  CODE TM><bit0   ( c -- c' )  \ exchange bit0 and TMdata
55      DP=SP   DPL INC   A,@DPTR MOVX      \ get c -> ACC
56      A RRC   TM><CY LCALL   A RLC
57      @DPTR,A MOVX                        \ put ACC -> c'
58      NEXT LJMP   END-CODE
59
60
61  \ ------------------------------------------------------------
62  \ TMC><C TMRD TMWR   exchange bytes w/ DOW     ALL 21:15 14APR96
63
64  CODE TM><C  ( c -- c' )     \ exchange byte and TMdata
65          DP=SP   DPL INC   A,@DPTR MOVX      \ get c -> ACC
66          B PUSH  B ,# 8 MOV                  \ bit count=8
67  1$:         A RRC   TM><CY LCALL   B , 1$ DJNZ  \ send a bit
68          A RRC                               \ get final bit
69          B POP
70          @DPTR,A MOVX                        \ put ACC -> c'
71          NEXT LJMP   END-CODE
72
73  : TMRD  ( -- c )    255   TM><C        ;
74  : TMWR  ( c -- )          TM><C   DROP ;
75
76
77  \ ------------------------------------------------------------
78  \ CRC8          =X^8+X^5+X^4+1                 ALL 21:17 14APR96
79
80  CODE CRC8   ( crc char -- crc' )  DP=SP         \ build CRC8
81              DPL INC   A,@DPTR MOVX  DPTR INC    \ charL ->ACC
82              R0,A MOV                            \  -> R0
83              DPL INC   A,@DPTR MOVX   R1,A MOV   \ crc -> R1
84              B ,# 8 MOV    A,R0 MOV  \ B=bit count; A=char
85  1$:             A,R1 XRL   A RRC            \ CY=char XOR crc.0
86                  A,R1 MOV   2$ JNC           \ NC: bit0 was 0
87                      A,# 24 XRL                  \ 18h feedbacks
88  2$:             A RRC   R1,A MOV            \ posititon new crc
89                  A,R0 MOV   A RR   R0,A MOV  \ next bit to bit0
90                  B , 1$ DJNZ                 \ NZ: more bits
91              A,R1 MOV   @DPTR,A MOVX   DPL DEC   \ put crc'
92              DP=>SP LJMP   END-CODE
93
94
95  \ ------------------------------------------------------------
96  \ .tm# TML     show DOW/DS2401 data            ALL 12:43 31JAN95
97  \ L. shows low byte as 2 HEX chars
98
99  : .tm#  TMEM? IF
100             CR
101             cTMRD TM><C     \ 33h | 0Fh is RD CMD
102             DROP
103             0 ( -- cCRC )
104             8 0 DO   TMRD DUP L. CRC8   LOOP
105             IF  7 EMIT ." CRC error " THEN  \ append moaning
106         THEN
107  ;
108
109  : TML   BEGIN .tm#   ESC? UNTIL ;  \ an endless loop shows DOW
110
111
112  \ ------------------------------------------------------------
```

Throw & Catch Exception

Michael Kalus

Wenn es darum geht, in ANS–Forth Ausnahmen des Programmablaufs (exception) zu behandeln, braucht man eine definierte Methode. Sie soll nicht nur zur Fehlerbehandlung springen, sondern auch die Stacks, besonders den Returnstack, sauber aufgeräumt hinterlassen. So eine Methode ist `throw` und `catch`. Damit wurde ein Multilevel-Exit verwirklicht, ein Konzept, das in C als setjmp() und longjmp(), und in LISP's ebenfalls als `CATCH` und `THROW` erfolgreich ist. Im Folgenden wird zunächst das übliche exit, dann das multilevel exit näher betrachtet. Sodann wird das Konzept des `throw` und `catch` beschrieben. Schließlich wird der Umgang mit den zugehörigen Meldungen, den exceptions, erläutert.

One level Exit

Strukturiertes Programmieren in Forth erfordert bekanntlich keine besondere Mühe, weil Forth selbst schon so angelegt ist. Jede Routine (word) wird über ein exit verlassen. Und der exit führt zum inneren Interpreter, der über den IP (instruction pointer) die nächste Routine ausführen wird (next).

Der übliche Weg ist also immer so, dass das Semikolon den regulären exit herstellt (1). Auch Verzweigungen ändern daran nichts (2). Selbst eine Konstruktion mit multiplen exits fällt auf die gleiche Ebene zurück wie das reguläre Wortende (3). Und auch im Fall, dass eine Schleife vorzeitig verlassen wird, landen wir im regulären Exit (4).

```
( 1) : spam ( -- )
         tuwas  ; <-- Exit.

( 2) : spam1 ( -- )
         tuwas0 0= IF tuwas1 THEN ; <-- Exit.

( 3) : spam2 ( -- )
         tuwas0 0=
         IF tuwas1 exit  \ <--
         ELSE tu-was-anderes THEN ; <-- Exit.

( 4) : sapm3 ( -- )
        10 0 DO
            i 5 = IF leave THEN
        LOOP ; <-- EXIT
```

Multilevel Exit

Nun kann es sein, dass, während ein Programm arbeitet, etliches auf dem Daten- und Returnstack aufgetürmt worden ist, und dann ein Fehler bzw. eine *Ausnahme* (exception) auftritt. Alles, was noch auf den Stacks ist, kann nun unbrauchbar sein, z. B. ein Stacküberlauf. Das soll aber nicht zum Systemabsturz führen, sondern das Programm muss möglichst unschädlich mit einer passenden Meldung zum Benutzer zurückkehren. Es muss also, unabhängig davon, wo Return- oder Datenstack gerade hingewachsen sind, zu einem definierten Grundzustand zurückgefunden werden. Klassischerweise genügte dazu noch das Abort, welches in die Hauptroutine namens QUIT zurücksprang, und dort auf neue Eingaben wartete — das war aber kein reguläres exit, sondern ein Notausgang.

Solche Notausgänge sind jedoch für komplexere Aufgaben nicht geeignet, da in den meisten Fällen weitergemacht werden muss im Programm, selbst dann, wenn eine Teilfunktion in eine Ausnahme läuft. Hier setzt das Konzept von `catch` und `throw` an. Es ist in der Lage, auch über mehrere Verschachtelungsstufen (nesting) hinweg zu einem sicheren vordefinierten Zustand zurückzukehren. So ein Sprung über mehrere reguläre exits hinweg wird multilevel exit genannt. Mitch Bradley hat um 1990 herum `catch` & `throw` in Forth formuliert, und so den multilevel exit auch für uns realisiert.

```
: spam0  ... ... ... #exc throw ... ;
: spam1  ... spam0 ... ;     |
: spam2  ... spam1 ... ;     |
...      V
: spamx  ... ['] sapm2 catch #exc-handling ;
```

Throw

Auch wenn `catch` und `throw` eigentlich nur im Zusammenhang gesehen werden können, ist das Konzept einfacher zu verstehen, wenn man sich zunächst auf `throw` konzentriert.

Im Grunde ist `throw` nur eine Verzweigung im Programmfluss, wie das if auch. Aber während if immer nur innerhalb einer Routine verzweigt, findet `throw` auch durch mehrere Ebenen hindurch sein Ziel. Wie das if auch, verzweigt `throw` je nach flag auf dem Stack. Findet `throw` eine Null oben auf dem Datenstack, verzweigt es nicht. In allen anderen Fällen stellt `throw` die Weichen für den Sprung hin zu einer vorher vereinbarten Stelle. Diese Stelle wird gewöhnlich im *Rahmen für die Behandlung von Ausnahmen* übergeben (catchframe). Durch diese Konstruktion ist also ein conditional multilevel exit möglich.

Catch

Damit das `throw` aber kein goto wird, sondern strukturiert bleibt, braucht es ein Gegenüber, welches den Ausnahmerahmen erzeugt. So wie das `then` dem `if` während der Compilation sagt, wo es hingeht, so legt auch das `catch` für das `throw` ein Sprungziel fest. Und es legt

darüber hinaus noch den Rahmen an, mit dem auch die Stackzeiger restauriert werden können. So wird für eine sichere Landung in der Ausnahmebehandlung gesorgt.

Es ist Sache des Programmierers, solch einen Rahmen zu definieren, mit dem Forth in einen abgesicherten Zustand zurückkehren kann, falls eine Ausnahmebedingung im System aufgetreten ist. Im ANS ist lediglich ein minimaler Rahmen (catchframe) vordefiniert. Darin werden neben dem Sprungziel auch die Zeigerpositionen für den Daten– und den Returnstack angegeben, und es wird eine Ausnahmenummer (exception number, `#exc`) übermittelt. Der Standard hat für eine ganze Reihe von Ausnahmezuständen des Forthsystems solche Nummern schon festgelegt. (Siehe Liste)

Throw ausprobieren

Wie immer wollen wir mit solchen Forthkonzepten auch spielen können. Klar geht das händisch, jedenfalls im Gforth.

```
-1 throw
```

liefert uns eine Meldung und das ok wie immer.

```
*the terminal*:7: Aborted
-1 throw
   ^^^^^
Backtrace:
  ok
```

Hier führte die Fehlernummer -1 zum Neustart. Da es am Terminal eingegeben wurde, bleibt der Backtrace natürlich leer. Tritt dagegen die Ausnahme in einem Programm auf, wird ermittelt, wo das gewesen sein könnte. Hier ein simpler screenshot dazu. (Die :14: in dem Beispiel bedeutet das es die 14. Zeile Input am Terminal gewesen ist, welche den Fehler auslöste.) Jede andere Fehlernummer kann natürlich ebenso ausprobiert werden.

```
: spam -1 throw ;  ok
: blah spam ;  ok
: buh blah ;  ok
buh
*the terminal*:14: Aborted
buh
^^^
Backtrace:
$1026590 throw
$10265B0 spam
$10265D0 blah
```

Bis hierher haben wir `throw` einzeln für sich ausprobiert. Genau genommen hatte es aber schon ein `catch`, das den Rahmen dafür abgegeben hat. Nur wurde jenes `catch` ja bereits im Kern des Forthsystems angelegt.

Daran ist sehr schön zu erkennen, das es bei so einem multilevel exit ja auch gar nicht darauf ankommt, wie das letztlich gemacht worden ist. Das `throw` findet auf jeden Fall spätestens auf der Systemebene einen `catch` frame und verhält sich entsprechend. So ist es möglich, die Standard–Ausnahmezustände des Systems für eigene Programme auszunutzen, ohne sich mit dem `catch` überhaupt näher befassen zu müssen. Daher kann auch das klassische abort oder das `abort"` damit einfach formuliert werden.

```
: abort  -1 throw ;
: spam ( #exc -- ) abort" na so was!" ;
```

Catch ausprobieren

Das Gegenstück zum `throw` ist das `catch`. `catch` baut für das `throw` immer einen sichernden Ausnahmerahmen auf, bevor es das Forthwort ausführt, das absturzsicher gemacht werden soll. So ist dafür gesorgt, das unter unserem Programmwort immer das sichernde Netz des `catch` liegt. Und tritt die Ausnahme tatsächlich ein, geht es hinter dem `catch` weiter. Ansonsten geht alles so, als wäre `catch` nicht da gewesen. Wie das if und then können `catch` und trow ebenfalls verschachtelt angewendet werden.

Erreicht wird der Sprung von `throw` hinter sein Gegenstück, das `catch`, über die Manipulation der execution tokens auf dem Returnstack. Die Fehlerbehandlung hingegen durch die Übergabe einer Ausnahmenummer `#exc`. Die wird nach dem Sprung hinter das `catch` zuoberst auf dem Stack sein, und damit wird weiter gemacht. Im günstigsten Fall ist `#exc` eine Null, und es gibt an dieser Stelle nichts weiter zu tun. Das folgende Beispiel dazu ist dem dpans entnommen. Weitere Beispiele finden sich im Gforth Manual.

```
: could-fail ( -- char )
    KEY DUP [CHAR] Q =  IF  1 THROW THEN ;

: do-it ( a b -- c)   2DROP could-fail ;

: try-it ( --)
    1 2 ['] do-it  CATCH  IF ( -- x1 x2 )
        2DROP ." There was an exception" CR
    ELSE ." The character was " EMIT CR
    THEN
;

: retry-it ( -- )
    BEGIN
      1 2 ['] do-it CATCH
    WHILE
       ( exception handling: )
       ( -- x1 x2) 2DROP
       ." Exception, keep trying" CR
    REPEAT ( char )
    ." The character was " EMIT CR
;
```

Am `retry-it` ist schön zu sehen, dass die Ausnahmebehandlung auch darin bestehen kann, im Programm weiter zu machen. Solange man *ausnahmsweise* das große Q drückt, erlaubt uns die Schleife im `retry-it`, weiterzumachen, bis wir irgend eine andere Taste erwischen. Dabei ist auch gut erkennbar, dass `throw` über drei Ebenen

hinweg wieder hinter dem `catch` aufsetzt, also mit der Ausnahmenummer dort in die Schleife eintritt.

Außerdem erkennt man, dass der Datenstack hinter dem `catch` derselbe ist wie vorher! Die beiden Einträge dort `( - x1 x2 )` sind nach wie vor vorhanden. Denn das `do-it` ist ja gescheitert. Diese beiden nun wertlos gewordenen Daten müssen also durch unsere Ausnahmeregelung (exception handling) entsorgt werden.

Als Beispiel mag auch dienen, wie im Gforth mit Dateien umgegangen werden sollte.

```
s" spam.in" r/o open-file throw Value fd-in
```

Das `throw` verarbeitet zunächst die Ausnahmenummer *bevor* der file–handler als Value gespeichert werden kann. Das korrespondierende `catch` wird vom Systemkern gestellt.

Übrigens sind `catch` und `throw` im Gforth defered words, und können durch eigene Prozeduren ersetzt werden.

Exception

Abgerundet wird das Conditional–Multilevel–Exit–Konzept für die Ausnahmebehandlung durch eine einfach zu handhabende Art, eigene Ausnahmenummern samt Meldung vergeben zu können. Dazu wird das Wort exception benutzt. Es fügt einen Meldungstext in die Ausnahmeliste ein und erzeugt eine fortlaufend absteigende Ausnahmenummer, im Gforth, angefangen bei -2050. `Throw` verhält sich damit grundlegend anders als `abort"`, welches ja an den Ort seiner Definition gebunden war.

Die Liste der Ausnahmemeldungen kann mit n .error inspiziert werden. Das errorlisting erzeugt eine Übersicht der selbst definierten und der im System schon vergebenen Meldungen.

screenshot:

```
s" spam1" exception .s  <1> -2050  ok
s" spam2" exception .s  <2> -2050 -2051  ok
s" spam3" exception .s  <3> -2050 -2051 -2052  ok
...

-2052 throw
*the terminal*:2: spam3
...

-2050 .error spam1 ok
   -7 .error Do-loops nested too deeply ok

: errorlisting ( -- )
   0 next-exception @  do cr i . i .error loop ;
```

Im Gforth compiliert exception die Meldungen in eine linked list und .error greift in diese Liste. Da es im ANS–Forth aber keine standardisierte Art und Weise gibt, wie Meldungen angelegt werden sollten, ist exception spezifisch für das Gforth. Eine funktional abgespeckte, aber von der Schnittstelle her kompatible Implementation gibt es übrigens in compat.zip der TU Wien bei Anton Ertl (1). Damit kann man Programme, die exception benutzen, auf allen Standard-Systemen laufen lassen, auch in kleineren Microcontroller–Boards, die keine eigenen Fehler–Listen enthalten.

So, und nun noch viel Vergnügen beim Erkunden.

Quellen:

Crook, Ertl, Kuehling, Paysan, Wilke. Gforth Manual. https://www.complang.tuwien.ac.at/forth/gforth/Docs-html/

exceptions (ANS compatible) http://www.complang.tuwien.ac.at/forth/compat.zip.

dpans. A.9. The optional Exception word set. Table 9.2 - THROW code assignments. http://www.taygeta.com/forth/dpans1.htm

Milendorf, M. CATCH and THROW. Sun Microsystems, Inc.

Sala, F. 2003. Catch und Throw. In: Forth–Magazin *Vierte Dimension*, Heft 4/2003, Seite 24.

—

dpANS:

```
Table 9.2 - THROW code assignments (System predefined)

      Code     Reserved for
      ----     ------------
       -1      ABORT
       -2      ABORT"
       -3      stack overflow
       -4      stack underflow
       -5      return stack overflow
       -6      return stack underflow
       -7      do-loops nested too deeply during execution
       -8      dictionary overflow
       -9      invalid memory address
```

```
-10     division by zero
-11     result out of range
-12     argument type mismatch
-13     undefined word
-14     interpreting a compile-only word
-15     invalid FORGET
-16     attempt to use zero-length string as a name
-17     pictured numeric output string overflow
-18     parsed string overflow
-19     definition name too long
-20     write to a read-only location
-21     unsupported operation (e.g., AT-XY on a
        too-dumb terminal)
-22     control structure mismatch
-23     address alignment exception
-24     invalid numeric argument
-25     return stack imbalance
-26     loop parameters unavailable
-27     invalid recursion
-28     user interrupt
-29     compiler nesting
-30     obsolescent feature
-31     >BODY used on non-CREATEd definition
-32     invalid name argument (e.g., TO xxx)
-33     block read exception
-34     block write exception
-35     invalid block number
-36     invalid file position
-37     file I/O exception
-38     non-existent file
-39     unexpected end of file
-40     invalid BASE for floating point conversion
-41     loss of precision
-42     floating-point divide by zero
-43     floating-point result out of range
-44     floating-point stack overflow
-45     floating-point stack underflow
-46     floating-point invalid argument
-47     compilation word list deleted
-48     invalid POSTPONE
-49     search-order overflow
-50     search-order underflow
-51     compilation word list changed
-52     control-flow stack overflow
-53     exception stack overflow
-54     floating-point underflow
-55     floating-point unidentified fault
-56     QUIT
-57     exception in sending or receiving a character
-58     [IF], [ELSE], or [THEN] exception
```

Bootmanager und FAT-Reparatur: Erste Fort(h)schritte

Fred Behringer

Sie haben ein paar unüberlegte Experimente mit dem unseligen FDISK aus DOS, Version sowieso, gemacht, vielleicht sogar mit dem vielgepriesenen Parameter FDISK /MBR, oder etwa sogar aus Windows heraus, und plötzlich sehen Sie nur noch die Meldung *Betriebssystem fehlt* — egal, ob über Diskette oder HD gebootet. Was machen Sie? Vielleicht können Ihnen meine Vorüberlegungen aus dem vorliegenden Artikel weiterhelfen? Erste amateurhafte Schritte zwar, aber immer noch besser als das vergebliche Herumjonglieren mit allen möglichen und unmöglichen *Reparaturprogrammen*. (Man möchte gern wissen, was passiert.)

In diesem Artikel werden 23 Forth–Worte entwickelt, die zeigen sollen, wie man sich in Forth *ganz leicht* einen Bootmanager für den PC nach eigenem Geschmack einrichten kann (könnte). Zunächst wird an fünf in sich geschlossenen Beispielen die grundsätzliche Vorgehensweise besprochen. Dann werden im Listing die 23 entwickelten Forth–Worte aufgezeigt. Sie sind mit vielen \–Kommentaren versehen. An das Listing schließt sich zum schnellen Zurechtfinden ein knapp gehaltenes Glossar (23 Zeilen) an. Und schließlich wird im Hauptteil des Textes der Hintergrund beleuchtet. Nur soweit, wie für die hier geschilderten allerersten Schritte nötig.

Es kann hier die Partitionstabelle aus dem MBR (*master boot record*) angezeigt werden. (Dabei handelt es sich natürlich um den MBR der im BIOS eingestellten Boot–Festplatte.) Damit können alle wesentlichen Festplatten–Informationen ermittelt werden. Somit kann (auch) der momentane Boot–Zustand eingesehen werden. Es können beliebige primäre oder logische Partitionen versteckt oder sichtbar gemacht werden. Es können beliebige primäre Partitionen aktiviert oder deaktiviert werden. Ein dem Forth–Geübten leicht von der Hand gehendes bisschen High–Level–Forth würde sicher genügen, um aus diesen Bestandteilen einen menügeführten Bootmanager zu machen. (Ich unterdrücke mein Verlangen, das im Vorliegenden schon zu versuchen.) Allerdings zunächst nur über zwei Stufen: Erst (ein normalerweise über DOS aufgerufenes) Forth (mit dem per `INCLUDE` eingebundenen Programm `BOOTMAST.FTH`) laden und den Bootmanager auf das gewünschte Betriebssystem einstellen, dann den PC mit diesen Einstellungen rebooten. Wer hierbei einen Schritt einsparen will, der vergleiche den interessanten Artikel von Carsten Strotmann im VD–Heft 2/2006 [CS] über den Aufruf eines DOS–freien Forth–Systems schon unmittelbar nach dem BIOS–Boot–Durchgang.

Es können aber mit den hier gezeigten Mitteln auch schon einige Dinge in einen Forth–Puffer gelesen, verändert und wieder zurückgeschrieben werden, nämlich: Der MBR (*master boot record*), die Bootsektoren der einzelnen Partitionen (auch der logischen Laufwerke der erweiterten Partition) und vor allen Dingen auch die Partitionstabellen (auch die der logischen Laufwerke!) .

Dazu sind keine Kunststücke nötig: In Forth geht alles! Allerdings sollte man sich bei alldem vorher natürlich genau überlegen, was man zu tun vorhat — wenn man keine Daten verlieren möchte.

In den folgenden Beispielen wird die Verwendung der hier zu entwickelnden Forth–Worte demonstriert. Diese Worte werden in dem dann anschließenden Listing einzeln näher besprochen.

Beispiel 1: Anzeige und Änderung des MBRs

```
getmbr              \ MBR von HD in den Forth-Puffer schreiben
showsectbuf         \ Pufferinhalt auf dem Bildschirm anzeigen
cc xx sectbuf + c! \ Byte-Offset xx auf Byte cc setzen
putmbr              \ Puffer in den MBR der HD zurueckschreiben
```

Beispiel 2: Partitionstabellen der erweiterten Partition

```
n getpart           \ laedt Tabelle des logischen Laufwerks n in den Puffer,
showsectbuf         \ zeigt sie am Bildschirm an (ggf. aendern wie bei BS 1),
n putpart           \ speichert Tabelle wieder zurueck.
```

Beispiel 3: Reparatur der Adresskette der erweiterten Partition

```
0f getpart          \ ergab bei mir im Reparaturfall ab Offset 1be (Achtung,
                    \ little endian):
                    \ 80 01 C1 FF 06 FE FF FF 9E 99 07 00 D1 EB 03 00
                    \ 00 00 C1 F0 05 FE FF FF 70 B7 D7 00 50 99 07 00

10 getpart          \ ergab ab Offset 1be:
```

```
                \ 80 01 C1 FF 06 FE FF FF 8E AD 03 00 D1 EB 03 00
                \ 00 00 C1 FF 05 FE FF FF 80 A3 DB 00 20 D8 07 00

                \ Das war (offensichtlich) falsch (Laufwerk S: (= 10) ist
                \ bei mir das letzte logische Laufwerk unterhalb der
                \ DOS-Grenze von 8 GB). Repariert habe ich zu:

                \ 80 01 C1 E0 06 FE FF EF 3F 00 00 00 D1 EB 03 00
                \ 00 00 C1 F0 05 FE FF FF 70 B7 D7 00 10 EC 03 00
                \ und:
                \ 80 01 C1 F0 06 FE FF FF 3F 00 00 00 D1 EB 03 00
                \ 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

                \ Beide Gruppen wurden nach dem Muster von Beispiel 1 ueber
                \ das Forth-Wort c! in den Puffer geschrieben. Der neue
                \ sectbuf-Inhalt wurde dann ueber:
E0C1 putpart    \ und:
F0C1 putpart    \ zurückgeschrieben. F0C1 entspricht bei mir der
                \ "Sektoradresse" des letzte logischen Laufwerks (S:) vor
                \ der DOS-8GB-Grenze. Die erweiterte Partition geht noch
                \ weiter und enthaelt bei mir noch Linux, aber DOS (und
                \ Win 3.11 und Win 95/98/ME) hoeren bei F0C1 mit ihrer Suche
                \ auf (2. Zeile lauter Nullen). Aeusserste Vorsicht vor dem
                \ "Putten" mit falschen "Sektoradressen" !!!

                \ C1 F0 usw. (wegen little endian zu lesen als F0C1) enthaelt
                \ die Spur und den Sektor. Weiteres zur Interpretation der
                \ Kombination aus Spur und Sektor siehe Kommentar bei
                \ (getsect) im Listing.

                \ Offensichtlich ist die Eintragung bei Offset 1be nicht
                \ kritisch: Man haette wohl 00 (statt 80) erwartet. Ich habe
                \ mich bei der Reparatur an die analogen Stellen in den
                \ Partitionstabellen der Laufwerke D: bis Q: gehalten.
```

Beispiel 4: Haupt–Partitionstabelle

```
showparttab     \ Anzeige mit Erlaeuterungen und Ermittlung des Bootzustandes
                \ Gegegebenenfalls im Sektorpuffer sectbuf nach eigenem Gusto
                \ aendern und per putmbr (mit aeusserster Vorsicht!) auf die
                \ Festplatte zurueckschreiben.
```

Beispiel 5: Bootmanager auf Diskette

```
                \ Als "Bootdiskette" rudimentaere DOS-Diskette einrichten:
                \ Forth-System draufladen, BOOTMAST.FTH (siehe Listing)
                \ hinzuladen, das Ganze ueber SAVE-SYSTEM als BOOT.COM (auf
                \ Diskette) abspeichern.
                \ Im BIOS zum Booten "1. Diskette, 2. HD" einstellen.
                \ Booten (per Diskette): Am Prompt beispielsweise boot 3 boot
                \ eingeben. DOS herunterfahren. Diskette herausnehmen. System
                \ (jetzt per HD) neu starten. Die 3 (in boot 3 boot) bewirkt
                \ auf meiner Anlage, dass dann Windows 98 (eingerichtet mit
                \ FAT16) gebootet wird (dritte Zeile in der Partitionstabelle
                \ des MBRs).

                \ Ich glaube niemandem, nicht einmal mir selbst.
                \ Professionellen Bootmanagern, mit Absicherungen gegen
                \ Fehleingaben, will ich keine Konkurrenz machen. Ich moechte
                \ lediglich wissen, was hinter den Kulissen geschieht. Eine
                \ vorsichtige Vorgehensweise koennte wie folgt aussehen:
```

```
showparttab         \ MBR in den Puffer (nach sectbuf) holen. Partitionstabelle
                    \ auf dem Bildschirm anzeigen. OK? Ja, dann
n (bootpart)        \ Vorbereitung im MBR-Puffer auf das Booten der Partition n.
                    \ OK? Ja, dann gaaanz vorsichtig neuen MBR auf die Festplatte
                    \ schreiben:
n bootpart          \ HD ist jetzt zum Booten (der neuen Partition) vorbereitet.
[Reset-Knopf]       \ Wirksam werden lassen.

                    \ Ist man sich seiner Sache sicher, dann kann man natuerlich
                    \ auch gleich n bootpart eingeben.
```

Listing

```
1   \ ***************************************************
2   \ *                                                 *
3   \ *  BOOTMAST.FTH                                   *
4   \ *                                                 *
5   \ *  Zutaten fuer FAT-Reparatur und Bootmaster unter *
6   \ *  Turbo-FORTH-83.          Auch fuer ZF geeignet. *
7   \ *                                                 *
8   \ *  Fred Behringer - Forth-Gesellschaft -  8.8.2008 *
9   \ *                                                 *
10  \ ***************************************************
11
12  \ =================================================
13  \ Bei Arbeiten mit ZF:
14  \ zf fload bootmast.fth     -    .fth nicht vergessen!
15  \ attributs off wegnehmen!  attributs in ZF unbekannt.
16  \ Ansonsten scheint auch unter ZF alles zu gehen.
17  \ =================================================
18
19  attributs off     \ Fuer den Fall, dass kein ANSI.SYS in der CONFIG.SYS ist.
20                    \ Bei Arbeiten mit ZF wegnehmen !
21  hex
22
23  210 allot here    \ Platz fuer mind. 1 Sektor = 512d Bytes
24  here 0f and -     \ sectbuf an Paragraphenanfang
25  200 -             \ Anfang des Sektorpuffers
26  constant sectbuf  \ Liefert Adresse des Sektorpuffers
27
28  \ Sektor lesen: cx = Spur/Sektor-Kombination
29  \ cx = Bits F-0 = FEDCBA98 76543210 : Spur = 76FEDCBA98 : Sektor = 543210
30  \    =               ch       cl    :        76(cl) &ch :          von cl
31
32  code (getsect) ( seite spur/sektor -- )
33            ds push \ ds --> es
34            es pop
35       80 # dl mov  \ dl = erste Festplatte
36            cx pop  \ Kombination aus Spur (track #) und Sektor
37            ax pop
38          al dh mov  \ dh = Seitennummer (head #)
39  sectbuf # bx mov  \ bx auf den Anfang des Puffers setzen.
40      201 # ax mov  \ Einen physikalischen Sektor lesen
41            13 int  \ HD-Interrupt aufrufen
42           next end-code
43
44  \ Sektor schreiben: cx = Spur/Sektor-Kombination (wie unter "Sektor lesen"):
45  \ Spur = ch mit vorangesetzten Bits 7-6 von cl, Sektor = Bits 5-0 von cl
46
47  code (putsect) ( seite spur/sektor -- )
```

```
48            ds push \ ds --> es
49            es pop
50       80 # dl mov  \ dl = erste Festplatte
51            cx pop  \ Kombination aus Spur (track#) und Sektor
52            ax pop
53          al dh mov  \ dh = Seitennummer (head #)
54  sectbuf # bx mov  \ bx auf den Anfang des Puffers setzen.
55      301 # ax mov  \ Einen physikalischen Sektor schreiben.
56            13 int  \ HD-Interrupt aufrufen
57           next end-code
58
59  \ 10 Bit Spur und 6 Bit Sektor --> 16 Bit Spur/Sektor
60  \ In dieser Codierung steht es im Master-Boot-Record
61  \ und so wird es in Int 13h, 2/3 in cx verlangt.
62
63  code sp,sc>spsc ( sp sc -- spsc )
64         ax pop     \ Sektoreingabe
65    3f # ax and     \ Sektor = 6 niederwertige Bits von al
66         bx pop     \ Spureingabe
67     6 # cl mov     \ Spur = Bit 6,7 von al vorn an bl
68       bh cl shl    \ um 6 Bit nach links
69       bh al or     \ Spurbits 6,7 nach Sektorbyte
70       bl ah mov    \ Beides in ax sammeln
71          ax push   \ und gemeinsam zum Stack
72        next end-code
73
74  \ Umkehrung von sp,sc>spsc. Weitere Erklaerungen dort.
75
76  code spsc>sp,sc ( spsc -- sp sc )
77         ax pop     \ Eingabe (und Aufbewahrung in ax)
78       ax bx mov    \ der Spur/Sektor-Kombination
79     6 # cl mov
80       bl cl shr    \ um 6 Bit nach rechts
81       bl dh mov    \ Spurbits 6,7 nach bits 0,1 von dh
82       ah dl mov    \ Spur dx zu 10 Bits ergaenzen
83          dx push   \ Spur auf Stack
84    3f # ax and     \ Sektor = 6 niederwertige Bits von al
85         ax push    \ Sektor auf Stack
86        next end-code
87
88  \ MBR lesen und nach sectbuf speichern
89  \ Die Partitionstabelle beginnt bei Adresse 1be.
90  \ Der MBR endet mit den Bytes 55 aa.
91
92  : getmbr ( -- ) 0 1 (getsect) ;
93
94  \ Inhalt von sectbuf (gaanz vorsichtig!) in den MBR der Festplatte schreiben.
95
96  : putmbr ( -- ) 0 1 (putsect) ;
97
98  \ Sektor-Puffer am Bildschirm anzeigen
99
100 : showsectbuf ( -- )    sectbuf 200 dump ;
101
102 \ Nur 100 Bytes anzeigen
103
104 : showsectbuf100 ( -- ) sectbuf 100 dump ;
105
106 \ Sektoradresse s-ad (Spur/Sektor-Kombination wie bei (getsect)) des
107 \ n-ten logischen Laufwerks (der erweiterten Partition) auf den Stack
```

```
108  \ holen und auch den zugehoerigen Sektor nach sectbuf speichern. Die
109  \ Partitionstabelle beginnt bei Parttab-Offset 1be; ansonsten hat der
110  \ Parttab-Sektor bis auf die beiden Bytes 55 aa am Ende nur Nullen.
111
112  \ Zur (besser aufbereiteten) Anzeige der Partitionstabelle des logischen
113  \ Laufwerks n kann man (nach entsprechender Um-Interpretation) auch
114  \ (showparttab) verwenden. Man verwechsle die Partitionstabellen der logischen
115  \ Laufwerke der erweiterten Partition nicht mit derem jeweiligen Bootsektor.
116  \ Die Partitionstabellen der logischen Laufwerke entsprechen dem entsprechenden
117  \ (und am selben Platz liegenden) Teil des MBRs (der gesamten Festplatte).
118
119  : getpart ( n -- s-ad )     \ n = 1 -> 1. logisches Laufwerk, usw.
120      getmbr
121      sectbuf 1b2 +           \ Ausgangsposition im Puffer
122      4 0                     \ 4 relevante Zeilen im MBR
123      do
124        10 + dup c@ 5 =       \ Schon erweiterte Partition?
125        if 2 - @ leave then   \ Ja, dann s-ad holen und raus.
126      loop                    \ ( n s-ad(1) )
127      begin                   \ k = 0 ...
128        0 over                \ ( n-k s-ad(k+1) 0 s-ad(k+1) )
129        (getsect)             \ ( n-k s-ad(k+1) )
130        swap 1 - >r           \ ( s-ad(k+1) )
131        sectbuf 1d0 + @       \ ( s-ad(k+1) s-ad(k+2)? )
132        dup 0=                \ ( s-ad(k+1) s-ad(k+2) fl )
133        if   drop 1           \ ( s-ad(k+1) 1 )
134        else nip  0           \ ( s-ad(k+2) 0 )
135        then                  \ ( s-ad(k+?) 0/1 )
136        r@                    \ ( s-ad(k+?) 0/1 n-k-1 )
137        -rot r>               \ ( n-k-1 s-ad(k+?) 0/1 n-k-1 )
138        0=                    \ ( s-ad(k+?) 0/1 fl )
139        or                    \ ( s-ad(k+?) fl )
140      until                   \ parttab(n) jetzt in sectbuf
141      drop sectbuf 1c0 + @    \ ( fl s-ad(n) )
142      swap if cr ." Letztes Ext-Laufwerk schon erreicht!" then ;
143
144  \ Partitionstabelle eines logischen Laufwerks (der erweiterten Partition) aus
145  \ dem Puffer sectbuf holen und an "richtiger Stelle" (nach s-ad) auf die Platte
146  \ zurueckschreiben. s-ad ist derjenige Wert auf dem Stack, der nach Aufruf von
147  \ getpart dort abgelegt wurde. s-ad enthaelt Spur und Sektor in der Codierung
148  \ des Interrupts 13h. Die 0 in putpart entspricht der Seitennummer 0.
149
150  \ Das Paar n getpart und putpart dient also der Reparatur einer verungluecken
151  \ Partitionstabelle eines logischen Laufwerks der erweiterten Partition - soweit
152  \ eine solche ueberhaupt vorhanden ist.
153
154  : putpart ( s-ad -- ) 0 swap (putsect) ;  \ Aeusserste Vorsicht !
155
156  \ Bootsektor des logischen Laufwerks n (1 = erstes Laufwerk der
157  \ erweiterten Partition usw) holen und in den Sektorpuffer schreiben
158
159  \ Der Bootsektor des logischen Laufwerks n der erweiterten Partition hat nichts
160  \ mit der Partitionstabelle des logischen Laufwerks zu tun! Voellig falsch waere
161  \ es, sich nach n getboot irgendwelche brauchbaren Daten per (showparttab)
162  \ anzeigen lassen zu wollen (siehe dort).
163
164  : getboot ( n -- ) getpart 1 swap (getsect) ;
165
166  \ Partition verstecken. Zunaechst nur in sectbuf (Sektorpuffer). MBR muss
167  \ schon per getmbr in den Puffer geschrieben worden sein. Zum
```

```
168  \ Wirksamwerdenlassen dann mit putmbr abschliessen! Vorsicht bei der erweiterten
169  \ Partition! Es ist die Frage, ob ein Verstecken der erweiterten Partition
170  \ sinnvoll ist. Das Verstecken von Partitionen mit zweistelligen
171  \ Dateisystemkennungen wird hier nicht erlaubt. Davon betroffen sind
172  \ insbesondere 82 (Linux swap) und 83 (Linux native). Windows ME auf FAT32-Basis
173  \ hat die Kennung 0C und wird von hidepart voll einbezogen. Windows XP hat (bei
174  \ der ueblichen NTFS-Basis) die Kennung 07 und wird von hidepart ebenfalls voll
175  \ einbezogen.
176
177  \ Achtung: Es wird bei hidepart, hideall und hideall-ext davon ausgegangen, dass
178  \ Linux, falls vorhanden, in der erweiterten Partition liegt, dass Linux also in
179  \ der Partitionstabelle des MBRs nicht in Erscheinung tritt. Andernfalls wuerde
180  \ der hier verwendete Mechanismus des Versteckens oder Sichtbarmachens nicht
181  \ greifen. Diese Dinge muessen unbedingt noch genauer untersucht werden.
182
183  \ Alle Operationen spielen sich "nur" im Sektorpuffer sectbuf ab. Sie muessen
184  \ dann noch per putmbr auf die Festplatte geschrieben und durch Neubooten
185  \ des Computers wirksam gemacht werden.
186
187  : hidepart ( n -- )          \ n wird vorsichtshalber auf [1..4] begrenzt.
188      1 - 3 and 10 * 1c2 +     \ n-1 mal Zeilenverschiebung (10) plus Offset
189      sectbuf + dup dup        \ im Puffer (dreimal).
190      c@ f0 and 0=             \ Keine Nicht-DOS-Kennung (wie etwa 83 bei Linux)
191      if   c@ 0f and 10 or     \ Unteres Nibble uebernehmen, 1 in oberes Nibble,
192           swap c!             \ Ergebnis nach sectbuf (Sektorpuffer) schreiben;
193      else 2drop               \ sonst Kennungsadresse aus dem Puffer wegnehmen.
194      then ;
195
196  \ Partition sichtbar machen. Zunaechst nur in sectbuf (Sektorpuffer). MBR muss
197  \ schon per getmbr in den Puffer geschrieben sein. Zum Wirksamwerdenlassen
198  \ mit putmbr abschliessen! Vorsicht bei der erweiterten Partition! Weiter wie
199  \ bei hidepart.
200
201  : unhidepart ( n -- )        \ n wird vorsichtshalber auf [1..4] begrenzt.
202      1 - 3 and 10 * 1c2 +     \ n-1 mal Zeilenverschiebung (10h) plus Offset
203      sectbuf + dup dup        \ im Puffer (dreimal).
204      c@ f0 and 10 =           \ Keine Nicht-DOS-Kennung (wie etwa 83 bei Linux):
205      if   c@ 0f and           \ Unteres Nibble uebernehmen, 0 in oberes Nibble,
206           swap c!             \ Ergebnis nach sectbuf (Sektorpuffer) schreiben;
207      else 2drop               \ sonst Kennungsadresse aus dem Puffer wegnehmen.
208      then ;
209
210  \ Alle Partitionen, auch die erweiterte, verstecken. Alle Vorsichtsmassnahmen
211  \ von hidepart werden uebernommen. Die Bearbeitung findet nur im Sektorpuffer
212  \ sectbuf statt. Der MBR muss vorher per getmbr dorthin gelegt worden sein. Um
213  \ die Aenderungen auf die Festplatte zu bringen, muss dann noch putmbr
214  \ eingesetzt werden.
215
216  : hideall ( -- )
217      4 0                      \ 4 zu bearbeitende Zeilen im MBR
218      do
219        i 1 + hidepart
220      loop ;
221
222  \ Alle Partitionen, mit Ausnahme der erweiterten, verstecken. Ansonsten alles
223  \ wie bei hideall.
224
225  : hideall-ext ( -- )
226      4 0                      \ 4 zu bearbeitende Zeilen im MBR
227      do
```

```
228         i 3 and 10 * 1c2 +
229         sectbuf + c@ 0f and  \ Erweiterte Partition?
230         05 <>
231         if
232           i 1 + hidepart     \ Nein, dann verstecken
233         else
234           i 1 + unhidepart   \ Ja, dann 05 in sectbuf schreiben
235         then
236       loop ;
237
238   \ Partition bootbar machen. Hat natuerlich fuer die erweiterte Partition
239   \ (normalerweise) keinen Sinn. Zunaechst nur in sectbuf (Sektorpuffer). MBR
240   \ muss schon per getmbr in den Puffer geschrieben worden sein. Zum
241   \ Wirksamwerdenlassen mit putmbr abschliessen!
242
243   : activatepart ( lw n -- )  \ lw = HD-Laufwerk (1...). Keine Begrenzung!
244       swap 7f + swap          \ Jetzt lw = 80... .
245       1 - 3 and 10 * 1be +    \ n = Partition. n-1 begrenzt auf [0...3].
246       sectbuf + c! ;          \ In Partitionstabelle (nach sectbuf) schreiben.
247
248   \ Partition nicht-bootbar machen. Mit deactivateall (siehe gleich) kann man
249   \ die gesamte Festplatte (welche?) ausschalten. Auch bei Linux?
250
251   : deactivatepart ( n -- )   \ Byte bei Offset 1be in Partitionstabelle auf 0
252       -7f swap                \ setzen: Also keine Laufwerkangabe noetig !?!
253       activatepart swap ;
254
255   \ Alle Partitionen auf nicht-bootbar (ID = 00) setzen. Nur im Sektorpuffer
256   \ sectbuf. Der MBR muss vorher in den sectbuf geholt werden. Wenn alle weiteren
257   \ Massnahmen erledigt sind, per putmbr auf der Platte wirksam werden lassen!
258
259   : deactivateall ( -- )
260       4 0                     \ 4 zu bearbeitende Zeilen im MBR
261       do
262         i 1 + deactivatepart
263       loop ;
264
265   \ Die im MBR enthaltene Partitionstabelle mit Erlaeuterungen aus dem im
266   \ Forth-Puffer gespeicherten MBR herausholen und am Bildschirm anzeigen.
267
268   \ Achtung: (showparttab) kuemmert sich nicht darum, ob im Puffer wirklich ein
269   \ Abbild des momentanen (oder wenigstens eines brauchbaren) MBRs liegt. Vor
270   \ Aufruf von (showparttab) muss man den MBR erst per getmbr von der Festplatte
271   \ in den Puffer holen. Das Forth-Wort showparttab (siehe weiter unten im
272   \ vorliegenden Listing) erledigt beides.
273
274   \ Eine .com-Datei von etwa demselben Funktionsumfang (mit Erlaeuterungen in
275   \ englischer Sprache) wurde mir am 20.12.2003 von Rolf Schoene
276   \ (Forth-Gesellschaft und damals Institut fuer Angewandte Mathematik der
277   \ TU-Muenchen) uebermittelt. Das Vorliegende praesentiert also das Ganze in
278   \ Forth. Schon allein fuer die ueberaus kompakte Moeglichkeit der Darstellung
279   \ in Forth hat sich der mit diesem Artikel verbundene Aufwand (fuer mich)
280   \ gelohnt.
281
282   : (showparttab) ( -- )
283       ." MBR-Partitionstabelle (Kopf 0, Spur 0, Sektor 1, Offset 01BE): " cr cr
284                05 1 do i 30 + emit ." :" space
285                10 0 do sectbuf 1be + i + 10 j 1 - * + c@  0 <# # # #> type
286                                    space         loop  cr
287                                                  loop
```

```
288       4 spaces 10 0 do b3 emit 2 spaces        loop  cr
289       4 spaces 0c 0 do b3 emit 2 spaces        loop  c0 emit
290                03 0 do c4 emit c4 emit c1 emit loop
291                03 0 do c4 emit                 loop
292                ."  Part.-Laenge (in Sektoren)"        cr
293       4 spaces 08 0 do b3 emit 2 spaces        loop  c0 emit
294                03 0 do c4 emit c4 emit c1 emit loop
295                08 0 do c4 emit                 loop
296                ."  Anzahl vorausgegangener Sektoren" cr
297       4 spaces 07 0 do b3 emit 2 spaces        loop  c0 emit
298                15 0 do c4 emit                 loop
299                ."  Nr des letzten Zylinders (0..7)"  cr
300       4 spaces 06 0 do b3 emit 2 spaces        loop  c0 emit
301                09 0 do c4 emit                 loop
302                ."  Nr des letzten Sektors (0..5), Zylinder (6..7)"           cr
303       4 spaces 05 0 do b3 emit 2 spaces        loop  c0 emit
304                25 0 do c4 emit                 loop
305                ."  Nr des letzten Kopfes"             cr
306       4 spaces 04 0 do b3 emit 2 spaces        loop  c0 emit
307            ."  01:FAT12, 04:FAT16<32MB, 05:erw.Part., 06:FAT16>32MB, 07:NTFS" cr
308       4 spaces 03 0 do b3 emit 2 spaces        loop  c0 emit
309                22 0 do c4 emit                 loop
310                ."  Nr des ersten Zylinders (0..7)"   cr
311       4 spaces 02 0 do b3 emit 2 spaces        loop  c0 emit
312                16 0 do c4 emit                 loop
313                ."  Nr des ersten Sektors (0..5), Zylinder (6..7)"           cr
314       4 spaces 01 0 do b3 emit 2 spaces        loop  c0 emit
315                2b 0 do c4 emit                 loop
316                ."  Nr des ersten Kopfes (0..5)"      cr
317       4 spaces                                       c0 emit
318                08 0 do c4 emit                 loop
319                ."  80:aktive Primaerpartition (Bootpartition - nur eine!),"
320                ."  00:inaktiv" cr cr
321                ." Achtung: little endian!" cr ." Die vier Bytes "
322                04 0 do sectbuf 1ca + i + c@  0 <# # # #> type space loop
323                ." in Zeile 1 stellen die Hexzahl "
324                04 0 do sectbuf 1ca + 3 + i - c@  0 <# # # #> type   loop
325                ."  dar, usw."                        cr ;
326
327   : showparttab ( -- )        \ Kommentare siehe (showparttab)
328         getmbr cr cr (showparttab) ;
329
330   \ Mit dem Booteinrichtungsprogramm n (bootpart) wird die Partitionstabelle im
331   \ Sektorpuffer sectbuf auf das Booten der Partition n vorbereitet. n wird durch
332   \ "wrapping" auf die Werte 1 bis 4 beschraenkt und stellt die Zeile in der
333   \ Partitionstabelle dar, deren Entsprechung gebootet werden soll. Wird
334   \ irrtuemlich ein Booten von der erweiterten Partition (so man eine eingerichtet
335   \ hat) verlangt, so bricht (bootpart) mit einer Fehlermeldung ab und das System
336   \ wartet auf eine neue Eingabe. Bevor (bootpart) vernuenftig arbeiten kann, muss
337   \ der MBR per getmbr in den Sektorpuffer sectbuf geholt worden sein. Damit die
338   \ Neueinstellungen auf die Festplatte geschrieben werden, muss abschliessend
339   \ putmbr eingesetzt werden.
340
341   \ Vorsicht mit putmbr, wenn man sich nicht ganz sicher ist, ob man das Resultat
342   \ von getmbr fuer den Fall aller Faelle irgendwo abgespeichert hat!
343
344   : (bootpart) ( n -- )       \ n = Zeile in der Partitionstabelle
345       dup 1 - 3 and 10 *
346       1c2 + sectbuf + c@ 0f and 05 =
347       if cr ."  Erweiterte Partition" drop exit then
```

```
348       hideall-ext            \ Alle Partitionen, ausser erweiterter, verstecken.
349       deactivateall          \ Alle Laufwerke inaktiv setzen
350       1 over activatepart    \ Laufwerk n in Boot-HD aktiv setzen
351       unhidepart             \ Laufwerk n in Boot-HD sichtbar machen
352       (showparttab) ;        \ Partitionstabelle anzeigen
353
354   \ Die vorausgegangenen Vorbereitungsschritte werden ueber bootpart zu einem
355   \ einzigen Schritt zusammengefasst. Die neue Partitionstabelle steht dann im
356   \ MBR der Festplatte. Zum endgueltigen Booten muss der Computer dann neu
357   \ gestartet werden.
358
359   : bootpart ( n -- )         \ Laufwerk n in Boot-HD endgueltig booten
360       getmbr                  \ MBR in den Puffer sectbuf holen
361       (bootpart)              \ Partitionstabelle vorbereiten
362       putmbr ;                \ Puffer sectbuf auf Boot-HD zurueckspeichern
363
364
365   \ Glossar
366
367   \ sectbuf        ( -- ad) Konstante, Adresse des Sektorpuffers
368   \ (getsect)      ( seite spur/sektor -- ) Sektor von HD nach sectbuf holen
369   \ (putsect)      ( seite spur/sektor -- ) Sektor von sectbuf nach HD schreiben
370   \ sp,sc>spsc     ( sp sc -- spsc ) Spur und Sektor zu 2 Bytes zusammenfassen
371   \ spsc>sp,sc     ( spsc -- sp sc ) Umkehrung von sp,sc>spsc
372   \ getmbr         ( -- ) 0 1 (getsect) , MBR von HD nach sectbuf holen
373   \ putmbr         ( -- ) 0 1 (putsect) , sectbuf als MBR auf HD schreiben
374   \ showsectbuf    ( -- ) sectbuf (200 Bytes) anzeigen - egal, was drin
375   \ showsectbuf100 ( -- ) sectbuf (100 Bytes) anzeigen - egal, was drin
376   \ getpart        ( n -- s-ad ) Partitionstabelle n von HD nach sectbuf holen
377   \ putpart        ( s-ad -- ) Ergebnis von getpart von sectbuf nach HD schreiben
378   \ getboot        ( n -- ) Bootsektor des Laufwerks n von HD nach sectbuf holen
379   \ hidepart       ( n -- ) Partition n mit Hi-Nibble 1 versehen - nur im sectbuf
380   \ hideall        ( -- ) Alle Partitionen im sectbuf mit Hi-Nibble 1 versehen
381   \ hideall-ext    ( -- ) Wie hideall, aber erweiterte Partition mit 05 versehen
382   \ unhidepart     ( n -- ) Partition n in sectbuf wieder sichtbar machen
383   \ activatepart   ( lw n -- ) HD-ID von Part. n von Platte lw auf 80+lw setzen
384   \ deactivatepart ( n -- )    HD-ID von Partition n auf 00 setzen
385   \ deactivateall  ( -- )    HD-ID aller Partitionen auf 00 setzen
386   \ (showparttab)  ( -- ) MBR-Partitionstabelle aus sectbuf am Monitor anzeigen
387   \ showparttab    ( -- ) Wie (showparttab), aber erst MBR von HD nach sectbuf.
388   \ (bootpart)     ( n -- ) sectbuf zum Booten von Partition n vorbereiten
389   \ bootpart       ( n -- ) MBR auf der HD zum Booten von Partition n vorbereiten
390
391   \ Ende des Listings
```

Hintergrund

Das Ziel im Titel ist hochgesteckt, aber eigentlich ist es auch hier nur wieder der berühmte *Weg, der das Ziel verkörpert*. Mit Forth im Rucksack traut man sich leicht auch auf den schwierigsten Weg — dem Ziel entgegen.

An sich will ich hier den immer wieder zu hörenden Spruch „In einem solchen Fall sind die Daten unrettbar verloren“ ad absurdum führen. Unüberlegtes Handeln kann aber doch auch im vorliegenden Artikel große Schwierigkeiten nach sich ziehen. Außerdem sind Fehler in meinen Überlegungen oder in den Programmen vorprogrammiert. Von blinden Experimenten, wobei die Betonung auf *blind* liegt, rate ich also dringend ab. Es sind auch noch genügend Fragezeichen stehengeblieben, die auf eine endgültige Behandlung warten.

Nichts ist praktischer als eine gute Theorie. Aber auch die beste Theorie nützt wenig, wenn sie nicht an brauchbaren praktischen Beispielen erklärt wird.

Das allgegenwärtige *praktische Beispiel* war bei mir ein zerschossenes Windows 98 auf dem PC unter FAT16 auf einem logischen Laufwerk von 1 GB in der erweiterten DOS–Partition. „Zerschossen? Da geht überhaupt nichts mehr“, hat man mir im Bekanntenkreis beschieden. Nun, ja! Es hat mich aber viel hartes Nachdenken, viel Nachschlagen und enorm viel Nerven gekostet. Genau besehen, ist es eigentlich unsinnig, sich dermaßen stark in eine Aufgabe aus einem Themenbereich zu knien, den

man durch die Beschäftigung mit der Aufgabe eigentlich erst erlernen möchte. „Wozu?“, war die Frage, die ich zu hören bekam. „Dieses Problem haben doch andere schon längst bearbeitet. Man braucht ja nur im Internet oder in der Literatur zu suchen.“ Nun, ich betreibe Forth als Steckenpferd und bin daher gar nicht so unfroh darüber, „dass es immer wieder was zu tun gibt“. Ich habe sehr viel dabei gelernt. Vor allen Dingen habe ich gelernt, dass man nicht in Panik geraten und blindwütig alle verfügbaren *Rettungssysteme* (von zum Teil fragwürdiger Herkunft) durchprobieren sollte. Zumindest dann nicht, wenn man sich voller Ungeduld noch keine Zeit gegönnt hat, die Beschreibungen ordentlich durchzulesen.

Es werden hier einige Forth–Worte für den Umgang mit der Hauptpartitionstabelle (dem MBR) und den Partitionstabellen und den Bootsektoren der logischen Laufwerke in der erweiterten Partition angegeben. Es werden nur die Verhältnisse bei FAT16 besprochen. Und schließlich wird ein Forth–Wort entwickelt, das man als Bootmanager verwenden kann, ein Bootmanager, den man mit Forth–Mitteln ganz leicht den eigenen Bedürfnissen und Erwartungen anpassen kann. Die eigentliche DOS–Dateiorganisation, die Dateizuordnungstabellen und alles, was zur FAT16–*Reparatur* nötig ist, soll einem späteren Bericht vorbehalten bleiben.

Weiter unten gebe ich einiges FAT16–Relevantes aus der sehr lesenswerten kurzen und prägnanten Zusammenfassung von Anton Zechner [AZ] wieder. (Fehler infolge von Änderungen des Textes gehen zu meinen Lasten.)

DOS und ähnliche Betriebssysteme können nur von primären Partitionen booten (die im Byte mit Offset 1be + n*10h (n = Partition) den Hexwert 80 (erste Festplatte, zweite = 81 usw.)) eingetragen haben müssen. Alle übrigen primären Partitionen müssen versteckt werden. Einige Betriebssysteme werden zum Verstecken mit einer 1 im höherwertigen Nibble unter Offset 1c2 (+ n*10h, n=0..3) versehen. Das gilt für das hier Gesagte, aber nicht für alle denkbaren Betriebssysteme. Im vorliegenden Artikel werden über die Forth–Worte hidepart und unhidepart nur solche Betriebssysteme berücksichtigt, die eine 1 (für versteckt) oder 0 (für sichtbar) im höherwertigen Nibble verwenden.

Das fürs Booten von Windows 98 nötige DOS (bei mir DOS 6.2) liegt natürlich (fürs Dual–Boot–Verfahren) auf einer kleinen primären FAT16–Partition. Als Bootmanager verwendete ich bisher XFDISK von Florian Painke und Ulrich Müller [PM]. Für die Reparaturarbeiten hat es sich gut gemacht, dass der Hauptteil des Windows–98–Systems auf einem logischen Laufwerk liegt: Ich kann so von überall her (Windows 95, Windows ME, Linux 9.1, FreeDOS, DOS 7.0, DOS 8.0) darauf zugreifen. Hätte mein zerschossenes Windows 98 vollständig auf einer primären Partition gelegen, dann hätte ich diese Partition *verstecken* müssen und hätte (beim Booten von einer anderen primären HD–Partition aus) keinen Zugriff mehr darauf.

An der hier zu besprechenden Computer–Anlage arbeite ich gern. Sie ist meine einzige (betriebsbereite) Anlage mit ISA–Slots für meine Transputer–Karten. (Und ein ZIP–Laufwerk habe ich dort auch noch zur Verfügung, was bei meinen anderen Anlagen nicht mehr der Fall ist.)

Die reparaturbedürftige Situation stellte sich wie folgt dar:

Beide Dateizuordnungstabellen (FAT16) von Windows 98 waren vollständig gelöscht (auf 0 gesetzt). Wahrscheinlich war `SCANDISK` von mir, nachdem es wieder mal unerwünscht losgerattert war, per Reset–Knopf am ordnungsgemäßen Zuendeführen seiner Arbeit gehindert worden.

Die Verzeichnisse und Unterverzeichnisse waren (in den FATs) fast alle als gelöscht markiert (im ersten Buchstaben ihres Namens mit einem E5h versehen) worden.

Die Dateien waren teils als gelöscht gekennzeichnet, teils nicht.

Ich hatte (und habe noch) im Netscape Navigator unter Windows 98 etwa 82 Megabyte an älteren empfangenen E–Mails und etwa 32 Megabyte an gesendeten E–Mails (mit Anhängen!) zu liegen, auf die ich nur ungern verzichten wollte. Eigenartigerweise war (fast) keine Datei wirklich angegriffen worden. Aber von einer Inbox– oder Sent–Sammeldatei des Netscape Navigators erwartet man natürlich nicht, dass sie nicht defragmentiert ist. Und das manuelle Zusammenstückeln der einzelnen Cluster ist eine Heidenarbeit. Um es gleich zu sagen, diese Arbeit habe ich noch nicht zum Abschluss bringen können.

Meine ersten Reparaturversuche waren:

Eine Analyse mit dem Partition–Manager von Mikhail Ranish [MR]. Dabei muss ich etwas falsch gemacht haben: Die Organisation der ersten primären Partition (nicht die Dateien selbst, nur die Grenzen von Bootsektor, FAT1, FAT2, Root–Directory und Datenbereich) wurden um genau einen Cluster (16 KB = 32 Sektoren) nach höheren HD–Speicherwerten hin verschoben. Es dauerte einige Zeit, bis ich verstand, dass das Wiedereinsetzen des ursprünglichen Bootsektors den Fall behoben hätte. Und noch länger hat es gedauert, bis ich dahinterkam, dass das mit einem ganz kleinen Eingriff in den Datenteil des Bootsektors (hier der ersten primären Partition) wieder hätte hingebogen werden können. Irgendwo habe ich in der Zwischenzeit gelesen, dass sich der Ranish–Partition–Manager [RP] in den freigeschaufelten Platz (in der ersten primären Partition) einnistet. Beinahe hätte ich wegen meiner anfänglichen Unwissenheit deswegen das zu Windows 95 führende DOS–System (der ersten primären Partition) aufgegeben. (Ich hasse Neuinstallationen an einem organisch gewachsenen System.)

Als Hilfsmittel für den sektorweisen Festplattenzugriff stand mir der Diskeditor DISKEDIT (ver 1.2) von Martin Kalisch [MK] (1992) zur Verfügung. Für das Auslesen (in eine Datei) und das Wiedereinlesen der Bootsektoren (nur der primären Partitionen) und des MBRs verwendete ich anfangs das Programm SSI.COM aus

dem *PC–Festplattenbuch* von Michael Thieser und Andreas Finkler [TF] (Markt&Technik 1996). (Bemerkung eines gutmeinenden Computerfreundes: „Was heißt hier Bootsektoren? Von Laufwerken der erweiterten Partition kann man nicht booten." Nun ja, jedem seine Meinung!) Natürlich hätte man die Bootsektoren, auch die der erweiterten Partition, genauso gut auch nach dem Schema `DEBUG -l 200 2 0 1 d 200` auslesen können. Aber woher die Daten für die Bootsektoren–Kette der aufeinanderfolgenden logischen Laufwerke (in der erweiterten Partition) nehmen? Und der MBR (mit der Haupt–Partitionstabelle) lässt sich wohl ohnehin nicht mit `DEBUG` (direkt) auslesen? Und woher die Partitionstabellen der gesamten Kette von logischen Laufwerken nehmen?

Und überhaupt: Warum soll man seine Zeit übermäßig mit dem Herumsuchen nach Programmen anderer Autoren verbringen (bei denen man meistens doch nicht genau herausbekommt, was sie eigentlich tun), wenn man sich die Dinge *ganz schnell* auch selbst machen kann? Forth macht's möglich — und es sollte hier ein erster Versuch dazu unternommen werden. Mit dem sprichwörtlichen Rad, das da nicht immer wieder neu erfunden zu werden braucht, hat das nur bedingt etwas zu tun. Mit *Learning by doing* schon eher. Es soll hier nicht über Weltneuheiten berichtet werden, sondern über anfangs vergebliche Versuche eines neugierigen Forth–Hobbyisten. Im Übrigen halte ich mich gern an Rafael Deliano, der sich im VD–Heft 1/2008 wiederum auf Charles Moore bezieht, welchen man sinngemäß mit den Worten zitieren kann: „Gebt mir nur die Idee, und ich programmiere mir den Rest selbst — in Forth."

PCTOOLS kann beim Wiedereinbringen von gelöschten Dateien helfen, zumindest bei nicht fragmentierten. Beim Versuch, auch gelöschte Verzeichnisse zu reparieren, kam aber laufend die Meldung *Sector not found* heraus. Das war für mich nicht sehr hilfreich. Und in Bezug auf PCTOOLS–Unterlagen musste ich zu mir selbst sagen: *Manuals not found* — oder *Manuals überhaupt nie gehabt.*

Zu allem Überfluss wurden bei den Versuchen mit Reparaturprogrammen (ich glaube, es war `TESTDISK`) auch noch die Laufwerke R: und S: (die letzten in der Kette der Laufwerke der erweiterten Partition vor der von DOS geforderten 8GB–Grenze) zu einem einzigen Laufwerk *zusammengelegt* — mit verheerenden Folgen, was die Partitions–Tabellen–Ketten–Parameter betrifft. Und außerden wollte ich die in `R:` und `S:` angehäuften vielen schönen Programme nicht verlieren. (Mit den im vorliegenden Artikel entwickelten Forth–Werkzeugen konnte ich das Problem schnellsten geradebiegen — siehe Beispiel 3.) Außerdem wollte ich das dann auf meiner Festplatte noch folgende SuSE–Linux 9.1 (5 GB *ganz hinten* in der erweiterten Partition) nicht leichtfertig durch Neuformatierungen an anderer Stelle der erweiterten Partition aufs Spiel setzen. Der FAT16–Teil vor der 8GB–Grenze (der erweiterten Partition) wird bei mir per `XFDISK /nowin9x` auf die Kennung 5 gesetzt und dadurch auch für DOS zugänglich gehalten, nicht nur beispielsweise für Windows ME (das bei mir auf einer primären Partition mit FAT32 *ganz weit hinten* liegt).

All diese Erkenntnisse und Überlegungen haben mich dazu geführt, mir die Hilfsmittel für das Operieren an der Dateisystem–Verwaltung selbst zu machen — es zumindest zu versuchen. Ohne Brimborium, nur das Nötigste, zielgerichtet in einer Sammlung kleinerer Hilfsbausteine, in Form von Forth–Worten, die nach Belieben in ein größeres Forth–Umfeld eingebaut werden könnten. Forth macht's möglich.

Ich habe mich im vorliegenden Artikel fürs Erste auf den MBR (*master boot record*) und darin auf die Hauptpartitionstabelle und dann auch noch auf die Partitionstabellen (*partition tables*) der erweiterten Partition konzentriert. Die Bootsektoren und das gezielte Lesen, Ändern und Schreiben (in Forth) von FAT16–Verzeichnissen und –Datensektoren behalte ich mir für später vor.

Als Nebenprodukt stelle ich mit dem vorliegenden Artikel einen voll („na ja") funktionsfähigen Bootmanager zur Verfügung, der von Forth unter DOS auf Festplatte oder Diskette aus operiert und weder installiert noch deinstalliert zu werden braucht. Bootmanager gibt es *wie Sand am Meer.* Etwa fünf der gebräuchlichsten liegen zur unmittelbaren Benutzung auf der sehr empfehlenswerten *Ultimate Boot–CD 3.4* [UB]. Ich traue jedoch (siehe oben) keinem Bootmanager mehr – es sei denn, ich habe ihn selbst vermurkst.

Die Vorgehensweise ist wie folgt: Man lade DOS. Am einfachsten über eine Diskette. Will man DOS von einem HD–Laufwerk aus laden, muss man sich zur Organisation etwas einfallen lassen. Das ist ein gesondertes Thema, das für das hier Gesagte nicht unbedingt lebensnotwendig ist. Vom geladenen DOS aus rufe man Forth mit dem hier besprochenen Paket von Forth–Worten auf, stelle die Festplatte per n Bootpart auf das einzuladende Betriebssystem n um und boote neu (*Affengriff* oder Reset–Knopf). An Umständlichkeit mutet das dem Benutzer auch nicht mehr zu als das (von mir immer noch gern praktizierte) Booten von Linux per `LOADLIN` von DOS aus.

Wenn man so will, kann man daraus auch per `SAVE-SYSTEM` eine eigenständige COM–Datei machen — und vergessen, dass man Forth zum Booten missbraucht. Vielleicht kann das als eine Ergänzung zu dem von Carsten Strotmann [CS] Gesagten betrachtet werden. Carsten erwähnt in seinem interessanten Artikel aus dem VD–Heft 2/2006 (über Forth am Stick) das Programm von Alexei Frounze (Russland 2000), das sich (ähnlich wie der weiter unten auch zu erwähnende Ranish–Partition–Manager) in den von DOS und fast allen anderen Betriebssystemen ungenutzten ersten Sektoren (auf Seite 0) einklinkt und ein (von DOS–Zugriffen gesäubertes) Forth–System zum Booten verwendungsfähig macht. Carsten erwähnt weiter, dass man das bootende Forth–Rumpfsystem mit den von Alexei Frounze bereitgestellten Mitteln auch als COM– oder EXE–Datei schon ganz am Anfang (noch vor dem sonst üblichen vom System aufgerufenen DOS) aufrufen und sich das Einbauen in

die ungenutzten Sektoren sparen kann. Das eigentliche Booten geschieht dann also aus einem schon bereitliegenden Dateisystem heraus.

Das im vorliegenden Artikel beschriebene Boot–Forth braucht nicht DOS–frei zu sein. Es setzt hinter dem schon in den Computer geladenen DOS an — so, wie ich es auch bei SuSE–Linux 9.1 beim Booten durch Aufruf von `LOADLIN` aus DOS heraus mache.

Zur Entwicklung habe ich Turbo–Forth (in der 16–Bit–Version) von Marc Petremann [MP1–3] verwendet. Die von mir im hiesigen Listing eingesetzte Assembler–Notation aus Turbo–Forth stimmt (wie ich nachgeprüft habe) mit der aus ZF überein. Mit anderen Worten, das vorliegende Programmpaket läuft auch unter ZF (von Tom Zimmer) — wenn man das ganz am Anfang des vorliegenden Artikels über `ATTRIBUTS` Gesagte beachtet.

Unter Windows 3.11/95/98/ME kann das Turbo–Forth–System ohne Schwierigkeiten aufgerufen werden und das hier entwickelte Forth–Programm–Paket arbeitet bestens. Das Ganze funktioniert auch unter FreeDOS. (Auf ein nicht wirksam werdendes `ANSI.SYS` in der `CONFIG.SYS` kann man verzichten, indem man in Turbo–Forth `attributs off` schaltet.)

Ein Abstecher zu XP: Windows XP erlaubt *keinen direkten Zugriff auf die Festplatte*. Man hat die Möglichkeit, diese Einschränkung zu *ignorieren*. Dann kommen aber beispielsweise bei `showparttab` auf dem Bildschirm lauter Nullen heraus. Turbo–Forth und die hier entwickelten Forth–Programme sind für den reinen DOS–Betrieb gedacht. Aber auch XP–Systeme beziehen ihre Boot–up–Daten über die Partitionstabelle im MBR. Um mir absolute Gewissheit zu verschaffen, habe ich *mal schnell* an einer XP–Maschine nachgeprüft, dass ich per USB–Diskette oder über ein *normales* Diskettenlaufwerk oder mit FreeDos unter KNOPPIX auf Live–CD das mich störende XP umgehen kann. Das geht, wie erwartet, gut.

Im schon erwähnten Artikel (Forth am Stil — Teil 1) von Carsten Strotmann aus dem VD–Heft 2/2006 [CS] wird Weiteres zum Thema *Einbau von Forth ohne DOS in den Bootprozess* angekündigt. Ich bin gespannt.

An dieser Stelle steht bei den Autoren, die sich mit solchen und ähnlichen Fragen beschäftigen, üblicherweise die Bemerkung, dass die vorgeschlagenen Operationen *äußerst systemgefährdend* seien. Nun ja, man kann sich ja beispielsweise den MBR vor jeder Veränderungsabsicht sicherheitshalber per `getmbr showsectbuf` (siehe Listing) anzeigen lassen und ihn mit Forth–Mitteln ganz leicht und bequem durch Abspeichern des Inhalts von `sectbuf` in eine Datei *sichern*.

Wie habe ich eigentlich die Entwicklung auf einem System erledigt, das dabei den XFDISK als Bootmanager verwendet, wobei ich von XFDISK auch bei den Entwicklungsarbeiten nicht lockerlassen wollte? Zunächst einmal habe ich beim Ausprobieren alle schreibenden Zugriffe auf die Festplatte ausgeklammert. Insbesondere habe ich (`putsect`) `putmbr putpart bootpart` fürs Erste vermieden.

Ansonsten deinstalliere man vom XFDISK–Bootmanager–Menü aus eben diesen Bootmanager–Teil. Das *Retten* der XFDISK–Menü–Einstellungen in eine Datei `CONFIG.XCF` ist dabei hilfreich. Nach Erledigung der vorzunehmenden Überprüfungen (`getmbr`, abändern, `putmbr` usw.) kann man dann den XFDISK–Bootmanager (unter Einbeziehung von `CONFIG.XCF`) wieder neu installieren.

Zu guter (Vor–)Letzt noch eine Schwierigkeit, bei welcher ich für Hinweise aus der Leserschaft dankbar wäre: Alles bisher Gesagte ist für ein Arbeiten mit FAT16 gedacht und funktioniert bestens. Bei mir bezieht sich der Kern der Überlegung auf das *externe* Umschalten des Bootverhaltens des Computers von Windows 95 zu Windows 98 (und zurück) unter Einbeziehung von DOS 6.2 und Windows 3.11 nebst den logischen Laufwerken `D:` bis `S:` der erweiterten Partition. Der gesamte unter FAT16 anerkannte Teil der erweiterten Partition liegt unterhalb von 8 Gigabyte. Im oberen Teil der erweiterten Partition liegt SuSE–Linux 9.1. Das rufe ich per LOADLIN von DOS aus auf. *Ganz hinten* auf der Festplatte (XFDISK macht's möglich) liegt Windows ME auf einer weiteren primären Partition. Die aber habe ich unter FAT32 eingerichtet. In den Bootmanager von XFDISK lässt sich ME prima einbinden und von dort aus kann es ohne Schwierigkeiten aufgerufen werden — auch wenn der Bootmanager von XFDISK für die Zeit des Ausprobierens deinstalliert und später wieder installiert wurde. Mit den im vorliegenden Artikel beschriebenen Hilfsmitteln (außerhalb von XFDISK, per n bootpart) kann ME (bei mir wenigsten) jedoch nicht gebootet werden. Ich habe es an einer anderen, ähnlich eingerichteten Anlage noch einmal versucht — mit demselben Misserfolg. Und schließlich habe ich an einer dritten experimentellen Anlage Windows 98 (natürlich nicht auf DOS 6.2 aufgesetzt) auf einer primären Partition unter FAT32 eingerichtet: Auch dieses Windows 98 ließ sich mit den Mitteln aus dem vorliegenden Artikel nicht booten. Der Bootmanager von XFDISK schafft das dagegen wunderbar. Was macht der Bootmanager von XFDISK, das ich mit meinen Überlegungen (noch) nicht schaffe? Spielt außer dem MBR auch noch der Bootsektor hinein? Habe ich bei meinen *Reparaturen* etwas übersehen? Aber mein Reparaturfall bezog sich ja nur auf eine einzige von mehreren daraufhin untersuchten Anlagen!

Die MBR–Eintragungen im zuletzt erwähnten Experiment unter *Erster Kopf, erster Zylinder, erster Sektor* waren sowohl bei Windows 98 wie auch bei Windows ME dieselben, nämlich `00 C1 FF` . Nebenbei gesagt, zeigten auch zwei XP–Anlagen (unter NTFS mit der Kennung `07`) eben diese Eintragungen, nämlich `00 C1 FF` . Als Untersuchungsgegenstände habe ich mir zum *Lüften dieser Geheimnisse* für demnächst Folgendes vorgenommen: Erweiterter Interrupt 13, FAT32, LBA statt CHS, Festplatten–Geometrie, Einbeziehung des Bootsektors.

Überflüssig zu erwähnen, dass alle hier beschriebenen Vorgänge beliebig automatisiert und menügesteuert eingerichtet werden können. Außerdem können wir als

Forth–Nutzer Parameter–Eingaben auch ganz unkompliziert einfach im Quelltext vornehmen. Der muss ja sowieso vom Forth–Compiler übersetzt werden.

Alexei Frounze [AF] verwendet MSDOS 6.22 und Windows 95. Ich befinde mich also in guter Gesellschaft und brauche mich nicht zu scheuen, nicht über XP, Vista und die allerneuesten MS–Erzeugnisse zu sprechen. Aber auch Linux in den allerneuesten Versionen und Zusammenstellungen (*distributions*) kann für meine Experimente außer Acht gelassen werden. A. Frounze fragt: Where do I get these compilers and assemblers from? Für mich war eine solche Frage schon immer gegenstandslos: Ich habe Forth. Zu Forth gehört normalerweise ein Assembler. Und wenn mir der nicht reicht, feile ich ihn mir einfach weiter zurecht. Keine große Systemveränderung! Einfach nur ein Vorspann zum sowieso einzuladenden Forth–Programm. Und experimentiere ich ausnahmsweise mal mit einem Forth–System, das keinen Assembler eingebaut hat, dann kann ich mir ja ganz leicht Hex–Zahlen als *Inline–Code* einbauen. Unter Forth geht alles! Ein bisschen Nachschlagen in den Assembler–Befehlslisten tut nicht weh, und bei Julian Noble ([JN1],[JN2]) steht beispielsweise, wie man den *Inline–Einbau* von Hexzahlen automatisieren und gegen Fehleingaben absichern kann.)

Das FAT–16–Dateiformat

Festplatten beginnen mit dem MBR auf Kopf 0, Zylinder 0 und Sektor 1. Der MBR enthält ein kleines Programm, das den Bootprozess einleitet. Für den vorliegenden Artikel wichtig ist die Partitionstabelle im MBR. Auch jedes Laufwerk in der erweiterten Partition hat eine Partitionstabelle! Auch diese Tabellen sind wichtig. Über die Bootsektoren soll in einem späteren Artikel berichtet werden.

Der MBR hat folgenden Aufbau (Angaben hexadezimal):

```
Offset  Größe      Funktion
   000  1BE Bytes  1. Stufe des Bootcodes
   1BE   10 Bytes  1. Eintrag in der Partitionstabelle
   1CE   10 Bytes  2. Eintrag in der Partitionstabelle
   1DE   10 Bytes  3. Eintrag in der Partitionstabelle
   1EE   10 Bytes  4. Eintrag in der Partitionstabelle
   1FE    2 Bytes  Signatur 55 AA
```

Die Partitionstabelleneinträge haben folgenden Aufbau (Auszug aus [TF]):

```
Offset Größe   Funktion
    00 1 Byte  80: bootfähig
               00: nicht bootfähig
               81: 2. Festplatte usw.
    01 1 Byte  Startkopf der Partition (0 bis n)
    02 1 Byte  Bit 0-5 Startsektor (1-63)
               Bit 6-7 = 8-9 vom Startzylinder
    03 1 Byte  Startzylinder (0-1023)
    04 1 Byte  OS-Typ:
               01 = FAT12
               04 = FAT16 (max. 32 MB)
               05 = erweiterte DOS-Partition
               06 = FAT16 (max. 2 GB)
               07 = HPFS/NTFS
               0A = OS/2-Bootmanager
               0B = FAT32 (CHS-Adressierung)
               0C = FAT32 (LBA-Adressierung)
               0E = FAT16 (LBA-Adressierung)
               0F = erweiterte Partition (LBA,
                    mehr als 1024 Zylinder)
               11 = versteckt FAT12
               14 = versteckt FAT16 bis 32MB
               16 = versteckt FAT16
               17 = versteckt HPFS / NTFS
               1B = versteckt WIN95 FAT32
               1C = versteckt WIN95 FAT32 (LBA)
               1E = versteckt WIN95 FAT16 (LBA)
               3C = Partition Magic
               50 = OnTrack DM
               52 = CP/M
               81 = Booten von Laufwerk D: (?)
               82 = Linux Swap
               83 = Linux native
               84 = OS/2 versteckt C:
               86 = NTFS volume set
               87 = NTFS volume set
               9F = BSD/OS
               A6 = Open BSD
               C1 = DRDOS/sec (FAT32)
               C4 = DRDOS/sec (FAT32(LBA))
               C6 = DRDOS/sec (FAT16(LBA))
               EB = BeOS fs
               EE = EFI GPT
               EF = EFI (FAT12/16/32)
    05 1 Byte  Endkopf der Partition (0 bis n)
    06 1 Byte  Bit 0-5 Endsektor (1 bis 63)
               Bit 6-7 = 8-9 vom Endzylinder
    07 1 Byte  Endzylinder (0 bis 1023)
    08 4 Bytes Anfangssektor der Partitionsdaten
               (absoluter LBA)
    0C 4 Bytes Größe der Partition in Sektoren
Offset Größe   Funktion
```

Die LBA–Adressierung (*large block array*) ist für Festplatten mit mehr als 1024 Zylindern vorgesehen, bei weniger Zylindern kann weiter die CHS–Adressierung verwendet werden (*cylinder head sector*). Bei der LBA–Adressierung wird statt Kopf, Zylinder, Sektor ein 32–Bit–DWORT–Wert für die Sektor–Adressierung verwendet.

Aus den Partitionsdaten kann die Position und Größe der logischen Laufwerke ermittelt werden. Dazu dienen die DWORD–Werte bei Offset `08` und `0C`. Handelt es sich bei der Partition um eine *erweiterte Partition*, so ist diese kein logisches Laufwerk, sondern eine weitere Unterpartition mit einem weiteren Master–Boot–Record (Partitionstabelle). Zu allen Positionen der LBAs ist der Anfangssektor der erweiterten Partition hinzuzuzählen (relative Angaben).

Aufbau einer Partition

Am Anfang jeder FAT16–Partition eines logischen Laufwerkes steht der Boot–Sektor. Er enthält Angaben über die Größe und Art der Partition. Anschließend kommt ein Bereich mit reservierten Sektoren. Diese können auch

eine Kopie des Bootsektors enthalten. Danach kommen die FAT–Sektoren. Um eine höhere Sicherheit zu erhalten, werden meist zwei FAT–Tabellen verwendet, für den Fall, dass eine davon beschädigt wird. Die FAT–Tabellen enthalten Informationen über die Cluster–Ketten, die in den einzelnen Dateien gespeichert sind. Nach den FATs folgt das Rootverzeichnis. Es enthält alle Dateien, die im Hauptverzeichnis stehen. (Bei FAT32–Partitionen ist kein Rootverzeichnis vorhanden, es ist durch eine Cluster–Kette ersetzt, deren Anfang im Boot–Sektor steht.) Zuletzt kommen die Datensektoren. Der Bereich ist in einzelne Cluster unterteilt. Jeder Cluster besteht aus ein oder mehreren Sektoren. Der erste Cluster hat die Nummer 2 und beginnt beim ersten Datensektor.

Vor dem immer wieder zitierten *Anfang* eines logischen Laufwerks einer FAT–Partition steht (auf Seite 0) die zugehörige Partitionstabelle.

Der Boot–Sektor

und alles damit Zusammenhängende soll in einem späteren Artikel besprochen werden.

Andere Disk–Editoren

[CW] CWDSKEDT (CW–Diskedit) 2.22 (1997–99) Christoph Walter. Freeware, DOS mit Mausunterstützung. Ohne eingeschalteten Maustreiber geht nichts. Gute Partitionstypen–Liste.
[MK] DISKEDIT 1.2 von Martin Kalisch (1992). DOS oder auch Windows (DOS–Box).

Andere Bootmanager

[ZS] DocsBoot+ 0.25ß. 1994 Zac Schroff. Residiert in Spur 0.
[UB] Ultimate Boot–CD 3.4. Begleit–CD zu PC–Professionell 9/2006.
[PM] XFDISK. Florian Painke und Ulrich Müller.
[ ] Kalle Gerwien: Retter in der Not — GRUB–Boot–Diskette erstellen. easyLINUX 11/2004, S.74–76.

Literatur und Internet–Adressen

[AZ] `members.inode.at/anton.zechner/az/FatFormat.htm`
[TF] Michael Thieser und Andreas Finkler: PC–Festplattenbuch, Markt&Technik– Verlag 1996.
[MR] Ranish Partition Manager liegt u.a. auch auf der Ultimate Boot–CD 3.4 . `http://www.ultimatebootcd.com/`
[CS] Carsten Strotmann: Forth am Stil. VD–Heft 2/2006.
[AF] Alexei Frounze. E–Mail: alexfru@chat.ru. Homepage: `http://alexfru.chat.ru`. Mirror: `http://members.xoom.com/alexfru` . Das Programmpaket von A.F. findet sich unter dem Namen BOOTPROG im PC–WELT–Sonderheft 1/2000. Von FAT32 ist bei A.F. nicht die Rede. Es geht nur um FAT12 (Floppy) und FAT16 (Festplatte).
[JN1] Julian Noble: A Call to Assembly 1–3. Forthwrite 113–115 (September 2001, November 2001, Januar 2002). Der besagte Micro–Mini–Assembler steht im zweiten Teil der Artikelfolge des Autors.
[JN2] Julian Noble: Ein Assembl(i)eraufruf 1–3. Vierte Dimension 2–4/2002. Übersetzung der Artikelfolge des Autors aus der Forthwrite. Übersetzer: Fred Behringer.
[MP1] Marc Petremann e.a.: Manuel de référence de Forth–83–standard. 1987.
[MP2] Marc Petremann e.a.: Turbo–Forth — Manuel d'apprentissage. Paris 1990.
[MP3] Marc Petremann e.a.: Turbo–Forth — Guide de référence. Paris 1990.


```
showparttab

MBR-Partitionstabelle (Kopf 0, Spur 0, Sektor 1, Offset 01BE):

1: 00 01 01 00 16 FE 3F 3F 3F 00 00 00 01 B0 0F 00
2: 00 00 01 80 05 FE FF FF 80 60 1F 00 80 A3 DB 00
3: 00 00 01 40 16 FE 3F 7F 40 B0 0F 00 40 B0 0F 00
4: 80 00 C1 FF 0C FE FF FF AD 9E B2 01 4E 01 B0 00
   |  |  |  |  |  |  |  |  |  |  |  |  |__|__|__|__ Part.-Laenge (in Sektoren)
   |  |  |  |  |  |  |  |  |__|__|__|__________ Anzahl vorausgegangener Sektoren
   |  |  |  |  |  |  |  |_______________________ Nr des letzten Zylinders (0..7)
   |  |  |  |  |  |  |_________ Nr des letzten Sektors (0..5), Zylinder (6..7)
   |  |  |  |  |  |_______________________________________ Nr des letzten Kopfes
   |  |  |  |  |_ 01:FAT12, 04:FAT16<32MB, 05:erw.Part., 06:FAT16>32MB, 07:NTFS
   |  |  |  |______________________________________ Nr des ersten Zylinders (0..7)
   |  |  |___________________ Nr des ersten Sektors (0..5), Zylinder (6..7)
   |  |________________________________________________ Nr des ersten Kopfes (0..5)
   |________ 80:aktive Primaerpartition (Bootpartition - nur eine!), 00:inaktiv

Achtung: little endian!
Die vier Bytes 01 B0 0F 00 in Zeile 1 stellen die Hexzahl 000FB001 dar, usw.
```

Mit der Zeit gehen — Keeping Track of Time

Michael Kalus

Laufzeitmessung mit Gforth; OSX, Linux, Windows.

Neulich begab es sich, dass bei den Projekt–Euler–Versuchen in Gforth ich auch ganz gern die Laufzeiten meiner Lösungsversuche gesehen hätte. Drei Zeiten stellt das OSX dem Gforth bereit: Das Wort `utime` liefert den Stand der Echtzeituhr (realtimeclock) als doppelt genauen Wert in Mikrosekunden auf dem Stack ab. Und mit `cputime` bekommt man zwei Zeiten vom System, `duser` und `dsystem`, beide ebenfalls doppelt genau.

Bei diesen Euler–Kalkulationen kann nun mittels `utime` die Zeit wiedergegeben werden, die seit dem Start eines Algorithmus bis zu dessen Fertigstellung vergangen ist. Und wenn es länger dauert und du inzwischen YouTube, Mail oder andere Vorgänge hattest, geht das natürlich in die Ausführungszeit ein. Der folgend angegebene runtime–test liefert ein kontinuierliches Bild der Zeiten, um dies zu studieren.

Es wird sichtbar, dass die usertime recht unabhängig von anderen Tasks ist, welche im Hintergrund oder sogar im Vordergrund ablaufen mögen, während die systime das nicht ist. Gforth erhält diese Zeiten vom Betriebssystem.

```
: cputime ...
#ifdef HAVE_GETRUSAGE
struct rusage usage;
getrusage(RUSAGE_SELF, &usage);
duser = timeval2us(&usage.ru_\texttt{utime});
dsystem = timeval2us(&usage.ru_stime);
...
```

```
The getrusage() function shall provide measures
of the resources used by the current process
... RUSAGE_SELF Returns information about the
current process.
```

So sollte es nun klarer sein, was `utime` und `cputime` auf den Stack geben und was man damit machen kann. Eine elegante Art, interaktiv damit umzugehen, hat Brian Fox beigesteuert. Sein `elapse` misst gleich die Ausführungszeit einer kompletten Kommandozeile.

Selbstkritik

Während mein Stück Programm kein so gutes Beispiel für Forthcode ist, weil es, ohne Teile faktorisiert zu haben, in einem durch geschrieben ist, hat Brian Fox sehr schön gezeigt, wie man es besser macht. Er hat die funktionellen Teile identifiziert und in eigene Worte gehüllt. Damit wird das letztendlich ausführende Wort in seiner Funktion gut transparent.

Zu meiner Entlastung kann ich nur sagen, wie dieser Lindwurm entstanden ist: Um die `for next`–Schleife herum wurde zunächst in zwei lokalen Variablen die Anfangs– und die Endzeit abgelegt. Dann experimentierte ich mit verschiedenen Ausgabeformaten und schließlich dachte ich, es könnte doch mal ganz instruktiv sein, alle Zeiten wiederholt darzustellen. Das ging mittels copy&paste und etwas Anpassung bequem, und so ist hier nichts weiter faktorisiert.

Forth eignet sich für beide Vorgehensweisen. Mal schnell was ausprobieren und studieren, ohne auf Eleganz zu achten, geht ebenso, wie mit der Zeit stabile gute Tools zu finden. Das ist ein Feature, das mich an Forth immer wieder so fasziniert.

Übrigens, da `utime` die Zeit doppeltgenau in Mikrosekunden angibt, bekommen wir etwas wie *die Zeit, die seit 1970* vergangen ist? Wer weiss es genauer?

```
1213138874991075 / 1000000 / 60 /
60 / 24 / 365 = 38,4683.. years.
```

Quellen

Gforth Manual; gforth-0.6.2.pdf
Usenet Forum: comp.lang.forth

Listing

```
1   \ Keeping track of Time
2   \ Study timers provided using gforth - Mac OSX (PowerBook G4)
3
4   vocabulary test   test definitions
5
6   : runtime-test ( -- )
7     page  0 3 at-xy
8     ."                        time     t2-t1       max       min .s  ratio"
9     0. 0. { D: utime0     D: utime1    }
10    0. 0. { D: usertime0  D: usertime1 }
11    0. 0. { D: systime0   D: systime1  }
12    0. 10000000000. { D: utmax D: utmin }
13    0. 10000000000. { D: usermax D: usermin }
```

```
14    0. 10000000000. { D: sysmax D: sysmin }
15
16    begin   \ permanent display times
17
18      utime    ( -- dutime )           to utime0
19      cputime  ( -- duser0 dsystem0 )  to systime0  to usertime0
20
21        1000000 for ( insert testword here )  next
22
23      cputime  ( -- duser1 dsystem1 )  to systime1  to usertime1
24      utime    ( -- dutime )           to utime1
25
26    \ formatted display of times
27     cr
28     0 0 at-xy ." user: "
29     usertime0 20 ud.r
30     usertime1 usertime0 d-
31     2dup usermax d> if 2dup to usermax then
32     2dup usermin d< if 2dup to usermin then
33     10 ud.r
34     usermax 10 ud.r  usermin 10 ud.r space .s
35     usermin d>f usermax d>f f/ f.
36
37     cr
38     0 1 at-xy ." sys : "
39     systime0 20 ud.r
40     systime1 systime0 d-
41     2dup sysmax d> if 2dup to sysmax then
42     2dup sysmin d< if 2dup to sysmin then
43     10 ud.r
44     sysmax 10 ud.r  sysmin 10 ud.r space .s
45     sysmin d>f sysmax d>f f/ f.
46
47     cr
48     0 2 at-xy ." ut  : "
49     utime0 20 ud.r
50     utime1 utime0 d-
51     2dup utmax d> if 2dup to utmax then
52     2dup utmin d< if 2dup to utmin then
53     10 ud.r
54     utmax 10 ud.r  utmin 10 ud.r space .s
55     utmin d>f utmax d>f f/ f.
56
57
58    key? until
59
60    0 5 at-xy  ;
61
62
63
64  \ Brian Fox in 2008
65  \ Elapsed timer for gforth compatible with Win32Forth
66
67  : ms@  ( -- n )
68         utime drop 1000 / ;
69  0 value start-time
70  : timer-reset    ( -- )
71                   ms@ to start-time ;
72  : .#"            ( n1 n2 -- a1 n3 )
73                   >r 0 <# r> 0 ?do # loop #> ;
```

```
74  : .elapsed      ( -- )
75                  ." Elapsed time: "
76                  ms@ start-time -
77                  1000 /mod
78                    60 /mod
79                    60 /mod 2 .#" type ." :"
80                            2 .#" type ." :"
81                            2 .#" type ." ."
82                            3 .#" type ;
83  : elapse        ( -<commandline>- )
84                  timer-reset interpret cr .elapsed ;
85
86  0 [if]
87  -----
88  Gforth 0.6.2, Copyright (C) 1995-2003 Free Software Foundation, Inc.
89  Gforth comes with ABSOLUTELY NO WARRANTY; for details type `license'
90  Type `bye' to exit
91  include elapsed.fs  ok
92    ok
93  elapse 5000 ms
94  Elapsed time: 00:00:05.000 ok
95  elapse 10000 ms
96  Elapsed time: 00:00:10.001 ok
97  -----
98  Brian Fox
99
100 [then]
101
102 : .. bye ;   words   cr cr
103
104
```

Theo Windges: *Mit der Zeit gehen*
`http://www.theo-windges.de/zeit-katalog.htm`
(Mit freundlicher Genehmigung des Künstlers)

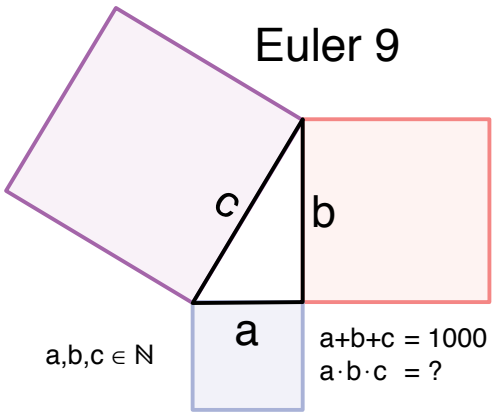
Eine weitere Lösung für Euler 9

Ulrich Hoffmann

Hier eine weitere Lösung für das Problem 9 des Euler–Projekts `http://projecteuler.net/` (vgl. *Projekt Euler - Problem 9* in Heft 2/2008 Seite 25). Diese Lösung kommt ohne das Ziehen der Quadratwurzel aus.

```
1   \ Euler 9                                   uho 2008-08-24
2
3   \ a + b + c = 1000
4   : a_b_c ( a b -- a b c )
5      2dup + 1000 swap - ;
6
7   \ a^2 + b^2 = c^2
8   : pytriple? ( a b c -- flag )
9      >r  dup *  swap  dup *  +  r>  dup *  = ;
10
11  : euler9? ( a b -- flag )
12     a_b_c pytriple? ;
13
14  : euler9 ( -- a b c )
15     500 dup 1 DO
16        dup I DO
17            J I euler9?
18            IF drop  J I a_b_c  UNLOOP UNLOOP EXIT THEN
19        LOOP
20     LOOP drop 0 0 0 ;
21
22  : .solution ( a b c -- )
23      dup IF
24        >r cr ." a=" over .  ." b=" dup .  ." c=" r@ .
25        cr ." a+b+c=" 2dup + r@ + .
26        cr ." a*b*c=" * r> * . EXIT THEN
27      drop drop drop cr ." No solution" ;
28
29  euler9 .solution
```

```
Gforth 0.6.2, Copyright (C) 1995-2003 Free Software Foundation, Inc.
Gforth comes with ABSOLUTELY NO WARRANTY; for details type ‘license’
Type ‘bye’ to exit
  ok
include Euler9.fs
a=200 b=375 c=425
a+b+c=1000
a*b*c=31875000  ok
```



Forth–Gruppen regional

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

München
Bernd Paysan
Tel.: (0 89) – 79 85 57
`bernd.paysan@gmx.de`
Treffen: Jeden 4. Mittwoch im Monat um 19:00, im Chilli Asia Dachauer Str. 151, 80335 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)

Mainz
Rolf Lauer möchte im Raum Frankfurt, Mainz, Bad Kreuznach eine lokale Gruppe einrichten.
Mail an rowila@t-online.de

Gruppengründungen, Kontakte

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

µP–Controller Verleih

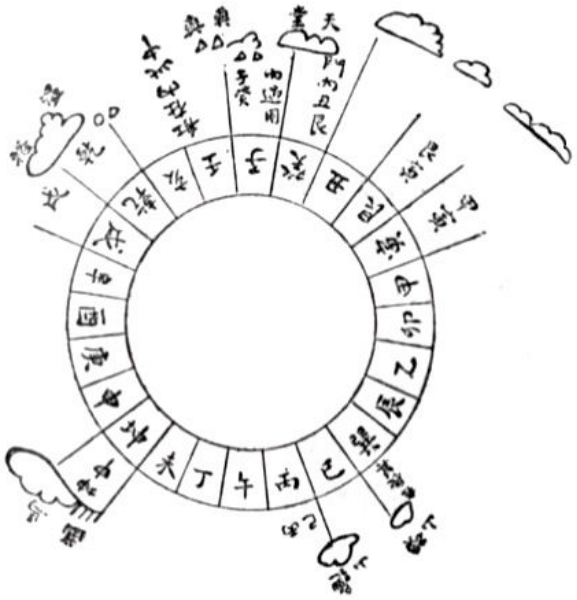
Carsten Strotmann
`microcontrollerverleih@forth-ev.de`
`mcv@forth-ev.de`

Spezielle Fachgebiete

FORTHchips (FRP 1600, RTX, Novix) — **Klaus Schleisiek–Kern**, Tel.: **(0 40) – 37 50 08 03** (g)

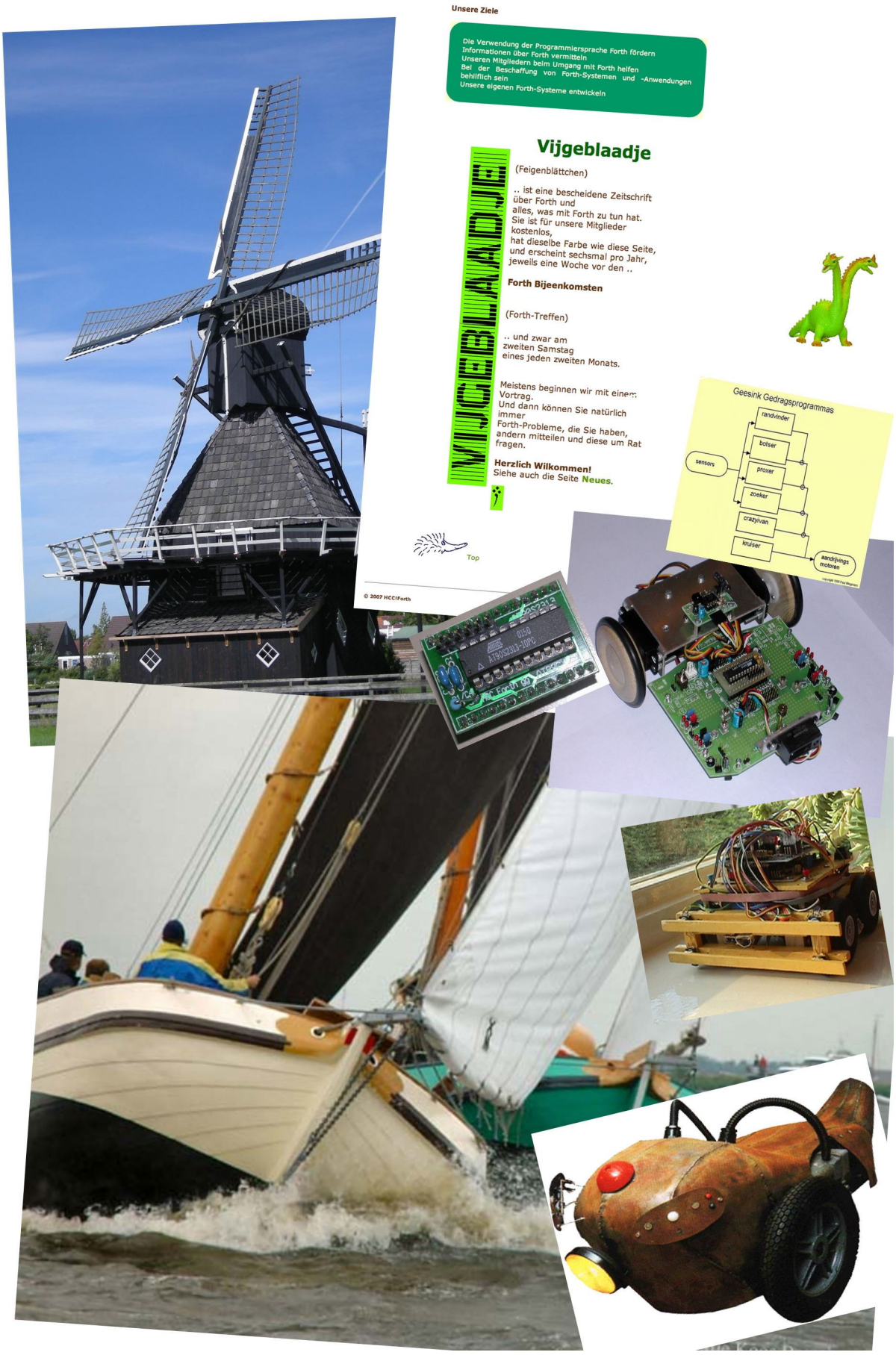
KI, Object Oriented Forth, Sicherheitskritische Systeme — **Ulrich Hoffmann**, Tel.: **(0 43 51) – 71 22 17** (p), Fax: **– 71 22 16**

Forth–Vertrieb
volksFORTH
ultraFORTH
RTX / FG / Super8
KK–FORTH
Ingenieurbüro **Klaus Kohl–Schöpe**
Tel.: **(0 70 44)–90 87 89** (p)



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 Forth–Gesellschaft e.V. und der VD finden Sie im Impressum des Heftes.



Impressionen zu Forth in den Niederlanden