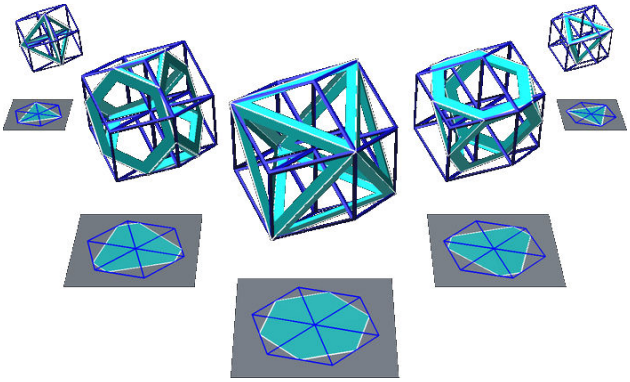
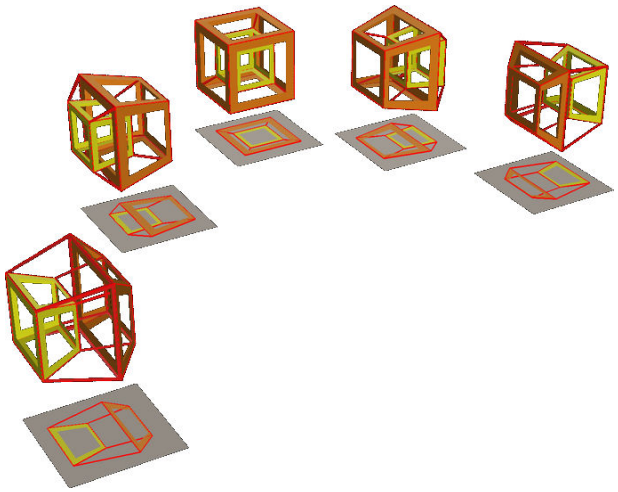
Vierte Dimension

2/2006

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 Forthgesellschaft e. V., zur Verfügung stellen kann.

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

Inhalt:

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

Anfragen bitte an
Martin.Bitter@t-online.de

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

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 Forthgesellschaft 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
Entwicklungsbüro Dr.–Ing. Egmont Woitzel

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

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

Ingenieurbüro
Dipl.–Ing. Wolfgang Allinger

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

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

Ingenieurbüro
Klaus Kohl

Tel.: 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 19 02 25
80602 München
Tel: (0 89) 1 23 47 84
E–Mail: secretary@forth-ev.de
direktorium@forth-ev.de
Bankverbindung: Postbank Hamburg
BLZ 200 100 20
Kto 563 211 208
IBAN: DE60 2001 0020 0563 2112 08
BIC: PBNKDEFF

Redaktion & Layout

Bernd Paysan, Ulrich Hoffmann
E–Mail: vd@forth-ev.de

Anzeigenverwaltung

Büro der Herausgeberin

Redaktionsschluss

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

Erscheinungsweise

1 Ausgabe / Quartal

Einzelpreis

4,00€ + Porto u. Verpackung

Liebe Leser,

Der Interims–Redaktion hat das Erstellen der VD 1/2006 so viel Spaß gemacht, dass sie gleich Material für eine zweite VD gesammelt — und auf dieselbe Weise zu einer VD zusammengesetzt hat. Ein hauptamtlicher, einzelkämpfender Editor ist nach wie vor nicht in Sicht, wird aber in dieser Form auch wohl nicht mehr gebraucht.

Was gebraucht wird, sind Inhalte. Die sollen natürlich von den aktiven Forthern kommen; Aufgabe der Redaktion ist es mehr, den Autoren einen Schubs zu geben, damit die Inhalte auch fließen. So weit klappt das im Moment schon ganz gut, auch wenn manche Lieferanten wie Henry Vinerts etwas verunsichert sind. Deshalb: Keine Angst, eure Beiträge sind selbstverständlich auch weiterhin erwünscht, auch wenn die Umfrageergebnisse andere Artikel bevorzugt haben.

Ach ja, Umfrageergebnisse: Hier ist die magere Ausbeute von 12 Stimmen. Das ist zwar nicht repräsentativ, aber wir werden auch für diese VD wieder eine Umfrage machen.

Artikel	Stimmen
Wie wir diese Vierte Dimension produziert haben	0 (0,00%)
Gehaltvolles	0 (0,00%)
Forth Metaprogramming: Regexps	1 (8,33%)
Forth von der Pike auf — Teil 2	4 (33,33%)
Forth und UTF–8	3 (25,00%)
CSV–Files lesen und verarbeiten	1 (8,33%)
Protokoll der Jahresversammlung 2005	0 (0,00%)
Lebenszeichen	0 (0,00%)
Bericht von der EuroForth 2005	0 (0,00%)
Hexadoku	3 (25,00%)

Die Artikelserie *Forth von der Pike auf* ist recht beliebt, aber es war auch die Sorge zu hören, dass sich die gesamte Serie mit ihren acht Teilen also über zwei Jahre hinziehen würde. Nun — dem haben wir Rechnung getragen und so erscheinen in dieser Ausgabe Teil 3 *und* Teil 4.

Die VD ist aber ein Kommunikationsorgan, und da gehören die Nachrichten aus dem Rest der Forth–Welt ebenso dazu wie die Artikel mit Listings, die unsere Leserschaft offensichtlich bevorzugt.

Damit noch mehr forthige Inhalte den Weg zu den Mitgliedern finden, haben wir auf der Vereins–Website ein Wiki (`http://www.forth-ev.de/wiki/`) installiert. Wer schreiben will, muss sich anmelden, Teilnahme ist aber auf alle Fälle erwünscht.

Bernd Paysan und Ulrich Hoffmann

Die Abbildungen von vierdimensionalen Hypercubes auf der Titelseite sind von `http://alem3d.obidos.org/en/`

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

Die Forth-Gesellschaft wird durch ihr Direktorium vertreten:

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

Kontakte: Direktorium@Forth-ev.de

Leserbriefe

Henry Vinerts über Forth

Dear Fred,

ich wollte dir nur sagen, dass mein Exemplar der Vierten Dimension gestern angekommen ist und ich versucht habe, es in einem einzigen Zug hintereinanderweg durchzulesen. Das geht nicht ganz so schnell, da ich das Deutsch teilweise nur mit dem Wörterbuch in der Hand lesen kann.

Heute will ich deine eigenen Worte aus deiner E–Mail vom 23. Dezember 2005 zitieren: „. . . und ich denke immer, ich muss mich für alles interessieren. Geht natürlich nicht immer."

Es geht auch für mich nicht immer! Ich werde langsam zu alt, Fred, um all das, was ich sehe und lese, zu behalten und einzuordnen.

Hieraus, und auch nach dem Lesen der neuen VD–Ausgabe, resultieren einige Zusatzbemerkungen zu den Antworten, die ich dir auf die Fragen in deiner E–Mail vom 5. Februar gab.

1. Es tut mir Leid, Bernds Editorial entnehmen zu müssen, dass du auf der Tagung 2006 nicht mehr zur Wiederwahl zur Verfügung stehst. Angesichts dessen neige ich dazu, deine Frage, ob ich weiterhin SVFIG–Reports schreiben werde, nicht mehr ganz so mit „gewiss doch" zu beantworten. Ich würde mal sagen, ich werde mich bemühen weiterzuschreiben, solange ich genügend interessantes Material über die SVFIG habe — und solange das neue Direktorium und der VD–Redakteur wirklich haben wollen, dass ich weitermache.

Wie ich dir schon gesagt hatte, hege ich wirklich Zweifel daran, ob sich die VD noch sehr lange wird halten können. Nach dem Durchlesen von Jens Wilkes „Zusammenfassung 2004" auf Seite 23 der VD 1/2006 ist mir nicht mehr ganz klar, wie die Herstellung und die Verteilung der gedruckten Hefte weiterhin in das FG–Budget passen sollen.

Ich muss auch noch auf deine oben wiedergegebene Bemerkung zurückkommen. Es gibt zu viele Dinge, die ich lernen und tun möchte, und Forth steht auf meiner Liste nicht immer ganz oben. Je älter ich werde, desto mehr muss ich mir die Richtungen, in welchen ich tätig werden möchte, sorgfältig einteilen.

2. Um deine zweite Frage (ganz und gar aus meinem eigenen Gefühl heraus) zu beantworten, möchte ich Carsten Strotmann (Seite 7 des Heftes 1/2006) zitieren: „Die gedruckte VD ist auch ideal, um Artikel in Ruhe auf Papier zu lesen." Was mich betrifft, so ist das so ziemlich der einzige Weg, wie ich den größeren Teil überhaupt lesen kann, da ich nicht beabsichtige, mich mit den allermodernsten Hochgeschwindigkeitscomputersystemen und teueren Breitbandübertragungsdiensten auszurüsten, und zudem, da ich Deutsch unbedingt ohne Hast und mit einem Wörterbuch neben mir lesen muss.

Wie ich sehe, sprichst du Bernd und Ulrich Komplimente zu ihrem LaTeX–Produktionsverfahren aus, und ich muss sagen, es ist wirklich gut gelungen. Die Frage ist aber, wie viel Zeit durch die Verwendung von LaTeX eingespart wurde und wie die Lernkurve aussieht, die der neue Redakteur wird durchlaufen müssen, bevor er eben dieses Verfahren wird anwenden können. Und zudem wird sich die Vereinszeitschrift ohne eingehende Artikel, Korrekturlesen und Übersetzungen wohl kaum füllen lassen.

„Alles Gute" to you!
Henry

Einen Tag später (Red.):

Dear Fred,

Ich fühle mich darüber geschmeichelt, dass du meinen Bemerkungen so viel Aufmerksamkeit zollst. Ich habe die Angewohnheit, immer genau das zu sagen, was ich meine, und das habe ich auch diesmal so gehalten. Wenn es der Forth–Sache irgendwie hilft, dann nur zu. Du hast völlige Freiheit, meine Berichte und Bemerkungen in jedweder Form zu verwenden, die dir nützlich erscheint. Nur eines: Füge bitte hinzu, dass die in meinen Zeilen ausgedrückten Meinungen nicht die der Silicon Valley Forth Interest Group (SVFIG) sind, sondern meine eigenen.

With my best wishes,
Henry

Übersetzt von Fred Behringer.

Freds Antwort

Lieber Henry,

ich weiß nicht, was die VD–Leserschaft dazu meint. Wir haben mehrfach versucht, Meinungen über deine Berichte einzuholen. Vergeblich. Aber das Problem der Rückkopplung ist nicht neu. Es bestand ja auch bei der Forthwrite. Ich selbst bin fest davon überzeugt, dass deine Berichte wichtig sind. Schließlich sind sie ja für viele das einzige Mittel, um zu erfahren, mit was sich die Erfinder und Hauptförderer von Forth in den USA zur Zeit beschäftigen. Außerdem sind deine subjektiv verzierten Berichte nicht nur in der hervorragenden Übersetzung von Thomas Beierlein, sondern auch im Original für mich erfrischend zu lesen. Du findest immer die richtigen Worte, das Lesen spannend und erfreulich zu gestalten. Mach, wenn es dir irgend möglich ist, weiter so!

Und was die gedruckte VD betrifft: Ich bin davon überzeugt, dass sie (vorläufig jedenfalls) nicht den Weg der Forthwrite oder der Forth Dimensions gehen wird. Wir halten uns eher an das Vijgeblaadje: Beschränken, Rationalisieren und ein bisschen Enthusiasmus. Wie Ulrich Hoffmann sagt (oder war ich das selbst?): „Einer für alle, alle für einen." Dem FG–Direktorium jedenfalls hat

das Anfertigen der ersten neueren in Gemeinschaftsarbeit herausgegebenen Ausgabe (alles andere als eine *Notausgabe*) so viel Freude bereitet, dass wir spontan beschlossen, zügig weiterzumachen. Material liegt (ungelogen) bis zum Heft 4/2006 schon vor. Selbstverständlich würden wir die Arbeit auch gern an jemanden abgeben, der (oder die) sich als Redakteur/in hervortun und die Verantwortung und vielfältige und vielschichtige Kleinarbeit (ehrenamtlich) übernehmen möchte. Ehrenamtlich! Friederich Prinz hat es uns jahrelang in bester Qualität vorgemacht — und wir weichen davon nicht mehr ab. Damit, Henry, ist auch die Budget–Frage geklärt: Es ist, breite Mitarbeit vorausgesetzt, durchaus machbar, die gedruckte VD in aller Ewigkeit fortzuführen. Wir werden diesen Punkt sicher auf der kommenden Tagung ansprechen. An der Mitgliederzahl scheitert es momentan auch nicht. Die ist, wie es aussieht, gefestigt.

Herzlichen Gruß
Fred

Mikroprozessor R8C/13, Mitgliederliste

Von Rolf Schöne, `secretary@forth-ev.de`

Ave alle,

die Zeitschrift *Elektor* lieferte mit ihrer Ausgabe vom Dezember 2005 ein kostenloses Board mit dem MC R8C/13 aus. Heinz Schnitter hatte die glorreiche Idee, für diesen Winzling ein Forth zu schreiben, um jungen wie älteren Lesern Forth nahezubringen. An den Jungen vor allem sollte uns gelegen sein, damit wir nicht irgendwann aussterben. Wir haben als ältestes Mitglied einen 84–Jährigen. Die beiden jüngsten sind Jahrgang 1982 und taufrische Studenten, aber auch schon 24 Jahre alt.

Dieses Forth für diesen Winzling ist fertig. Heinz Schnitter hat einen Assembler geschrieben, Bernd Paysan das Forth, das im mühsam herauszusuchenden Quelltext unter `http://www.forth-ev.de/wiki/doku.php` (ff.) erhältlich ist.

Mein Vorschlag: Macht es doch Neueinsteigern nicht gar so schwer, gebt ihnen lieber eine .MOT–Datei, die sie mit dem gewohnten FDT auf den MC flashen können. Das Motorola HEX–Record–Format (darum handelt es sich) kennt ihr sicher, seine Erzeugung ist nicht schwer und zum Beispiel in `http://www.buchanan1.net/hex_format.shtml` beschrieben.

Die aktuelle Mitgliederliste wird es mit der VD 3/2006 geben, wenn die Turbulenzen des Jahreswechsels endlich bereinigt sein werden.

Salve, Rolf

N — Das Spiel

Hier etwas das die Forth Hobby Menschen unter uns interessieren könnte:

`http://www.harveycartel.org/metanet/`

Dort findet man eine ausführliche Erklärung wie ein jump&run–Spiel programmiert werden kann. Das Tutorial A (Basic Collision Detection and Response), Tutorial B (Grid-Based Collision Detection and Raycasting) und der Beyond HitTest(): Collision Detection (download slides) sind sehr instruktiv und anschaulich gemacht. Die Autoren meinten dazu ferner:

> "The source code is written in actionscript, but should be easily understandable (and might seem somewhat pseudocode-like) to c or java programmers."

Und ich möchte ergänzen auch in Forth. Und an anderer Stelle:

> "Metanet Software is devoted to creating fun, innovative, unique games. Our first game [N] has been released and is available for download in the games section: it's 100% free, and can be played on Mac or PC. N received the Audience Choice award at the 2005 IGF awards, and the Audience Sparky at the 2006 Slamdance Guerilla Gamemaker Competition. Thanks to everyone who voted! You guys rock. . . . Metanet Software was founded in 2001 by Raigan Burns and Mare Sheppard."

Viel Vergnügen, Michael

Elektor Hexadoku–Wettbewerb

Sehr geehrter Herr Behringer,

herzlichen Glückwunsch! Sie haben mit Ihrer Lösung den Hauptpreis gewonnen: Das E–blocks Starter Kit Professional im Wert von 365 Euro.

Wir wünschen Ihnen viel Freude an Ihrem E–blocks Starter Kit und weiter viel Spaß mit Elektor und dem Hexadoku.

Mit freundlichen Grüßen,

Elektor–Redaktionssekretariat

Antwort von Fred Behringer

Liebe Elektor–Redaktion,

über den Hauptgewinn in Form eines E–Blocks–Starter–Kit–Professionals im Werte von 365,– Euro freue ich mich sehr. Natürlich ist mir klar, dass ich nicht der Einzige bin, der eine richtige Lösung des zweiten Hexadoku–Rätsels eingeschickt hat. Aber umso mehr freue ich mich darüber, dass auch das Glück nachgeholfen und mir einen Anstoß zu weiteren Arbeiten gegeben hat. Ich spiele mit dem Gedanken, mir „von dem gesparten Geld" Ihr neuestes FPGA–Projekt mit Entwicklungsumgebung anzuschaffen.

```
                                                                  Wert Zeile Spalte
    0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F
  +-----------+-----------+-----------+-----------+                 1     D     1
0 | E  4  D  1| 3  0  7  A| C  2  5  6| 9  F  8  B| 0  --> 0        7     D     5
1 | C  6  5  9| 4  D  E  2| 1  8  B  F| 7  0  3  A| 1  --> 0        6     F     4
2 | 3  F  7  B| 1  8  6  5| A  D  9  0| E  2  4  C| 2  --> 0        9     C     4
3 | 8  A  0  2| C  9  B  F| E  3  4  7| 1  6  5  D| 3  --> 0        B     8     B
  +-----------+-----------+-----------+-----------+   - 0  - 0  - 0  - 0  1     B     B
4 | A  8  4  0| 7  6  D  1| 9  B  E  C| F  3  2  5| 4  --> 0        B     1     A
5 | 9  B  3  C| 5  2  4  0| F  7  A  8| D  1  E  6| 5  --> 0        A     2     8
6 | 5  2  E  7| 8  A  F  B| D  1  6  3| C  9  0  4| 6  --> 0        5     A     8
7 | 1  D  F  6| E  3  C  9| 2  5  0  4| 8  A  B  7| 7  --> 0        0     F     1
  +-----------+-----------+-----------+-----------+   - 0  - 0  - 0  - 0  6     A     0
8 | 0  5  1  4| D  E  A  6| 3  9  C  B| 2  8  7  F| 8  --> 0        D     A     3
9 | 7  3  B  E| F  4  9  C| 0  A  8  2| 5  D  6  1| 9  --> 0        8     7     C
A | 6  C  A  D| 2  1  3  8| 5  F  7  E| 4  B  9  0| A  --> 0
B | 2  9  8  F| B  5  0  7| 6  4  D  1| 3  C  A  E| B  --> 0
  +-----------+-----------+-----------+-----------+   - 0  - 0  - 0  - 0
C | 4  7  6  8| 9  F  1  D| B  E  3  A| 0  5  C  2| C  --> 0
D | B  1  C  3| A  7  5  4| 8  0  2  D| 6  E  F  9| D  --> 0
E | D  E  9  A| 0  C  2  3| 4  6  F  5| B  7  1  8| E  --> 0
F | F  0  2  5| 6  B  8  E| 7  C  1  9| A  4  D  3| F  --> 0
  +-----------+-----------+-----------+-----------+   - 0  - 0  - 0  - 0
    0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F
    0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
```

Abbildung 1: Hexadoku — eine Lösung

Wir haben in der deutschen Forth–Gesellschaft (`www.forth-ev.de`) eine Arbeitsgruppe, die alles theoretisch Nötige vorbereitet hat, um aus einem FPGA (von Anfang an war ein ALTERA–Typ in die engere Wahl gezogen worden) einen Forth–Prozessor zu machen. Forth ist bekanntlich die Sprache des Rapid–Prototypings schlechthin, genügend schnell, kompakt und nicht nur compilierend, sondern auch interpretativ einsetzbar.

An Ihrer Zeitschrift Elektor war ich schon immer stark interessiert. (In früheren Zeiten hatte ich auch Ihre Schwesterzeitschrift Elektuur laufend gelesen und später eine ganze Reihe von Arbeiten aus dem Holländischen übersetzt.) Eines muss ich Ihnen lassen: Seit dem Dezemberheft mit dem R8C–Projekt ist mein Interesse an Ihrer Zeitschrift sprunghaft gestiegen.

Auch für den R8C hat sich in der Forth–Gesellschaft eine Arbeitsgruppe gebildet, die sich bemüht, auf dem R8C ein Forth–System hochzuziehen. Die Arbeiten sind, wie mir berichtet wurde, weit fortgeschritten.

Und mit dem Hexadoku, mit dem Sie ja ab der Januar–Ausgabe großen Spürsinn bewiesen haben, haben wir uns in der Forth–Gesellschaft ebenfalls beschäftigt. Martin Bitter, Lehrer an einer Integrativen Schule, hat Hexadoku für sich als Mittel entdeckt, den lernschwachen unter seinen Schülern spielerisch das logische Denken beizubringen. Er hat in unserer vierteljährlich erscheinenden Vereinszeitschrift *Vierte Dimension*, die auch bei unseren Forth–Freunden in Holland, England und Amerika in Rezensionen oder im Original gelesen wird, ein Forth–Programm zur Lösungshilfe veröffentlicht. Ich lasse Ihnen gern ein Heft–Exemplar zuschicken.

Ich danke Ihnen.

Mit freundlichen Grüßen
Fred Behringer

Hexadoku von Martin Bitter

Von Bernd Paysan, `bernd.paysan@gmx.de`

> An alle: Ist das überhaupt geeignet?

Klar ist das geeignet. Ich habe jedenfalls schnell eine Lösung gefunden (siehe Abbildung 1). Ein automatisches Spielen ist mit einem Backtracking–Algorithmus (siehe mein Vier–Gewinnt–Spiel) natürlich möglich. Damit der Computer möglichst schnell fertig wird, braucht er natürlich eine Zug–Heuristik. Gezogen wird zuerst auf den Spielfeldern, die möglichst wenig Freiheitsgrade haben (also erst mal alle mit 2 Möglichkeiten), und dabei werden die Züge bevorzugt, die die niedrigste Zahl automatischer Züge zur Folge haben (da ist die Wahrscheinlichkeit am niedrigsten, in eine Sackgasse zu laufen — man löst nur ein lokales Problem, und hält sich möglichst viel offen) — so mache ich das jedenfalls von Hand.

Ankündigung EuroForth 2006

Anton Ertl

Die 22. EuroForth wird vom 15. – 17. Semptember 2006 im Sidney Sussex College in Cambridge (England) stattfinden. Am Tag davor (also 14. 9. Nachmittag und 15. 9. Vormittag) findet wie letztes Jahr das Forth 200x Standardisierungstreffen statt. Die Konferenzsprache ist Englisch.

Auf der Konferenz werden wie immer Vorträge gehalten und in Workshops interessante Themen nach Wahl der Teilnehmer diskutiert.

Neben dem eigentlichen Konferenzprogramm kann man wie immer ein starkes Begleitprogramm erwarten, z. B. eine Stadtbesichtigung, ein Konferenz–Bankett, und Staken auf dem Fluss Cam. Die EuroForth in Oxford (1997) war die meistbesuchte aller Zeiten.

Call for Papers

Artikel über alle Aspekte von Forth (Anwendung, Implementation, Erweiterung, etc.) und verwandte Themen sind erwünscht.

Artikel können entweder als begutachtete (refereed) Artikel eingereicht werden (academic stream), oder als nicht begutachtete Artikel (industrial stream). Man kann natürlich auch einen Vortrag halten, ohne einen Artikel einzureichen, oder auch einfach so auf die Konferenz fahren.

Artikel sollen per Email (an Anton Ertl, `anton@mips.complang.tuwien.ac.at`) eingereicht werden, und zwar in Form von Postscript– oder PDF–Dateien. Das Format für die Endversion ist A4 mit 25mm Rand auf allen Seiten, unnummeriert.

Das Programm–Kommittee, das die Begutachtung der begutachteten Artikel vornimmt, besteht derzeit aus:

- M. Anton Ertl, TU Wien (Vorsitz)
- Phil Koopman, Carnegie Mellon University
- Jaanus Pöial, Estonian Information Technology College, Tallinn
- Bradford Rodriguez, T–Recursive Technology
- Reuben Thomas
- Sergey N. Baranov, Motorola ZAO, Russia
- Ulrich Hoffmann, Heidelberger Druckmaschinen AG

Daten

28. 6.	Einreichfrist für zu begutachtende Artikel (academic stream)
26. 8.	Benachrichtigung, ob der begutachtete Artikel angenommen wurde
4. 9.	Einreichfrist für die Endversion bzw. für nicht begutachtete Artikel
14. 9. – 15. 9.	Forth 200x Standardisierungstreffen
15. 9. – 17. 9.	EuroForth

Die Einreichfristen sind streng; zu spät eingereichte Artikel können nicht mehr bearbeitet werden.

Adressen

Aktuelle Informationen zur Konferenz finden sich auf ihrer Homepage (`http://dec.bournemouth.ac.uk/forth/euro/ef06.html`). Den Call for Papers und Informationen über das Einreichen von Artikeln findet man auf `http://www.complang.tuwien.ac.at/anton/euroforth2006/`. Informationen über Forth 200x und das Standardisierungstreffen findet man auf `http://www.forth200x.org/forth200x.html`.

Die Konferenz wird von Janet Nelson (`jen@micross.co.uk`) organisiert. Artikel sollten bei Anton Ertl (`anton@mips.complang.tuwien.ac.at`) eingereicht werden. Es gibt eine Mailing–Liste für die Konferenz (`http://groups.yahoo.com/group/euroforth/`), für die man sich über `euroforth-subscribe@yahoogroups.com` anmelden kann.

Vorschau auf das Programm der Forthtagung im Mai 2006

Michael Kalus

Das Programm hat allmählich Form angenommen und ich danke allen, die schon einen Beitrag angekündigt haben. Vorgesehen sind Vortragsblöcke ab Freitag Nachmittag sowie am Samstag. Die Reihenfolge liegt noch nicht fest und kann, wie es Tradition ist, ja auch im Verlauf abgewandelt werden. Bitte weitere Themenvorschläge zusenden an: `michael.kalus@onlinehome.de`

Angekündigt sind bisher (jüngste Meldung oben auf dem Stack):

Vorträge

- *Forth 200x;* Ertl
- *Der Forth–Stammbaum;* Ertl
- *Status VolksForth;* Strotmann
- *VolksForth–MSDOS direkt vom USB Stick booten;* Strotmann
- *Zugriff auf USB–Flash–Speicher in Embedded Systemen;* Strotmann
- *Making of "swappi";* Bitter
- *Flags, Interrupts und Semaphore — auf der Suche nach Vereinheitlichung;* Schleisiek
- *uCore;* Schleisiek
- *Tsunami;* Schleisiek
- *T4, ein neuer Ansatz für embedded Forth;* Völker
- *Forth–eV wiki;* Paysan
- *Galois–Felder und Kodierung;* Storjohan
- *PIC, die neue 16– und 32–Bit–Familie;* Peter
- *Debug a Forth Application;* Krüger
- *forth–ev.de, Stand der Dinge;* Kalus

Workshops

- *Rund um die VD;* Paysan, Hoffmann
  Die Technik rund um die derzeitige VD wird diskutiert. Ziel ist auch, das Redaktionsteam zu vergrößern.
- *forth–wiki;* Paysan, Kalus
  Der Aufbau des wiki wird diskutiert. Brainstorming zur Verzeichnisstruktur. Ausbauplan des Inhalts.
- *Forth–Projekt Elektor–R8C;* Paysan, Schnitter
- *Forth für ein embedded system selber machen;* Hoffmann und Peter
  Es wird nachgedacht darüber, welche Strukturen Forth ausmachen und welche Word–Sets mindestens gebraucht werden, um eine Steuerung zu realisieren. Und welche Word–Sets dazukommen sollten, um die Steuerung debuggen zu können. Schließlich wird die Fähigkeit bedacht, das System auch zu erweitern. Multitasking ist im Ansatz zu berücksichtigen.

Exkursionen

- Wanderung durch das Muttental zur Zeche Nachtigal.
- Abbau und Streckenvortrieb untertage in einem Bergwerk (auf 6 begrenzte Teilnehmerzahl)

Im Beiprogramm (optional) ist angedacht:

- Wuppertaler Schwebebahn.
- Altstadt Hattingen.
- Burg Blankenstein oder Essen im Schloss Kemnade.
- und auf besonderen Wunsch das *Varieté et cetera* mit: *Das Wunder von Bochum* (Abends, da können alle mit!)

Am Sonntag ist wieder die Vereinsversammlung.

Wichtige TOPs sind schon mal:

- Vorstandswahl
- Redaktion der VD

Natürlich tagt auch wieder der Drachenrat und dabei wird wieder der letzte Hüter des Swap ... aber lassen wir das.

mka

Umwandlung von HEX in BCD–Zahl in High–Level–Forth
Oder: Schlag nach bei forthwiki.

Bernd Paysan, Michael Kalus

Dieser kleine Beitrag geht auf einen thread in `de.comp.lang.forth` zurück. Gleichwohl etliche Mikroprozessoren eine Instruktion haben, die diese Umwandlung unterstützt, wird exemplarisch gezeigt, was zu tun ist, wenn eine solche Unterstützung fehlt oder aus Gründen der Portabilität des Forth solche Instruktion gar nicht genutzt werden soll.

Ausgangspunkt war die Frage, ob ein minimaler Forthkern für Steuerungen mit einer spartanischer Ausgabe in Hex auskommt, oder ob die Umwandlung in eine andere Zahlenbasis nötig ist, und ob dann die Division implementiert sein muss. B. Paysan zeigte, wie eine dezimale Zahlenausgabe ohne Divison in High–Level–Forth gemacht werden kann.

Nehmen wir ein 32–Bit–PC–Forth als Grundlage. (Bei 64 Bit muss man die Maske halt mit 16ern/8ern/4ern/2ern auffüllen). Ihr findet den Code auch im Forth–Wiki.

2* und bcd2*

Wenn Du das `bcd2*` in Listing 1 durch ein ganz ordinäres `2*` ersetzt, siehst Du, was da eigentlich passiert: Du schiebst die Bits von der einen Zahl in die andere Zahl. Durch das `bcd2*` änderst Du dabei die Codierung der Zahl von binär zu BCD. Wenn man — wie hier — den `bcd2*` emulieren muss, ist das kein großer Gewinn gegen die Division, aber wie gesagt: Auf fast allen Controllern ist das nicht nötig, weil die ja so einen BCD–Befehl bereits eingebaut haben.

Exkurs: Die BCD–Codierung

Um eine Zahl als BCD–Zahl darzustellen, wird jede dezimale Ziffer (0 bis 9) durch jeweils 4 Bit, also ein Halbbyte (Nibble), im Dualsystem dargestellt (0000_2 bis 1001_2). Die übrigen sechs Werte, die mit 4 Bit darstellbar sind (1010_2 bis 1111_2), stellen keine gültigen BCD–Zahlen dar (Pseudotetraden). Sie werden in manchen Systemen zur Codierung von Vorzeichen, Überträgen oder Kommata verwendet.

Zur Codierung von Zahlen mit mehr als einer Dezimalziffer werden die BCD–Darstellungen der einzelnen Ziffern hintereinander gesetzt (zum Beispiel wird die Zahl 2687 als 0010 0110 1000 0111_2, beziehungsweise ohne trennende Leerzeichen als 0010011010000111_2 dargestellt). Mit einem Byte (8 Bit) können also zwei Dezimalziffern dargestellt werden. Werden die 4 Bits einer BCD–Zahl jeweils in den niederwertigen Bits codiert und die restlichen 4 Bits mit Nullen aufgefüllt, so spricht man von einer ungepackten BCD–Zahl. Werden beide Hälften eines Bytes mit je einer BCD–Zahl belegt, so nennt man dies entsprechend eine gepackte BCD–Zahl.

Quellen:
Bernd Paysan in `de.comp.lang.forth`; Mo 13 Mär. 2006 14:30
Wikipedia, *BCD-code*

mka

```
1   decimal
2   : hex. ( x -- )  36 emit  base @  swap hex u.  base ! ;
3
4   : bcd2* ( bcd1 -- bcd2 )
5     dup $88888888 and 2/ dup 2/ or swap 2* swap + \ forward carry correction
6     dup $88888888 and over $44444444 and 2 pick $22222222 \ BCD correction
7     and 2* or 2* and 2/ dup 2/ or + ;
8
9   : hex>bcd ( u -- bcd )
10    0 8 cells 0 DO  bcd2* over 0< - swap 2* swap  LOOP  nip ;
11
12  \ test-code:
13  \ decimal 12345678  hex .s \ Ausgabe: <1> BC614E  ok
14  \ hex>bcd hex.             \ Ausgabe: $12345678  ok
15  \
16  \ decimal 98765432  hex .s \ Ausgabe: <1> 5E30A78  ok
17  \ hex>bcd hex.             \ Ausgabe: $98765432  ok
```

Listing 1: Umwandlung nach BCD

Forth am Stil — Teil 1

Carsten Strotmann

Die Idee

Anfang dieses Jahres veröffentlichte die Zeitschrift c't zwei interessante Artikel über das Installieren und Starten von Windows[1]– und Linux[2]–Betriebssystemen von einem USB Stick. Diese Artikel regten mich zu der Überlegung an, ob nicht auch ein kleines Forth–System direkt von einem USB–Stick gestartet werden kann. Es wäre doch praktisch, ein mobiles Forth *in der Tasche* mitzunehmen und direkt starten zu können (ohne dass erst ein umfangreiches Betriebssystem wie Unix/Linux oder Windows gestartet werden muss).

USB–Sticks sind aus Kompatibilitätsgründen mit FAT16– oder FAT32–Dateisystem formatiert, da diese Dateisysteme von einer Vielzahl von Betriebssystemen unterstützt werden (DOS, Windows 9x, Windows NT/2000/XP, Linux, Solaris, MacOS 9/X, OS/2, ...). Das FAT–Dateisystem wurde 1977 von Bill Gates und Marc McDonald für das Microsoft–Disk–BASIC entworfen, speziell zur Verwaltung der zur damaligen Zeit erhältlichen Disketten (160 KB). Nachdem das FAT–Dateisystem Anfang der 80er Jahre in MS–DOS eingebaut wurde und im Laufe der Zeit die Disketten, und später die Festplatten, immer größer wurden, musste auch das Dateisystem an die neuen Speichergrößen angepasst werden. Wie oftmals bei *historisch gewachsenen* Systemen geschah dies nicht ohne Verluste. Obwohl FAT32 die neuste Variante des FAT–Dateisystems ist (FAT32 wurde Mitte der 90er Jahre mit Windows 95 OSR2 eingeführt), merkt man dem System noch die Vergangenheit als Disketten–Dateisystem an. Im FAT–Dateisystem werden Sektoren zu Clustern zusammengefasst. Die Anzahl der Sektoren pro Cluster hängt von der Gesamtkapazität des Speichermediums ab. Da USB–Speichersticks vom PC wie Festplatten verwaltet werden, befindet sich am Anfang des Mediums ein Bootsektor mit der Partitionstabelle. Dieser Bootsektor mit Partitionstabelle (512 Byte) verbraucht jedoch den vollen ersten Cluster eines mit FAT32 formatierten Mediums, da die FAT–Partition erst am Anfang des zweiten Clusters beginnen kann.

Auf meinem Test–USB–Stick (Apple iPod Shuffle 512MB) beginnt die FAT Partition auf Sektor 45. Abzüglich des Bootsektors sind das 44 Sektoren zu je 512 Byte, die nicht genutzt werden und auf dem USB–Stick brachliegen. Platz genug, um dort ein Forth mitsamt den Zusatzprogrammen zu *verstecken*.

[1] Windows: XP von USB–Laufwerken booten, S. 188, c't 2/2006

[2] Linux auf dem USB–Stick installieren, S. 212, c't 3/2006

[3] `http://alexfru.chat.ru/epm.html#bootprogs`

[4] `http://home.teleport.com/~brainy/interrupts.htm`

Der Plan:

- ein PC–*real–mode*–Forth, bootbar direkt von USB–Sticks
  - zuerst VolksForth 8086
  - später „Begin–Again“ (OpenBIOS)
- kompatibel mit aktueller PC–Technologie und auf Intel–CPU basierenden Apple–Macintosh–Rechnern
- Forth–Unterstützung für den Zugriff auf FAT32–Laufwerke
  - grundlegende Dateiverwaltungsbefehle wie `copy`, `delete` und `rename`
  - ASCII–Textbetrachter
  - ASCII–Texteditor
  - Forth–Entwicklungssystem
- Forth und Hilfsprogramme *versteckt* in ungenutzten Sektoren des FAT–Dateisystems

Der Bootloader

Das Forth–System soll direkt vom Bootsektor aus geladen werden. Das Bootprogramm im Bootsektor muss daher die Partitionstabelle lesen und das Forthsystem in den nicht von Partitionen belegten Sektoren finden. Dies ist meist einfach, da die freien Sektoren direkt mit Sektor 1 nach dem Bootsektor beginnen. Wer im Internet nach Bootloadern für die PC–Plattform sucht, wird schnell an mehreren Stellen fündig, inklusive Beschreibungen und Quellcode. Ich habe mich bei diesem Projekt für den Bootloader von Alexei Frounze[3] entschieden. Dieser Bootloader kann MS–DOS–COM– und –EXE–Programme direkt laden und starten, was die Entwicklung beschleunigt, da VolksForth schon als MS–DOS–COM–Datei vorliegt und nicht jedesmal die Forth–Programmdatei in die freien Sektoren kopiert werden muss, sondern das System auch direkt aus dem Dateisystem *gebootet* werden kann.

Auf die Sektoren des USB–Speichers muss mit Hilfe der erweiterten BIOS–Schnittstellen[4] für Festplatten zugegriffen werden, da in den meisten BIOS–Versionen die regulären BIOS–Schnittstellen nur auf Festplatten und nicht auf USB–Speicher zugreifen.

VolksForth–Anpassungen

Das VolksForth wird nun direkt von Bootloader im Bootsektor geladen (wie Forth–Systeme in der „guten alten Zeit“). Daher mussten alle Abhängigkeiten zu MS–DOS aus dieser Version des VolksForth entfernt werden. Das Forth–System wird direkt, Sektor für Sektor, auf den USB Stick zugreifen und das FAT32–Dateisystem mitsamt der Dateistrukturen interpretieren. Derzeit (Stand März 2006) kann die Partitionstabelle und der FAT–Bootsektor gelesen und interpretiert werden. Auch können Verzeichnisse gelesen und aufgelistet werden, sowie Dateien geöffnet werden.

Da der Treiber für das FAT32–Dateisystem auch für das MicroUSB[5]–Projekt verwendet wird, ist dieser Treiber unabhängig vom VolksForth gehalten und kann als Modul auch in anderen Forth–Systemen und Projekten eingesetzt werden.

Entwicklung

Programme, die direkt über BIOS–Funktionen auf die Festplatte zugreifen, sollten nicht direkt auf dem Entwicklungsrechner ausgetestet werden. Neben den langen „Turn–Around“ Zeiten (Reboot, USB–Boot Test, Betriebssystem laden, Forth–Kernel ändern, auf USB–Stick speichern, Reboot ...) kann auch ein Programmierfehler die gesamte Festplatte durcheinanderwürfeln und wichtige Dateien zerstören. Aus diesem Grund führe ich die Programmtests innerhalb einer VMWare Workstation[6] VirtuellenMaschine durch. VMWare simuliert eine virtuelle PC–Hardware innerhalb eines gestarteten PC–Betriebssystems. Programme innerhalb der VMWare können nicht auf die Festplatte des Wirt–Systems zugreifen. Für die Entwicklung kann auch der kostenlose VMWare Player verwendet werden. Das VMWare–BIOS unterstützt USB–Sticks sowie die erweiterten BIOS–Funktionen zu Lesen und Schreiben von Sektoren. Leider kann VMWare derzeit noch nicht von USB–Speichern booten. Für den Test der Programme boote ich daher das modifizierte VolksForth von Diskette und greife dann auf den USB–Speicher zu. Verläuft der Test erfolgreich, so wird das VolksForth von der Diskette auf den USB–Stick übertragen und in einem richtigen PC gebootet. Für den Boot von einem USB–Stick müssen ggf. die Boot–Einstellungen im BIOS angepasst werden, bei einigen BIOS–Versionen muss auch „USB–Keyboard“[7]–Unterstützung aktiviert werden. Informationen zum Booten von USB–Speichern finden sich im Internet[8].

Ausblick

Bisher ist das VolksForth angepasst und kann sowohl von Diskette wie auch vom USB–Stick gebootet werden. Derzeit arbeite ich an den FAT32 Routinen. Mein Plan ist es, bis zum Sommer eine fertig benutzbare Version auf der VolksForth–Webseite[9] bereitzustellen. Über die weitere Entwicklung werde ich im zweiten Teil dieses Artikels in der VD 3/2006 berichten.

[5] `http://www.microusb.org/`

[6] `http://www.vmware.com/`

[7] `http://rz-obrian.rz.uni-karlsruhe.de/knoppix-usb/`

[8] `http://www.weethet.nl/english/hardware_bootfromusbstick.php`

[9] `http://volksforth.sf.net/`

Forth von der Pike auf — Teil 3 und 4

Ron Minke

Die hier mit freundlicher Genehmigung der HCC–Forth–gebruikersgroep wiederzugebende achtteilige Artikelserie erschien in den Jahren 2004 und 2005 in der Zeitschrift *Vijgeblaadje* unserer niederländischen Forth–Freunde. Übersetzung: Fred Behringer.

Hier sind nun der dritte und vierte Teil des Versuchs, ein Forth–System auf die Beine zu stellen, dessen Voraussetzung überhaupt nix, oder auf gut Deutsch *from scratch*, lautet.

Nach den drei Entscheidungen, die wir im zweiten Teil getroffen haben, wollen wir uns jetzt dafür interessieren, wie wir Daten auf den Returnstack legen können. Für Returnstack–Operationen gibt es im AVR–Prozessor eigens zwei Maschinencode–Befehle:

`PUSH`	legt Daten auf den Returnstack
`POP`	holt Daten vom Returnstack

Diese Befehle arbeiten so, als wenn in einem Unterprogrammaufruf CALL (in Pseudocode) stünde:

```
PUSH:
   MOV (AVR_SP), Rn     lege Daten von Register Rn ab
   DEC AVR_SP           zeige auf die neue leere Stelle

POP:
   INC AVR_SP           erhöhe Pointer, um an die Daten zu kommen
   MOV Rn, (AVR_SP)     kopiere Daten, lass alten Wert stehen
```

Wie wir sehen, zeigt der Pointer auf einen freien Platz. Wir hatten uns aber vorgenommen (Entscheidung 3), genau das in unserer virtuellen Forth–Maschine **nicht** zu tun. Doch noch ist nicht alles verloren: Mit den oben stehenden PUSH– und POP–Befehlen können wir alle Returnstack–Operationen verwirklichen. Innerhalb der virtuellen Forth–Maschine ist es weniger von Belang, *wo* der Pointer genau hinzeigt, solange wir nur an die Daten kommen. Wollen wir tatsächlich wissen, wo der Pointer hinzeigt, dann müssen wir den Offset zwischen dem angezeigten freien Platz und dem Platz der eigentlichen Daten in Rechnung ziehen. Zum Glück beträgt dieser Offset nur 1 Byte. Unser Vorgehen, für den Forth–RP den AVR–SP zu wählen, stellt sich also als gangbar heraus.

Entscheidung 4: Das Forth–RP wird dem AVR–SP–Register zugewiesen.

Der Interpreter–Pointer

Das nächste Register der virtuellen Forth–Maschine, für das wir eine AVR–Lösung suchen wollen, ist der Interpreter–Pointer. Der IP zeigt auf den nächsten Befehl (Forth–Definition, Wort) der in unserem Forth–Programm ausgeführt werden soll. Der IP steuert die Reihenfolge der Ausführung auf dieselbe Weise, wie der Maschinenprogramm–Counter PC die Reihenfolge in einem Assembler–Programm bestimmt. Das wichtigste Codestück, das den IP verwendet und steuert, ist das Stück Code für NEXT (lesen Sie sich Teil 1 dieser Serie noch einmal durch). Sodann wird IP unter anderem in den Codeteilen verwendet, die Sprunganweisungen ausführen (BRANCH, LOOP u. dgl.). Der Vollständigkeit halber wiederholen wir das Stück Pseudo–Assembler–Code für NEXT:

`NEXT:`	IP zeigt auf das als nächstes auszuführende Wort
`MOV W,(IP)`	kopiere den Inhalt von IP, die CFA des als nächstes auszuführenden Wortes, ins W-Reg.
`INC IP`	Lass IP auf das Wort *nach* dem momentanen Wort zeigen, um dann dort ohne Umwege fortzufahren.
`MOV PC,(W)`	Führe den Code–Interpreter, dessen Adresse nun im W–Register sitzt, aus. Dieser Wert kommt aus dem Codefeld des momentan auszuführenden Wortes (indirekter Sprung).

Welches der AVR–Registerpaare W, X, Y und Z können wir hier nun am vorteilhaftesten einsetzen? Die Wahl wird eigentlich durch die letzte Zeile im Pseudoassemblercode bestimmt:

```
MOV PC,(W)
```

Hier wird indirekt auf eine Adresse gesprungen, die im Virtuellen–Forth–Register **W** steht. Das einzige AVR–Registerpaar, das indirekte Sprünge zulässt, ist das Z–Registerpaar R30–R31 (Genaueres findet man im Befehlssatz auf der Atmel–Website). Wir benötigen das Z–Registerpaar als Zwischenschritt, um indirekt springen zu können. Leider verfügt der AVR nicht über Maschinenbefehle, die ein Registerpaar in einem einzigen Zug laden können. Also müssen wir das in zwei Schritten tun. Hierzu definieren wir **ZL** (= R30) als den unteren Teil (low) des Z–Registerpaares und **ZH** (= R31) als den oberen Teil (high). Wir brauchen auch ein W–Registerpaar zur Zwischenlagerung. Dafür wollen wir vorläufig das W–Registerpaar des AVRs wählen. Ob diese Entscheidung richtig war, werden wir später sehen (dass sich im AVR–Chip ein Registerpaar befindet, das auch W heißt, ist Zufall). Das W–Registerpaar spalten wir in zwei Teile auf, **WL** (= R24) und **WH** (= R25).

Wenn wir den gesamten Pseudocode für NEXT hinschreiben, bekommen wir:

Pseudocode		Assemblercode	NEXT–Routine
MOV W,(IP)	(1)	`Ld WH,Rn`	indirekt hereinholen
INC IP	(2)	`Inc Rn`	
	(3)	`Ld WL,Rn`	indirekt hereinholen
	(4)	`Inc Rn`	
	(5)	`Mov Raaa,WH`	Pointer setzen
	(6)	`Mov Rbbb,WL`	
MOV PC,(W)	(7)	`Ld ZH,Raaabbb+`	indirekt, auto incr
	(8)	`Ld ZL,Raaabbb`	
	(9)	`IJmp`	indirekter Sprung

Uff, das ist ein schönes Stück Code! Um es in den Griff zu bekommen, nochmal alles der Reihe nach. Die Assembler–Befehle in den Zeilen 1–4 holen indirekt die Stelle herein, wo die *nächste* Befehlsdefinition zu finden ist. Gleichzeitig wird der Pointer so erhöht, dass er auf die darauffolgende Definition zeigt. Die Zeilen 5 und 6 kopieren den hereingeholten Wert in ein Registerpaar Raaabbb. Das kann das X–, das Y– oder das Z–Registerpaar sein. Auf welches die Wahl fällt, werden wir gleich sehen. Da unser Forth eine indirekt gefädelte Version (das klassische Modell) ist, müssen wir abermals einen indirekten Wert hereinholen, wenn wir erfahren wollen, wo der eigentliche Maschinencode für diese Definition steht.

Die Zeilen 7 und 8 holen diesen indirekten Wert herein und legen ihn in das Z–Register. Wissen Sie es noch (Entscheidung 2): Erst das obere Byte hereinholen, dann das untere. Man beachte, dass dabei von der Auto–Inkrement–Funktion Gebrauch gemacht wird, so dass wir das Registerpaar Raaabbb nicht selbst zu erhöhen brauchen. Und schließlich wird über den Befehl IJmp in Zeile 9 der indirekte Sprung vollzogen. Der zum gerade auszuführenden Forth–Wort gehörende tatsächliche Maschinencode macht sich nun an seine Arbeit.

Wir müssen noch festlegen, welches der in Frage kommenden Registerpaare X, Y und Z wir für das eben verwendete Registerpaar Raaabbb einsetzen wollen. Dabei müssen wir auch daran denken, dass die Register IP und SP der virtuellen Forth–Maschine noch endgültig festgelegt werden müssen. Große Auswahl haben wir eigentlich nicht, das Z–Registerpaar haben wir bereits für indirekte Sprünge verwendet. Bleiben für die Zuweisung an IP und SP das X– und das Y–Registerpaar übrig (welches zu welchem, werden wir sogleich sehen). Zur Darstellung von Raaabbb bleibt also nur noch das Z–Registerpaar übrig. ??? Aber das Z–Registerpaar hatten wir ja gerade verwendet...??? Wir befreien uns aus dieser Situation, indem wir *beides* tun!

Entscheidung 5: Wir verwenden das Z–Registerpaar für Raaabbb *und* auch für indirekte Sprünge.

Durch wohlüberlegten Umgang mit dieser Kombination entsteht der folgende AVR–Maschinencode (wir benötigen dabei allerdings zwei Zwischenregister, für welche wir R0 und R1 nehmen).

Pseudocode		Assemblercode	Next–Routine
MOV W,(IP)	(1)	`Ld WH,Rn`	indirekt hereinholen
INC IP	(2)	`Inc Rn`	
	(3)	`Ld WL,Rn`	indirekt hereinholen
	(4)	`Inc Rn`	
	(5)	`Mov ZH,WH`	Pointer setzen
	(6)	`Mov ZL,WL`	
MOV PC,(W)	(7)	`Ld R0,Z+`	ind, oberes Byte, auto incr
	(8)	`Ld R1,Z`	ind, unteres Byte
	(9)	`Mov ZL,R1`	kopiere (Z) zurück nach Z
	(10)	`Mov ZH,R0`	
	(11)	`IJmp`	indirekter Sprung

Wie sich herausstellt, gibt es im Befehlssatz der **MEGA**–AVR–Prozessorserie einen Befehl, der ein Register**paar** in einem einzigen Zug kopieren kann, so dass die Zeilen 5 und 6, bzw. 9 und 10 zu einem einzigen Mov zusammengefasst werden können. Für die kleinen AVR–Prozessoren trifft das jedoch nicht zu.

Wir können aber auch von den Zwischenregistern noch etwas abknapsen. Bedenkt man, dass ZL auch ein gewöhnliches Register ist (und zwar R30) und dass man es auch als solches verwenden kann, können wir eines der Zwischenregister einsparen. Der Code:

MOV PC,(W)	(7)	`Ld R0,Z+`	ind, oberes Byte, auto incr
	(8)	`Ld ZL,Z`	ind, unteres Byte
	(10)	`Mov ZH,R0`	kopiere nur das obere Byte
	(11)	`IJmp`	indirekter Sprung

ist erfreulicherweise einen Befehl kürzer! Und angesichts der Tatsache, dass NEXT das am häufigsten gebrauchte Stückchen Forth–Code ist, nehmen wir das auch noch gern mit. Das Register R0 dient nach wie vor als Zwischenregister.

In der vorigen Folge hatten wir versucht, den Pointern IP und SP je ein AVR-Registerpaar zuzuordnen. Beim Untersuchen der Möglichkeiten dazu hatten wir die Verwendung des AVR-Z-Registers festgelegt. Wir verwenden es als Notizblock, als einen Platz zum schnellen Zwischenspeichern, mit dem eigentlichen Ziel, einen indirekten Sprung auszuführen.

Zuordnung der Forth-Register SP und IP

Dann wird es nun also Zeit, uns zu überlegen, welche Register wir den Pointern IP und SP zuordnen können. Wir haben noch die AVR–Registerpaare W, X und Y übrig. Für den Zugriff auf die Worte (à 16 Bits) im gesamten Forth–System wäre eine Auto–Inkrement–Funktion bequem. Das AVR–System ist 8 Bits breit, so dass wir die Daten so oder so in zwei Schritten hereinholen müssen. Das Register W hat keine Auto–Inkrement–Funktion, fällt also weg. Sodann würde es uns sehr zupass kommen, wenn wir bei Zugriffen auf den Datenstack nicht nur das oberste Element (eigentlich ja das unterste, der Stack steht Kopf) erreichen könnten, sondern auch die Daten von den Elementen weiter oben auf dem Datenstack. Der AVR–Befehlssatz hat dafür vorgesorgt: Das Y– und das Z–Registerpaar können (in beschränktem Umfang) auch

Daten mit einem Extra–Offset hereinholen. Und das ohne Rechenleistung. Der Offset sitzt ganz *normal* im Opcode. Das Z–Registerpaar haben wir bereits vergeben. Bleibt uns also das Y–Registerpaar. Es folgt ein Beispiel, um das noch etwas deutlicher zu machen:

Stackposition	Wert
5	Wort 3 unteres Byte
4	Wort 3 oberes Byte
3	Wort 2 unteres Byte
2	Wort 2 oberes Byte
1	Wort 1 unteres Byte
SP → 0	Wort 1 oberes Byte

Der Datenstack–Pointer SP zeigt auf einen Platz im RAM–Speicher. Wie vereinbart, steht dort das obere Byte eines Wortes. Wir greifen etwas vor und setzen Y auf den Wert von SP. Dieser Wert wird für die momentanen Erklärungen als Basiswert festgehalten. Mit dem Maschinenbefehl

```
Ld R4,Y
```

holen wir uns das obere Byte von Wort 1 und legen es ins Register R4. Und jetzt, ohne Extraberechnung: Mit dem Maschinenbefehl

```
Ldd R5,Y+3
```

laden wir auf einen Schlag das untere Byte von Wort 2 ins Register R5. Der hier verwendete Offset von 3 wird im Opcode automatisch verarbeitet. Es liegt nun also sehr nahe, dem Forth–Datenstack–Pointer SP das Registerpaar Y zuzuordnen... Das ist am Ende jener Platz, mit welchem das gesamte Forth–System arbeitet: Das System ist stack–orientiert.

Entscheidung 6: Der Forth–Datenstack–Pointer SP wird dem AVR–Registerpaar Y zugeordnet.

Nun haben wir nur noch den Interpreter–Pointer IP übrig. Und es bleiben nicht mehr viel AVR–Registerpaare zu verteilen...

Wir hatten bereits gesehen, dass eine Auto–Inkrement–Funktion für einen Pointer außerordentlich bequem ist. Für das Registerpaar, das wir für IP verwenden wollen, wäre diese Funktion auch sehr willkommen. AVR–Registerpaare mit Auto–Inkrement–Funktion sind X, Y und Z. Davon haben wir das Y– und das Z–Paar bereits vergeben. Es bleibt uns also keine Wahl mehr!

Entscheidung 7: Der Forth–Interpreter–Pointer IP wird dem AVR–Registerpaar X zugeordnet.

Wir können nun den (beinahe) endgültigen Code für NEXT zusammenstellen. Die Zeilennummerierung wurde unmittelbar vom Code oben übernommen. Jene Zeilennummern, die hier nicht mehr vorkommen, wurden dadurch eingespart, dass wir unsere Entscheidungen anpassten und pfiffige Code–Lösungen verwendeten.

Pseudocode		Assemblercode	NEXT–Routine
MOV W,(IP)	(1)	`Ld WH,X+`	indirekt, auto–increment
INC IP	(2)	`Ld WL,X+`	indirekt, auto–increment
	(5)	`Movw ZL,WL`	kopiere Pointer
MOV PC,(W)	(7)	`Ld R0,Z+`	ind, oberes Byte, auto incr
	(8)	`Ld ZL,Z`	ind, unteres Byte
	(10)	`Mov ZH,R0`	kopiere nur das obere Byte
	(11)	`IJmp`	indirekter Sprung

Das Einzige, was wir uns noch überlegen müssen, ist die Frage, ob unsere Wahl des AVR–Registerpaares W für das Forth–Register W die richtige Wahl war. Um das beurteilen zu können, müssen wir im Forth–Prozess noch einen Schritt weitergehen, nämlich zur Behandlung von High–Level–Worten.

Die Behandlung von High–Level–Worten

In einem High–Level–Wort, das aus einer :–Definition besteht, enthält das Parameterfeld der Definition eine Liste mit Adressen (mit CFAs, wie man in Teil 1 nochmal schnell nachliest) von anderen Worten, die ausgeführt werden sollen. Die Verarbeitungs–Routine dieser High–Level–Forth–Worte muss diese Adressenliste in der richtigen Reihenfolge abarbeiten. Das geschieht im Adressen–Interpreter DOCOL. DOCOL verwendet den Interpreter–Pointer IP auf dieselbe Weise, wie der AVR–Programmzähler PC die Maschinenbefehle verarbeitet. Anders gesagt, der IP läuft durch die Adressenliste im Parameterfeld so, wie der Programmzähler durch die Folge von Maschinenbefehlen läuft. Wenn die CFA auf eine andere High–Level–Definition zeigt, muss IP verwendet werden, um durch die neue Liste von Adressen zu laufen. Den alten Wert von IP bewahren wir auf dem Returnstack (daher der Name) auf, um später wieder zurückkehren zu können. Damit wird IP wieder zur Verarbeitung der neuen Liste frei. Auf diese Weise bildet der Returnstack eine Erweiterung von IP, so dass es möglich wird, ein weiteres High–Level–Wort aus einem anderen heraus aufzurufen. Die maximale Anzahl von Worten, die sich eines aus dem anderen heraus aufrufen können (die Nesteltiefe) hängt ausschließlich vom Platz auf dem Speicher ab, den der virtuelle Forth–Computer dem Returnstack zur Verfügung stellt.

Am Ende einer :–Definition muss die Kontrolle wieder an das aufrufende Wort zurückgegeben werden. Das besorgt der EXIT–Code. Die Rückkehradresse hatten wir auf dem Returnstack (dessen Bezeichnung jetzt klar wird) aufbewahrt. Wir können den gesamten Sachverlauf in Pseudocode fassen:

`DOCOL:`	Das W–Register zeigt auf die CFA des momentan gerade ausgeführten Wortes
`DEC RP`	Schaffe Platz auf dem Returnstack.
`MOV (RP),IP`	Setz die Adresse des als nächstes auszuführenden Wortes auf den Returnstack; wir benötigen IP zum Durchlaufen der neuen CFA–Liste.
`INC W`	Lass W auf die PFA des laufenden Wortes zeigen, auf die erste Adresse in der Liste mit CFAs.
`MOV IP,W`	Kopiere diese Adresse des ersten Wortes aus der neuen CFA–Liste nach IP, bereit zur Verwendung durch NEXT.
`NEXT`	Führe den Code für NEXT aus (siehe oben, siehe auch Teil 3), um dieses neue Wort auszuführen.

Man beachte, dass wir zwei dieser Befehle durch einen einzigen ersetzen können:

```
DEC RP
MOV (RP),IP   → PUSH IP
```

Der PUSH–Befehl erledigt auf einen Schlag beide Dinge zugleich.

Wenn wir auf diese Weise die gesamte Liste von High–Level–Worten durchgearbeitet haben, müssen wir wieder dorthin zurückkehren, wo wir hergekommen sind. Dafür sorgt der EXIT–Code oder das ; am Ende unserer Forth–Definition.

`EXIT:`	Die Rückkehradresse liegt auf dem Returnstack.
`MOV IP,(RP)`	Stelle vom Returnstack aus die Adresse des als nächstes auszuführenden Wortes wieder her.
`INC RP`	Gib den frei gewordenen Platz auf dem Returnstack wieder zurück.
`NEXT`	Führe NEXT aus, um dort fortzufahren, wo wir nach Ausführung dieses Wortes verblieben waren.

Auch hier können wir zwei dieser Befehle durch einen einzigen ersetzen:

```
MOV IP,(RP)   → POP IP
INC RP
```

Der POP–Befehl erledigt auf einen Schlag beide Dinge zugleich.

Im nächsten Teil übersetzen wir den oben stehenden Pseudocode in AVR–Maschinenbefehle. Ein funktionierendes, im klassischen Sinne aufgebautes AVR–Forth rückt näher!

— Wird fortgesetzt —

R8C–Forth

Bernd Paysan

In der Dezember–Ausgabe der Elektor[1] wurde ein Renesas[2]–R8C[3]–Board[4] von Glyn[5] verteilt. Die Software von Glyn beinhaltet einen C–Compiler, aber interessanter ist es natürlich, ein Forth auf der CPU laufen zu lassen.

Aufgabenverteilung

- Heinz Schnitter schreibt den Assembler
- Bernd Paysan passt Gforth–EC an den R8C an

Status 09apr2006

Das System ist inzwischen reif für den Beta–Test.

- Das Forth sagt OK über die UART, kann LEDs und LCD ansteuern, legt selbstdefinierte Programme per Default (sehr knappen) RAM ab. Umschalten zwischen RAM und ROM mit den Wörtern `ram` und `rom`.
- Das Daten–Flash kann mit `flashc!` und `flash!` programmiert werden.
- Gforth ist so anpasst, dass Code Flash–gerecht abgelegt wird (also jedes Bit genau einmal gelöscht wird). Immediate– und Compile–only–Bit sind also zunächst gesetzt (bedeutet natürlich „nicht immediate“ und „nicht compile–only“), und werden bei Bedarf gelöscht. Ebenso ist `:dovar` an einer Adresse, die das Ändern auf andere Doer (mit `DOES>`) ermöglicht ($C0FE).

[1] `http://www.elektor.de`
[2] `http://eu.renesas.com/`
[3] `http://eu.renesas.com/fmwk.jsp?cnt=r8ctiny_series_landing.jsp&fp=/products/mpumcu/m16c_family/r8c_tiny_series/`
[4] `http://www.elektor.de/Default.aspx?tabid=109`
[5] `http://www.glyn.de`

- Gforth EC R8C erlaubt das Anlegen von bis zu 4k Forth–Programmen im Datenflash. Ein `savesystem` speichert eine Kopie des RAMs in dieses Flash; damit sind auch Turnkey–Applikationen möglich. Das Datenflash kann öfter neu programmiert werden als das Programmflash, damit ist auch mehr Platz für Experimente. Ein `empty` löscht das gespeicherte System wieder.
- Der Assembler ist komplett.
- Der Rest des im ROM zur Verfügung stehenden Platzes ist mit nutzerfreundlichen Fehlermeldungen gefüllt
- Es gibt ein Terminal–Programm, das auch das Nachladen von Dateien über ein einfaches Handshake erlaubt (Xon/Xoff scheitert am Polling–Intervall des USB–Controllers)
- Eine Adaption von tt.fs passt in die 4K ROM.

Download

Die aktuelle Entwicklerversion von Gforth[6] kann über pserver–CVS gelanden[7] werden. Für Windows–Nutzer, die keine Lust haben, Cygwin zu installieren, gibt's einen Snapshot[8] in Form einer `setup.exe` der aktuellen Entwicklerversion. Auch das Gforth–R8C[9] selbst findet sich dort als S–Record–Datei.

Benutzung

Installiert man zunächst Cygwin[10] und dann Gforth aus den Sourcen, sollte man vor dem `./configure && make` noch die libffi[11] installieren.

Installiert man obige `setup.exe`, werden alle benötigten Libraries mitinstalliert. Das Menü enthält noch einen Punkt „Bash“, der eine Shell aufruft. Alle Anweisungen, die nun folgen, gehen von einer frisch geöffneten Shell aus (für Cygwin–Nutzer: Im Gforth–Verzeichnis).

ROM–Image erzeugen

Mit

```
./build-ec r8c
```

erzeugt man eine Datei `rom-r8c.mot` (sie sollte der `gforth-r8c.mot` von oben entsprechen). Es wird auch noch ein `data-r8c.mot` erzeugt, mit dem man ein zerschossenes Daten–Flash wieder in Ordnung bringen kann. Das `rom-r8c.mot` lädt man in's Flash. Danach kann man ein Terminal mit 38400 Baud 8N1 anschließen (etwa das Hyperterm, das in Windows eingebaut ist — Verbindung auf „COMx“ setzen), und interaktiv mit dem Forth arbeiten. Die Kommunikation findet über dieselbe serielle Schnittstelle statt, mit der man das Flash programmiert — kein Umstöpseln nötig. Das Nachladen von Dateien geht mit einem normalen Terminalprogramm nicht, da Xon/Xoff zu langsam reagieren.

Spielen Sie ein wenig, etwa

```
1 2 + .
lcdpage
s" Hello World" lcdtype
adc@ .
```

Forth–Terminal

Mit

```
./gforth arch/r8c/terminal.fs
```

startet man das in Gforth eingebaute Terminal. Es nimmt per Default die Verbindung über COM2 auf — wenn nötig, die Datei editieren, oder — bei Verwendung eines USB–Adapters — die Schnittstelle über den Gerätemanager auf COM2 festlegen. Unter Linux wird `/dev/ttyUSB0` verwendet, auch hier kann man entsprechend durch Editieren eine andere Schnittstelle festlegen.

Hier kann man nun auch

```
include arch/r8c/tt.fs
```

eingeben, und das adaptierte Tetris–for–Terminals von Dirk Zoller laden (dauert etwas — der R8C compiliert den Code schließlich selbst). Starten mit

```
tt
```

Links

- Peter Ouwehand beschreibt das HD44780 LCD[12].
- Thread zum Programmieren des Daten–Flashs[13]
- Thomas Fischl hat einen Flasher für Linux und Mac OS X[14] an den R8C angepasst.

[6] http://www.jwdt.com/~paysan/gforth.html
[7] http://www.complang.tuwien.ac.at/forth/gforth/cvs-public/
[8] http://www.jwdt.com/~paysan/gforth-0.6.2-20060409.exe
[9] http://www.jwdt.com/~paysan/gforth-r8c.mot
[10] http://www.cygwin.com
[11] http://loreley.ath.cx/cygwin/cygwin-1.5/libffi-cygwin-standalone/libffi-2.00-beta-1.tar.bz2
[12] http://home.iae.nl/users/pouweha/lcd/lcd0.shtml#hd44780
[13] http://www.elektor.de/default.aspx?tabid=29&view=topic&forumid=23&postid=3645
[14] http://www.fischl.de/thomas/elektronik/r8c/r8c_flasher.html

Forth erzeugt automatisch Hexadoku–Rätsel–Vorgaben

Fred Behringer

Bei Sudoku (Vorstufe zu Hexadoku) werden üblicherweise etwa 30 Elemente vorgegeben. Diese können nicht beliebig gewählt werden. Sie müssen *zulässig* sein, d.h., aus einer *zulässigen* Gesamtmatrix stammen. Es soll eine Analyse darüber versucht werden, wie man es (mit Hilfe von Forth) schafft, praktisch beliebig viele voneinander verschiedene zulässige Hexadoku–Rätsel zu erzeugen. *Erzeugen*, als Vorgabeschema, *nicht* unbedingt schon *lösen!* Allerdings soll auch über das (schnelle) Lösen gesprochen werden, und zwar bei einer extrem spärlich verteilten geringen Anzahl von vorgegebenen Elementen.

Bedienungsschnellanleitung

Dinge, die in dieser Schnellanleitung noch unklar bleiben, werden, so hoffe ich, aus dem eigentlichen Textteil dieses Artikels heraus klarer (man fange am besten mit **Einleitung und Ziel** an, zu lesen). Die Eingabe `hexa` ⟨ret⟩ liefert (auf dem Bildschirm) beliebig viele (vom Programm *zufällig* ausgewählte) Hexadoku–Vorgaben, pro Knopfdruck (⟨ret⟩) je eine. Die vorherige Eingabe `loes? on` ⟨ret⟩ liefert, solange sie nicht durch `loes? off` ⟨ret⟩ umgestellt wurde, neben der Vorgabematrix jeweils auch immer die zugehörige zulässige Matrix (die *Lösung*). Voreingestellt ist `loes? on`.

Ausgangspunkt im Hauptprogramm `hexa` ist die *kanonische Matrix* (siehe unten). Schon beim ersten Weiterschreiten (nach eingeleitetem `hexa` ⟨ret⟩) ist das Tableau (das Feld `matrix`) völlig undurchsichtig geworden. Von der ursprünglich durch `kanon` systematisch aufgebauten zulässigen `matrix` ist nichts erkennbar Systematisches mehr übrig geblieben. Das Weiterschreiten (von *zulässiger* `matrix` zu *zulässiger* `matrix`) geschieht über einen beliebigen Tastendruck (unter Aussparung der Returntaste). Den Erzeugungsprozess zum Aufbau von zulässigen Matrizen verlassen kann man mit der Returntaste oder mit ⟨q⟩ oder mit ⟨Q⟩. ⟨q⟩ oder ⟨Q⟩ ist als Hommage an MinForth (siehe weiter unten unter **Forth–Programme**) gedacht.

Kein Autor ist gegen Denk– oder Programmierfehler gefeit: Es könnten beim Benutzer nach vielen Schritten im `hexa`–Prozess Zweifel an der immer noch vorliegenden Zulässigkeit von `matrix` aufkommen. Man breche den Prozess über die Returntaste ab und prüfe die gerade erreichte `matrix` über `mok?` ⟨ret⟩. Natürlich liegt hier ein Teufelskreis vor: Für diese Prüfung muss man sich auf das Prüfprogramm `mok?` verlassen können. Das lässt sich aber durch eine einmalige strenge logische Überlegung ein für alle Mal erledigen.

Das Hauptprogramm `hexa` liefert (zumeist jedenfalls) eine reproduzierbare Serie von Ergebnissen. Das ist nicht weiter schlimm, da man ja den Prozess `hexa` und den damit verbundenen Pseudozufallsprozess beliebig weit forttreiben kann, bevor man an eine Übernahme der Ergebnisse (der gelieferten Rätselvorgaben) denkt. Man kann aber auch das Programm `hexa` umformulieren und es von der (an sich beliebig eingebbaren) Anfangszahl für den Pseudozufallsprozess (im Programm habe ich 2000 gewählt) abhängig werden lassen. (Man beachte auch das unter **Forth–Programme** über MinForth Gesagte.)

Voreingestellter Wert für die Abfragevariable `kanon?` ist `kanon? on` . Stellt man auf `kanon? off` um und leitet erst dann den Erzeugungsprozess per `hexa` ein, so wird man sich wundern: Schon die erste Matrix ist dann im Allgemeinen nicht zulässig (mit mok? nachprüfen!). Zum ordnungsgemäßen Start braucht der Erzeugungsprozess die Einstellung `kanon? on` . Es wird dann mit der (zulässigen) kanonischen Matrix begonnen und alle nachfolgenden Matrizen sind ebenfalls zulässig. Bricht man den Erzeugungsprozess dann aber irgendwann durch Drücken der Returntaste ab und gibt wieder `hexa` ⟨ret⟩ ein, so wird wieder mit der kanonischen Matrix gestartet. Will man das (was durchaus wünschenswert sein kann) verhindern, so kann man den Erzeugungsprozess über die Returntaste kurz unterbrechen und dann `kanon? off` ⟨ret⟩ eingeben. Wenn man daraufhin wieder `hexa` ⟨ret⟩ eingibt, so fährt der Erzeugungsprozess mit der bis dato bereits erreichten (zulässigen) Matrix fort.

An Bildschirm–Anzeigemöglichkeiten aus dem Stand heraus (außerhalb eines eingeleiteten `hexa`) stehen zur Verfügung: `.m` für die momentane Matrix (Rätsellösung), `.v` für die zugehörige Vorgabematrix (zu stellendes Rätsel) und `.m&v` für beides zusammen. Mit `cpy-m` kann man die Matrix auf den Platz für die Vorgabematrix kopieren, mit `xch-m` tauscht man Matrix und Vorgabematrix gegeneinander aus.

Wegen einer praktischen Inbetriebnahme unter verschiedenen Forth–Systemen sehe man weiter unten im Abschnitt **Forth–Programme** nach.

Einleitung und Ziel

Magische Quadrate ($n \times n$–Matrizen) faszinieren den Betrachter seit Menschengedenken. Nicht immer ist es die Summe der (aus natürlichen Zahlen bestehenden) Elemente, die über Zeilen, Spalten und Diagonalen hinweg konstant bleiben soll. Verlangt man, dass in einer 9×9–Matrix jede Ziffer von 1 bis 9 in jeder Zeile, in jeder Spalte und in jeder 3×3–Teilmatrix genau einmal auftritt, dann hat man es mit einer Sudoku–Matrix zu tun.

Sudokus (su = Ziffer, doku = einmal) sind seit einigen Jahren erst in Amerika, dann in Japan (daher der Name) und jetzt auch bei uns in Deutschland beliebt geworden. Ein neuer Volkssport, sozusagen: Eine gewisse Anzahl von Elementen (meist um die 30 herum) werden vorgegeben, der Rest muss *erraten* werden. Sudoku–Rätselsammlungen (mit durchschnittlich 200 Rätseln und auf den letzten Seiten angegebenen Lösungen) wurden zuhauf fünfeuroweise auf den Markt geworfen.

Die Zeitschrift Elektor (und in Holland deren Mutterblatt Elektuur) hat ihre Leser mit dem Begriff *Hexadoku* begeistert: Eine 16×16–Matrix, die in sechzehn 4×4–Teilmatrizen aufgeteilt ist. (*Aufteilung* heißt Überdeckung mit 4×4–Teilmatrizen, die mit den Elementen der Gesamtmatrix auskommt.) Vorgegeben ist eine gewisse Anzahl von Elementen, der Rest soll *erraten* (irgendwie gefunden) werden. In jeder Zeile, in jeder Spalte und in jedem Teilquadrat soll jede Hexadezimalziffer von 0 bis F genau einmal vorkommen (hier also auch die *0,* bei *Sudoku* fängt es *erst bei 1* an.)

Martin Bitter hat sofort erkannt, dass Sudoku (besser: Hexadoku) ein willkommenes Mittel sein kann, den lernschwachen unter den Schülern seiner Integrativen Schule spielerisch das logische Denken beizubringen. Er hat im VD–Heft 1/2006 ein Forth–Programm zur Lösungshilfe veröffentlicht.

(Aus Amerika hört man von *unserem Reporter* Henry Vinerts, dass John Rible, erwähnt in comp.lang.forth, auf einem kürzlichen SVFIG–Treffen über ein vollständiges Lösungsverfahren in Forth vorgetragen hat. John Rible stützt seine Lösung auf einen Rekursive–Backtracking–Algorithmus von Robert Spykerman, den John um einen Brute–Force–Backtracking–Solver ergänzt hat. Bernd Paysan hat in einer persönlichen Mitteilung an das Redaktions–Team der VD sofort nach Bekanntwerden von Martins Anstrengungen angedeutet, wie ein vollständiges Lösungsverfahren aussehen kann. Fred Behringer hat das Hexadoku–Rätsel aus dem Elektor–Heft 2/2006 mit Bleistift, Papier und Radiergummi gelöst und damit den Elektor–Hauptpreis, das E–Blocks–Starter–Kit–Professional, gewonnen.)

In den Sudoku–Büchern werden die *Rätsel* mit *weniger Vorgaben* als *schwer* bezeichnet, die mit *mehr Vorgaben* als *leichter*. *Das kann man, so behaupte ich, so generell nicht sagen.* Die kleinste Anzahl von Vorgaben ist 0. Das müsste dann also ein schweres Rätsel sein. *Null* liefert aber ein *ganz ganz leichtes* Spiel. Ich gebe dafür gleich anschließend eine *Lösung* an (*eine* Lösung — mit der Eindeutigkeitsfrage müsste man sich noch gesondert befassen), die ich als Ausgangspunkt für weitere Überlegungen nehmen und als *kanonisch* bezeichnen möchte. Matrizen mit einem *einzigen* vorgegebenen Element, das sieht man schnell ein, lassen sich *ebenfalls ganz einfach* behandeln. Zwei vorgegebene Elemente bereiten keine Schwierigkeit. Vier vorgegebene Elemente, das wird man sich mit dem weiter unten Gesagten ebenfalls leicht überlegen können, liefern verhältnismäßig schnell eine Lösung, wenn sie sich gleichmäßig auf vier waagerecht angeordnete, vier senkrecht angeordnete oder vier diagonal angeordnete Teilmatrizen verteilen. Ein bisschen Überlegung braucht man dazu allerdings schon.

Dass Vorgaben nicht willkürlich gemacht werden können, ist klar: Zweimal die 1 in ein und derselben Zeile oder in ein und derselben Spalte wäre ja beispielsweise schon eine unzulässige Vorgabe.

Frage 1: Wenige Vorgaben sind leicht, viele Vorgaben sind ebenfalls leicht. Gibt es eine mittlere Vorgabenbelegung, die einer schwersten Schwierigkeitsstufe zugeordnet werden kann?

Ich erwähne diese Frage nur, weil sie mit der kanonischen Belegung eng zusammenhängt. Auch will ich mir über vollständige, computerübertragbare Lösungsverfahren (Algorithmen) keine Gedanken machen. Die Eindeutigkeitsfrage (Bedingungen an die Vorgaben, welche eine eindeutige Lösung garantieren) wäre ebenfalls interessant. Was mich hier interessiert aber ist die folgende

Frage 2: Wie kann man automatisch eine (im praktischen Sinne) beliebige Anzahl von verschiedenen Hexadoku–Rätseln erzeugen?

(Martin möchte sich ja schließlich nicht jede Woche ein neues Hexadoku–Rätselbuch, so es solche überhaupt je geben wird, kaufen müssen. Dass man nach erfolgreicher Beantwortung der obigen Frage den Computer dazu befähigen könnte, automatisch, ohne Zutun des Menschen, Bücher (Hexadoku–Rätselbücher) zu schreiben und auf den Markt zu werfen, ist ein interessanter und sicher diskutierwürdiger Nebeneffekt. Haben wir damit die KI–Frage gelöst (der Computer als Bücherschreiber) und eine erfolgversprechende KI–Begriffsbestimmung gefunden?)

Definitionen und Bezeichnungen

Matrix sei eine abkürzende Bezeichnung für eine 16×16–Matrix mit den Hexadezimalziffern 0 bis F als *Elementen.* Zeilen– und Spaltenzählungen *beginnen bei 0!* Matrizen seien aufgeteilt in 4×4–Teilmatrizen. Jede solche Teilmatrix heiße *Viererquadrat.* Die Viererquadrate einer Matrix seien *blockzeilenweise* mit 0 bis F durchnummeriert: Das Viererquadrat 9 ist von links gesehen das zweite (*Längsstreifen* 1), von oben gesehen das dritte (*Querstreifen* 2) Quadrat. Auch die Zählung der Viererquadrate *beginnt bei 0!* Von *Reihe* sprechen wir, wenn wir uns nicht genau festlegen wollen, ob wir Zeile oder Spalte meinen. Eine waagerechte Anordnung von vier Viererquadraten möge *Querstreifen* heißen. Eine senkrechte Anordnung von vier Viererquadraten heiße *Längsstreifen.* Soll es offen bleiben, ob Quer– oder Längsstreifen gemeint ist, sprechen wir einfach von *Streifen.*

Eine *zulässige Matrix* ist eine Matrix (im obigen Sinne), bei welcher in jeder Zeile, in jeder Spalte und in jedem Viererquadrat jede der *Hex*(adezimal)*ziffern* von 0 bis F genau einmal vorkommt. Elemente, die (noch) nicht bekannt sind, bezeichnen wir mit dem *Leerzeichen* (Space). (Space tritt als Platzhalter, als variables Element, als Variable auf.) Ein *zulässiger Streifen* ist ein Streifen einer

0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
4	5	6	7	8	9	A	B	C	D	E	F	0	1	2	3
8	9	A	B	C	D	E	F	0	1	2	3	4	5	6	7
C	D	E	F	0	1	2	3	4	5	6	7	8	9	A	B
1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	0
5	6	7	8	9	A	B	C	D	E	F	0	1	2	3	4
9	A	B	C	D	E	F	0	1	2	3	4	5	6	7	8
D	E	F	0	1	2	3	4	5	6	7	8	9	A	B	C
2	3	4	5	6	7	8	9	A	B	C	D	E	F	0	1
6	7	8	9	A	B	C	D	E	F	0	1	2	3	4	5
A	B	C	D	E	F	0	1	2	3	4	5	6	7	8	9
E	F	0	1	2	3	4	5	6	7	8	9	A	B	C	D
3	4	5	6	7	8	9	A	B	C	D	E	F	0	1	2
7	8	9	A	B	C	D	E	F	0	1	2	3	4	5	6
B	C	D	E	F	0	1	2	3	4	5	6	7	8	9	A
F	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E

Abbildung 1: Die kanonische Matrix

zulässigen Matrix. Eine *Vorbelegung* ist eine willkürliche Auswahl von Elementen einer Matrix, deren restliche Elemente mit Space (20h) gekennzeichnet sind. *Eine zulässige Vorbelegung* ist eine Vorbelegung, die aus (mindestens) einer zulässigen Matrix durch Weglassung von Elementen (Belegung mit Space) hervorgeht.

Eine *Transformation* ist (hier) eine (eineindeutige, also bijektive) Umordnung der Elemente einer Matrix. Eine *zulässige Transformation* ist eine Transformation, die eine zulässige Matrix in eine zulässige Matrix überführt.

In der vorliegenden Arbeit geht es um zulässige Transformationen, die sich aus beliebig zusammengewürfelten zulässigen Transformationen zusammensetzen.

Feststellungen

Die meisten der jetzt folgenden Feststellungen sind leicht einzusehen und benötigen keinen Beweis. Bei anderen reicht eine kurze Erläuterung oder ein Beispiel. Die Feststellungen lassen sich in zwei Klassen einteilen, allgemeine Feststellungen (die für beliebige Belegungen gelten) und solche, die sich (wie bei *kanonischen* Matrizen, siehe unten) auf eine bestimmte, festverankerte Belegung beziehen. Letztere können selbstverständlich mit Auge und Gehirn (mit Bleistift und Papier) auf Richtigkeit überprüft werden: Das Auge braucht *nur* 256 Elemente zu überfliegen. Aber gerade weil die Überprüfung der sehr anspruchslosen Regeln in den durchaus überschaubaren Matrizen so einfach ist, lässt sich das auch *per Knopfdruck* überaus leicht erledigen.

Ich gebe weiter unten ein Forth–Programm an, welches nicht nur die zufallsmäßige Erzeugung von Hexadokus erledigt, sondern auch die Überprüfung auf Zulässigkeit (einer irgendwie per Hand geänderten zulässigen Matrix).

Feststellung 1: In einer zulässigen Vorbelegung ist in jeder Zeile und in jeder Spalte keine der auftretenden Ziffern mehr als einmal vertreten.

Feststellung 2: Eine zulässige Matrix bleibt zulässig, wenn in einem Querstreifen zwei Zeilen vertauscht werden.

Feststellung 3: Eine zulässige Matrix bleibt zulässig, wenn in einem Längsstreifen zwei Spalten vertauscht werden.

Feststellung 4: Eine zulässige Matrix bleibt zulässig, wenn in ihr zwei Querstreifen vertauscht werden.

Feststellung 5: Eine zulässige Matrix bleibt zulässig, wenn in ihr zwei Längsstreifen vertauscht werden.

Vertauscht man generell in einer Matrix M deren Zeilen und Spalten, dann erhält man bekanntlich die zu M *transponierte* Matrix M^T.

Feststellung 6: Bekanntlich ist $M^{TT} = M$.

Feststellung 7: Ist M eine Matrix im obigen Sinne und V das Vierquadrat n mit den (zeilenweise zu lesenden) Elementen $0, 1, 2, 3, 4, 5, 6, 7, 8, 9, \mathrm{A}, \mathrm{B}, \mathrm{C}, \mathrm{D}, \mathrm{E}, \mathrm{F}$, dann ist das Vierquadrat m mit den Elementen $0, 4, 8, \mathrm{C}, 1, 5, 9, \mathrm{D}, 2, 6, \mathrm{A}, \mathrm{E}, 3, 7, \mathrm{B}, \mathrm{F}$ von M^T die Transponierte V^T von V.

Beweis: Betrachtung der in üblicher Weise doppeltindizierten Elemente von M und Auszählung der Indizes: Vertauschung von Zeilen und Spalten in der Gesamtmatrix M läuft auf eine Vertauschung von Zeilen und Spalten auch in den Vierquadraten hinaus. ∘

Feststellung 8: Die Transponierte einer zulässigen Matrix ist eine zulässige Matrix.

Beweis: Mit den Zeilen/Spalten einer zulässigen Matrix erfüllen auch die Spalten/Zeilen ihrer Transponierten die Kriterien der Zulässigkeit. Alles Weitere folgt aus Feststellung 7. ○

Generierverfahren

Feststellung 9: Die Matrix (im obigen Sinne) in Abbildung 1 (Seite 20) ist eine zulässige Matrix. Wir wollen sie wegen ihres einfachen Aufbaus die *kanonische Matrix* nennen.

Beweis: Einfach durchprobieren, Zeile für Zeile, Spalte für Spalte, Viererquadrat für Viererquadrat! ○

Diese Matrix baut sich wie folgt auf: Die (Hex–)Ziffern der ersten Zeile (Zeile 0) in natürlicher Reihenfolge hinschreiben. Eine Kopie der so gebildeten ersten Zeile in die zweite Zeile (Zeile 1) schreiben und diese dann um 4 Spalten nach links rotieren (was links herausfällt, rechts anfügen). Eine Kopie der so gebildeten zweiten Zeile (Zeile 1) in die dritte Zeile (Zeile 2) schreiben und diese wieder um 4 Spalten nach links rotieren lassen. Und schließlich die dritte Zeile (Zeile 2) in die vierte Zeile (Zeile 3) kopieren und die vierte Zeile (Zeile 3) um vier Spalten nach links rotieren. Damit ist der erste Querstreifen (Querstreifen 0) der aufzubauenden kanonischen Matrix gebildet.

In die fünfte Zeile (Zeile 4) schreibe man eine Kopie der ersten Zeile (Zeile 0) und rotiere sie um 1 nach links (um 1, nicht um 4!). Dann folgerichtig weiter mit der sechsten, siebten und achten Zeile (jeweils die vorhergehende kopieren und um 4 Spalten nach links rotieren lassen). Bei der neunten Zeile (Zeile 8) wiederum die fünfte (nicht die erste!) nehmen und um 1 Spalte nach links rotieren. Und so weiter und so fort.

Feststellung 10: Aus der kanonischen Matrix aus Feststellung 9 erhält man wieder eine Art kanonischer Matrix (einfach aufgebaut und zulässig), wenn man in Feststellung 9 das Wort links durch das Wort rechts ersetzt. Ich will die zulässige Matrix aus Feststellung 9 daher auch *linkskanonisch* und die, um die es in der vorliegenden Feststellung 10 geht, *rechtskanonisch* nennen.

Beweis: Hinschreiben und Zulässigkeit nachprüfen! ○

Feststellung 11: Aus einer zulässigen Matrix erhält man wieder eine zulässige Matrix, wenn man die Zeilen oder/und die Spalten in umgekehrter Reihenfolge hinschreibt.

Beweis: Feststellungen 2 bis 5. *Spalten in umgekehrter Reihenfolge hinschreiben* ist dasselbe wie *äußerste Spalten miteinander vertauschen, dann zweitäußerste* usw. Entsprechend für Zeilen. ○

Bezeichnung: Geht man von der kanonischen Matrix aus und schreibt die Zeilen (Spalten) wie in Feststellung 11 invertiert hin, dann wollen wir die resultierende Matrix die *zeileninvertierte* (*spalteninvertierte*) *kanonische Matrix* nennen.

Randomisierung

Anzahl der Vorgaben: Bei Sudoku sind etwa 30 Vorgabe–Elemente üblich. Die Zeitschrift *Elektor* hat in ihren Ausgaben vom Januar, Februar, März und April 120, 122, 119, 111 Elemente vorgegeben. 128 Elemente beispielsweise scheint also eine brauchbare Zahl von Vorgaben zu sein. 128 ist genau die Hälfte von 256, der Gesamtzahl an Matrixelementen. Das legt für die Randomisierung eine Zufallsverteilung von *gerade* und *ungerade* nahe.

Einsparung eines echten Zufalls: Man weiß, wie (Pseudo)zufallszahlen zu konstruieren sind, die den bekannten statistischen Verfahren zur Überprüfung auf Gleichverteilung standhalten. Hier ist das aber sicher nicht nötig: Ob gleichverteilt oder nicht, interessiert uns herzlich wenig, Hauptsache, die Lösung (*eine* Lösung) lässt sich nicht schon aus der Struktur der Vorgabeelemente heraus sofort erraten. Naheliegend ist also die Vorgabe einer Zahl (*Seed*), ab der im Programmspeicher die aufeinanderfolgenden Bytes abgefragt und mit 1 geANDet werden. Bei Ergebnis 0 wird das gerade an der Reihe befindliche der parallel dazu laufenden Matrixelemente *gelöscht* (mit Space belegt), bei 1 bleibt es stehen.

Das gibt eine ungefähre Vorgabebelegung mit 50% aller Elemente, also mit etwa 128d Elementen. In gewisser Bandbreite kann man das noch experimentell durch geeignete Wahl der Pseudozufalls–Seed–Zahl (im Programm habe ich willkürlich 2000 gewählt) variieren (es kann ja ganze Strecken im RAM geben, die mit geraden Bytes, z.B. 0, belegt sind). Möglichkeiten, die vorgegebene Anzahl von Elementen zu reduzieren, bestehen darin, dass man `AUSWAHL` (siehe Programm) auf eine schon getroffene Auswahl anwendet, und das vielleicht sogar mehrfach. Man experimentiere! Die Reduzierungsrate lässt sich schwer vorhersagen. Experimente haben jedoch gezeigt, dass schon nach etwa 10 solcher wiederholten `AUSWAHL`en praktisch kein Belegungselement mehr übrig ist. (Man beachte aber auch das unter **Forth–Programme** über MinForth Gesagte!)

Löseverfahren bei kleiner Zahl von Vorgaben

Ich will nur auf ein paar Lösungs– und Prüfverfahren eingehen, die bei 30 (Sudoku) oder 128 (Hexadoku) vorgegebenen Elementen der entsprechenden `MATRIX` nicht mehr anwendbar oder, da nur in Ausnahmefällen auftretend, nicht mehr so wichtig sind. Generelle Lösungsverfahren, so interessant sie sein mögen, sollen hier nicht angegangen werden.

Feststellung 12: Wird in einer Matrix (im obigen Sinne) ein (und nur ein) Element vorgegeben, dann lässt dieses sich (zumindest wie folgt) zu einer zulässigen Matrix ergänzen: O.B.d.A. möge das vorgegebene Element den Wert 7 haben und im Viererquadrat m liegen. Man betrachte die kanonische Matrix.

Vielleicht stimmen Zeile und Spalte des vorgegebenen Elementes genau mit jenem Element im Viererquadrat m überein, das den Wert 7 hat. Wenn nicht, dann:
Es muss im Viererquadrat m genau ein Element mit dem Wert 7 geben. Es möge in Zeile i und Spalte j liegen.
Man vertausche die Zeile des vorgebenen Elementes mit Zeile i und die Spalte mit Spalte j. Dann ist nach Feststellung 2 und 3 alles erledigt. ○

Feststellung 13: Die Lösung von Feststellung 12 ist alles andere als eindeutig, da ja schon die rechtskanonische, die zeileninvertierte kanonische, die spalteninvertierte kanonische Matrix und schließlich die Transponierte aller genannten Matrizen offensichtlich unterschiedliche Lösungen liefern.

Feststellung 14: Hat man bis zu vier vorgegebene Elemente, die sich gleichmäßig so verteilen, dass in den zugehörigen Längs– und Querstreifen kein weiteres Vorgabeelement liegt (beispielsweise bei Diagonalanordnung), dann gelangt man zu (mindestens) einer Lösung, wenn man in Bezug auf jedes Vorgabeelement so wie in Feststellung 12 vorgeht.

Beweis: Offensichtlich. Kein Streifenpaar beeinflusst ein anderes Streifenpaar. ○

Ist man im unten stehenden Programm mit der Anzahl der Vorbelegungselemente (die man beispielsweise über `VORGABE` erhalten hat) zufrieden, nicht aber mit deren Verteilung (weil sie sich beispielsweise alle in einer Ecke der `MATRIX` zusammenballen), dann kommt man eventuell mit folgender Feststellung weiter.

Feststellung 15: Eine zulässige Vorbelegung liefert wieder eine zulässige Vorbelegung, wenn man auf sie eine zulässige Transformation (siehe **Definitionen und Bezeichnungen**) anwendet.

Beweis: Offensichtlich. Die zulässige Transformation entstammt einer zulässigen Matrix (durch Weglassung von Elementen hervorgerufen). Der Begriff der zulässigen Transformation bezieht sich an sich auf eine zulässige Matrix, ist aber ansonsten eine (wiederholbare) Matrixoperation. Man betrachte statt der zulässigen Vorbelegung (welche Leerelemente enthält, genauer, enthalten kann) die (genauer, eine) zugehörige zulässige Matrix, wende die zulässige Transformation auf ebendiese zulässige Matrix an und ersetze die zuvor irgendwie gekennzeichneten ursprünglichen Leerelemente wieder durch Leerelemente. ○

Feststellung 16: Nimmt man eine der Hexzahlen $0, 4, 8$ oder C und addiert sie modulo 10h zu sämtlichen Elementen einer zulässigen Matrix, so erhält man wieder eine zulässige Matrix.

Beweis: Addition modulo 10h einer der genannten Konstanten zu den Hexzahlen $0 \ldots$ F führt aus eben diesem Bereich nicht hinaus. (Eine solche Konstantenaddition ist umkehrbar eindeutig.) In jeder Zeile, in jeder Spalte und in jedem Viererquadrat ist eine solche Addition gleichbedeutend mit einer Umordnung der betreffenden Elemente. Die Zulässigkeitsbedingungen bleiben also insgesamt erhalten. ○

Bemerkung 1 zu 16: Man kann in Feststellung 16 beliebige durch 4 teilbare nichtnegative (Hex–)Zahlen nehmen: $8 + 14 =$ C mod 10h etc. (Auch negative Hexzahlen tun der Überlegung keinen Abbruch: Beweis!)

Bemerkung 2 zu 16: Zu jeder zulässigen Vorbelegung erhält man unmittelbar drei weitere zulässige Vorbelegungen mit exakt gleichen Vorgabepositionen: Addition modulo 10h von $4, 8$ oder C zu sämtlichen Elementen.

Feststellung 17: Bei der Addition modulo 10h von 4 zu sämtlichen Elementen der *kanonischen* Matrix läuft das Ergebnis auf eine Linksrotation (siehe Programmteil) sämtlicher Zeilen um 4 Spalten hinaus. Das gilt auch, wenn man 4 durch andere durch 4 teilbare Zahlen ersetzt. Insbesondere läuft eine Addition modulo 10h um C Spalten auf eine Rechtsrotation um 4 Spalten hinaus.

Beweis: Anhand der unten stehenden Forth–Programme leicht nachweisbar. ○

Eindeutigkeit und Ausschließung

Bei den in der Zeitschrift Elektor bisher zum *Raten* veröffentlichten Hexadokus handelt es sich ganz offensichtlich um Probleme mit *eindeutigen* Lösungen, auch wenn das nicht gesagt wird. Wir haben aber schon gesehen, dass eine gefundene Hexadoku–Lösung bei *wenigen Vorgaben alles andere als eindeutig* zu sein braucht. In den Sudoku–Büchern wird als (unter anderen) mögliches Lösungsverfahren das *Ausschließungsprinzip* empfohlen. Wo es geht, ist es prima und macht Spaß: Angenommen, in einem Viererquadrat fehlen noch zwei Elemente, 0 und 7. Man setze die 0 im Geiste versuchsweise in das eine der in Frage kommenden beiden Kästchen. Ein schneller Blick möge uns zeigen, dass sowohl in der zugehörigen Zeile wie auch in der zugehörigen Spalte schon eine 0 vorkommt. Also bleibt für das betreffende Kästchen nur die 7 — und für die 0 gibt es anschließend nur noch eine einzige verbleibende Möglichkeit. Voilà!

Wenn es aber immer so einfach wäre, wäre das Hexadokulösen einfach wirklich zu einfach. Die Erfahrung zeigt, dass man auf mittlerer Lösungsstrecke meist mehr als nur eine einzige noch verbleibende Möglichkeit in Betracht ziehen muss. Und aus diesen Möglichkeitenkombinationen ist dann durch logische Überlegung eine einzige Restmöglichkeit herauszufiltern. Das gilt für *eindeutig gegebene Aufgabenlösungen.* Was aber, wenn es mehr als nur eine Lösung gibt? Die Matrix in Abbildung 2, die sich aus der kanonischen Matrix durch Weglassen der letzten beiden Spalten ergibt, stellt eine durchaus zulässige Vorgabematrix dar, eine Vorgabematrix mit 224 aus 256 Elementebelegungen. So etwas könnte man im Endstadium eines Lösungsprozesses vorfinden. Und der Rest wäre nach allem, was man hört, einfach.

0	1	2	3	4	5	6	7	8	9	A	B	C	D		
4	5	6	7	8	9	A	B	C	D	E	F	0	1		
8	9	A	B	C	D	E	F	0	1	2	3	4	5		
C	D	E	F	0	1	2	3	4	5	6	7	8	9		
1	2	3	4	5	6	7	8	9	A	B	C	D	E		
5	6	7	8	9	A	B	C	D	E	F	0	1	2		
9	A	B	C	D	E	F	0	1	2	3	4	5	6		
D	E	F	0	1	2	3	4	5	6	7	8	9	A		
2	3	4	5	6	7	8	9	A	B	C	D	E	F		
6	7	8	9	A	B	C	D	E	F	0	1	2	3		
A	B	C	D	E	F	0	1	2	3	4	5	6	7		
E	F	0	1	2	3	4	5	6	7	8	9	A	B		
3	4	5	6	7	8	9	A	B	C	D	E	F	0		
7	8	9	A	B	C	D	E	F	0	1	2	3	4		
B	C	D	E	F	0	1	2	3	4	5	6	7	8		
F	0	1	2	3	4	5	6	7	8	9	A	B	C		

Abbildung 2: Die letzten beiden Spalten der kanonischen Matrix werden gelöscht.

Wirklich einfach? Es ist klar, dass es zwei mögliche Lösungen gibt, die eine der beiden Spalten zuerst und dann die andere, oder umgekehrt. Im Beispiel aus Abbildung 2 kann man den Rest erraten: Zeile für Zeile erst das kleinere Element, dann das größere, oder umgekehrt. Was aber, wenn man eine ähnlich bis auf zwei Spalten vollbesetzte Vorgabematrix nimmt, die nicht direkt aus der kanonischen Matrix stammt, sondern aus einer zuvor gehörig durchgerüttelten kanonischen Matrix? Man müsste im schlimmsten Fall 2 hoch 16 Kombinationen durchprobieren, in jeder Zeile ein Elementepaar, so oder so herum. Natürlich fällt immer dann eine Möglichkeiten–Zweierpotenz weg, wenn man beim Durchprobieren in ein und derselben Spalte auf eine schon vorhandene Ziffer stößt. Und ebenso natürlich braucht man nacheinander jeweils nur immer ein Viererquadrat des betreffenden Längsstreifens zu betrachten. Aber das tut der Überlegung (mit der sehr großen Zahl von Kombinationsmöglichkeiten) keinen Abbruch. Und was, wenn man (zur Verwirrung des Betrachters) ganz am Anfang erst die letzten beiden Spalten der kanonischen Matrix löscht (wie im Beispiel in Abbildung 2) und dann die Matrix unter Zuhilfenahme von Feststellung 1 bis 14 *durchrüttelt?*

Das Wissen um die eventuell vorliegende Lösungseindeutigkeit (ob oder ob nicht) wäre (wie man hier sieht) auch aus praktischen Gründen ein erstrebenswertes Ziel. (Es gibt nichts Praktischeres als eine gute Theorie!)

Aufgabe 1: Lösungsexistenz (Zulässigkeit) vorausgesetzt, gebe man hinreichende oder/und notwendige Kriterien für die Eindeutigkeit der angestrebten Lösung an.

Aufgabe 2: Man lasse die gefundenen Kriterien in ein Forth–Löseprogramm einfließen.

Forth–Programme

Entwickelt habe ich das gesamte Programm in Turbo–Forth von Marc Petremann (16 Bit) unter DOS 6.2. Die Lauffähigkeit getestet habe ich außerdem auch im Subroutine–Threaded–DPMI–Forth von Rick van Norman (32 Bit, ANS) und in ZF von Tom Zimmer (16 Bit). Das System von Rick van Norman ist ein volles ANS–System. Ich habe in meinem Programm darauf geachtet, dass keine nicht–ANS–konforme Bestandteile verwendet werden. Den für das System von Rick van Norman benötigten DPMI–DOS–Extender erreicht man am besten, indem man mein Programm in der DOS–Umgebung von Windows aufruft. Auf Lauffähigkeit getestet habe ich mein Programm in der genannten Umgebung unter Windows 3.11, Windows 95, Windows 98 und Windows ME. Ich habe darauf geachtet, keines derjenigen ANS–Forth–Worte zu verwenden, die in ZF oder Turbo–Forth nicht akzeptiert werden. Andererseits habe ich keine Worte aus ZF oder Turbo–Forth verwendet, die nicht in ANS enthalten sind. Zur Sicherheit habe ich einige Worte (Beispiele `ON` und `OFF`), auf die Gefahr hin, doppeltgemoppelt zu haben, noch einmal definiert.

Insbesondere habe ich mich bei der Ausgabe von `MATRIX` auf den Bildschirm beschränkt. Die Ausgabe in eine Datei oder auf den Drucker wird der geneigte Leser (die geneigte Leserin) mit den Mitteln des von ihm (ihr) vorgezogenen Forth–Systems selbst ergänzen. Entsprechendes gilt für die Eingaben (hier nur direkt über die Tastatur).

In Turbo–Forth und in dem Forth–System von Rick van Norman lädt man mein Programm per `include hexadoku.fth` ins (schon aufgerufene) Forth–System, in ZF per `fload hexadoku.fth` . (Natürlich muss dabei das Programm als DOS–Datei unter dem Namen `hexadoku.fth` vorliegen (bei anderer Namensgebung entsprechend anders aufzurufen) und im gleichen Verzeichnis liegen wie das aufrufende Forth–System. Eine Unterscheidung zwischen Groß– und Kleinschreibung ist in keinem der genannten drei Forth–Systeme nötig.) Mein Programm läuft bestens auch im jeweils abgesicherten Modus und auch, wenn (wie mir unbeabsichtigt passierte) nur der amerikanische Tastaturtreiber eingeschaltet ist (Forth ist sehr genügsam!).

Von Anfang an wollte ich die Lauffähigkeit des Programms `hexadoku.fth` auch unter dem unterprogramm–gefädelten ANS–Forth–System *MinForth* von Andreas Kochenburger testen, das ganz normal schon unter DOS (bei mir 6.2) läuft. Es gelang zunächst nicht herauszufinden, wie *Include*–Dateien einzuladen sind. Schließlich führte mich ein genaueres Betrachten der von `words` ⟨ret⟩ ausgegebenen Liste darauf, dass `fload hexadoku.fth` ⟨ret⟩ das erledigt. `fload` scheint kein ANS–Wort zu sein, jedenfalls wird es im bekannten ANS–Forth–Buch von Conklin und Rather nicht genannt. Aber auch schon im ANS–Forth–Dokument X3J14 dpANS–6 — June 30, 1993 ist es nicht verzeichnet. Man lernt nie aus! Also: Mein Programm läuft auch unter MinForth.

MinForth akzeptiert zwar erwartungsgemäß auch `key`, liefert aber dabei für die Returntaste keinen Rückgabewert (0Dh) auf dem Stack. Mit `ekey` wäre es in MinForth gegangen, hätte aber in Turbo–Forth und ZF gestört. Um das Programm im allerletzten Moment ohne Fehlergefahr abzuändern, habe ich mich damit beholfen, dass ich nicht nur die Returntaste für den Aussprung aus `hexa` bereitstellte, sondern auch ⟨q⟩ und ⟨Q⟩. Eine weitere Schwierigkeit bei MinForth ist die durch `vorgabe` erzeugte Anzahl der vorgegebenen Elemente. Sie beträgt bei MinForth zwischen 40 und 50. Das ist offenbar zu wenig. Statt 2000 müsste wohl in `hexa` eine geschickt gewählte andere Zahl verwendet werden. Ich überlasse das dem/der Leser/in.

Das zeitraubende (time–consuming) Prüfprogramm `mok?` lässt sich in MinForth (auch in der DOS–Box von Windows ME) wirklich sehr viel Zeit. Am schnellsten (praktisch sofort) geht es im System von Rick van Norman (von mir getestet in der Vollbild–DOS–Box von Windows ME).

```
\ HEXADOKU.FTH

HEX

\ In Turbo-Forth und ZF vorhanden, jedoch nicht ANS
\ -------------------------------------------------

: ON ( ad -- ) -1 SWAP ! ;
: OFF ( ad -- ) 0 SWAP ! ;


\ Matrixaufbau
\ ------------

\ Alle Bearbeitungen (Zeilen-/Spalten-Vertauschung etc) nur fuer MATRIX.
\ Bei VORGABE wird zunaechst MATRIX nach V-MATRIX kopiert und die Vorgabewerte
\ werden in MATRIX gesammelt.
\ XCH-M tauscht MATRIX und V-MATRIX gegeneinander aus.

\ Hexadoku-Matrix, zeilenweise, pro Element ein Byte
VARIABLE MATRIX 0FF ALLOT

\ Vorgabe-Matrix, vom selben Aufbau wie MATRIX
VARIABLE V-MATRIX 0FF ALLOT

\ MATRIX nach V-MATRIX kopieren
: CPY-M ( -- ) 100 0
   DO MATRIX I + C@ V-MATRIX I + C! LOOP ;

\ MATRIX mit V-MATRIX vertauschen
: XCH-M ( -- ) 100 0
   DO MATRIX I + C@ V-MATRIX I + C@ MATRIX I + C! V-MATRIX I + C! LOOP ;

\ MATRIX zu etwa 50% mit vorgegebenen Elementen belegen
\ seed = Anfangsbyte im RAM fuer Zufallsauswahl
\ Kleinerer Belegungsprozentsatz gelingt durch mehrfache Anwendung
: AUSWAHL ( seed -- )
   100 0 DO DUP I + C@ 1 AND 0= IF 20 I MATRIX + C! THEN LOOP DROP ;
```

```
39
40  \ Vorher MATRIX sichern
41  : VORGABE ( seed -- ) CPY-M AUSWAHL XCH-M ;
42
43
44  \ MATRIX-Elemente
45  \ ---------------
46
47  \ Adresse ad[ij] von Element a[ij] in MATRIX, i = Zeile, j = Spalte
48  : AD[IJ] ( i j -- ad[ij] ) SWAP 10 * + MATRIX + ;
49
50  \ Hole Element a[ij] von MATRIX
51  : A[IJ]@ ( i j -- a[ij] ) AD[IJ] C@ ;
52
53  \ Speichere Element a[ij] nach MATRIX
54  : A[IJ]! ( a[ij] i j -- ) AD[IJ] C! ;
55
56
57  \ Zeilen und Spalten
58  \ ------------------
59
60  \ Hole Zeile  i von MATRIX, letztes Element zuerst auf Stack, I=j !
61  : A[I.]@ ( i -- a[i.] ) 00 0F DO DUP I A[ij]@ SWAP -1 +LOOP DROP ;
62
63  \ Hole Spalte j von MATRIX, letztes Element zuerst auf Stack, I=i !
64  : A[.J]@ ( j -- a[.j] ) 00 0F DO I OVER A[IJ]@ SWAP -1 +LOOP DROP ;
65
66  \ Speichere  Zeile i von MATRIX, juengstes Stack-Element zuerst ins RAM, I=j !
67  : A[I.]! ( a[i.] i -- ) 10 00 DO SWAP OVER I A[IJ]! LOOP DROP ;
68
69  \ Speichere Spalte j von MATRIX, juengstes Stack-Element zuerst ins RAM, I=i !
70  : A[.J]! ( a[i.] j -- ) 10 00 DO SWAP OVER I SWAP A[IJ]! LOOP DROP ;
71
72
73  \ Rotationen und Konstantenaddition
74  \ ---------------------------------
75
76  \ Linksrotation der Zeile i von MATRIX um 1 Spalte
77  : ROL1[I.] ( i -- ) >R R@ 0 A[ij]@ R@ A[i.]@ DROP R> A[i.]! ;
78
79  \ Linksrotation der Zeile i von MATRIX um 4 Spalten
80  : ROL4[I.] ( i -- ) 4 0 DO DUP ROL1[I.] LOOP DROP ;
81
82  \ Addition von 4*n zu saemtlichen Elementen von MATRIX
83  : ADD4 ( n -- ) 4 *
84      100 0 DO DUP MATRIX I + C@ + 10 MOD MATRIX I + C! LOOP DROP ;
85
86  \ Rechtsrotation der Zeile i von MATRIX um 1 Spalte
87  : ROR1[I.] ( i -- ) >R R@ A[i.]@ R@ 0F A[ij]@ R> A[i.]! DROP ;
88
89  \ Rechtsrotation der Zeile i von MATRIX um 4 Spalten
90  : ROR4[I.] ( i -- ) 4 0 DO DUP ROR1[I.] LOOP DROP ;
91
92
93  \ Bildschirmdarstellungen
94  \ -----------------------
95
96  \ Spaltensprung
97  : J+ ( i -- ) DUP 0> SWAP 4 MOD 0= AND IF SPACE THEN ;
98
99  \ ASCII --> Ziffernausgabe
100 : ZIFF ( n i -- ) J+ DUP 0F U> IF EMIT SPACE ELSE . THEN ;
101
102 \ Zeilensprung
103 : I+ ( i -- ) DUP 0> SWAP 4 MOD 0= AND IF    CR THEN ;
104
```

```
105  \ Bildschirmdarstellung von MATRIX
106  : .M ( -- ) 10 0 DO I I+ CR I A[I.]@ 10 0 DO I ZIFF LOOP LOOP ;
107
108  \ Bildschirmdarstellung von V-MATRIX
109  : .V ( -- )  CR 10 0
110     DO I I+ 24 SPACES 10 0
111        DO V-MATRIX J 10 * + I + C@ I ZIFF
112        LOOP CR
113     LOOP ;
114
115  \ Bildschirmdarstellung von MATRIX & V-MATRIX
116  : .M&V ( -- ) CR 10 0
117     DO 10 0 DO   MATRIX J 10 * + I + C@ I ZIFF LOOP 5 SPACES
118        10 0 DO V-MATRIX J 10 * + I + C@ I ZIFF LOOP CR I 1+ I+
119     LOOP ;
120
121
122  \ Abwandlungen von MATRIX
123  \ -----------------------
124
125  \ Kanonische Matrix --> MATRIX
126  : KANON ( -- )
127     00 0F DO I -1 +LOOP 00 A[I.]!
128     10 01 DO I 4 MOD 0=
129             IF   I 04 - A[I.]@ I A[I.]! I ROL1[I.]
130             ELSE I 01 - A[I.]@ I A[I.]! I ROL4[I.]
131             THEN
132           LOOP ;
133
134  \ Linkskanonische Matrix --> MATRIX
135  : KANONL ( -- ) KANON ;
136
137  \ Rechtskanonische Matrix --> MATRIX; entspricht KANON mit ROR statt ROL.
138  : KANONR ( -- )
139     00 0F DO I -1 +LOOP 00 A[I.]!
140     10 01 DO I 4 MOD 0=
141             IF   I 04 - A[I.]@ I A[I.]! I ROR1[I.]
142             ELSE I 01 - A[I.]@ I A[I.]! I ROR4[I.]
143             THEN
144           LOOP ;
145
146  \ Transponierte von MATRIX, a[ij] <--> a[ji] fuer i < j
147  : TRANSP ( -- )
148     10 0 DO
149     10 0 DO I J < IF I J A[IJ]@ J I A[IJ]@ I J A[IJ]! J I A[IJ]! THEN
150          LOOP
151          LOOP ;
152
153  \ Viererquadrate und Streifen
154  \ ---------------------------
155
156  \ Aus Zeile/Spalte mach Viererquadrat  (0...f)
157  : IJ>V ( i j -- v )  4 / SWAP 4 / 4 * + ;
158
159  \ Aus Zeile/Spalte mach Querstreifen   (0...3)
160  : IJ>Q ( i j -- q )  DROP 4 / 4 * ;
161
162  \ Aus Zeile/Spalte mach Laengsstreifen (0...3)
163  : IJ>L ( i j -- l )  NIP  4 / 4 * ;
164
165  \ Aus Quer/Laengs  mach Viererquadrat  (0...f)
166  : QL>V ( q l -- v )  SWAP 4 * + ;
167
168  \ Aus Viererquadrat mach Quer/Laengs   (0..3, 0..3)
169  : V>QL ( v -- q l )  4 /MOD SWAP ;
170
```

```
171  \ Tausche Zeile  i1 in MATRIX gegen Zeile  i2
172  : XCHI ( i1 i2 -- )
173     >R >R R@ A[I.]@ R> R@ SWAP >R A[I.]@ R> A[I.]! R> A[I.]! ;
174
175  \ Tausche Spalte j1 in MATRIX gegen Spalte j2
176  : XCHJ ( j1 j2 -- )
177     >R >R R@ A[.J]@ R> R@ SWAP >R A[.J]@ R> A[.J]! R> A[.J]! ;
178
179  \ Kehre Zeilenfolge  in MATRIX um ( i <--> 0f-i )
180  : INVERSI ( -- )  8 0 DO I 0F I - XCHI LOOP ;
181
182  \ Kehre Spaltenfolge in MATRIX um ( j <--> 0f-j )
183  : INVERSJ ( -- )  8 0 DO I 0F I - XCHJ LOOP ;
184
185  \ Tausche Querstreifen   q1 in MATRIX gegen Querstreifen   q2
186  : XCHQ ( q1 q2 -- ) 4 MOD SWAP 4 MOD
187     4 0 DO 2DUP 4 * I + SWAP 4 * I + XCHI LOOP 2DROP ;
188
189  \ Tausche Laengsstreifen l1 in MATRIX gegen Laengsstreifen l2
190  : XCHL ( l1 l2 -- ) TRANSP XCHQ TRANSP ;
191
192
193  \ Hauptprogramm: Folge von Hexadokus
194  \ ----------------------------------
195
196  \ Zaehler
197  VARIABLE INDEX
198
199  \ Bildschirmanzeige mit zugehoeriger Loesung (LOES? = ON/OFF)?
200  VARIABLE LOES?
201  LOES? ON       \ Default
202  \ Start von HEXA bei kanonischer MATRIX?
203  VARIABLE KANON?
204  KANON? ON      \ Default
205
206  \ Beliebig viele Vorgabematrizen erzeugen und anzeigen.
207  \ Bei LOES? OFF nur Vorgaben, bei LOES? ON auch Loesungen.
208  \ KANON? ON : Start mit kanonischer MATRIX
209  \ Return-Taste oder [q] oder [Q] = raus, andere Taste = naechstes Bild.
210  : HEXA ( -- )
211     ." Taste oder ( [ret] oder [q] oder [Q] ) druecken!" CR
212     2000 INDEX ! ( seed zu Beginn der Folge )
213     KANON? @ IF KANON THEN
214     BEGIN
215       KEY DUP
216       0D ( [ret] ) = OVER 71 ( q ) = OR SWAP 51 ( Q ) = OR IF EXIT THEN
217       5 INDEX +!
218       INDEX @ C@ 1 AND IF TRANSP  THEN 3 INDEX +!
219       INDEX @ C@ 1 AND IF INVERSI THEN 3 INDEX +!
220       INDEX @ C@ 1 AND IF INVERSJ THEN 3 INDEX +!
221       INDEX @ C@ INDEX @ 1+ C@    XCHQ 3 INDEX +!
222       INDEX @ C@ INDEX @ 1+ C@    XCHL 3 INDEX +!
223       INDEX @ C@                  ADD4 3 INDEX +!
224       INDEX @ C@ 4 / 4 * 10 MOD DUP
225       INDEX @ 1+ C@ 4 MOD + SWAP INDEX @ 2 + C@ 4 MOD + XCHI 5 INDEX +!
226       INDEX @ C@ 4 / 4 * 10 MOD DUP
227       INDEX @ 1+ C@ 4 MOD + SWAP INDEX @ 2 + C@ 4 MOD + XCHJ 5 INDEX +!
228       INDEX @ VORGABE LOES? @ IF .M&V ELSE .V THEN
229       CR ." Weiter mit Taste, raus mit [ret] oder [q] oder [Q] !"
230     AGAIN ;
231
232
233  \ Zulaessigkeitspruefungen
234  \ ------------------------
235
236  \ Tritt n in Zeile i mindestens zweimal auf ? Meldung, wenn ja.
```

```
237  \ Funktioniert auch fuer Leerstellen (n=20h)
238  : 2*INI? ( n i -- )
239     INDEX OFF 10 * MATRIX +
240     10 0 DO 2DUP I + C@ = IF 1 INDEX +! THEN LOOP 2DROP
241     INDEX @ 1 > ABORT" Mindestens zweimal!" ;
242
243  \ Tritt n in Spalte j mindestens zweimal auf ? Meldung, wenn ja.
244  \ Funktioniert auch fuer Leerstellen (n=20h)
245  : 2*INJ? ( n j -- ) TRANSP 2*INI? TRANSP ;
246
247  \ Tritt n im Viererquadrat v mindestens zweimal auf ?
248  \ Funktioniert auch fuer Leerstellen (n=20h)
249  : 2*INV? ( n v -- )
250     INDEX OFF V>QL SWAP 10 * + 4 * MATRIX +
251     4 0 DO 2DUP I + C@ = IF 1 INDEX +! THEN LOOP 10 +
252     4 0 DO 2DUP I + C@ = IF 1 INDEX +! THEN LOOP 10 +
253     4 0 DO 2DUP I + C@ = IF 1 INDEX +! THEN LOOP 10 +
254     4 0 DO 2DUP I + C@ = IF 1 INDEX +! THEN LOOP 10 + 2DROP
255     INDEX @ 1 > ABORT" Mindestens zweimal!" ;
256
257  \ Ist MATRIX zulaessig? Meldung, wenn nicht.
258  : MOK? ( -- )
259     10 0 DO 10 0 DO J I 2*INI? LOOP LOOP
260     10 0 DO 10 0 DO J I 2*INJ? LOOP LOOP
261     10 0 DO 10 0 DO J I 2*INV? LOOP LOOP ;
```

Lebenszeichen

Berichte aus der FIG Silicon Valley: *Henry Vinerts*

Dezember 2005

Lieber Fred,

dank der Notiz, die Birgit Prinz an meinen *Lebenszeichen*–Bericht in der VD 3/4–2005 angehängt hat, kenne ich jetzt einige alte chinesische Weisheiten, die ich mit all denen teilen werde, die zu viele Dinge zu tun und nicht genug Zeit dafür haben. Wie Du siehst, lese ich jetzt das Buch, welches Glen Haydon auf dem September–Treffen 2005 der SVFIG erwähnte. Es ist *The Importance of Living* (*Weisheit des lächelnden Lebens* in German) von Lin Yutang, schon 1937 geschrieben und immer noch sehr aktuell. Dies gilt insbesondere für alte Männer wie mich, die nach jeder Entschuldigung suchen, das Leben in Faulheit zu genießen und nicht jede Aufgabe zu beenden, die sie begonnen haben. Birgit, es gibt mir *viel Spaß beim Lesen*. Sobald ich es ausgelesen habe, werde ich mich viel besser fühlen — trotz all der Bücher, die ich gekauft, aber nicht gelesen habe, obwohl ich nicht jeden Brief beantwortet, meinen Schreibtisch und meine Schränke nicht aufgeräumt habe, und obwohl ich nicht genug Forth gelernt habe, um aus meinem Novizen–Status herauszukommen. Und es gibt da noch viel mehr...

Die chinesische Philosophie, schreibt Lin Yutang, besteht in Folgendem: „Man muss mit dem Glauben beginnen, dass es keine Katastrophen auf dieser Welt gibt und dass es neben der erhabenen Kunst, Dinge zu tun, die noch erhabenere Kunst gibt, Dinge ungetan zu lassen.“ „Vom chinesischen Standpunkt aus“, sagt er, „ist der Mann, der klugerweise nichts tut, der kultivierteste.“ Gebt mir ein Glas Wein, und dieses Zitat und ich sollte in der Lage sein, den eingebildeten Forderungen dieses *Rattenrennens*, das wir Leben in diesem Land der Gelegenheiten nennen, zu widerstehen.

Gut, nun zum SVFIG–Treffen vom 17. Dezember 2005. Wie zu erwarten war, hat dieses Treffen wohl den Rekord der geringsten Teilnehmerzahl gebrochen. Es waren nicht mehr als 9 von uns anwesend. Es war die reguläre Kerntruppe, wie ich sie nennen möchte. Wir hörten eine Stunde eher auf, aber es war ein gutes Treffen. Ich habe eine Seite voller Notizen von nur drei Sprechern. Natürlich fehlte es Dr. Ting nicht an Dingen, über die er uns in der Morgensitzung berichtete. Er hat seine eigene Webseite (`http://www.offete.com`) mittels Yahoos *Site Builder*–Software eingerichtet. Ich lade euch ein, Sie für all die Details zu besichtigen, die ich auslassen werde, da ich beginne, meine neue berechtigte Faulheit zu üben. Ting zeigte uns den ARM Chip aus Taiwan, auf dem sein eForth läuft, sprach über eForth v.2, welches er auf seinem eigenen Chip nutzt, da es subroutine–threaded so viel schneller läuft. Er beendete seine Präsentationen mit der Beschreibung eines hand–gelöteten DSO–Boards (Digital Speicher Oszilloskop), welches wir uns näher anschauen konnten. Ich habe dieses Projekt schon in früheren Berichten erwähnt. In der Nachmittags–Sitzung demonstrierte Dave Jaffe eine Anwendung von *Photo Story for Windows*, die ich ganz interessant fand. Die Software kann frei heruntergeladen werden und kann auf einer DVD eine erzählte Geschichte mit ausgewählter Hintergrundmusik in der Art eines Photo–Albums erzeugen. Ein reguläres Windows XP mit Media Player, einem DVD–Brenner und einem Scanner von dem die JPG–Bilder importiert werden können, ist alles, was man neben dem eigenen Talent, eine Geschichte zu erzählen und zu präsentieren, braucht. Das Programm bietet Editor–Funktionen, die eine Umordnung der Bilder erlauben und die Synchronisation der Anzeigedauer der Bilder mit den begleitenden Texten oder der Hintergrundmusik unterstützen. Das scheint ein gutes Projekt für alte Forther zu sein, die ihre Lebensgeschichte für die Nachwelt zusammenstellen wollen.

Kein SVFIG–Treffen wäre ohne Kevin Apperts Beschreibung von *Cool things I saw on the Web lately* denkbar. (Ich bin nicht sicher, wie ich das Adjektiv *cool* übersetzen soll, welches von der momentan herrschenden Generation hier im Tal benutzt wird, *attraktiv, interessant, nett...*. Ich hoffe, es ist in Dr. Beierleins Wörterbuch (Sorry, Henry, in German it is only *cool*. Imported from Overseas `;-)`.) Kevin ist ein ziemlicher Experte darin, im World Wide Web alles zu finden, was auch immer er wissen will. Falls ich es nicht falsch verstanden habe, hat er unseren Webmaster, Dave Jaffe, überredet, Links zu Google.com auf die FIG/SVFIG Webseite aufzunehmen. Ich sollte hier aber aufhören und selbst nachsehen.

So, das war es, Leute, für 2005. Ich hoffe, dass das Jahr 2006 uns günstige *Verbesserungen/Updates* in unser gesamtes Leben bringt.

Henry.

Übersetzer: Thomas Beierlein

Januar 2006

Lieber Fred,

ich weiss nicht, ob es Samuel Johnson oder Karl Marx war, der den Ausspruch *Der Weg zur Hölle ist mit guten Absichten gepflastert* geprägt hat, aber ich hatte gute Vorsätze, rechtzeitig zu schreiben, um Dr. Beierlein in der Zeit für die Übersetzung zu erreichen, in der er nicht zu beschäftigt war. Das war vor nahezu vier Wochen. Meine Bequemlichkeit tut mir Leid. Vielleicht kann ich aushelfen, indem ich versuche, mein Motto selbst ins Deutsche zu übersetzen. Wie klingt das für Dich? *Saumseligkeit ist die Kunst, den Schritt mit gestern zu halten*[1].

[1] *Saumseligkeit ist die Kunst, mit gestern Schritt zu halten* sounds better in German. Th. Beierlein

Nach Aussagen von Don Marquis (1876–1937), einem amerikanischen Humoristen und Zeitungsschreiber, wurde Archys ursprüngliche Maxime *Saumseligkeit ist die Kunst, mit gestern Schritt zu halten* auf Dons Schreibmaschine von einer Kakerlake namens Archy geschrieben, die in ihrem Vorleben ein *Poet der freien Verse* war. Archy hat das nachts geschrieben, indem sie auf den Tasten der Schreibmaschine, immer eine nach der anderen, herumsprang — natürlich ohne Großbuchstaben oder Satzzeichen. Archys Freund, Mehitabel, hat Marquis' Geschichten weitere Farbe hinzugefügt. Heute bringen mich diese Charaktere — wie ein Straßenkater in seinem neunten Leben, eventuell eine Wiederauferstehung von Cleopatra — zurück zu einem Kommentar von unserem John Carpenter, den er vor langer Zeit auf dem SVFIG–Treffen am 19. August 1998 machte (Ja, ich habe meine alten Notizen immer noch!). John sagte, falls C–Programmierer wie Hunde wären, dann sind Forth–Programmierer die *verwilderten Straßenkatzen*. Wenn ich John das nächste Mal treffe, werde ich ihn fragen, ob er sich an diese Bemerkung erinnert und was ihn zu diesem Ausspruch bewogen hat.

Soviel zu meiner Einleitung. Nun, da ich meine Tastatur aufgewärmt habe, lasst mich schauen, was für Notizen ich auf dem letzten SVFIG–Treffen gemacht habe. Bevor ich beginne und einiges von dem wiederhole, was Dave Jaffe inzwischen auf unserer Webseite veröffentlicht hat, möchte ich dich auf folgende URLs aufmerksam machen: `http://www.forth.org/svfig/kk/01-2006.html` und `http://www.forth.org/svfig/kk/02-2006.html` enthalten die Notizen für das Januar– und Februar–Treffen.

Das Treffen am 28. Januar begann mit einer Gruppe von ungefähr einem Dutzend Leuten und erweiterte sich auf 15 um die Mittagszeit. Dr. Ting war gleich früh da, um stolz sein Weihnachtsgeschenk zu zeigen, das er sich selbst gemacht hatte: das LEGO Mindstorms Roboter Kit, Version 2.0. Hätte ich gewusst, dass er eins haben wollte, hätte ich ihm meines überlassen. Dies liegt seit ca. 6 Jahren in meinem Schrank und wird dort wohl bleiben. Der RCX–Robot ruht in Frieden und wartet auf ein neues Lebenselixir von Ralph Hempel oder irgendeinem anderen. Vielleicht ist dies dann Ting selbst, obwohl es so aussieht, als ob das neue Mindstorms–NXT–Robot–Kit, welches LEGO für die Markteinführung im nächsten August vorbereitet, den alten 8–Bit–Hitachi–betriebenen Baustein für immer im Roboter–Friedhof ruhen lassen wird. Offensichtlich war sich Ting nicht des neuen NXT bewusst, als er sein Weihnachtsgeschenk kaufte. Als er Ralph anrief, um mehr über das pbForth zu erfahren, war Ralph so überrascht, dass er als Antwort nur fragen konnte, ob es wirklich der Ting am anderen Ende der Leitung war. Nun bin ich gespannt, ob Ting einen Weg findet, diesen alten Roboter mit seinem eForth laufen zu lassen, auch wenn es eventuell bedeutet, dass er einige Organverpflanzungen am LEGO–Stein vornehmen muss.

Dave Jaffe setzte seine Serie von *Mehr über neue Produkte — Präsentation in PDF mit Weblinks* fort und beschrieb kürzlich eingeführte Dinge, die mit Forth in eingebetteten Systemen benutzt werden können. Unter den improvisierten Hinweisen und Vorschlägen aus dem Puplikum zu damit zusammenhängenden Weblinks, verlängerte sich sein Informations–Austausch zum Teil über die Mittagspause hinaus. Ich kann feststellen, dass ein immer größerer Teil der SVFIG–Mitglieder zu den Treffen mit ihren eigenen *Masseninformations–Waffen (weapons of mass information)* kommt, hauptsächlich Laptops, die sie an die Internetzugänge unseres Gastgebers, Cogswell College, anschließen können. Vor vier Jahren (genau am 21. Dezember 2001) schrieb ich in meinem Bericht an die VD, dass ich mich in der Bibliothek des College wie der einzige Passagier in einem Flugzeug mit 10 Piloten und Navigatoren fühlte, welches im Internet herumkreiste. Das war das erste Mal, dass wir von unseren SVFIG–Treffen aus direkt auf das Internet zugreifen konnten. Hätten wir heute überhaupt noch Treffen, wenn ein solcher Zugang nicht möglich wäre? Oder wie sähe es aus, wenn uns das College seine Beamer nicht zur Verfügung stellen würde?

Der Höhepunkt des Januar–Treffens war für mich die Präsentation von Dirk De Mol, einem eingeladenen Sprecher von National Instruments. Dirk ist nicht nur ein interessanter Redner — darüber hinaus ist er ein erfahrener Astronom und LabVIEW–Programmierer. Sein Vortrag erklärte die Eigenheiten und Vorteile von LabVIEW als Steuersprache für das vorgesehene South African Large Telescope (SALT). LabVIEW, welches von ASYST abstammt, gibt es nun schon eine ganze Weile. Es ist eine datenfluss–orientierte Sprache, in der man mit Diagrammen direkt in Maschinensprache programmieren kann. Sie benutzt keine sequentiellen Textanweisungen (John Carpenter nannte es in diesem Zusammenhang *Bilder–FORTH*).

Vielleicht war es Zufall, dass parallel zu Tings Bemühungen um die Mittagszeit, seine Vorführungen des zögerlichen LEGO–RCX–Roboters zu beenden, Dirk mit eintraf und uns eifrig darüber informierte, dass das neue verbesserte LEGO Mindstorms Kit 95 Dollar kosten wird. Weiterhin informierte er uns, dass sein 16–Bit–Prozessor eine Bluetooth–Verbindung haben wird, mittels NXT gesteuert wird und dass NXT unter LabVIEW läuft.

Die Fülle der Informationen über SALT und andere vergleichbare Teleskope, die uns Dirk präsentierte, war äußerst interessant, aber zuviel für mich, um sie jetzt hier zusammenzufassen. Ich musste den Raum während des letzten und nicht weniger interessanten Vortrages verlassen. Er wurde von Jay McKnight gehalten, unserem eigenen Langzeit–Forther, Eigner des Magnetic Reference Laboratory. Damals im Jahre 1973 wurde Jay als Experte von Richter Sirica engagiert, um zweifelsfrei festzustellen, dass $18\frac{1}{2}$ Minuten der berühmten Nixon/Watergate–Bänder gelöscht waren. Ich vermute, dass mehr als die Hälfte unserer Gruppe sich daran erinnert, auch wenn Forth nicht mit der Amtsenthebung unseres Präsidenten zu tun hatte. Nebenbei, weiß jemand, wann Forth das erste Mal Schlagzeilen auf globalem Niveau machte? War es die in Forth programmierte Argo, als sie die Titanic 1985 gefunden hatte?

Ich sehe, dass Dave im Web ungefähr zwei Seiten mit Notizen vom SVFIG–Treffen am 25. Februar veröffentlicht hat. Da ist meinerseits nicht mehr allzuviel hinzuzufügen. Um 10 Uhr waren wir nur zu 8 und nicht mehr als 4 bis 5 kamen im Laufe des Tages noch hinzu. Es war genug Kaffee, Brötchen und Backwaren für alle vorhanden. Ich glaube, nur Dr. Tings große Teekanne war am Ende des Tages leer. Ting füllte den Morgen mit Berichten über seine laufenden Projekte und tauschte technische Tips und Details mit denjenigen aus, die ihn verstanden und in der Lage waren, zur Diskussion beizutragen. Ich erfuhr, dass kein Chinese von außerhalb nach Hong Kong darf, dass die Grenze schwer bewacht wird und dass das Welthauptquartier für Hacker von Mobiltelefonen in einer Stadt nördlich von Hong Kong liegt; jedoch kann ich den Namen der Stadt nicht aussprechen.

Dave Jaffe präsentierte *Teil 3* seiner Sammlung interessanter Dinge aus dem Web. Kevin Appert fügte danach eine lange Liste eigener Links hinzu, die — natürlich — sofort begutachtet wurden, soweit ausreichendes Interesse der Zuhörer vorlag. Einige weitere brachten Dinge von persönlichem oder allgemeinem Interesse zur Sprache und das Treffen bekam einen zumeist unstruktierten, informellen Ablauf. Teilweise wurde an zwei oder drei Laptops gleichzeitig gearbeitet und gegooglete Dinge aus dem Web zum Vorschein gebracht.

John Peters hatte die Möglichkeit, uns sein *Hauptgericht* in Forth–Programmierung zu zeigen. Es geht um elektronische Kostenvoranschläge, die er nun online direkt von einer Baustelle aus mittels VNC bearbeiten kann.

Der Tag endete damit, dass wir ein Album signierten, welches Dave zum Andenken an Henrik Thurfjell vorbereitet hatte.

Henry.

Übersetzer: Thomas Beierlein

März 2006

Lieber Fred,

Ich habe hier noch einen Bericht über SVFIG für dich, ehe ich zu meiner zweiten Hüftoperation muss, die mich wahrscheinlich bis zum Juni von den Treffen fern hält. Ich hoffe dass bis dahin die Crew des Schiffs Die Vierte Dimension einen neuen Kapitän gefunden hat, der es tapfer, manchmal sogar einhändig, forth–segeln wird.

Die reguläre Crew des SVFIG–Schiffs versammelte sich wieder am 25. März 2006 unter Führung von Kapitän George Perry. Die Offiziere kamen alle — sogar John Hall, der erste Kapitän des einst so stolzen Flagschiffs FIG, stieß im Laufe des Tages zu uns.

Wie es fast immer zur Tradition geworden war, begann das Treffen wieder mit einem Vortrag des Chefingenieurs des Schiffs, C. H. Ting. Er beschrieb sechs der Projekte an denen er zur Zeit arbeitet, wobei die aktuelle Version (5.20) seines eForth das Hauptwerkzeug in seinem Werkzeugkasten ist. Im Februar letzten Jahres schrieb ich Friederich über Tings Vorlesungen zu chinesischer Musik und der vereinfachten Musik–Notation, die die Chinesen um 1900 herum entwickelten und die heute noch benutzt wird. Im Wesentlichen unterscheidet es sich von der westlichen Notation darin, dass es anstelle von Noten, platziert auf einem fünflinigen Raster, Zahlen und Spezialzeichen benutzt, die wie ein Textstring aneinander gereiht werden. Nun, es scheint, dass unser erfindungsreicher Chefingenieur der Erste ist, der versucht ASCII als Werkzeug zur Digitalisierung chinesischer Musik zu verwenden. Er hat jetzt sein eigenes System zur Transkribierung der chinesischen Musiknotation in ASCII–Textdateien entwickelt, mit dem er über MIDI die Musik über die Lautsprecher eines Computers wiedergeben kann. Wir hörten Ausschnitte aus 27 Songs, die Ting innerhalb dreier Tage aus den *Canaan Hymnal* (Nov. 1998, ISBN 957–9642–55–9) umgesetzt hat. Er hat sogar seine eigene Harmony auf der Basis der ersten Noten durch zusätzliche Akkorde hinzugefügt.

Unter den anderen Projekten, die Ting davon abhielten, die restlichen 760 Songs im Buch zu rekodieren, erfasste vor allem eines unsere Aufmerksamkeit, da er extra dafür bezahlt wurde, Forth in einer Schaltung eigenen Designs einzusetzen. Es handelt sich um einen Laderegler für Batterien mit einem breiten Spannungsbereich. Übrigens erinnerte mich die Lade/Entladekurve, die Dr. Ting an die Tafel zeichnete, an die S–Kurven aus dem Buch *Vorhersagen* von Theodore Modis, welches ich gerade kürzlich zu lesen begann, als ich erkannte, dass das Leben von Forth einer ebensolchen S–Kurve folgen mag.

SVFIGs Zahlmeister John Rible kam aus Santa Cruz, Ken Morley hatte einen längeren Weg von seinem Heim, ca. 150 km entfernt, zurückgelegt. Glen Haydon kam von seinem Seeblick–Hügel in La Honda herunter. Die eher lokaleren *Decksarbeiter* — Herman Griffin, Jay McKnight, Tom Gregory, Alan Furman — waren auch da, bereit im World Wide Web zu segeln, falls Sie nicht anderweitig beschäftigt waren. Natürlich vergesse ich Dave Jaffe, unseren Webmaster, nicht, der in den alten Tagen der Seefahrt wohl als Schiffsfunker gedient hätte. Und ich möchte den Chefmaat des Kapitäns, Kevin Appert, erwähnen, um darauf hinzuweisen, dass durch seine Bemühungen unser Programm mit Gesprächen und Präsentationen interessanter Besucher, wie z.B. Dave Wyland, den er uns nach dem Essen vorstellte, bereichert wurde.

Daves Spezialgebiet ist die Robotik. Er sagte, dass seine Anfänge mit Forth bis zur Geburt des *Homebrew Robotics Club* zurückbasieren (Ich glaube es war ein Ableger des Homebrew Computer Clubs im Silicon Valley in den 70ern. Er sagte, dass er sich daran erinnert, den Namen *Forth Dimensions* für die Zeitschrift vorgeschlagen zu haben, welche, wie wir alle mit Bedauern wissen, 1999 starb. Dabei hatte sie nicht einmal die 21 erreicht. Das Thema seines Vortrages war *SIFT, FPGAs, and Forth*. Wieder einmal zu reichhaltig und zu fortgeschritten für mich, um es mit den Worten eines Laien zu erklären. Er erwähnte, dass der Text seiner Präsentation im Web verfügbar ist. Dort gibt es auch mehr über SIFT (Scale–Independent Feature Transform), welches auf David Lowe von der University of British Columbia

zurückgeht. Ich habe soviel mitbekommen: Falls es genug Computer–Wissenschaftler gibt, die sowohl Tierdressur als auch Gaußsche Filter, FFT und Laplace–Operatoren beherrschen, dann mag da Hoffnung sein, dass wir eines Tages einen Roboter herstellen können, der das Kommando *Los, hol mir mal ein Bier aus dem Kühlschrank* befolgen kann.

Der Tag schloss mit Dave Jaffe, der mehr von seinen neuen Computererzeugnissen aus dem Web präsentierte, mit John Rible, der über Chuck Moores letzte Bemühungen bei Intelasys berichtete, und mit Kevin Apperts Kommentaren zu interessanten Webseiten. Die nächsten beiden Treffen sind jeweils für den dritten Samtag im April und Mai geplant. Es kann sein, dass du erst im Juni wieder von mir hörst. Viel Glück dabei, Forth in Deutschland am Leben zu erhalten!

Mit besten Wünschen,

Henry

Übersetzer: Thomas Beierlein

Gehaltvolles

zusammengestellt und übertragen von *Fred Behringer*

VIJGEBLAADJE der HCC Forth–gebruikersgroep, Niederlande Nr. 54, Februar 2006

Das Vijgeblaadje erscheint in jedem geraden Monat um den Ersten herum.

Verandering *Ron Minke*

Ron stellt sich mit drei Forth–Zeilen als neuer Redakteur vor.

Vooraankondiging: de ALV *Die Redaktion*

Die nächste Jahresversammlung findet (fand) am 8. April 2006 statt.

Nieuwigheden *Ron Minke*

Ron geht auf den R8C/13-Prozessor ein, der über die holländische Zeitschrift Elektuur nun auch den holländischen Enthusiasten bekanntgemacht wurde. Er erwähnt „onze Duitse Forth collega's," die bereits damit begonnen haben, ein Forth-System darauf hochzuziehen. Und er bringt eine Beschreibung der Möglichkeiten des R8Cs in der Darstellungsweise seiner Artikelserie „Forth von der Pike auf" (deutsche Übersetzung ab VD–Heft 3–4/2005): *Allgemeines — der Chip — die Peripherie — die Register — was bringt uns dieser Prozessor?*

De lichtkrant *Paul Wiegmans*

Ein ausgedientes LED–Laufschrift–Gerät der Marke Surtronic. Der Autor möchte (oder hat?) ein ATS–Board mit einem selbstgeschriebenen Forth–Programm zur Ansteuerung verwenden(t). Er zeigt ein Blockschema und deutet an, „was nötig war, um das Display zum Leben zu erwecken."

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

HCC-Forth-gebruikersgroep.

Für 10 € pro Jahr schicken wir Ihnen 5 oder 6 Hefte unserer Vereinszeitschrift 'Het Vijgeblaadje' zu. Dort können Sie sich über die Aktivitäten unserer Mitglieder, über neue Hard- und Softwareprojekte, über Produkte zu günstigen Bezugspreisen, über Literatur aus unserer Forth-Bibliothek und vieles mehr aus erster Hand unterrichten. Auskünfte erteilt:

Willem Ouwerkerk
Boulevard Heuvelink 126
NL-6828 KW Arnhem
E-Mail: `w.ouwerkerk@kader.hobby.nl`

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

Format–Hinweise für Autoren

Ulrich Hoffmann

Die Vierte Dimension wird seit Ausgabe 1/2006 (`http://www.forth-ev.de/filemgmt_data/files/4d2006-01.pdf`) mit Hilfe von LaTeX produziert. Anzeigen, Artikel und Leserbriefe können in verschiedenen Formaten eingeliefert werden. Die folgenden Informationen beschreiben die bevorzugten Formate und geben einige Hinweise, durch welche die Aufbereitung vereinfacht wird.

Allgemeine Hinweise

Zeichenkodierung

Wenn möglich, so sollten Dokumente im Format Unicode UTF–8 kodiert werden. Wenn dies nicht sicher möglich ist, dann ist es besser, Dokumente im *Hausformat* einzuliefern und das Format (oder die Plattform) mit anzugeben. Die Daten werden dann vor weiterer Bearbeitung nach UTF–8 konvertiert.

Hervorhebungen

Hervorzuhebende Texte sollten am besten *kursiv* oder **fett** geschrieben werden.

Bitte auf Unterstreichungen grundsätzlich verzichten.

Obwohl das S p e r r e n von Text möglich ist, ist das eher eine veraltete Art, Texte hervorzuheben, und sollte vermieden werden.

Auch Text, der GANZ groß geschrieben ist, sollte wenn möglich nicht verwendet und stattdessen besser *kursiv* oder **fett** hervorgehoben werden.

Verwendung von Anführungszeichen

Im Textsatz werden unterschiedliche Arten von Anführungszeichen verwendet: Öffnende und schließende Anführungszeichen treten u. a. in „deutscher“, “englischer” und «französischer»Variation auf.

Bitte in eingelieferten Dokumente Anführungszeichen nur dann verwenden, wenn tatsächlich zitiert wird oder wörtliche Rede auftritt. Statt Begriffe, die nicht ganz *passen*, in Anführungszeichen zu setzten, sollten diese lieber *kursiv* gesetzt werden.

Binde– und Gedankenstriche

Typographisch gibt es Trennstriche in mehreren Längen — für die Worttrennung am Zeilenende, als Gedankenstrich und zum Bilden zusammengesetzter Hauptwort–Ketten — die alle ihre Berechtigung haben. In LaTeX oder LyX werden sie mit -, -- oder --- erzeugt. In der Vierten Dimension werden die Trennstriche wie in diesem Text verwendet.

Für andere eingelieferte Dokumente ist es das Einfachste, keine Versuche zu unternehmen, um diese unterschiedlichen Trennstriche zu erzeugen. Bitte stattdessen immer das einfache Minuszeichen (U+002D) - verwenden. Die typographischen Trennstriche werden während des Redigierens eingefügt.

Graphiken

Da die Vierte Dimension gedruckt wird, ist es sinnvoll, dass Bilder normalerweise in einer Auflösung von mindestens 150dpi vorliegen. Bitte immer zu eventuell eingebetteten Bildern auch die Bilder in eigenen Files einliefern. Alle gängigen Graphik–Formate können verwendet werden. Während des Redigierens werden sie in Encapsulated-Postscript-Files (`.eps`) gewandelt.

ASCII–Art

ASCII–Art kann gerne verwendet werden. Die betreffenden Stellen werden im `Schreibmaschinestil` nicht–proportional gesetzt.

Programmlistings

Listings sind willkommen. Sie werden zum leichteren Zitieren in späteren Leserbriefen mit Zeilennummern versehen. Bitte nach Möglichkeit den Quelltext auch zusätzlich in einem eigenständigen File einliefern, damit er unverändert auf `www.forth-ev.de` zur Verfügung gestellt werden kann.

LaTeX

LaTeX–Dokumente können ohne wesentliche Nachbearbeitung direkt übernommen und dabei mit dem Stil der Vierten Dimension versehen werden. Die Dokumente bleiben eigenständig und können sowohl im Rahmen der Vierten Dimension als auch als einzelnes Dokument gesetzt werden.

Wenn Nicht–Standardpakete verwendet werden, ist es von Vorteil, hervorzuheben, wo diese Pakete zu beziehen sind.

Es gibt viele gut unterstützte TeX–Distributionen, etwa für teTeX für Unix (`http://www.tug.org/teTeX/`) oder MiKTeX für Windows (`www.miktex.org`).

Weitere Informationen finden sich unter `http://de.wikipedia.org/wiki/TeX`.

LyX

LyX (`www.lyx.org`) ist ein einfaches Textverarbeitungsprogramm auf Basis von LaTeX, das nach dem Prinzip *What You See is What You Mean* arbeitet. Während des Erstellens von LyX–Dokumenten wird eine der späteren Ausgabe angenäherte graphische Darstellung des Dokuments präsentiert, die das bequeme Ändern des Dokuments erlaubt. LyX ermöglicht es so, dass Dokumente

auch ohne umfangreiche LaTeX–Kenntnisse erzeugt werden, die dennoch die Funktionalität und Qualität von mit LaTeXgesetzten Dokumenten erreichen können.

Fortgeschrittene Nutzer behalten mit der Quelltextbearbeitungsmöglichkeit zudem volle Kontrolle über ihr Dokument, können aber weiterhin Standardaufgaben elegant in der graphischen Oberfläche erledigen.

LyX–Dokumente können ohne wesentliche Nachbearbeitung direkt in die Vierte Dimension übernommen werden. Wenn Nicht–Standardpakete verwendet werden, bitte ausdrücklich unter Angabe der Bezugsquelle auf sie hinweisen.

Einen komfortablen Installer für Windows findet man hier: `ftp://ftp.lyx.org/pub/lyx/contrib/LyXWinInstaller`.

Text

Text eignet sich hervorragend für Nachrichten und Leserbriefe. Absätze sollten durch Leerzeilen voneinander getrennt werden. Hervorhebungen können beispielsweise durch umschließende ***Sterne*** oder */Slashes/* ausgedrückt werden. Während des Redigierens werden sie **fett** bzw. *kursiv* gesetzt. Überschriften stehen am besten als einzelner Absatz. Sie mit ASCII–Art zu unterstreichen, ist nicht nötig.

Textverarbeitungs–Dokumente (.doc, .odt, .sxw)

In Dokumenten aus Textverarbeitungen (Word, Star–Office, OpenOffice) sollten insbesondere die oben beschriebenen Allgemeinen Hinweise beachtet werden. Star–Office– und OpenOffice–Dokumente können mit Hilfe von writer2LaTeX (`http://www.hj-gym.dk/~hj/writer2latex/`) nach LaTeX konvertiert werden. Dies kann auch für Word–Dokumente geschehen, wenn diese durch OpenOffice in dessen Format konvertiert werden.

Bitte keine Tabulatoren zum Formatieren der Texte verwenden. Die Formatierung in der Vierten Dimension ist typischerweise unterschiedlich und dann sind Tabulatoren hinderlich.

Tabellen sollten unabhängig von der Formatierung immer zusammenstehen. Das Beste ist, wenn Sie nicht explizit selbst umgebrochen sind: Mehrzeilige Tabellen–Einträge bitte in *einer* Zelle ablegen und nicht selbst–umgebrochen in mehreren untereinander stehenden Zellen.

Andere Formate

Andere Formate, etwa XML–basierte wie DocBook, TEI, oder ähnliche, werden gerne akzeptiert. Bitte weitere Details mit der Redaktion der Vierten Dimension unter `vd@forth-ev.de` abstimmen.

Forth–Gruppen regional

Moers

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

Mannheim

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

München

Bernd Paysan
Tel.: (0 89) – 79 85 57
`bernd.paysan@gmx.de`
Treffen: jeden 4. Mittwoch im Monat im Indischen Restaurant „Lakschmi“ in Oberschleißheim (gleich an der S–Bahn):
Am Stutenanger 4
85764 Oberschleißheim

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** 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 Forthgesellschaft und der VD finden Sie im Impressum des Heftes.

Aufruf zur Forthtagung 2006

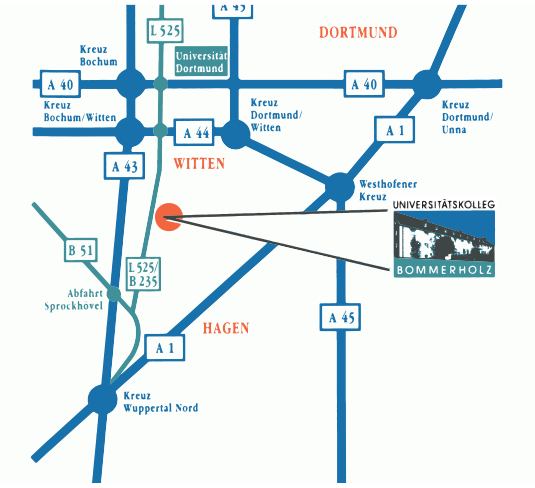
Ankündigung und Call for Papers

Die nächste Jahrestagung der Forthgesellschaft wird in der Nähe von Witten stattfinden. Witten, am Südrand des Ruhrgebietes zwischen Bochum und Dortmund gelegen, bewirbt sich wie die ganze Region um den Titel der Kulturhauptstadt 2010. Menschen einander näher zu bringen durch Kultur, mit diesem Ziel präsentieren sich jährlich unterschiedliche europäische Städte als internationale Kulturzentren und völkerverbindende Begegnungsorte. Im Jahr 2010 will das Ruhrgebiet, mit seinen 53 Städten und Gemeinden, zum kulturellen Austausch einladen. Ein kultureller Ballungsraum, der in seiner Dichte einzigartig in Deutschland und Europa ist. Aus dem einstigen Industrierevier ist inzwischen eine vielseitige Region entstanden. Wir als Forthgesellschaft sind schon vielseitig von Anfang an. Die Art dieser Programmiersprache, ihre Gestaltbarkeit, bringt das mit sich.

Forthtagung in Witten
Fr. 12.–So. 14. Mai 2006

Bommerholzer Str. 60
58456 Witten–Bommerholz

Bei der Suche nach einem Tagungsort hier in der Gegend fiel die Wahl auf das Kolleg der Universität Dortmund, weil es so gut ausgestattet ist und mit all den Annehmlichkeiten aufwarten kann, die wir auf unseren Forthtagungen inzwischen erwarten. Gepflegte Unterbringung, gute Küche und Gastlichkeit in einem Haus für uns, ungestört, auch bis tief in die Nacht nicht ohne Getränke. Im voll ausgestatteten Tagungsraum, mit WLAN und DSL ins Internet, kann alles, was die Teilnehmer erfahrungsgemäß so mitbringen, betrieben werden. Beamer, Tafeln, Ausstellung, kleinere Räume für Gruppen, eben alles da. Dabei liegt das Haus am Wald, hat eine Umgebung, die zum Spazieren einlädt, es gibt Museen, Shopping, Wellness, Theater. Aber auch moderne Fabriken wie Opel oder das Edelstahlwerk sind hier ansässig. Nur die rauchenden Schlote der Zechen sind verschwunden; Hochöfen, Kokereien, Krupp sind nicht mehr hier. Die Region ist seitdem im Umbruch und ist es noch.



Die Forthtagung macht also nun auf ihrer Wanderschaft durch Deutschland wieder Station tiefer im Westen. Auch Forth hat hier Spuren hinterlassen in etlichen Köpfen und Projekten. Wir wollen auf die Spurensuche gehen und auch diesen Teil der Forth–Geschichte beleuchten, Forth — vom Kult zur Kultur. Alle, die dazu beitragen möchten, sind herzlich dazu eingeladen. Dann aber soll es weiter gehen in die Bereiche Datenschutz und Privacy und ich hoffe, es finden sich dazu Beiträge auch aus der Forth–Perspektive. Dieses Thema hat einzelne in der FG in diesem Jahr sehr beschäftigt und beschäftigt uns ja alle außerordentlich im Lande. Und natürlich wird die Kunst, Forth zu implementieren, gepflegt, auf neuen Wirten und als eigens gefertigte Chips. Ich hoffe, es gibt eine Ausstellung dazu. Sodann gebührt den Applikationen in Forth unsere Aufmerksamkeit und — last, but not least — die Frage, wie wir als Gesellschaft Forth hier bekannt machen und bereit stellen können, und wie Forth derzeit international aufgenommen wird. Weitere Vorschläge sind willkommen.

So rufe ich Euch auf, Beiträge nun einzureichen und sich frühzeitig anzumelden.

Euer M.Kalus.

Das Anmeldeformular findet Ihr unter `http://www.forth-ev.de/`

Michael Kalus
+49 234 7731226 oder 0177 5635235 (AB)
e–Mail: `michael.kalus@onlinehome.de`