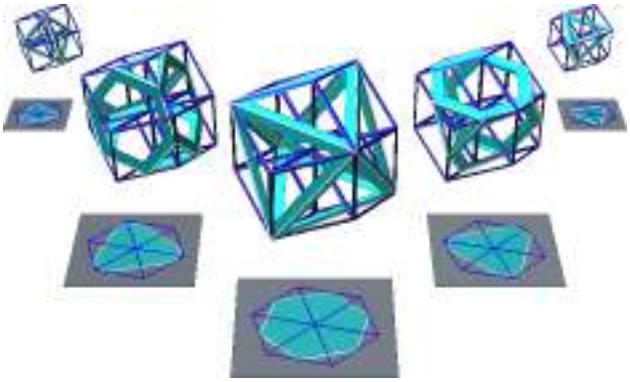
VIERTE DIMENSION

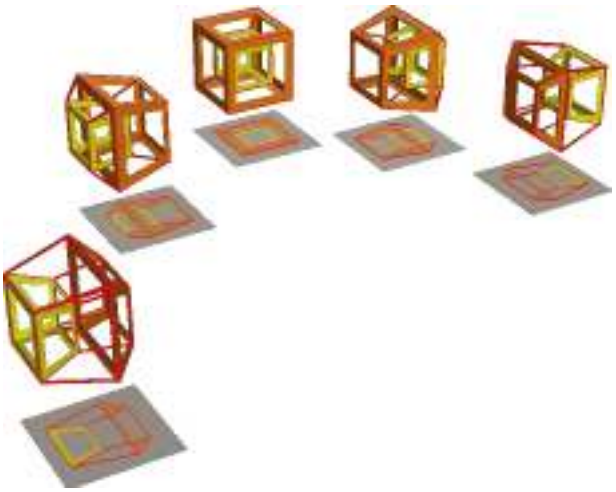
1/2006

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.

Inhaltsverzeichnis

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,

Diese Ausgabe der Vierten Dimension wird von einer Interims–Redaktion, bestehend aus den Direktoren Ulli Hoffmann und Bernd Paysan mit tatkräftiger Unterstützung von Fred Behringer (Übersetzungen, Lektorat) zusammengestellt. Eine dauerhafte Lösung kann das aber nicht sein, und wir hoffen natürlich, wieder einen Redakteur oder ein Team zu finden, das diese Aufgabe wahrnimmt.

Um den Zeitverbrauch beim Gestalten einer VD zu reduzieren, setzen wir ein Satzsystem (LaTeX) ein; die Arbeit des zukünftigen Teams soll sich damit mehr auf Inhalte und Mobilisierung derselben als auf den eigentlichen Satz der Zeitung mit einem DTP–Programm konzentrieren.

Die zur Produktion nötigen Dateien liegen als Subversions–Repository auf dem Web–Server, und sind somit einer kollaborativen Bearbeitung (zumindest in der Theorie) zugänglich. Es muss sich also nicht ein Editor allein die Nächte um die Ohren schlagen, um die VD zu Papier zu bekommen.

Aber all das löst ein Problem nicht: Die VD muss auch mit Inhalten gefüllt werden. Wenn die nicht kommen, kann auch ein zukünftiges Editoren–Team, das sich die Arbeit teilt, keine VD produzieren.

Noch eine Personalie: Fred Behringer, derzeit dienstältester Direktor, scheidet zur Jahres–Versammlung als Direktor aus, steht also für eine Wiederwahl nicht zur Verfügung. Gesucht wird also nicht nur ein neuer Redakteur, sondern auch ein neuer Direktor. Wir danken Fred an dieser Stelle für sein Engagement als Direktor der Forth–Gesellschaft, und drücken natürlich die Hoffnung aus, dass er sich auch ohne Amt weiter engagiert.

Um die Vereins–Website mehr mit der VD zu verknüpfen, hat Carsten Strotmann auf Seite 7 einen Vorschlag gemacht, den wir zumindest teilweise umzusetzen versuchen: Abstimmen über die Artikel dieser VD wird man da mit dem Erscheinen der VD können.

Bernd Paysan

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

Wie wir diese Vierte Dimension produziert haben

Ulrich Hoffmann

Ich weiß nicht, ob Sie's schon wussten? Aber diese Vierte Dimension ist auf eine neue Art und Weise entstanden. Das vorliegende Heft ist Ergebnis einer Mannschaftsleistung und ist innerhalb weniger Tage um die Jahreswende entstanden. Wir haben dazu eine Reihe von technischen Hilfsmitteln eingesetzt, die ich im Folgenden kurz ansprechen möchte.

Einleitung

Diese Vierte Dimension ist eine Ausgabe, für die das Direktorium kommissarisch die Rolle des Editors eingenommen hat. Damit überbrücken wir den Zeitraum bis zur Jahrestagung, auf der wir über die Zukunft der Vierten Dimension diskutieren wollen. Es wäre schön, wenn wir dann die Redaktion in neue Hände geben könnten.

In ihrer langen Geschichte hat die Vierte Dimension schon viele Editoren (ihnen allen sei hier nocheinmal ein ausdrücklicher Dank ausgesprochen) und noch mehr Publishing–Systeme erlebt (von Schneid– und Kleb, über Ventura–Publisher, PressWorks, Pagemaker, Star–Office und anderen zu MS–Publisher). Alle diese Systeme haben gemeinsam, dass der Schwerpunkt der Arbeit auf das Layout der Seiten gelegt wird. Als Programmierer haben wir natürlich mehr Spaß daran, uns auf Inhalte zu konzentrieren als stundenlang Layoutanpassungen machen zu müssen.

Bernd Paysan und ich haben neben Forth auch schon seit langem mit dem Textsatzsystem LaTeX gearbeitet. Bernd hat mit der Neuauflage von *Thinking Forth* (`http://thinking-forth.sourceforge.net/`) schon gezeigt, wie man mit einer Mannschaft eindrucksvoll ein Buch produzieren kann. Nachdem klar war, dass Friederich Prinz als letzte Amtshandlung die Vierte Dimension 3/4–2005 als Doppelheft herausbringen würde, schlug Bernd also vor, zwischen den Jahren das Heft 1–2006 nach neuer Technik im Direktorium gemeinsam zu produzieren.

Gemeinsames Arbeiten

Der Schlüssel dieser VD–Produktion ist das Zusammenarbeiten in einer Mannschaft, frei nach dem Motto *Einer für alle, alle für eine VD* haben Fred Behringer und Thomas Beierlein Briefe und Artikel übersetzt und zusammengefasst. Bernd hat den LaTeX–Stil für die Vierte Dimension entworfen und die technische Infrastruktur aufgesetzt. Bernd und ich haben das Rohmaterial der Autoren entgegengenommen und die notwendigen LaTeX–Auszeichnungen ergänzt.

Server der Forth–Gesellschaft

Jede Gruppe, die zusammenarbeiten will, muss sich austauschen können. Für Abstimmung der Aufgaben und das Anliefern von Roh–Artikeln haben wir Email verwendet. Für das entstehende Heft selbst bietet es sich aber an, alle Informationen an einer Stelle zu sammeln, damit von allen *online* darauf zugegriffen werden kann.

Zum Glück betreibt die Forth–Gesellschaft seit Frühjahr 2005 einen eigenen Server. Auch beim Gestalten unserer Homepage `www.forth-ev.de` haben wir gute Erfahrunge gemacht, die Arbeit auf mehrere Schultern zu verteilen. Auf `www.forth-ev.de` kann jeder Interessierte Neuigkeiten über Forth erfahren *und* bereitstellen. Ich würde mir wünschen, dass das von noch mehr Forthern genutzt werden würde.

Aber — der Server bietet uns eben neben der Homepage zusätzlich auch die Möglichkeit, weitere Dienste zu nutzen, die man in einem einfachen Web–Angebot nicht findet. Bernd hat hier eine zentrale Ablage für Forth–Projekte auf Basis des Versionskontroll–Systems Subversion (`http://subversion.tigris.org/`) eingerichtet.

Versionsverwaltung mit Subversion

Das Versionsverwaltungs–System Subversion erlaubt es ganz generell, Sammlungen von Dateien und Verzeichnissen zu verwalten. Das Besondere ist, dass alle Versionen der Dateien (und Verzeichnisse) aufbewahrt werden und die Subversion–Benutzer so in der Lage sind, quasi rückwärts in der Zeit zu reisen und auf alte Änderungsstände zuzugreifen. Subversion ist ein Client–Server–System, das über das Internet arbeiten kann. Der Subversion–Server (auf www.forth-ev.de) verwaltet die Daten in einem sogenannten *Repository*.

Subversion–Clients gibt es unter anderem für Linux, MacOS, Windows. Auf diese Weise können Subversion–Benutzer, die mit ganz unterschiedlichen Computern arbeiten, auf die gleichen Daten im Repository zugreifen.

Programmierer setzen Subversion normalerweise ein, um Quellcode zu verwalten. Das tun wir auch. Aber wir können dort auch bestens die entstehende Vierten Dimension verwalten, wenn wir sie nur als Programm begreifen. Wir sehen ein Heft als Forth–Programm an, das in einzelne Module eingeteilt ist. Hier sind das etwa Artikel, Rubriken, das Impressum oder die feststehenden Umschlagsseiten, die alle getrennt voneinander bearbeitbar sind.

Statt also die Daten des Heftes als eine monolithische, proprietäre Datei eines Publishers abzulegen, halten wir sie modular in einem Internet–fähigem Versionsverwaltungs–System in einem offenen, gut dokumentierten Klartextformat.

Aber mit welchem Klartextformat lassen sich Druckprodukte angemessen beschreiben?

Automatisches Setzen mit LaTeX

Das von Donald Knuth entwickelte Textsatzsystem TeX und seine auf Leslie Lamport zurückgehende Erweiterung LaTeX (siehe beispielsweise `www.dante.de` oder `www.tug.org`) erzeugen hochwertig gesetzte Dokumente aus Text–Beschreibungen.

Ähnlich wie in HTML werden dabei die Textstücke ausgezeichnet, um ihre Bedeutung innerhalb eines Textes zu kennzeichnen. Damit gebe ich über den Inhalt hinaus Informationen über die gewünschte Formatierung des Textes und seine Struktur an, die beim Setzen berücksichtigt wird.

Wenn ich etwa `\emph{hervorgehoben}` in der Beschreibung notiere, dann wird der entsprechende Teiltext *hervorgehoben* dargestellt. Wie aber diese Hervorhebung tatsächlich im Druck erscheint, ob *kursiv* oder **fett**, legt erst der umfassende Stil fest. So erhält man ein einheitliches Schriftbild.

Wenn ich erreichen möchte, dass mein Text zweispaltig gesetzt wird, so schließe ich ihn in `\begin{multicols}{2}` und `\end{multicols}` ein. Die passende Formatierung inklusive Worttrennung übernimmt TeX. Auch sorgt es dafür, dass selbstständig ein Inhaltsverzeichnis entsteht. Ändert sich der Titel eines Artikels, wird er im Inhaltsverzeichnis ebenfalls geändert erscheinen.

TeX ist *kein* what–you–see–is–what–you–get–System[1]. TeX–Texte werden typischerweise mit einem ASCII–Editor geschrieben und dann mit dem TeX–Prozessor und weiteren Transformatoren in das gewünschte Ausgabeformat (etwa letztlich Postscript oder PDF) gebracht.

LaTeX ist ein Makro–Paket, das die in TeX eingebaute Makrosprache verwendet. Es definiert wiederkehrende Auszeichnungen für Standardtexte — beispielweise Briefe, Artikel, Berichte, Bücher oder eben Zeitschriften. Diese Auszeichnungen machen es leichter, solche Standardtexte zu setzen. Für LaTeX wiederum gibt es eine große Anzahl von Erweiterungen und Verfeinerungen, so genannte Stile, in denen etwa die oben angesprochenen Makros für Hervorheben und spaltenweises Setzen schon vordefiniert sind.

TeX oder LaTeX zu lernen, ist keine einfache Sache, aber es lohnt sich. Mit TeX werden umfangreichste Bücher gesetzt und wissenschaftliche Periodika produziert. Ich selbst würde für ein textorientiertes Dokument, das mehr als ein paar Seiten besitzt, nichts anderes einsetzen. Die Literatur zu LaTeX ist umfangreich und frei verfügbare Systeme gibt es für alle gängigen Plattformen.

Für die Vierte Dimension hat Bernd, ausgehend vom dtk–Stil, einen eigenen vd.cls–Stil definiert. Die wiederkehrenden Teile der Vierten–Dimension werden einmal entsprechend ausgezeichnet. Ist ersteinmal so ein Stil definiert, dann wird das Setzen von Texten fast zum Kinderspiel.

Artikel können problemlos in LaTeX angeliefert werden. Für einfache Texte benötigt ein Autor oft nur wenige Auszeichnungsarten, um seine Artikel zur formulieren. Diese sind schnell gelernt.

ASCII–Texte (Emails, Leserbriefe) müssen ausgezeichnet werden. Dies ist nur dann ein wenig aufwändiger, wenn der Text Anführungszeichen enthält, die in LaTeX eine Sonderrolle spielen. Typographisch gibt es viele Arten von Anführungszeichen.

Andere Texte, die etwa mit Word oder OpenOffice[2] verfasst wurden, müssen auch entsprechend konvertiert werden. Das Einfachste ist, sie als Text abzulegen und dann auszuzeichnen.

Alles in allem entlastet TeX beim Layout und erlaubt es, dass wir das Schwergewicht mehr auf den Inhalt legen können.

Fazit

Wir haben mit der angerissenen Technik vergleichsweise schnell die aktuelle Ausgabe produzieren können. Jeder wählt die Werkzeuge, die ihm am besten liegen; das ist der Grund, warum wir alle mit Forth arbeiten.

> Aber das Direktorium will nicht dauerhaft die Rolle des Editors übernehmen: Die Forth–Gesellschaft sucht einen neuen Editor für die Vierte Dimension.

Der zukünftige Editor ist natürlich frei in der Wahl seiner Werkzeuge. Welche Technik aber auch immer eingesetzt wird: Unsere Vierte Dimension lebt von den Artikeln ihrer Autoren und von denen kann auch der zukünftige Editor jede Menge gebrauchen.

Schreibt Artikel, wir haben alle was davon!

[1] Es gibt aber grafische Frontends wie LyX, deren Output selbstverständlich eingebunden werden kann — *Bernd*

[2] Auch mit writer2LaTeX konvertierbar — *Bernd*

Leserbriefe

Wettbewerb: I–EMIT

Im Heft 3/2004 der Vierten Dimension hatte ich schon mal auf GForth hingewiesen, das auch unter Mac OS X läuft — bei `http://fink.sourceforge.net/download/` wird man fündig. Dieses Forth läuft einfach gut. Allerdings an Zeichen gebunden im Terminalfenster.

Was fehlt, ist eine Möglichkeit, von Forth aus ein Bild auf den Bildschirm an beliebige Stelle zu platzieren. Also quasi so etwas wie ein

```
: i-emit ( i -- ) \ image-handler is given
    \ emit image to cursor position
    cursor@ display! ;
```

Das Bildchen soll schon fertig in einer Datei vorliegen, auf die der image–handler dann verweist. So wäre es möglich, Bildchen übereinander zu legen, eine Collage zu machen. Das ließe sich weiter spinnen. Nun, wer wagt sich da heran und veröffentlicht mal, wie so was geht? In Forth unter Windows? Meinetwegen. In GForth unter Linux? Besser. In Gforth unter Mac OS X? Mein persönlicher Favorit natürlich! Retro–Forth Freunde sind natürlich auch eingeladen, auf einem Atari kommt das bestimmt auch gut. Alle Interessierten seien dabei auch auf die Möglichkeit hingewiesen, ihren Beitrag auf unsere website zu geben: `http://www.forth-ev.de/` oder für die ganz Schnellen: Zeigt es beim Forthtreffen, 12. - 14. Mai 2006 in Witten.

Viel Erfolg! Als Prämie hat sich das Direktorium einen attriaktiven Sachpreis ausgedacht.

Euer Michael Kalus

Abstimmung über VD–Artikel

Von: Carsten Strotmann <`carsten@strotmann.de`>

Was haltet ihr von der Idee mit einer Leserabstimmung über Geeklog?

Den Club, welchen ich dort angesprochen habe, ist der ABBUC (Atari Bit Byter User Club), die Web–Umfragen seht ihr hier ⟶
`http://www.atari-portal.net/modules.php?name=Surveys`

Mein Diskussionsvorschlag:

- die Redaktion der VD ist identisch mit der Redaktion der Webseite
- Artikel, die von Autoren eingereicht werden, erscheinen sofort in einem nur Mitglieder zugänglichen Bereich auf der Webseite
- diese Artikel können über Geeklog diskutiert werden
- alle 3–4 Monate (oder wenn genügend Artikel vorhanden sind) werden die neuen Artikel per DTP gesetzt und als gedruckte „VD“ an die Mitglieder versendet
- gleichzeitig wird eine PDF– oder HTML–Version öffentlich zugänglich gemacht (download)

Vorteil:

- höhere Aktualität für Mitglieder (Artikel erscheinen erst Online)
- gedruckte VD beinhaltet schon Kommentare und Feedback zu den Artikeln (aus dem Web)
- die gedruckte VD erreicht weiterhin die Mitglieder, welche nicht tagtäglich „online“ sind
- die gedruckte VD ist auch ideal, um Artikel in Ruhe auf Papier zu lesen (ich wollte darauf nur ungerne verzichten)

Ciao

Carsten

Lokale Gruppe Mainz?

Von: Rolf Lauer <rowila@t-online.de>

Nach der letzten Mitgliederliste aus der VD–2005/2 sind einige aus dem Bereich Frankfurt, Mainz, Bad Kreuznach. Gibt es aus dieser Gegend Leute, die mitlesen und vielleicht Lust hätten eine lokale Gruppe zu gründen?

Gruss

Rolf

R8C im Elektuur

Tag Fred,

auch von Almere ein vor allen Dingen gesundes 2006!

Meine Antwort hat noch etwas auf sich warten lassen, da ich momentan sehr beschäftigt bin (nicht etwa Urlaub!).

Doch ganz schnell Zeit genommen, um deine Übersetzung zu lesen. Die ist prima, ich habe nichts hinzuzufügen oder anzumerken. Und außerdem ist es höchst interessant, den eigenen Text in einer anderen Sprache nachzulesen. Den Text kannst du ohne Weiteres an Ulrich schicken.

Und noch eine Reaktion auf deine Mail über den R8C–Prozessor: Der wurde in der niederländischen Elektuur in der Januar–Ausgabe „vorgestellt“. Ein kurzer Artikel, mit der Ankündigung, dass sie diesem Prozessor in weiteren Artikeln viel Aufmerksamkeit schenken werden und

dass sie eine Möglichkeit anbieten werden, den Chip über die Firma Glyn preisgünstig zu beziehen. Vielleicht wird das Platinchen auch hier „verschenkt“. Die Idee, darauf Forth laufen zu lassen, ist wieder eine neue Herausforderung. Was mich betrifft: Ich bin noch zu sehr mit dem AVR beschäftigt, um dem R8C zu viel Aufmerksamkeit zu widmen. Auf den ersten Blick auf die Website des Herstellers sieht der Befehlssatz ein bisschen so wie der eines 68000 aus. Jedenfalls gibt es darüber genügend Dokumentationen.

Bis weiterhin, bis zu den nächsten Mails,

Hartelijke groeten, Ron

Forth für Windows CE

Von: Patrick Mauritz <oxygene@studentenbude.ath.cx>
Datum: 2005–11–07 13:36

Hallo,

nachdem jetzt schon die zweite Frage zu PocketPC Forth in der VD stand, und ich demnächst ein funktionierendes ARM system haben werde, wollte ich doch mal darauf antworten.

Windows CE ist seit ca. 2002 auf ARM CPUs normiert, volksForth wird daher vermutlich keine große Hilfe sein, da es für z80 und ähnliche (zB 8088/8086) entwickelt wurde. Auch wenn es portabel ist, ARM ist strikt 32bit, vF ist eher für 8bit cpus.

Ich werde voraussichtlich Ende des Monats einen GP2X (www.gp2x.com) erhalten, der ebenfalls ARM basiert ist (sogar 2 cores), und habe vor, dort Forth ans Laufen zu bekommen.

Mir ist kein reines ARM Forth bekannt, abgesehen von einem, dass auf Chuck Moores machine forth basiert. Der Source scheint allerdings verlorengegangen zu sein, und das beste, was ich noch habe finden können ist das Paper, in dem der Autor die Unterschiede zu und die Probleme mit MF beschreibt.

(http://dec.bournemouth.ac.uk/forth/euro/ef99/thomas99a.pdf)

Daher werde ich als Basis wohl BeginAgain hernehmen, das Forthsystem, welches ich urspruenglich für OpenBIOS entworfen habe mit dem Ziel, auf allen 32bit+ Systemen einsetzbar zu sein. Auf ARM wird BA allerdings nativ eher langsam laufen, da die Fähigkeiten und Schwächen der CPU von kaum einem C compiler berueckschtigt werden, und auch das VM design da wenig Spielraum lässt für „gute“ compiler. Vermutlich wird jedes andere portable Forth ähnliche Schwächen aufweisen. (gforth, pforth und MPE VFX haben auch „arm support“)

Das Hauptproblem bei WinCE wird sein, die Anbindung an die Systemlibraries hinzubekommen.. Linuxbasierte (wie der gp2x) oder Systeme ohne Betriebssystem haben es da definitiv leichter. Ansonsten bleibt da nur noch zu testen, inwiefern ein portables Forth schnell genug ist oder nicht, und was man dagegen machen kann/will..

Gruß

Patrick Mauritz

La Casita

Liebe Forth–Gesellschaft e. V.

Mit Computern habe ich erst sein kurzem etwas zu tun, mit Programmiersprachen eigentlich gar nichts. Warum also schreibe ausgerechnet ich an die Forth–Gesellschaft e. V.?

Der Grund dafür scheint schon fast erfunden, aber ich versichere, dass er nicht meiner Fantasie entspringt. Während dem Aufbau eines Ausbildungszentrums in Mar del Plata, Argentinien, war ich oft mit einem Freund namens Pablo Reda in Kontakt, der sich vom Inhalt seines Informatikstudiums abgewendet und Forth zugewendet hatte. Ich brauchte eine Weile, bis ich verstand, worum es sich dabei handelte, jedoch da wir in unserem Ausbildungszentrum LA CASITA sehr interessiert an ungewöhnlichen Ansätzen sind, wurde Pablos Idee mit Freude aufgenommen. Er hatte vor, mit Forth und zusammen mit den Kindern und Jugendlichen Lernspiele und –programme zu entwickeln. Bei seinen Recherchearbeiten stiess er irgendwann auf www.forth-ev.de und da er kein Wort [*Deutsch, die Red.*] verstand, bat er mich, ihm den Inhalt dieser Seite zu erklären.

Dies ist also mein etwas ungewöhnlicher Zugang zur Forth–Gesellschaft e. V. Über Argentinien bin ich also nach Deutschland gelangt, aber eigentlich bin ich von Liechtenstein und studiere in der Schweiz. Durch meine Fachhochschule (www.hyperwerk.ch) und die Projektarbeit in Liechtenstein wurde ich ständig damit konfrontiert, wie sehr die Technologien immer mehr ins Alltagsleben eingreifen. Während sich schon kleine Kinder schnell mit diesen Veränderungen zurechtfinden, sind Lehrpersonen und Eltern oft total überfordert. Auf der einen Seite begann ich mich also zu fragen, welche Antworten es auf diese Veränderungen der gesellschaftlichen Strukturen gibt und auf der anderen Seite war ich mit ganz anderen Realitäten in Argentinien und Bolivien konfrontiert, wo zumindest die Kinder und Jugendlichen, mit denen ich arbeite, kaum genug zu Essen geschweige denn einen Computer zu Hause haben. Sie geben jedoch jeden Centavo, den sie irgendwoher kriegen können, aus, um ein paar Minuten in einem Internetcafé zu verbringen. Das Interesse und die Lernbereitschaft in diesem Bereich sind also trotz grösster Armut vorhanden und die Chancen im Berufsleben könnten sich durch Kenntnisse im Umgang mit Computern multiplizieren.

Das System der CASITA ist, dass dort jede(r) unterrichten kann, was ihm/ihr am meisten liegt und dies trifft auch auf die vorhandenen Computer zu. Pablo ist einer der vielen jungen Menschen, welche die Möglichkeit haben werden, umzusetzen, was sie beschäftigt und Pablo

will ausprobieren, ob die Jugendlichen mit Forth arbeiten wollen und ob dies vielleicht sogar eine andere Lern– bzw. Denkstruktur in den Jugendlichen anregen könnte.

Im Juni wird Alejandra García Morillo, die Koordinatorin der CASITA aus Mar del Plata anreisen und im Rahmen eines Kongresses über ihre Erfahrungen im Ausbildungszentrum berichten. Ich würde mich freuen, wenn uns ein Mitglied der Forth–Gesellschaft e. V. ebenfalls mit seiner Anwesenheit beehren und über die Erfahrungen in der pädagogischen Arbeit mit Forth berichten könnte.

Herzliche Grüsse aus dem Ländle

Laura

Noch einmal Argentinien

Ist das ein Zufall? Da schreibt einerseits Patrick Mauritz über seine Pläne und Wünsche zu einem Forth für Arm–Prozessoren und WindowsCE, das er selbst entwickeln will, ... und andererseits nimmt Laura Hilti Kontakt zur Forth–Gesellschaft auf, weil ein Programmierer Jugendlichen aus dem sozialen Elend helfen will: mit einem selbst entwickelten Forth, das auf PocketPCs laufen soll! Zweimal PocketPC und zweimal Forth — in Gegenden, die doch recht weit voneinander entfernt sind.

Ein wenig zu LA CASITA und Pablo Reda: La Casita heißt 'Häuschen'. Und darum geht es auch. Mithilfe von Spendengeldern und in Eigenarbeit haben begeisterte und kompetente Menschen ein kleines Haus umgebaut und wieder bewohnbar gemacht. Es liegt nahe des Stadtzentrums und soll für benachteiligte Menschen eine Hilfe sein, zu besseren Lebensumständen zu kommen. Helfer aus der ganzen Welt können dort in relativer Sicherheit wohnen, während sie in den Räumen der Casita an sozialen Projekten arbeiten.

Einer der einheimischen Helfer ist Pablo Reda. Er ist ausgebildeter Programmierer und arbeitet ehrenamtlich seit einigen Jahren mit Jugendlichen aus den benachbarten Elendsvierteln. Die Jugendlichen kennen das Internet und nutzen es (Internetcafés). Normalerweise leben sie von den Erträgen, die das Durchmustern einer Müllhalde oder das Sammeln vom Pappkartons in der ganzen Stadt ihnen bringt.

Bisher hat Pablo Reda versucht, ihnen einen sinnvollen Umgang mit „`Computacion`“ zu vermitteln.

Er möchte, dass diese Jugendlichen ihr Selbstbewusstsein erhöhen, indem sie diese moderne Technik nicht nur konsumieren, sondern auch (etwas) beherrschen. Dazu hat er bisher BASIC und Logo eingesetzt und gezeigt, wie man damit Musik, Videos, Bilder und Texte verfasst und bearbeitet. Computer wurden zerlegt und wieder zusammengebaut.

Doch das war nicht alles. Es musste mehr sein! Manche Kinder und Jugendliche (9–16 Jahre) zeigen eine hohe Auffassungsgabe und Kreativität. Trotzdem war (ist) es sehr schwer, ihnen Programmiersprachen beizubringen.

Pablo Reda sah sich um und stieß auf Forth! (Wir kennen die Vorteile von Forth und können gut verstehen, wie Pablo zu seinem Entschluss kam.)

Im Moment entwickelt er, ausgehend von ColorForth (!) reda4, ein Forth für (jetzt kommt's:) PocketPCs. Es soll einfach werden und es soll eine gewisse künstlerische Arbeit ermöglichen (Kreativität). Geplant ist, dass die Jugendlichen einfache Spieleklassiker (`Ping Pong`, `Space Invaders`, `Moon Patrol`, `Pacman`) und eigene Spiele entwickeln und dabei eine Menge über „`Computacion`“ lernen.

Eine erste Version ist lauffähig. Pablo hofft, dass er bis Februar eine Portierung für 'normale' PCs zur Verfügung hat!

(to be continued) Martin Bitter

Forth-Programmierer gesucht

Von: Dr. Heinrich Möller <`hmoeller@data-al.de`>

Unsere Firma entwickelt ein Praxisverwaltungsprogramm für Ärzte, das in über 1000 Praxen im Einsatz ist. Das System ist zum Teil in Win32For geschrieben. Wir suchen zur Festanstellung (unbefristet) einen Software–Entwickler, der nicht nur FORTH beherrscht, sondern auch zumindest einige der folgenden Voraussetzungen erfüllt:

Kenntnisse:

- Betriebssysteme: Windows, Linux(optional)
- Objektorientierte Programmierung
- Programmiersprachen C++; optional Java, Smalltalk, Skriptsprachen, z.B. Python, Ruby
- Kenntnisse der Windows API
- COM/Automation
- IDE: VisualStudio, Eclipse (optional)
- (X)HTML, CSS, JavaScript
- XML, XSLT, XML Schema
- Datenbanken, z.B. PostgreSQL, MySQL
- MS–Office, VBA–Programmierung

Sonstige Anforderungen:

- Methodische Arbeitsweise

hohe Einsatzbereitschaft
Lernfähigkeit
Kreativität
gute Englischkenntnisse

Der Firmenstandort ist 89231 Neu–Ulm
Weitere Informationen unter `www.data-al.de`
Kontakt: `hmoeller@data-al.de`

Die Russische–Bauern–Multiplikation

Zu „Die Russische–Bauern–Multiplikation“ von Fred Behringer (VD 3/4–2005, S.7–11)

Hallo Fred,

In deinem Artikel über das russische Malnehmen schreibst du: „Es ist mir leider nicht gelungen, in High–Level–Forth einen doppeltgenauen Überlaufschutz bei der Addition zweier vierfachgenauen Zahlen zu verwirklichen. Also konnte ich `Q+` nicht aus schon bestehenden `D+`–Anteilen zusammensetzen.“

Zugegeben, es ist nicht von übergroßer Bedeutung, aber du kannst ein Carry allein durch Zusammenzählen erzeugen, ohne dabei Maschinencode zu verwenden.

```
: +CARRY  ( x y -- z carry )     0 TUCK D+ ;

: D+CARRY ( dx dy -- dz carry )
  ROT 2>R           \ xlo ylo     \r: yhi xhi
  +CARRY 0
  2R> +CARRY
  D+ ;
```

Hast du erst einmal `D+CARRY`, dann hast du auch `Q+`:

```
: Q+ ( qx qy -- qz )
  2>R 2SWAP 2>R D+CARRY 0 2R> D+ 2R> D+ ;
```

Könntest du bitte selbst nachprüfen, ob sich kein Fehler eingeschlichen hat? Ich habe nicht allzu lange darüber nachgedacht, da ich diese Zeilen gern noch in dem vom FG–Direktorium gerade bearbeiteten VD–Heft 1/2006 veröffentlicht sehen wollte.

De hartelijke groeten,

Albert

Antwort von Fred Behringer

Hallo Albert,

vielen Dank für deine Bemerkungen. Es geht tatsächlich (alles in High–Level–Forth). Nebenbei gesagt, du kennst meine Einstellung: Ich verwende gern Forth–Assembler als (mir) leicht zugängigen Allerweltsassembler und High–Level–Forth als leicht programmierbaren Organisator, der die eigentlich interessanten Assembler–Teile zusammenhält. Dabei befinde ich mich im Gleichklang mit beispielsweise Julian Noble `<jvn@virginia.edu>`: „Ein Assembl(i)eraufruf“, VD 2/2002, s.19–24. Julian schreibt:

„Was tun aber die Mitglieder der Forth–Gemeinde in allererster Linie (wenn sie nicht gerade in Forth programmieren)? Sie versuchen, die anderen Programmierer, die immer noch mit Sprachen geringerer Qualität herumwursteln, zu Forth umzuerziehen. Schon seit langem bin ich zu der Überzeugung gelangt, dass wir bei unseren Bekehrungsversuchen eine ganze Menge verschenken, indem wir viel zu stark die Vorzüge des High–Level–Teils von Forth anpreisen, seine Erweiterbarkeit, die Abstraktionskraft, die Einfachheit, die Eleganz und so weiter und so fort. Ich denke heute, es wäre wohl besser, Forth dem ungläubig Außenstehenden über den Assembler näherzubringen.“

Gut, dass du mir den Anstoß gegeben hast, alles noch einmal zu überprüfen: Bei `2AND` habe ich zu schnell überlegt. Man könnte bei `2AND` als Ergebnis einen doppeltgenauen Wert vermuten. Er ist bei mir aber nur einfachgenau — und sollte auch nur einfachgenau sein. Es wäre wohl besser gewesen, eine dem Umstand angemessene neue Bezeichnung einzuführen. Es ist im Artikel an dieser Stelle nichts falsch, aber man führe vielleicht doch besser folgende Ersetzungen ein:

```
\ : 2AND ROT AND -ROT AND OR ;
: D-UNGERADE? DROP 1 AND 0<> ;

\ B 2@ 1. 2AND IF C 4@ A 4@ Q+ C 4! THEN
  B 2@ D-UNGERADE? IF C 4@ A 4@ Q+ C 4! THEN
```

(Bei `D-UNGERADE?` steht `0<>` eigentlich nur der Schönheit wegen da.)

Und noch eine Warnung: Wenn man die Eingaben dezimal macht, könnte man bei höheren Werten meinen, die Ergebnisse seien falsch. Sind sie nicht! In Bezug auf die Aufteilung nach höherwertigem und geringerwertigem Anteil ist das Dezimalsystem mit dem Hexadezimalsystem nicht kompatibel: Man kann nicht einfach sagen, der höherwertige Dezimalanteil entspricht nach Umwandlung genau dem höherwertigen Hexadezimalanteil. Die *Schnittstellen* der Stellenwertsysteme müssen übereinstimmen. Bei Binär– und Hexadezimalsystem geht's.

Zur Umgehung dieser Schwierigkeit bei gleichzeitiger Überprüfung deines Forth-Wortes `Q+` mit allergrößten Dezimaleingaben kann man Folgendes machen:

```
65535 65535 65535 65535 UD*Q^^ [ret]
65535 65535     0     0 Q+     [ret]
65535 65535     0     0 Q+     [ret]
```

Ergebnis:

```
65535 65535 65535 65535
```

Das kann man *im Kopf* nachrechnen und sofort bestätigen.

Herzlichen Gruß

Fred

Gehaltvolles

zusammengestellt und übertragen von *Fred Behringer*

VIJGEBLAADJE der HCC Forth–gebruikersgroep, Niederlande Nr. 52, Oktober 2005

Cursorbesturing in FORTH — Gerard van der Sel

Vor einiger Zeit wurde in den Windows–Systemen das Programm Server32 eingeführt, vom dem kürzlich die Version 2.0 herauskam (siehe `http://www.forth.hccnet.nl`). Über dieses Programm kann die ATS–Platine der Forth–gebruikersgroep mit Windows–PCs kommunizieren. Damit besteht die Möglichkeit, Quelltexte von der Festplatte auf die ATS–Platine zu befördern. Hierbei bietet das im Server32 enthaltene Terminal dem Benutzer Rückkopplungsmöglichkeiten. An sich sind das Standardfunktionen, die der Server unter DOS auch schon hatte. Der Server32 hat jedoch noch eine weitere Möglichkeit: Cursor–Ansteuerung von der ATS–Platine aus. Dabei kommt der Terminal–Emulator VT52 ins Spiel. Dem Autor reichten die Ansteuerungsbefehle nicht. Er präsentiert ein paar in Forth geschriebene Zusätze.

Forth, meer dan een programmeertaal — De Schoolmeester

(voor gevorderden)

Een Forthadept in 't Zeeuwse Sas,
die zich in een komma verlas,
crashte finaal.
Het werd hem fataal.
De artsen bij de lijkschouwing zeiden:
de dood was niet te vermijden
daar de man zelf Forth geworden was.

Und hier ein Übertragungsversuch:

(für Fortgeschrittene)

Ein Forth–Adept aus dem hohen Norden,
ein Komma war ihm zum Verhängnis geworden,
der crashte total.
Für ihn war's fatal.
Die Ärzte sagten,
der Tod des Beklagten
war unvermeidbar.
Es sei nicht bestreitbar,
dass Forth er am Ende war selber geworden.

Forth vanaf de grond — een praktijksituatie — Ron Minke

Eine Studie aus der Praxis: Eine Tastatur soll an ein Intercom–System extern angeschlossen werden. Die Tastatur gehört zum ursprünglichen System und soll ein anderweitiges System seriell ansteuern. Die Tasten sind beleuchtet und eine kurze Beschreibung ist vorhanden. Zur Anwendung kommt das vom Autor in den vorangegangenen acht Artikelteilen besprochene AVR–Forth–von–der–Pike–auf–System ohne externes RAM. 24 Volt Speisespannung für die Lämpchen.

Wie schon angekündigt:

Ron Minke wird ab dem nächsten Mal die Redaktion des Vijgeblaadjes übernehmen. Wir von der Forth–Gesellschaft wünschen Ron gutes Gelingen bei diesem bestimmt nicht immer leichten Unterfangen.

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.

VIJGEBLAADJE der HCC Forth–gebruikersgroep, Niederlande Nr. 53, Dezember 2005

Byteforth voor beginners — voor gevorderden — Ernst Kouwe

„ByteForth für Anfänger — für Fortgeschrittene: Im Schulunterricht habe ich jeden Freitag Nachmittag „Roboter–Club," wo die Schüler sich unter anderem mit Ushi und dem AVR–ByteForth beschäftigen. Sie finden die Ansteuerung der Ports des AVR unbequem. Darum habe ich zwei neue Datenkonstruktionen gemacht: INPORT und OUTPORT. INPORT liest die Spannung an den Pins des Ports ab und OUTPORT legt dort eine bestimmte Spannung an. Einer Lese- oder Schreibaktion vorausgehend, wird das Port auf Lesen oder Schreiben gesetzt. OUTPORT kann auch ablesen, auf welchen Wert das Port zuletzt gesetzt wurde. OUTPORT kennt die Präfixe CLEAR SET INCR DECR PUSH POP TO +TO TOGGLE . Hier will ich nur TO besprechen. Neben INPORT und OUTPORT habe ich auch INPIN und OUTPIN definiert, die ich hier aber außer Betracht lasse."

Und weiter geht es mit: Die AVR–Portstruktur ... Initialisierung der Ports ... Präfix oder ! und @ — VALUE oder VARIABLE ... Das Definieren von Präfixen ... ein Beispiel.

Der Autor dankt Willem Ouwerkerk und Mark van der Wilk.

Verandering — Albert Nijhof

„Al jaren ben ik de redakteur van het Vijgeblaadje, maar dat gaat nu veranderen: Ron Minke heeft toegezegd om die taak van me over te nemen. ... (*Rezensent: siehe Übersetzung weiter unten*)

Tenslotte

> Een Forthprogrammeur te Harderwijk
> zette z'n buurman aardig in de zeik.
> Diens Dolfinariumprogramma, geschreven in C,
> dat het na vier maand nog steeds niet dee,
> herschreef hij binnen 't uur in Forth.
> 't Werkte meteen en 't was tien keer zo kort.
> De groeten en tot kijk.

(Uw Vijgredakteur van de afgelopen jaren)"

Und hier ein Übertragungsversuch:

„Seit Jahren bin ich Redakteur des Vijgeblaadjes, was sich aber jetzt ändern soll: Ron Minke hat zugesagt, diese Aufgabe von mir zu übernehmen. Wir haben verabredet, dass das jetzt das letzte Vijgeblaadje ist, bei dem ich noch mitwirke.

Die Initiative für diese Veränderung ging von mir aus. Solch eine Aufgabe sollte nicht zu lange bei ein und derselben Person liegen. Die Kontinuität kann mitunter gar zu groß werden und ein neues Herangehen wirkt oft Wunder, so wie man das weiter unten bei „Zum Schluss" lesen kann.

Aus genau demselben Grund habe ich vor einigen Monaten auch einen Nachfolger für das Versorgen unserer Website gesucht. Paul Wiegmans war bereit, das auf sich zu nehmen.

Von hier aus wünsche ich meinen Nachfolgern viel Erfolg.

Zum Schluss

> Ein Forth–Programmierer aus Helgoland
> beschämte den Nachbarn, ganz kurzer Hand:
> Dessen Delfinarium–Programm, das in C
> nach vier Monat' Arbeit nicht lief (o je!),
> übertrug er nach Forth, und im Handumdreh'n
> lief bestens es dann — und war doch nur zehnmal so kurz! Also denn: Auf Wiederseh'n!

(Euer Vijgeblad–Redakteur der vergangenen Jahre)"

Een I^2C EEPROM operating system — Robert Henneke

„Im Gegensatz zu den meisten Computern müssen unsere kleinen Freunde ohne Hintergrundspeicher auskommen. Natürlich können wir mit einem PC als Terminal von Download- und Upload–Funktionen Gebrauch machen, um Datenbestände zu laden und aufzubewahren. Ich hatte aber Bedarf an einem einfachen Hintergrundspreicher. ... Da kamen mir die I2C–EEPROMs gerade recht ... Was benötigen wir? ... Record–Typen ... Einteilung auf der Platte ... Software...

Wenn man so etwas macht, entdeckt man, warum beispielsweise MSDOS das tut, was es tut. Warum Platten fragmentieren und was man dagegen tun kann. Kurzum, recht langsam. Das System ist für einen 8052 mit genügend (minimal 8K) externem RAM für Daten und Code gedacht. Für einen AVR ist es weniger geeignet, da der AVR kein Programm–RAM hat."

Forth Metaprogramming: Regexps

Bernd Paysan

Der Artikel beschreibt einen Regexp–Compiler in Forth. Es kann gezeigt werden, dass die Umsetzung eines solchen Compilers nicht nur relativ einfach ist, sondern dass das Ergebnis auch weit performanter ist als vergleichbare Libraries für C, da die Regexps direkt in Maschinensprache umgesetzt werden.

Der Quelltext zeigt auch, wie man in Forth „Metaprogramming“ macht, also dem Compiler neue Tricks beibringt. Zentrales Wort ist hier `POSTPONE`, in der Form `]] [[`, um mehrere Wörter hintereinander zu compilieren.

Einleitung

Was sind Regexps?

Reguläre Ausdrücke (Regexps) sind aus der Unix–Welt bekannt. Programmiersprachen wie Perl, Python oder Ruby verwenden reguläre Ausdrücke an vielen Stellen; und die Programme `sed`, `grep` und `awk` sind sogar fast nur damit beschäftigt, reguläre Ausdrücke zu verarbeiten.

In der Theorie sind reguläre Ausdrücke endliche, nichtdeterministische Automaten. Man kann sie also entsprechend behandeln:

Theoretisch: Durch die Transformation des nichtdeterministischen Automaten in einen deterministischen.

Praktisch: Durch Backtracking.

Die praktische Behandlung besteht dabei in erster Linie aus einer Umfornung des Regexp–Pattern in einen Byte–Code, der stückweise ausgeführt werden kann. Schlägt ein Teilpattern fehl, wird zum letzten erfolgreichen Punkt zurückgesprungen, und ein alternatives Teilpattern ausprobiert.

Beispiele

Ein paar Regexps in PCRE–Syntax:

- `M[ae][ijy]er` (Namen suchen)
- `\d\d:\d\d:\d\d` (Uhrzeit)
- `\(\d*\)\d*/\d*` (Telefon)

Dasselbe in meiner Syntax

- ```
  charclass [ae] 'a +char 'e +char
  charclass [ijy] 'i +char 'j +char 'y +char
  : name ( addr u -- flag )
    (( ‘ M [ae] c? [ijy] c? ‘ e ‘ r )) ;
  ```
- ```
  : time (( \d \d ‘ : \d \d ‘ : \d \d )) ;
  ```
- ```
  : phone ( addr u -- flag )
    (( ‘ ( {** \d **} ‘ ) {** \d **}
       ‘ / {** \d **} )) ;
  ```

Elemente regulärer Ausdrücke

Ein regulärer Ausdruck besteht aus folgenden Elementen:

Zeichen, Zeichengruppen Hier werden einzelne Zeichen gematcht

‘ ⟨*char*⟩ für Zeichenkonstanten,

⟨*class*⟩ `c?` für Zeichenklassen und

=" ⟨*string*⟩" für String (mehere Zeichenkonstanten hintereinander)

Wiederholungen Hier werden Pattern mehrmals wiederholt

`{**` ⟨*pattern*⟩ `**}` Gierige Schleife, mindestens kein mal

`{++` ⟨*pattern*⟩ `++}` Gierige Schleife, mindestens einmal

`{*` ⟨*pattern*⟩ `*}` Genügsame Schleife, mindestens kein mal

`{+` ⟨*pattern*⟩ `+}` Genügsame Schleife, mindestens einmal

Alternativen Hier werden verschiedene Pattern angeboten, von denen eines matchen muss

`{{` ⟨*pattern*⟩ `||` ⟨*pattern*⟩ `... }}` Pattern–Auswahl

Um den Pattern–Compiler einzuschalten setzt man das ganze Pattern in doppelte runde Klammern:

`((` ⟨*pattern*⟩ `))`

Basis

Hilfswörter

Implementiert werden die Regexp üblicherweise als threaded code. In Forth können wir also den vorhandenen Compiler verwenden.

Da die einzelnen Elemente einen Entscheidungsbaum aufbauen, also Sprünge machen, müssen wir sie als Macro inlinen. Dazu verwene ich `]]` und `[[`, alles dazwischen wird mit `POSTPONE` compiliert.

```
: [[ ; \ token to end bulk-postponing
: ]] BEGIN >in @ ' [’] [[ <> WHILE
          >in ! postpone postpone
  REPEAT drop ; immediate
```
```

FORK JOIN

Eine neue Kontrollstruktur brauchen wir: `FORK .. JOIN`. Das entspricht `AHEAD .. THEN`, mit dem Unterschied, dass `FORK` einen Unterprogrammaufruf macht, der zum `JOIN` springt.

Und eine Möglichkeit, `LEAVE` auch ausserhalb einer Zählschleife zu verwenden:

`BEGIN .. LEAVE .. DONE`

Damit bauen wir dann unsere Kontrollstrukturen auf.

Zeichenklassen

Zeichenklassen werden als Bitmaps realisiert. Ist das Bit an der Zeichenposition gesetzt, ist es in der Zeichenklasse. Zeichenklassen können mit einzelnen Zeichen, Zeichengruppen und anderen Klassen aufgefüllt werden; auch können Zeichenklassen subtrahiert werden (etwa: Alle druckbaren Zeichen außer den Interpunktionszeichen).

```
0 Value cur-class
: charclass ( -- )
  Create here dup to cur-class
  $20 dup allot erase ;
: +char ( char -- )  cur-class swap +bit ;
: -char ( char -- )  cur-class swap -bit ;
: ..char ( a b -- )
  1+ swap ?DO  I +char  LOOP ;
: or! ( n addr -- )  dup @ rot or swap ! ;
: +class ( class -- )
  $20 0 ?DO @+ swap
        cur-class I + or!  cell +LOOP  drop ;
: and! ( n addr -- )  dup @ rot and swap ! ;
: -class ( class -- )
  $20 0 ?DO  @+ swap invert
        cur-class I + and!  cell +LOOP  drop ;
```

Zeichenklassen abfragen

Zeichen abfragen muss schnell gehen (Performance!). Deshalb ist hier auch etwas Assembler angebracht. Wir lesen das Zeichen an der aktuellen Position aus, erhöhen diese Adresse um eins, und gucken, ob das Zeichen auch in der Zeichenklasse zu finden ist — wenn ja, wird *true* zurückgegeben.

```
Code char? ( addr class -- addr' flag )
     DX pop
     .b DX ) CX movsx  DX inc  DX push
     CX AX ) bt  b makeflag
     Next end-code macro :dx :f T&P
```

Wird das Zeichen nicht als Teil der Klasse erkannt, springt der Code aus dem aktuellen Block und startet das Backtracking. Die Zeichenabfrage muss alos wie folgt generiert werden:

```
: c? ( addr class -- )
  ]] char? 0= ?LEAVE [[ ; immediate
: -c? ( addr class -- )
  ]] char? ?LEAVE [[ ; immediate
```

Alle Wörter zwischen `]]` und `[[` werden mit `POSTPONE` compiliert. Man kann sich das auch als Makro–Expansion vorstellen.

Ein paar Standard–Zeichenklassen definieren wir uns noch, damit der Nutzer es einfacher hat:

```
charclass digit '0 '9 ..char
charclass blanks 0 bl ..char
charclass letter 'a 'z ..char 'A 'Z ..char
charclass any 0 $FF ..char #lf -char
```

Und hier die Makros, um die Standard–Zeichenklassen abzufragen:

```
: \d ( a -- a' ) ]] digit c? [[ ; immediate
: \s ( a -- a' ) ]] blanks c? [[ ; immediate
: .? ( a -- a' ) ]] any c? [[ ; immediate
: -\d ( a -- a' ) ]] digit -c? [[ ; immediate
: -\s ( a -- a' ) ]] blanks -c? [[ ; immediate
```

Zeichen und Strings

Einzelne Zeichen werden mit ‘ ⟨*char*⟩ abgefragt:

```
: ‘ ( -- )
  ]] count [[ char ]] Literal <> ?LEAVE [[ ;
                                        immediate
```

Strings vergleicht man mit ="

```
: $= ( addr1 addr2 u -- f ) tuck compare ;
: ,=" ( addr u -- ) tuck
  ]] dup SLiteral $= ?LEAVE Literal + noop [[
;
: =" ( <string>" -- ) '" parse ,=" ; immediate
```

Stacks

Der Loop–Stack dient dazu, die Unterblöcke aufzunehmen (`BEGIN .. DONE`). Diese Daten liegen zunächst auf dem normalen Stack herum, wo sie uns beim weiteren Arbeiten aber stören. Dieser Teil muss je nach Forth–System angepasst werden; in bigFORTH wird eine Zelle belegt, in Gforth drei.

```
Variable loops  $40 cells allot
: loops> ( -- addr )
  -1 loops +! loops @+ swap cells + @ ;
: >loops ( addr -- )
  loops @+ swap cells + ! 1 loops +! ;
: BEGIN, ( -- ) ]] BEGIN [[ >loops ;
: DONE, ( -- )
  loops @ IF loops> ]] DONE [[ THEN ]] noop [[
;
```

Der Variablen–Stack nimmt die Variablen auf, die auf geparste Substrings zeigen. Solche Substrings werden mit runden Klammern eingeschlossen, dadurch ergibt sich ein Stack als natürliche Repräsentation des aktuellen Zustands während des Parsens.

```
Variable vars   &18 cells allot
Variable varstack 9 cells allot
Variable varsmax
: >var ( -- addr ) vars @+ swap 2* cells +
  vars @ varstack @+ swap cells + !
  1 vars +! 1 varstack +! ;
: var> ( -- addr ) -1 varstack +!
  varstack @+ swap cells + @
  1+ 2* cells vars + ;
```

Der Datenstack wird auch zur Laufzeit benutzt, und zwar für die Stringadressen. Ganz oben liegt die Adresse, die gerade untersucht wird, darunter Adressen, zu denen per Backtracking zurückgesprungen werden kann. Auf dem Returnstack liegen die Adressen für's Backtracking. Wird `TRUE` zurückgeliefert, war die Sucher bis hierher erfolgreich, bei `FALSE` muss eine andere Alternative durchsucht werden.

Start und Ende

Diese Wörter behandeln Start und Ende des Strings. Diese Adressen werden als globale Variablen gehalten, damit während der Auswertung der Regexp nur ein Wert (die aktuelle Position) auf dem Stack herumgeschoben wird.

```
0 Value end$
0 Value start$
: !end ( addr u -- addr )  over + to end$
                           dup to start$ ;
: $? ( addr -- addr flag )
  dup end$ u< ; macro
: ^? ( addr -- addr flag )
  dup start$ u> ; macro
: ?end ( addr -- addr )
  ]] dup end$ u> ?LEAVE [[ ; immediate
: \^ ( addr -- addr )
  ]] ^? ?LEAVE [[ ; immediate
: \$ ( addr -- addr )
  ]] $? ?LEAVE [[ ; immediate
```

High–Level

Regexp Block

Der ganze Reguläre Ausdruck wird in einen `FORK .. JOIN` Block geklammert — wenn der mit `EXIT` beendet wird, springt `AHEAD` heraus.

```
: (( ( addr u -- )
  vars off varsmax off loops off
  ]] FORK AHEAD BUT JOIN !end [[ BEGIN, ;
                                    immediate
: )) ( -- addr f )
  ]] ?end drop true EXIT [[
  DONE, ]] drop false EXIT THEN [[ ; immediate
```

Gierige Schleifen

Der Algorithmus für die sogenannten „greedy loops“ ist ziemlich einfach:

1. Bis zum Ende des Strings scannen
2. Den Rest des Pattern ausprobieren
3. Beim ersten Erfolg aufhören

Wir können die minimale Anzahl, die das Pattern vorkommen muss, fest einstellen; üblicherweise ist das entweder 0 oder 1 mal. Vor jedem Parsen des wiederholten Patterns kopieren wir die Laufadresse mit `DUP`, und nehmen sie hinterher mit `DROP` wieder weg, wenn der Rest des Pattern nicht passt.

```
: drops ( n -- ) 1+ cells sp@ + sp! ;

: {** ( addr -- addr addr )
  0 ]] Literal >r BEGIN dup [[ BEGIN, ; immediate
' {** Alias {++ ( addr -- addr addr ) immediate
: n*} ( sys n -- )
  >r ]] r> 1+ >r $? 0= UNTIL dup [[
  DONE, ]] drop [[
  r@ IF  r@ ]] r@ Literal u< [[
         ]] IF r> 1+ drops false EXIT THEN [[
  THEN
  r@ ]] r> 1+ Literal U+DO FORK BUT [[
  ]] IF I' I - [[ r@ 1-
  ]]    Literal + drops true UNLOOP EXIT [[
  ]]    THEN  LOOP [[
  r@ IF  r@ ]] Literal drops [[ THEN
  rdrop ]] false EXIT JOIN [[ ; immediate
: **} 0 postpone n*} ; immediate
: ++} 1 postpone n*} ; immediate
```

Genügsame Schleifen

Hier testen wir zuerst den Rest der Regexp, und nur, wenn das fehlschlägt, das Pattern in der Schleife (bei `{+` gehört dieses Pattern zum „Rest“).

```
: {+ ( addr -- addr addr )
  ]] BEGIN [[ BEGIN, ; immediate
: +} ( addr addr' -- addr' )
  ]] dup FORK BUT IF drop true EXIT [[
  DONE, ]] drop false EXIT THEN *} [[ ; immediate
: {* ( addr -- addr addr )
  ]] {+ dup FORK BUT IF drop true EXIT THEN [[
;
                          immediate
: *} ( addr addr' -- addr' )
  ]] dup end$ u> UNTIL [[
  DONE, ]] drop false EXIT JOIN [[ ; immediate
```

Der Such–Prefix macht aus dem Pattern–Matchen eine Pattern–Suche. Statt eine Klasse abzufragen, in der alle Zeichen drin sind (die also ohnehin nie springt), erledigt `1+` den Job schneller.

```
: // ( -- ) ]] {* 1+ *} [[ ; immediate
```

Alternativen

Eine Alternative nach der anderen testen wir ab, indem wir zunächst die erste Variante ausprobieren, und beim

Fehlschlag die nächste. Haben wir bei einem Pattern Erfolg, springen wir gleich an's Ende der Alternativen. Unterwegs müssen wir uns noch um die gerade gescannten Variablen kümmern — es könnten ja in einem Zweig welche verwendet werden, und im anderen nicht.

```
: {{ ( addr -- addr addr )
  0 ]] dup BEGIN [[ vars @ ; immediate
: || ( addr addr -- addr addr )
  vars @ varsmax @ max varsmax !
  ]] nip AHEAD [[ >r vars !
  ]] DONE drop dup [[ r> ]] BEGIN [[ vars @ ;
                              immediate
: THENs ( sys -- )
  BEGIN dup WHILE ]] THEN [[ REPEAT drop ;

: }} ( addr addr -- addr addr )
  vars @ varsmax @ max vars !
  ]] nip AHEAD [[ >r drop
  ]] DONE drop LEAVE [[ r> THENs ; immediate
```

Variablen

Beim Scannen interessieren immer wieder Teilstrings, die nachher verarbeitet werden. Wir speichern Anfangs– und End–Position in globalen Variablen.

```
: \( ( addr -- addr )  ]] dup [[
  >var ]] ALiteral ! [[ ; immediate
: \) ( addr -- addr )  ]] dup [[
  var> ]] ALiteral ! [[ ; immediate
: \0 ( -- addr u )  start$ end$ over - ;
: \: ( i -- )
  Create 2* 1+ cells vars + ,
DOES> ( -- addr u ) @ 2@ tuck - ;
```

```
: \:s ( n -- ) 0 ?DO  I \:  LOOP ;
9 \:s \1 \2 \3 \4 \5 \6 \7 \8 \9
```

Zusammenfassung

Performance

Ein Vergleich mit PCRE, aus dem Shootout Benchmark (12000 Iterationen, 2GHz Athlon64, beides 32–Bit–Programme), ergibt:

- GCC, mit PCRE: 1.483s
- bigFORTH: 0.147s

Das ist ein bequemer Faktor 10. Dazu ist die Datei nur 171 Zeilen lang, während PCRE etwa mit 5000 Zeilen zu Buche schlägt.

Der Quelltext ist Teil von bigFORTH, und dort in der Datei `regexp.fs` zu finden.

Ausblick

Was könnte man noch machen?

- Eine automatische Umsetzung der PCRE–Syntax ist denkbar

  Syntaxvorschlag

  `: phone ( addr u -- f ) ~"` ⟨*regexp*⟩`" ;`

- Die bigFORTH–Teilname am Shootout könnte wieder belebt werden. Gforth ist dort auf einem guten Platz, obwohl das verwendete Debian–Packet nicht optimal übersetzt worden ist.
- Ein Gforth–Port existiert schon; die Bitmaps werden durch Bytemaps ersetzt.

Forth von der Pike auf — Teil 2

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 ist der zweite Teil des Versuchs, ein Forth–System auf die Beine zu stellen, dessen Voraussetzung "überhaupt nix", oder auf gut Deutsch "from scratch", lautet.

16–Bit–Register auf dem AVR

In diesem Teil werden wir versuchen, die Register des AVR–Prozessors auf die Register der virtuellen Forth–Maschine abzubilden. Zuallererst müssen wir uns klar machen, dass wir drauf und dran sind, ein 16–Bit–Forth auf einem 8–Bit–Prozessor hochzuziehen. Wir werden also alle Register mit Register–*Paaren* des AVR–Prozessors koppeln müssen (die zum Glück in genügender Zahl vorhanden sind).

Zunächst noch schnell rekapitulieren, welche Register wir nötig haben:

PC	Maschinenprogrammzähler	Der Motor des Mikro–Controllers.
SP	Datenstack–Pointer	Zeigt auf das zur Zeit oberste Element des Datenstacks.
RP	Returnstack–Pointer	Zeigt auf das zur Zeit oberste Element des Returnstacks.
IP	Interpreter–Pointer	Zeigt auf die nächste Anweisung (Forth–Definition), die ausgeführt werden soll; überwacht die Reihenfolge der Ausführung.
W	Wort–Pointer	Zeigt auf die Definition, die gerade ausgeführt wird; nötig, um den Parameterteil dieser Definition anzuspringen.

Die Wahl des Registers PC ist einfach: Der PC! Der Programmzähler eines Mikro–Controllers ist das einzige "Register", das das "Runnen" des eigentlichen Programms, des Low–Level–Programms, steuert.

Der AVR–Prozessor hat 32 frei verfügbare 8–Bit–Register an Bord: R0 bis R31. Zum Glück haben die Prozessor–Entwickler gut nachgedacht: 8 Register können zu 4 Registerpaaren zusammengekoppelt werden. Das sind:

```
R24 – R25   W,
R26 – R27   X,
R28 – R29   Y,
R30 – R31   Z.
```

Um die richtige Wahl treffen zu können, müssen wir uns die verschiedenen Eigenschaften der Registerpaare ansehen.

Der Datenstack

Wir beginnen unsere Zuordnungssuche mit SP, dem Datenstack–Pointer. Wir wollen Daten auf den Datenstack legen. Hier begegnen uns schon vier Wahlmöglichkeiten:

	Pseudo–Code	Was der tut
1.	INC SP MOV (SP),data	Datenstack wächst nach oben, erst Pointer anpassen, danach dann Daten ablegen.
2.	MOV (SP),data INC SP	Datenstack wächst nach oben, erst Daten ablegen, danach dann Pointer anpassen.
3.	DEC SP MOV (SP),data	Datenstack wächst nach unten, erst Pointer anpassen, danach dann Daten ablegen.
4.	MOV (SP),data DEC SP	Datenstack wächst nach unten, erst Daten ablegen, danach dann Pointer anpassen.

Die Entscheidung darüber, ob der Datenstack nach oben oder nach unten wachsen soll, stellen wir noch etwas zurück. Allerdings wäre es besonders schön, wenn wir keine separaten Befehle INC oder DEC benötigen würden. Am liebsten hätten wir einen Befehl, der automatisch auch gleich noch inkrementiert bzw. dekrementiert. Beim AVR–Prozessor steht diese Möglichkeit für die Registerpaare X, Y und Z tatsächlich zur Verfügung. Mit dem Befehl ST (store)

```
ST X,R5
```

setzen wir den Inhalt des Registers R5 an die Stelle, wohin der Inhalt des X–Registerpaares zeigt. (Ausführliche Informationen finden sich auf der ATMEL–Webseite.)

Der Befehl

`ST X+,R5` (Auswahlmöglichkeit 2)

macht dasselbe, aber danach wird in einem einzigen Zug auch noch der Inhalt des X–Registerpaares um 1 erhöht (post increment). Und gratis ist das obendrein: Die Post–Increment–Aktion kostet *keinen* Extra–Maschinentakt.

Der Befehl

`ST -X,R5` (Auswahlmöglichkeit 3)

arbeitet genauso, jedoch wird erst der Inhalt des X–Registerpaares um 1 erniedrigt, (pre decrement), bevor R5 auf dem dann angewiesenen Platz aufbewahrt wird.

Nun müssen wir uns noch damit beschäftigen, was der SP–Pointer macht: SP zeigt auf das oberste (oder unterste, je nach Richtung) Element des Datenstacks. Wollen wir auf dem Stack neue Daten ablegen, dann muss *erst* Platz gemacht werden, bevor wir die Daten abspeichern können. Von den oben genannten vier Möglichkeiten bleibt also nur die mit Nummer 3 übrig. Damit haben wir uns für ein Nach–unten–Wachsen des Datenstacks entschieden.

Unsere virtuelle Forth–Machine ist 16 Bit breit, so dass wir nun für den 8–Bit–AVR–Prozessor zum folgenden Code kommen: (Daten stehen in R4 und R5 bereit)

```
ST -X,R4
ST -X,R5
```

Man beachte, dass wir damit bereits den Maschinencode für das Forth–Wort "!" (store) gemacht haben.

Aber gemach! Wir wollen unsere Wahl der AVR–Register–Zuweisung bequem gestalten: Sowohl das X– wie auch das Y– und das Z–Register haben die oben genannte Autodekrement–Eigenschaft. Die endgültige Wahl für SP muss also noch etwas zurückstehen.

Wir bekommen es jetzt noch mit etwas anderem zu tun. Es muss noch beschlossen werden, was zuerst aufzubewahren ist: Das obere Byte oder das untere Byte der 16–Bit–Zahlenwerte. Oder auch: Wie wollen wir die Daten auf dem Stack gelagert sehen? Dem Forth–System selbst ist diese Frage egal. Wir müssen uns also nach einem anderen Kriterium umsehen. Die Entscheidung darüber, welche Wahl wir treffen, bleibt noch einen Augenblick lang offen (na ja ... so dringend ist das noch nicht).

Der Returnstack

Erst noch ein weiteres Forth–Register: Der RP. Der Returnstack heißt so, weil die virtuelle Forth–Maschine ihn dazu verwendet, die Rückkehr–Adressen (return = zurück) aufzubewahren, diejenigen Adressen, wo besagte virtuelle Maschine weiterarbeiten soll, *nachdem* ein High–Level–Wort vollständig ausgeführt worden ist. Wenn ein High–Level–Forth–Wort ein schon früher definiertes anderes Forth–Wort aufruft, wird die Adresse des *nächsten* Wortes in die Wortliste auf dem Returnstack eingereiht. Diese Adresse wird wiederhergestellt, sobald das gerade eben aufgerufene Wort vollständig abgearbeitet ist. Das Programm kann dann vom Verzweigungspunkt aus weitergehen.

Der erste Gedanke, der aufkommt, ist der, ob man den RP nicht an den Maschinen–Stack–Pointer SP (nun, beim AVR heißt der nun einmal so) koppeln sollte. Dieses AVR–SP–Register hat genau dieselbe Funktion wie die, die wir bei der virtuellen Forth–Maschine haben wollen: Es zeigt auf die aufbewahrten Adressen, wo der Programm–Counter weiterarbeiten soll, wenn ein Unterprogrammaufruf beendet ist. Auto–Inkrement, bzw. Auto–Dekrement sind auch eingebaut. Beim Aufrufen eines Maschinensprach–Unterprogramms wird gleichzeitig die unmittelbar folgende Adresse auf den Returnstack gelegt. Wenn das Unterprogramm fertig ist, wird diese Adresse wieder in den Programm–Counter PC zurückgeschrieben, so dass das Programm weiterlaufen kann, als ob da nichts geschehen wäre. Also genau das, was wir brauchen.

Was passiert da nun genau? Angenommen, ein willkürlich herausgesuchtes Stück AVR–Maschinencode ruft ein Unterprogramm auf (CALL). Verfolgt man den sich ergebenden CALL–Code (siehe ATMEL–AVR–Befehlssatz auf deren Website), so bekommt man die Sequenz (in Pseudo–Code, in Bytes):

```
MOV (AVR-SP),unteres-Byte-momentaner-PC + 1
DEC AVR-SP
MOV (AVR-SP),oberes-Byte-momentaner-PC + 1
DEC AVR-SP
MOV PC,(momentaner-PC)
```

Wir sehen hier, dass der Returnstack nach *unten* wächst und dass *erst* die Daten abgespeichert werden, bevor Platz gemacht wird. Der AVR–SP zeigt also offenbar auf den ersten freien Platz auf dem Stack. So haben sich die Entwickler bei ATMEL das jedenfalls vorgestellt.

Festlegungen

Nun wissen wir zumindest einiges. Beim Codieren der virtuellen Forth–Maschine wollen wir es uns so einfach wie möglich machen. Wir halten uns an die oben genannte Arbeitsweise.

Wir treffen drei Entscheidungen:

1. Die Forth–Stacks SP und RP wachsen *nach unten.*
2. Das untere Byte eines 16–Bit–Wortes wird zuerst auf den Stack gelegt, darunter dann das obere Byte.
3. Der Forth–Pointer zeigt auf das *obere* Byte eines 16–Bit–Wortes (und also — in Abweichung von der Arbeitsweise des AVR–SPs — nicht auf den leeren Platz unter dem Wort). Das entspricht der Wahlmöglichkeit 3 bei der obigen Besprechung der Datenstack–Zuweisung.

Nun denn... jetzt kommt allmählich etwas Struktur in die virtuelle Forth–Maschine.

— Wird fortgesetzt —

Forth und UTF–8

Bernd Paysan

Was ist UTF–8

UTF–8 ist ein von Ken Thompson entwickeltes Encoding für den Unicode–Zeichensatz, der sich in der Unix–Welt durchgesetzt hat. UTF–8 hat folgende Eigenschaften:

- Kompatibel mit ASCII
- Zeichen variabler Länge
- Nächstes und vorheriges Zeichen jederzeit erkennbar

Mit variablen Zeichenlängen kann Forth nicht von Haus aus umgehen. Das XCHAR–Wordset ist deshalb ein Vorschlag, der eine Umstellung von Forth–Programmen in einer UTF–8–Umgebung möglich macht. Auch andere Codierungen mit variabler Länge sind implementierbar.

bigFORTH und die nächste Version von Gforth implementieren diesen Vorschlag.

UTF–8 Encodings

Die Länge eines UTF–8–Zeichens erkennt man am ersten Byte.

7 Bit `0xxxxxxx`

11 Bit `110xxxxx 10xxxxxx`

16 Bit `1110xxxx 10xxxxxx 10xxxxxx`

21 Bit `11110xxx 10xxxxxx 10xxxxxx 10xxxxxx`

26 Bit `111110xx 10xxxxxx 10xxxxxx 10xxxxxx 10xxxxxx`

31 Bit `1111110x 10xxxxxx 10xxxxxx 10xxxxxx 10xxxxxx 10xxxxxx`

Forth–Befehle für erweiterte Zeichen

Forth–Befehle auf Zeichenebene:

XC–SIZE `( xc -- n )` gibt die Länge eines Unicode–Zeichens in Bytes zurück

```
base @ hex
80 Value maxascii
: xc-size ( xc -- n )
    dup      maxascii u< IF
       drop 1  EXIT
    THEN \ special case ASCII
    $800  2 >r
    BEGIN  2dup u>=  WHILE
       5 lshift r> 1+ >r
       dup 0= UNTIL  THEN
    2drop r> ;
```

XC@+ `( xcaddr -- xcaddr' u )` Entspricht COUNT für XCHAR–Zeichen

```
: xc@+ ( xcaddr -- xcaddr' u )
    count  dup maxascii u< IF
       EXIT
    THEN  \ special case ASCII
    7F and  40 >r
    BEGIN   dup r@ and  WHILE  r@ xor
            6 lshift r> 5 lshift >r >r count
            3F and r> or
    REPEAT  r> drop ;
```

XC!+ `( xc xcaddr -- xcaddr' )` Abspeichern eines Unicode–Zeichens als XCHAR

```
: xc!+ ( xc xcaddr -- xcaddr' )
    over maxascii u< IF
       tuck c! char+  EXIT
    THEN \ special case ASCII
    >r 0 swap  3F
    BEGIN  2dup u>  WHILE
            2/ >r  dup 3F and 80 or
            swap 6 rshift r>
    REPEAT  7F xor 2* or  r>
    BEGIN   over 80 u< 0= WHILE
            tuck c! char+  REPEAT  nip ;
```

XC!+? `( xc xcaddr u -- xcaddr' u' flag )` Abspeichern eines Unicode–Zeichens als XCHAR mit Prüfung, ob es im Rest des Strings Platz hat.

```
: xc!+? ( xc xcaddr u -- xcaddr' u' flag )
    >r over xc-size r@ over u< IF
        ( xc xc-addr1 len r: u1 )
        \ not enough space
        drop nip r> false
    ELSE
        >r xc!+ r> r> swap - true
    THEN ;
```

XCHAR+ `( xaddr -- xaddr' )` Vorwärts zum nächsten XCHAR–Zeichen

```
: xchar+ ( xcaddr -- xcaddr' )
    xc@+ drop ;
```

XCHAR– `( xaddr -- xaddr' )` Rückwärts zum vorherigen XCHAR–Zeichen

```
: xchar- ( xcaddr -- xcaddr' )
    BEGIN  1 chars - dup c@ C0 and
           maxascii <>  UNTIL ;
```

XSTRING+ `( xaddr u -- xaddr' u' )` Vorwärts zum nächsten XCHAR–Zeichen am Ende eines Strings

```
: xstring+ ( xcaddr u -- xcaddr u' )
    over + xchar+ over - ;
```

XSTRING– `( xaddr u -- xaddr' u' )` Rückwärts zum vorherigen XCHAR–Zeichen am Ende eines Strings

```
: xstring- ( xcaddr u -- xcaddr u' )
    over + xchar- over - ;
```

+XSTRING `( xaddr u -- xaddr' u' )` Vorwärts zum nächsten XCHAR–Zeichen am Anfang eines Strings

```
: +xstring ( xc-addr1 u1 -- xc-addr2 u2 )
    over dup xchar+ swap - /string ;
```

–XSTRING `( xaddr u -- xaddr' u' )` Rückwärts zum vorherigen XCHAR–Zeichen am Anfang eines Strings

```
: -xstring ( xc-addr1 u1 -- xc-addr2 u2 )
    over dup xchar- swap - /string ;
```

X–WIDTH `( xcaddr u -- n )` Berechnung des Platzes auf einem fixed–width–Terminal — die Tabelle des zugrundeliegenden WCWIDTH lasse ich hier aber weg.

```
: wcwidth ( xc -- n )
    wc-table #wc-table over + swap ?DO
        dup I 2@ within IF
           I 2 cells + @  UNLOOP EXIT
        THEN
    3 cells +LOOP  1 ;
: x-width ( xcaddr u -- n )
    0 rot rot over + swap ?DO
        I xc@+ swap >r wcwidth +
    r> I - +LOOP ;
```

Forth–Befehle für's Terminal

XKEY `( -- xc )` Lesen eines XCHAR–Zeichens vom Terminal

```
: xkey ( -- xc )
    key dup maxascii u< IF
        EXIT
    THEN  \ special case ASCII
    7F and  40 >r
    BEGIN  dup r@ and  WHILE  r@ xor
           6 lshift r> 5 lshift >r >r key
           3F and r> or
    REPEAT  r> drop ;
```

XEMIT `( xc -- )` Schreiben eines XCHAR–Zeichens auf's Terminal

```
: xemit ( xc -- )
    dup maxascii u< IF
       emit  EXIT
    THEN \ special case ASCII
    0 swap  3F
    BEGIN  2dup u>  WHILE
           2/ >r  dup 3F and 80 or
           swap 6 rshift r>
    REPEAT  7F xor 2* or
    BEGIN   dup 80 u< 0= WHILE
            emit  REPEAT  drop ;
```

CHAR und [CHAR] müssen noch angepasst werden:

```
: char  ( "name" -- xc )
    bl word count drop xc@+ nip ;
: [char] ( "name" -- rt:xc )
    char postpone Literal ; immediate
```

- Kein extra TYPE nötig, wohl aber ein angepasstes ACCEPT.
- Automatische Anpassung, was ein XC ist abhängig von den Umgebungsvariablen, bei ISO–Latin bleibt ein XC ein Byte.

```
80 Constant utf-8
100 Constant iso-latin-1
: set-encoding  to maxascii ;
: get-encoding  maxascii ;
base !
```

Weitere Forth–Befehle nötig?

Beim Zugriff auf Dateien mit verschiedenen Encodings will man eventuell eine transparente Umcodierung.

Beim Wechsel des Systems zwischen verschiedenen Encodings (Wörter, eincompilierte Strings mit Umlauten) besteht ebenfalls ein prinzipielles Problem.

Lösungen:

- Ignorieren? Wir gehen einfach davon aus, dass das System erst mal nur mit ASCII gefüllt wurde, und dann nur eine Umgebung (UTF–8 oder ISO–LATIN) verwendet wird; kein Mix erlaubt.
- Recodieren von Dateien in READ–LINE und WRITE–LINE?

CSV–Files lesen und verarbeiten

Ulrich Hoffmann

Auch im Zeitalter des Datenaustausches über XML ist es immer noch wichtig, mit CSV-Files (Comma Separated Values) umgehen zu können. Dieser Artikel stellt einige Definitionen vor, die es erlauben, CSV-Files zu parsen und auf ihre Bestandteile zugreifen zu können.

Comma Separated Values sind ein minimales Datenformat, um Datensätze auszutauschen. Jeder Datensatz steht in einem CSV–File in einer eigenen Zeile. Die einzelnen Felder des Datensatzes sind durch Kommata getrennt. So weit, so einfach. Problematisch wird es, wenn ein Feld selbst ein Komma enthält. Dies darf nämlich nicht mit einem Feldtrennungs–Komma verwechselt werden. Felder dürfen daher in Anführungszeichen gesetzt werden, um kenntlich zu machen, dass die im Feld möglicherweise vorkommenden Kommata keine Feldtrenner sind. So weit, so einfach. Was aber, wenn ein Feld auch Anführungszeichen enthält. Nun — um ein Anführungszeichen darzustellen, muss das Feld selbst in Anführungszeichen stehen und das Anführungszeichen doppelt geschrieben werden. Hmm — nicht mehr so einfach. Ein paar Beispiele können das vielleicht erhellen:

Die Zeile

```
eins, zwei, drei
```

beschreibt einen Datensatz mit 3 Feldern, die die Werte `eins`, `zwei` und `drei` haben. Soll nun das zweite Feld ein Komma enthalten, notieren wir

```
eins, "zwei, zwei", drei
```

.

Bei

```
eins, zwei, "drei "" drei"
```

enthält das dritte Feld ein Anführungszeichen: Es hat den Wert `drei " drei`.

Das ist eigentlich doch nicht so schwer. Leider immer noch zu schwer für einige Programme. So exportiert OpenOffice Dezimalzahlen ohne Anführungszeichen, auch wenn die deutsche Zahlenformatierung mit Dezimalkomma gewählt ist. Oops.

CSV–Files sind nicht standardisiert und es gibt zahlreiche Dialekte. Statt des Kommas kann zum Beispiel als Feldtrenner auch ein beliebig anderes Zeichen, etwa ein Semikolon oder das Tab–Zeichen (Ctrl-I, 9), zum Einsatz kommen.

Leider besitzen CSV–Files keine Redundanz, so dass Daten im falschen Format meist ohne Fehlermeldung eingelesen und verarbeitet werden. Häufig haben alle Datensätze innerhalb eines CSV–Files die gleiche Anzahl von Feldern. Das kann man für eine einfache Plausibilitätsprüfung ausnutzen. Aber verlassen kann man sich darauf im Allgemeinen nicht.

Um CSV–Files mit Forth zu bearbeiten, lesen wir sie zeilenweise ein und zerlegen jeden Datensatz gemäß der obigen Regeln in Felder. Zunächst benötigen wir einige grundlegende Funktionen für die zeichenweise Bearbeitung von Strings, die wir Forth–typisch auf dem Stack durch ihre Anfangsadresse und ihre Länge darstellen.

```
: c++ ( caddr0 u0 -- caddr1 u1 )
    swap char+ swap 1- ;

: c@+ ( caddr0 u0 -- caddr1 u1 char )
    over c@ >r  c++  r> ;

: cappend ( caddr0 u0 char -- caddr1 u1 )
    >r  2dup chars +  r> swap c!  1+ ;
```

Das Wort `c++` ändert die Stringadresse und Länge auf dem Stack so, dass ein String repräsentiert wird, bei dem gegenüber dem ursprünglichen String das erste Zeichen fehlt. Dazu muss der ursprüngliche String mindestens ein Zeichen lang sein.

Mit `c@+` kann ein String zeichenweise durchlaufen werden. `c@+` liefert das erste Zeichen des Strings und Adresse und Länge des Reststrings ohne das erste Zeichen. Auch bei `c@+` muss der String mindestens ein Zeichen enthalten.

`cappend` hängt ein Zeichen an einen String an. Natürlich muss an der betreffenden Stelle im Speicher Platz für das zusätzliche Zeichen sein.

Gut — nachdem wir das Grundsätzliche geklärt haben, definieren wir Variablen und Tests für den Feldtrenner `sep` und das Anführungszeichen `quot`:

```
Variable sep  char , sep !
Variable quot char " quot !

: sep? ( char -- flag )  sep @ = ;
: quot? ( char -- flag )  quot @ = ;
: -quot? ( char -- char flag )  dup quot? 0= ;
```

Nun widmen wir uns dem zeichenweisen Lesen von Strings, die möglicherweise Anführungszeichen, wie oben beschrieben, haben können. Hierfür ist eine Variante von `c@+` nützlich, die auch mit leeren Strings umgehen kann. Dann soll das Wort `?c@+` einen speziellen Wert `#nochar` liefern, um anzuzeigen, dass kein Zeichen gelesen werden konnte:

```
-1 Constant #nochar

: ?c@+ ( caddr0 u0 -- caddr1 u1 x )
    dup  IF   c@+   EXIT THEN    #nochar ;
```

Für die Behandlung der Anführungszeichen und die spätere Erkennung von Feldtrennern ist es wichtig, ob Zeichen außerhalb der Anführungszeichen oder innerhalb gelesen werden. Deshalb hat das folgende Wort `getc` das

zusätzliche Flag `?out` (outside) als Argument und Resultat.

```
: getc ( caddr0 u0 ?out -- caddr1 u1 ?out' c )
( 1)   >r ?c@+  -quot? IF r> swap EXIT THEN
( 2)   drop r> IF  0  #nochar EXIT THEN
( 3)   ?c@+ -quot? swap ;
```

`getc` versucht, das nächste Zeichen aus dem durch `caddr0 u0` beschriebenen String zu lesen. Stößt es dabei auf ein Anführungszeichen oder das Ende des Strings, so kann es sein, dass kein nächstes Zeichen ermittelt werden kann und `#nochar` als Resultat geliefert wird. `getc` liefert außerdem den möglicherweise um das nächste Zeichen verkürzten String und ein eventuell modifiziertes `?out`. Die drei Zeilen von `getc` regeln dabei Folgendes:

1. Unabhängig, ob `getc` innerhalb oder außerhalb von Anführungszeichen liest, werden *normale* Zeichen einfach extrahiert und `?out` bleibt unverändert. Am Ende des Strings wird `#nochar` geliefert.
2. Behandelt das Auftreten eines Anführungszeichens, wenn Zeichen außerhalb von Anführungszeichen gelesen werden. Dann nämlich wechselt `?out` in den Zustand *innerhalb*. Ein Zeichen wird dann nicht geliefert.
3. Behandelt das Auftreten eines Anführungszeichens, wenn Zeichen innerhalb von Anführungszeichen gelesen werden. Folgt ein weiteres Anführungszeichen, steht es also doppelt, so ist das als nächstes gelieferte Zeichen ein Anführungszeichen selbst. `?out` bleibt dann im Zustand *innerhalb*. Folgt aber kein weiteres Anführungszeichen, steht das ursprüngliche Anführungszeichen also allein, dann wechselt `?out` nach *außerhalb* und das folgende Zeichen wird als nächstes gelesenes Zeichen geliefert. Auch hier wird durch `?c@+` die Behandlung des String–Endes berücksichtigt und möglichwerweise `#nochar` geliefert.

Puh — ganz schön kompliziert, was sich in den drei Zeilen alles tut. Schauen wir uns lieber an, wie `getc` verwendet wird. Wir definieren das Wort `next-csv`, das uns aus einem Datensatz (gegeben durch `caddr0 u0`) das nächste Feld heraussucht und nach `caddr1` legt.

```
: next-csv
   ( caddr0 u0 caddr1 -- caddr2 u2 caddr1 u1 )
    0 >r >r  -1 ( caddr u ?outside )
    BEGIN ( caddr u ?outside )
        over
    WHILE ( caddr u ?outside )
        getc ( caddr' u' ?outside char )
        dup #nochar =
        IF drop
        ELSE
          2dup sep? and  \ outside and sep
          IF ( caddr' u' -1 sep )
              2drop r> r> EXIT THEN
          r> swap r> swap  cappend >r >r
        THEN
        ( caddr' u' ?outside )
    REPEAT ( caddr 0 ?outside )
    drop r> r> ;
```

Das herausgesuchte Feld hat die Länge `u1`. Außerdem verkürzt `next-csv` den Datensatz (`caddr2 u2`). Um seine Aufgabe zu erledigen, liest `next-csv` zeichenweise mit `getc` den Datensatz. Das Ende des Feldes ist gefunden, wenn ein Separator (Komma) gefunden wird, der nicht innerhalb von Anführungszeichen steht. Die gelesenen Zeichen werden mit `cappend` an den String bei `caddr1` angehängt, das zwischenzeitlich mit `u1` auf dem Returnstack geparkt wird. Wird wegen gelesener Anführungszeichen von `getc` mal kein Zeichen geliefert, wird einfach weiter probiert. Ist der Datensatz erschöpft, so bricht die `WHILE`–Schleife ab und der letzte Wert ist gefunden (`u2` ist dann null).

So — wie kann man also die einzelnen Felder eines Datensatzes ausdrucken? Als Beispiel hier das Wort `print-fields`, das das erledigen kann:

```
: print-fields ( caddr u -- )
    BEGIN
      dup
    WHILE
       ." - " pad next-csv ( trim )
       2dup . .  ." >" type ." <" cr
    REPEAT 2drop ;
```

Die einzelnen Werte werden nach `pad` gelesen und dann zusammen mit ihrer Anfangsadresse und Länge von `>` und `<` eingerahmt ausgegeben.

Auch ein Test–Datensatz ist schnell erzeugt:

```
: $line: ( <name> <$> -- )
   create  [char] $ word count
   here over 1+ chars allot place ;

$line:␣record␣␣eins,␣"␣zwei␣,␣zwei␣"␣␣,␣drei$
```

Ein `record count print-fields` gibt damit schließlich so etwas aus wie:

```
-␣5␣211492␣>␣eins<
-␣21␣211492␣>␣␣zwei␣,␣zwei␣␣␣<
-␣5␣211492␣>␣drei<
␣ok
```

Wir sehen also, dass die Leerzeichen innerhalb der Felder erhalten geblieben sind. Das mag nicht immer sinnvoll sein. `print-fields` enthält dafür den auskommentierten Aufruf des Wortes `trim`, das führende und abschließende Leerzeichen entfernt. `trim` zu definieren, bleibt als Aufgabe für den interessierten Leser und führt hoffentlich zu inspirierenden Leserbriefen...

Fazit: Das Bearbeiten von CSV–Files mit Forth ist nicht wirklich schwer. Der Kern sind die beiden Worte `getc` für die Behandlung der Anführungszeichen innerhalb der Felder und `next-csv`, um die Felder aus einem Datensatz herauszulösen. Hat man die Felder erst einmal als Forth–String zu packen, kann man sie beliebig transformieren. Werkzeugkiste auf — Definitionen rein — Werkzeugkiste zu. □

Protokoll zur Jahresversammlung 2005 der Forth–Gesellschaft

Jens Wilke

10. April 2005, Gut Frohberg, Schönnewitz 9, Krögis bei Meißen

Begrüßung

Ulli Hoffmann eröffnet die Versammlung um 9 Uhr und begrüßt die Mitglieder.

Wahl Protokollführer

- Als Protokollführer wird Jens Wilke vorgeschlagen und ohne Gegenstimmen gewählt.

Wahl des Versammlungsleiters

- Als Versammlungsleiter wird Heinz Schnitter vorgeschlagen und ohne Gegenstimmen gewählt.
- Ulrich Hoffmann übergibt die Leitung der Versammlung an Heinz Schnitter.
- Heinz Schnitter stellt die Beschlussfähigkeit der Versammlung fest. Es sind 25 wahlberechtigte Vereinsmitglieder anwesend.

Übergabe der Drachens

Ewald Rieger, der amtierende Drachenträger, hält eine Laudatio für den vom Drachenrat gewählten nächsten Drachenträger. Der Drachenträger für dieses Jahr ist Hannes Reilhofer, der sich, gerade in den Anfängen der Forth–Gesellschaft, besonders engagiert hat.

Ergänzungen TOP Vorschläge

Vorschlag Martin Bitter: Weitere Aktionen in punkto Softwarepatente von der Forth–Gesellschaft. Dieser Punkt wird unter Verschiedenes diskutiert.

Bericht des Direktoriums

Rund um die VD (Fritz)

Fritz berichtet davon, dass es immer schwieriger wird, Material für die VD zu bekommen, und bittet inständig, dass alle Tagungsbeiträge und weitere interessante Beiträge möglichst schnell an ihn geschickt werden.

Kassenstand

Rolf Schöne stellt den Jahresabschluss (Einnahmen und Ausgaben des Vereins) 2004 vor.

Zusammenfassung 2004

Kategoriebeschreibung	01.01.04-31.12.2004
EINNAHMEN	
Verkauf Vierte Dimension	1921,78
Überschuss Jahrestagung	7529,00
Beitrag für lfd. Jahr	2875,95
Porto aus Abos	12,88
Spenden, Wohltätigkeit	609,36
Steuererstattungen	985,86
GESAMT EINNAHMEN	13934,83
AUSGABEN	
Arbeitsmittel Büro	1257,76
Büroausgaben	120,22
Kontoführungsgebühren	155,27
Druckkosten VD	1167,30
Frachtkosten für VD	32,60
Kosten der Internetpräsenz	240,00
Kosten für Ausrichtung Jahrestagung	8813,07
Porto, Postgebühren:	
Portokosten für Versand VD	973,95
Portokosten Verwaltung	396,21
GESAMT Porto, Postgebühren	1370,16
Ausgaben zur Unterstützung des Vereinsziels	1313,36
Verschiedene Ausgaben	128,76
Versand VD	206,90
Kosten für Vereinsverwaltung	966,16
Werbungskosten	567,63
Kategorielos: Ausgaben	0,00
GESAMT AUSGABEN	16339,19
GESAMT EINNAHMEN - AUSG.	-2404,36
UMBUCHUNGEN	
AUF Mehrwertsteuerkonto	-2197,92
VON Mehrwertsteuerkonto	135,32
VON Postbank Business Festgeld	2000,00
GESAMT UMBUCHUNGEN	-62,60
GESAMTSUMME	-2466,96

Im letzten Jahr wurden zahlreiche außerordentliche Ausgaben getätigt: Band- und Beamermiete für die Jahrestagung, Anstecknadeln, Tassen, Notebook für das Forth–Büro, Anschaffung von uCore–Boards.

Dadurch wurde mit einem Minus von etwa 2.300,– Eur abgeschlossen.

Der diesjährige Kassenprüfer, Adolf Krüger, berichtet, dass alle Buchungen nachvollziehbar sind, und erklärt den Kassenabschluss ohne Beanstandung für korrekt.

Mitgliederentwicklung

Im Jahre 2004 sind 8 Mitglieder ausgetreten und 7 neue Mitglieder hinzugekommen. Für 2005 sind schon 5 Austritte zu verzeichnen.

Antrag: „Das Direktorium wird ermächtigt, unter besonderen Umständen Mitglieder vom Mitgliedsbeitrag freizustellen“. Der Antrag wurde ohne Gegenstimmen angenommen.

Zum 6.4.2005 gibt es 128 Mitglieder.

Bericht Ulli Hoffmann über neue Webpräsenz

Weiteres Vorgehen: Kündigung der Präsenz bei Schlund und Partner und Einrichtung der Präsenz auf einem Root–Server bei Strato. Der Einsatz eines „Root–Servers“ ermöglicht höhere Flexibilität und ermöglicht neue Projekte.

Bericht von Fred bzgl. Kontakte zu Amerika, England und Holland

Es bestehen weiterhin Kontakte zur Gruppe um Henry Vinerts im Silicon Valley. Es wurden silberne Anstecknadeln und fünf Ehrungen mit goldenen Anstecknadeln nach Amerika gesandt.

In England gibt es derzeit strukturelle Probleme. Es gibt aktuell etwa 80 Mitglieder. Die Redaktion der Forthwrite konnte von Chris Jakeman nicht weitergeführt werden und wird aktuell für jeweils drei Ausgaben von wechselnden Redakteuren übernommen. Der Chairman der englischen Forth–Gesellschaft, Jeremy Fowell, wurde Mitglied bei unserer deutschen Forth–Gesellschaft.

Die Kontakte zu den Holländern reißen ebenfalls nicht ab. Albert Nijhof, Redakteur, und Willem Ouwerkerk, Vorsitzender der holländischen Forth–Gesellschaft, waren auf der Jahrestagung 2004 und Albert Nijhof und Albert van der Horst waren auf dem LinuxTag. Fred Behringer hat ein Forthbuch von Albert Nijhof aus dem Holländischen übersetzt, das nun über den mv–Verlag Münster im Rahmen von „Books on Demand“ vertrieben wird.

Bericht über Projekte

Auf dem LinuxTag war das „Tingel–Tangel“ von den holländischen Kollegen ein reiner Publikumsmagnet. Außerdem wurden auf dem LinuxTag zwei neue Mitglieder aus dem OpenBIOS–Projekt für die Forth–Gesellschaft hinzugewonnen.

Ulli Hoffmann berichtet vom VD–Scan–Projekt. Aktuell sind alle VDs eingescannt. Einige VDs müssen aber noch mit OCR behandelt und die Größen der Daten reduziert werden.

Bernd Paysan: Es wurde „Thinking Forth“ von Leo Brodie eingescannt und wieder als Buch aufgelegt. Die Rechte von „Thinking Forth“ liegen komplett bei Leo Brodie und sind unter die Creative Common Lizenz gestellt. Bernd möchte „Thinking Forth“ an den aktuellen ANSForth–Standard anpassen und in weitere Sprachen übersetzen. „Thinking Forth“ ist jetzt als „Book on Demand“ über Amazon zu beziehen.

Über das uCore–Projekt gibt es Unklarheiten über den Projektverlauf bzw., wie die Beschlüsse der Forthtagung 2004 auszulegen gewesen sind und ob der Vorstand seine Vorgaben korrekt umgesetzt hat.

Die Sachlage wurde von allen Beteiligten erläutert. Von den in der Jahresversammlung 2004 für das Projekt veranschlagten 3.500,– Eur für 12 Boards wurden 2.784,09 Eur ausgegeben und von Klaus Schleisiek wurden 6 Boards geliefert.

Entlastung des Direktoriums

- Das Direktorium wurde entlastet mit 20 Ja–Stimmen, 4 Enthaltungen und 1 Gegenstimme

Wahl des Direktoriums

Folgende Kandidaten werden vorgeschlagen und stellen sich zur Wahl:

- Fred Behringer
- Ulli Hoffmann
- Bernd Paysan
- Ewald Rieger
- Martin Bitter
- Egmont Woitzel

Antrag: Die Wahl wird abgehalten, indem jeder seine drei Wunschkandidaten auf einen Zettel schreibt. Der Antrag wurde einstimmig angenommen.

Die Auszählung der Wahl wird von Thomas Dammann und Thomas Beierlein vorgenommen. Die Auszählung der Stimmen ergab:

Fred	Ulli	Bernd	Ewald	Martin	Egmont
12	22	17	11	10	3

Ulli, Bernd und Fred nehmen die Wahl an.

Weiterführung Prozessorprojekte

Aufgrund der Probleme beim uCore–Projekt diskutiert die Versammlung über das generelle Vorgehen von Entwicklungsarbeiten. Ulli Hoffmann schlägt vor, dass die Kommunikation allgemein verbessert werden sollte. Klaus Schleisiek schlägt vor, dass man Projektplan, Kostenplan und auch Vorauszahlung für solche Vorhaben machen sollte.

Heinz Schnitter macht darauf aufmerksam, dass ein Verkauf von Boards durch den Verein aufgrund der Gemeinnützigkeit nicht möglich ist. Klaus Schleisiek vertrat die Meinung, dass dies nicht korrekt ist, benannte aber keine konkreten Belege dafür.

Bernd gibt einen technischen Überblick über die unterschiedlichen Entwicklungen bei den FPGA–Herstellern: Xylinx, Altera, Actel. Bernd hält es nicht für sinnvoll, eine Platinenentwicklung zu starten, wenn man nicht in

diesem Bereich ohnehin schon beruflich involviert ist. Es kann sein, dass innerhalb seines Jobs eine solche Aufgabe ohnehin auf ihn zukommen wird. In diesem Fall würde er dann auch „nebenbei“ ein Board für den b16 entwickeln, höchstwahrscheinlich auf der Basis von Actel–FPGAs.

Jens Wilke schlägt vor, existierende Prototypenboards ebenfalls in Erwägung zu ziehen, anstatt Eigenentwicklungen durchzuführen, und diese in den Mikrocontrollerverleih aufzunehmen. Eine Möglichkeit wäre dabei das Board aus dem C–One–Projekt.

Verschiedenes / Forthbücher

Sobald Bernd wieder Zeit hat, wird er das Projekt „Starting Forth“ wieder angehen. Die Herangehensweise soll identisch sein wie beim „Thinking Forth“. Projekt: Erst soll das Werk auf dem bisherigen Stand per OCR in den Computer übernommen werden, dann möchte er schrittweise das Buch auf den aktuellen Stand bringen.

Um 12:19 Uhr wird die Versammlung von Heinz Schnitter geschlossen.

Nächste Tagung: Die nächste Tagung wird von Michael Kalus im Harz durchgeführt.

Zum Thema „Software–Patente“ wird nach dem Mittagessen diskutiert, welche Aktionen die Forth–Gesellschaft unternehmen könnte.

Lebenszeichen

Berichte aus der FIG Silicon Valley: *Henry Vinerts*

7. Dezember 2005

Lieber Fred,

hier sende ich einen weiteren meiner Berichte über „lebendes Forth im Silicon Valley“. Wenn er dir gefällt, leite ihn an jeden der interessierten „Forther“ auf eurer Seite des „großen Teiches“ weiter. Ich weiß, dass Friederich seinen Posten als Editor der Vierten Dimension quittiert hat und ich bedauere dies. Ich nehme an, dass ein Teil dieses Bedauerns aus der Erkenntnis stammt, dass wir mit zunehmendem Alter weniger weit voraus schauen können als rückwärts auf unserer Reise durch das Leben. Letzten 29. Juni war es 10 Jahre her, dass ich meinen ersten Brief an Friederich schrieb. Viele Briefe folgten und fanden ihren Weg in die Vierte Dimension. Vielleicht spiegeln sich meine Gefühle an diesem Punkt in einigen Gedanken von Goethe wieder, die er vor vielen Jahren schrieb:

> Briefe hebt man auf, um sie nie wieder zu lesen; man zerstört sie zuletzt einmal aus Diskretion, und so verschwindet der schönste, unmittelbarste Lebenshauch unwiederbringlich für uns und andre.

Nun, passenderweise, bringt mich der Zug meiner Gedanken zu den wundervollen Übersetzungen meiner Briefe durch den „freundlichen Professor aus Mittweida.“ Thomas Beierlein, ich danke dir sehr für die unzweifelhaft harte Arbeit meinen kalifornischen Slang in einwandfreies, gelehrtes Deutsch zu konvertieren. (Und natürlich vergib mir bitte, dass ich gespannt bin ob du mir das Original des obigen Zitates von Goethe geben kannst. *[Ich habe lange gesucht, aber es gefunden. TB]*)

Samstag der 19. November 2005 war der fünfzehnte jährliche SVFIG–Forth–Tag an dem ich teilnahm, seitdem ich die Forth–„Kultur“ betrat (Ich mag das Wort seitdem ich „von Kult zu Kultur“ auf eurer Webseite gelesen habe). Es war eine erfreuliche Überraschung für mich, eine ansehnliche Gruppe vorzufinden, die alle aufgrund ihres Interesses an Forth zusammengefunden hatten. Wir begannen eine Stunde eher als üblich und zögerten so lange, bis wir eine Stunde später als üblich aufhörten. Im Gruppenbild (http://www.forth.org/svfig/kk/Forth-Day-2005.jpg), welches kurz vor Chuck Moores Rede gemacht wurde, fehlt mehr als ein halbes Dutzend der Leute, die im Laufe des Tages kamen und gingen. Es ist herzerwärmend, dass viele der Teilnehmer in der FIG waren, seit sie 1978 in North Carolina gegründet wurde. Ich würde sie gern alle benennen, aber ich werde warten, bis unsere Organisatoren die Namen für das Gruppenbild liefern. Ich möchte jedoch erwähnen, dass ich sehr froh war, Dave Boulton und Bill Muench zu erkennen, die ich beide seit unserem SVFIG–Treffen vor vielen Jahren nicht gesehen hatte.

Nach der Begrüßung durch den Veranstaltungsleiter George Perry fand der erste Vortrag über neue Produkte für embedded Forth–Systeme statt. Unser Webmaster Dave Jaffe, der sie vorstellte, hat eine Aufstellung davon auf der SVFIG–Webseite veröffentlicht, gefolgt von einer Zusammenfassung der Vorträge der anderen Redner. Ich sollte versuchen, sie alle kurz zu erwähnen, und dabei, wie üblich, meine Ignoranz gegenüber den Computerwissenschaften und den letzten Hard– und Softwareprodukten offenbaren, indem ich spezifische technische Details vermeide und meine Zeilen auf Notizen persönlichen Interesses und Verständnisses begrenze.

Als Nächster kam Dr. Ting, um seine „Forth Briefmarke“ vorzustellen. Dies ist ein Single–Chip–Computer, der auf dem AduC7020 von Analog Devices basiert. Bob Nash folgte mit einer Marktbeschreibung von Rick Van Normans Swift Forth, allerdings als erfahrener Nutzer und

Abbildung 1: Forth Day 2005, Henry im Swap–Shirt links neben Chuck Moore

nicht als Verkäufer für Forth, Inc. Ich werde euch später sagen, warum ich jetzt eine frei nutzbare DemoVersion des Systems (von `http://www.forth.com/`) heruntergeladen habe.

George Perry beschrieb Carbon MacForth (`http://www.macforth.com/`). John Hall, der vor kurzem bei Apple aufgehört hat, fügte hinzu, dass Forth die erste kommerzielle Sprache war, die mit dem Macintosh verkauft wurde, und das MacForth mit OS 8 and 9 arbeiten kann. Wie ihr seht, hatte Kevin Appert gefordert, dass alle Vortragenden ihre Präsentationen auf Forth–bezogene Themen begrenzen. Jedermann hat dies befolgt, außer Ting, der, wie ich in meinen Notizen vermerkt habe, jedermann seine traditionellen Forth–Tags–Grillspeisen präsentierte. Aber ich beschwere mich nicht, wie es auch kein anderer tat. Danke, Dr. Ting!

Es war keine Überraschung, dass das Mittagessen sowie die folgenden „Einführungen, Bekanntmachungen, Gerüchte und Anekdoten“ länger als geplant dauerten. Es war daher eine gute Sache, dass sich unser Gastgeber, Cogswell College, bereit erklärte, uns über die reguläre Endzeit von 16 Uhr hinaus bleiben zu lassen.

John Rible gab einen kurzen Bericht als Schatzmeister. Lasst mich aber zu seinem regulären Vortrag springen, da alle drei restlichen Redner zusammen in der gleichen Firma (Intelesis in Cupertino) arbeiten. Ihr habt vielleicht von dem Logik-Puzzle, genannt Sudoku, gehört. Dabei müssen fehlende Zahlen so in einem 9–mal–9 Gitter ergänzt werden, dass jede Zeile, jede Spalte und jedes der neun 3–mal–3 Quadrate innerhalb des Gitters genau einmal jede der Zahlen von 1 bis 9 enthält. Seine Popularität hat sich über die Welt verbreitet und es gibt verschiedene Schwierigkeitsgrade, abhängig davon, wie viele Zahlen in welchen Plätzen des Gitters vorgegeben sind. Für das schwierigste dieser Puzzle habe ich mehrere Stunden per Hand gebraucht. Daher kann ich gut erkennen, welche Herausforderung es für einen Programmierer ist, ein Programm für einen Computer zu schreiben, dies zu lösen. John Rible (und eine Anzahl seiner Vorgänger) haben genau dies getan. John zeigte uns seine Lösung (in Swift Forth, welches ihr im Web unter `www.forth.com` findet), die mir sehr clever und elegant erschien. Und deswegen habe ich mir die Evaluations Version von Swift Forth besorgt. Ich hatte allerdings noch nicht genügend Zeit, um Johns Code auszuprobieren. Vielleicht sollte ich es auch einfach sein lassen, da meine Frau die Sudoku Puzzle aus der lokalen Tageszeitung jeden Tag gelöst hat und kein Interesse daran hat, dass ein Computer der Herausforderung den Spaß nimmt.

Jeff Fox, Michael Montvelishsky, und Chuck Moore haben schon eine ganze Zeit lang zusammengearbeitet. Jeff hielt den ersten Vortrag. Wieder einmal zu schwierig für mich! Ich habe mir notiert, dass er über Linda, Transputer, Forthlets, ColorForth, ANSForth, Gforth usw. gesprochen hat.

Michaels Vortrag war ebenso weit über meinem Horizont. “Using the Forth multiprocessor chip.” “Parallel paradigm.” “Discrete Fourier transforms vs. FFT.” waren die Schlagworte.

Chuck beendete das Treffen mit den letzten Geschichten aus seinem Leben und seiner Arbeit. Chip Design, unter Nutzung seines eigenen Werkzeugs (OKAD, ColorForth), scheint nach wie vor seine Hauptbeschäftigung zu sein. Zur Zeit ist es der Forth Multiprozessorchip, welchen er uns erklärte. Eine Reihe von Leuten sind schon damit beschäftigt, Anwendungen dafür zu schreiben. Jeff zeichnete Chucks Rede auf Video auf und wird diese wahrscheinlich wie gewohnt als DVD verfügbar machen.

Meine abschließenden Gedanken für heute sind eigentlich eine Frage an die jüngere Generation. Haben wir ein Zeitalter erreicht, in welchem Computer–Wissenschaft und Informationstechnologie es höchst unwahrscheinlich machen, jede Spur der Erinnerung an das, was Goethe das Allerschönste nannte, unwiederbringlich zu zerstören. Und werden unsere Kinder und ihre Nachfahren fragen, „Ach, wer bringt die schönen Tage, jene holde Zeit zurück?“

Euch allen Frieden und Freude im Neuen Jahr wünschend,

Henry.

Übersetzer: Thomas Beierlein

Bericht von der EuroForth 2005

Anton Ertl

Die EuroForth 2005 fand vom 20.10.2005–23.10.2005 in Santander in Nordspanien statt, und wurde von Federico de Ceballos recht gut organisiert.

Santander ist ein traditionelles Seebad, und die Konferenz fand im Hotel Santemar in der Nähe des Strandes und des Casinos statt. Auch wenn es schon etwas zu kalt zum Schwimmen war, hatten wir doch recht warmes und schönes Wetter, das wir bei einigen Spaziergaengen geniessen konnten. Weiters genossen wir die ausgezeichneten landesüblichen Speisen und Getränke.

Diese Konferenz war eher schwach besucht (8 Teilnehmer), was aber ihrer Intensität keinen Abbruch tat. Wir hatten genügend interessante Forth–Themen zu diskutieren, sodass uns eher die Zeit ausging als dass uns langweilig wurde.

Am Tag vor der eigentlichen Konferenz fand das Treffen zum Forth 200x Standard <`http://www.complang.tuwien.ac.at/forth/ansforth/forth200x.html`> statt. Fast alle Teilnehmer der Konferenz nahmen auch an diesem Treffen teil. Dabei wurden die Vorschläge diskutiert, die den RfD/CfV–Prozess durchlaufen hatten <`http://www.complang.tuwien.ac.at/forth/ansforth/rfds.html`>:

- deferred
- extension–query
- parse–name
- defined

Die Vorschläge wurden nach einiger Diskussion so überarbeitet, dass sie in ein neues Standard–Dokument integriert werden können, und schließlich alle angenommen.

Weiters wurde bei dem Standard–Treffen diskutiert, welche weiteren Vorschläge geplant sind. Und der Standardisierungsprozess wurde diskutiert: Wie werden die neuen Vorschläge in das Standard–Dokument integriert? Wie gut funktioniert der RfD/CfV–Prozess? Sollen wir den neuen Standard durch eine offizielle Standardisierungsorganisation wie ANSI oder ISO absegnen lassen, und wenn ja, welche? Und einige Verwaltungsfragen wie Vorsitz, Redakteur des Standard–Dokuments, und das nächste Treffen. Ein genauerer Bericht (in Englisch) über das Treffen ist auf <`http://www.complang.tuwien.ac.at/forth/ansforth/meetings/2005.html`> zu finden.

In der eigentlichen Konferenz gab es dann im offiziellen Programm folgende Vorträge:

- M. Anton Ertl, David Gregg: Stack Caching in Forth
- M. Anton Ertl, Bernd Paysan: Xchars or Unicode in Forth
- Angel Robert Lynas, Bill Stoddart: SuDoku Solver Case Study: from specification to RVM–Forth (part I)
- N.J. Nelson, C. Williams: First experiences with Microcore
- N.J. Nelson, K.B. Swiatlowski: Self Documenting Sequences
- Federico de Ceballos: Simplicity in Forth
- Stephen Pelc: XML, SOAP and Web Services in Forth

Die Proceedings der Konferenz sind auf <`http://www.complang.tuwien.ac.at/anton/euroforth2005/papers/`> verfügbar.

Die Vorträge wurden teilweise ausführlich diskutiert, und sorgten teilweise auch im informellen Teil und in den Workshops für Diskussionen.

Die Themen, die in den Workshops behandelt wurden, habe ich nicht aufgezeichnet und sie sind mir daher nicht mehr alle präsent, aber u.a. ging es um die Themen Unicode und Benchmarks; dabei wurde nicht nur diskutiert, sondern auch recht viel am Laptop gearbeitet, demonstriert, und Dateien ausgetauscht; insgesamt eine recht intensive Erfahrung.

Die Vorteile, die so ein persönliches Treffen ab und zu bieten kann, wurden anhand des Unicode–Themas deutlich: Im Vorfeld wurde das Thema im Rahmen eines RfD von Bernd Paysan im Netz diskutiert, wobei Stephen Pelc dazu tendierte, man müsse Unicode–Zeichen als 16–bit oder 32–bit–Einheiten fixer Größe im Speicher halten, und in dieser Ansicht nicht viel Bewegung zeigte; der Xchars–Vorschlag geht aber (vereinfacht gesagt) in die Richtung, im Speicher eher mit Strings aus 8–bit–Einheiten (chars) zu arbeiten, wobei ein Unicode–Zeichen (xchar) aus mehreren dieser Einheiten bestehen kann. Nach längeren Diskussionen änderte Stephen Pelc seine Ansicht radikal und kam zu der Überzeugung, dass man Unicode–Zeichen und andere internationale Zeichensätze sehr gut in Form von Strings aus einfacheren Einheiten darstellen kann, und dass man sogar ohne einzelne Unicode–Zeichen auf dem Stack auskommt (was Xchars noch vorsieht); die Anwesenheit zweier Forth–Programmierer, die Erfahrung mit internationalisierten Anwendungen haben, hat bei diesem Meinungsumschwung wahrscheinlich auch eine große Rolle gespielt.

Die nächste EuroForth wird von Janet Nelson organisiert und findet voraussichtlich vom 15.9.2006–17.9.2006 in Cambridge statt, wieder mit einem Standardisierungstreffen davor am 14./15.9.2006. Ich hoffe auf zahlreiches Erscheinen. Näheres zu vergangenen und zukünftigen EuroForth–Konferenzen findet man auf <`http://dec.bournemouth.ac.uk/forth/euro/`>.

Hexadoku

Martin Bitter

Auf der Website des Vereins gab es einen Hinweis zu einem interessanten Hardwareprojekt der Zeitschrift Elektor: ein Microkontrollerboard, das der Ausgabe 12/2005 beilag (leider vergriffen). Es gibt Überlegungen, dazu ein Forth zu entwickeln. Diesem Hinweis folgend, klickte ich mich also zu `http://www.elektor.de/` — wirklich interessant! Dabei stieß ich auf ein Preisrätsel in Form eines Hexadoku. Hexa-watt? Dokumentation im Hexcode? Nein — ein Hexadoku ist die Abwandlung eines Sudokus. (Wie dekliniert man Sudoku?) Die Regeln sind schnell erklärt: Ein Sudoku ist ein Spielfeld aus 9 Zeilen und 9 Spalten. Zusätzlich ist es in drei mal drei (9) Quadrate eingeteilt. In diesem Spielfeld gilt es nun, die Ziffern 1–9 so zu verteilen, dass jede Ziffer in jeder Reihe, in jeder Spalte und in jedem Quadrat einmal — und nur einmal! — vorkommt. Fertig! Kurze Regel, langes Spiel. Es gibt $9! \times 722 \times 27 \times 27.704.267.971$ Möglichkeiten vgl. `http://de.wikipedia.org/wiki/Sudoku`. Zum Teil ausgefüllte Sudokus sind in Japan als Denkspiele sehr beliebt, sollen aber auch bei uns schon recht bekannt sein.

Ein Hexadoku ist ein erweitertes Sudoku. Statt 9 Zeilen und Spalten gibt es je 16. Statt 9 Unterquadrate sind es ebenfalls 16. Und statt der Ziffern von 1–9 werden die Hex-Ziffern 0–F verwendet. Die Anzahl gültiger Hexadokus ist mir unbekannt.

Regeln verstanden — Spielplan ausgedruckt und losgelegt ...

Bald wurde das Kombinieren und Ausprobieren langweilig. Das muss sich doch mit dem Rechner abkürzen lassen! Andererseits: à la brute force im Backtracking, das ist doch unter der Ehre! Oder? Aber eine Hilfe, die den stupiden Teil der Arbeit erledigt — die kann man (ich) schon benutzen!

Das war der Anlass. Es entstand das Programm Hexadoku (Quellcode `hexadoku.f`) mit bigforth unter Linux. Es sollte aber auch in anderen Umgebungen funktionieren. Vielleicht muss man die Worte zum Dateihandling anpassen. Für das Folgende mag es hilfreich sein, sich das Hexadoku aufzumalen (siehe Quelltext) oder auszudrucken (`start`).

Mein Vorgehen in kurzen Worten: Nachdem die Rätselaufgabe (als 16 Strings) gespeichert ist, kann sie mit `neu` in das eigentliche Spielfeld (Array) übertragen werden. Unbesetzte Zellen werden mit FF gekennzeichnet. *Über* dem Spielfeld liegen weiter 17 Felder (`ebene`). Das erste Feld wird mit Nullen, das zweite mit Einsen, ... das 16. (F.) Feld mit Fs gefüllt. Das 17. Feld wird mit Nullen belegt.

Jetzt wird überprüft, welche Ziffer in einer Zelle des Spielfeldes gesetzt ist. In der Zelle in Zeile 2 Spalte 0 ist das die 3. Das bedeutet, in der 4. Ebene, die den Wert 3 repräsentiert, werden alle Zellen der Zeile 2 und alle Zellen der Spalte 0 als *verbrannt* (FF) gekennzeichnet. Die Zelle 2 0 gehört zum Unterquadrat 1. Also werden auch alle Zellen der 4. Ebene, die zu diesem Unterquadrat gehören, markiert. Dies geschieht für alle Zellen des Spielfeldes. Übrig bleiben 16 Ebenen, die nur dort Werte von 0–F enthalten, wo diese auch für die betreffende Spielfeldzelle noch erlaubt sind. Mit dem Wort `? ( Zeile Spalte -- )` kann man sich anzeigen lassen, welche Ziffern das sind. Bsp.: `2 1 ?` zeigt an, dass hier noch die Ziffern `0 4 A F` möglich sind.

Das war der Plan — soweit so gut!

Aber ein wenig Hilfe mehr ist erlaubt. Oder?

Also weiter: Nun wird gezählt, wie viele erlaubte Ziffern es zu jeder Zelle noch gibt. Dazu werden einfach alle Zellen *über* einer unbesetzten Spielfeldzelle betrachtet und gezählt. Das Ergebnis findet sich in der 17. Ebene und kann mit `.v` betrachtet werden. Bsp.: `.v` zeigt nun an, dass zu der Zelle 2 1 noch 4 erlaubte Zahlen existieren. Auch das ist eine Hilfe. Aber noch weiter: Wenn nun schon mal gezählt ist, dann kann man doch überprüfen, bei welchen Zellen nur noch ein Wert möglich ist, und diesen dann auch gleich eintragen lassen. Das macht das Wort `automatik` (Kurzform `au`).

Natürlich muss man eine Spielfeldzelle auch per Hand setzen können, falls die Automatik nicht weiterweiß. Das macht `z!`, dem man den Wert und die Koordinaten übergibt. Bsp.: `0 2 1 z!` . Netterweise läuft ohne weiteres Zutun die Automatik an.

Zum Schluss bleibt dann nur noch, das jeweilige Ergebnis schön formatiert auszugeben. Das entsprechende Wort `.ebene` gibt das aktuelle Spielfeld formatiert aus und zeigt einiges an Statistik an: zur leichteren Orientierung die Zeilen- und Spaltenindizes, die Anzahl der freien Zellen pro Zeile, Spalte und Unterquadrat, die Anzahl der automatisch gesetzten Spielfeldzellen und die Anzahl der noch möglichen Kombinationen, soweit sie berechenbar ist (in den allermeisten Fällen führt dies zu einem Überlauf — sobald hier ein vernünftiger Wert angezeigt wird, ist man der Lösung schon recht nahe (subjektiv)).

Das alles hilft schon ganz schön! Aber bequemer geht es, wenn man Spielstände speichern, ändern und zurückrufen kann. Hier habe ich mich entschieden, die gemachten Eingaben zu speichern. Dazu wird ein Stack angelegt, auf den mit den Worten `z` und `n` zugegriffen werden kann. `z` steht für zurück und `n` für noch einmal. Dieser Zug-Stack kann mit `save_zuege` gespeichert und mit `lade_zuege` wieder eingelesen werden.

Trotz alledem habe ich bisher nicht die Lösung gefunden (Zeitmangel natürlich!). Es kann aber sein, dass das Programm einigen Lesern der Vierten Dimension Spaß und

Kurzweil im Umgang mit Hexadokus bereitet. (Es ist für den privaten Gebrauch entstanden, gar nicht kommentiert und enthält immer noch Bugs und Unschönheiten, wie zum Beispiel, dass nur noch eine freie Zelle in einer Spalte/Zeile nicht als Fehler erkannt wird.)

Auch wenn es nicht so ganz sportlich ist: Vielleicht kann `hexadoku.f` ja als Ausgangspunkt für eine vollautomatische Lösung dienen?

```
hex     \ Diesmal ist es einfacher, alles im Hexmodus zu betreiben

12 Value spielebenen    \ So viele Ebenen brauche ich

Create Aufgabe  \ ab hier wird die Aufgabe gespeichert

 ," E      A  56 F8 "
 ,"  65   E 18 F 03A"
 ," 3 7B  65 D   2  "
 ," 8     B  34   5 "

 ,"    07  19     2 "
 ," 9B   2 0F7 8D  6"
 ," 5 E7  FBD16 C   "
 ,"  D 6 3  2 0  A 7"

 ,"     D     C 287 "
 ," 73BE  9C0A82  6 "
 ,"   A  1    7E B9 "
 ," 29    0 64D   A "

 ," 476  F     A0  2"
 ," B C3A 5480   EF "
 ," DE9 0C2 4 F5  18"
 ," F  5 B    19 4D3"


Create Spielfeld        \ Jetzt folgt der Speicherbereich fuer das Spiel

  10 10 spielebenen * *         \ so viele Bytes braucht's schon

  allot here value spielfeldende
\ ---------------------------------------------- Verwaltung -------------------------
\ Da es nur ein Spielfeld gibt, wird mit absoluten Adressen gerechnet

: ebene ( n -- adr )    \ gibt die Anfangsadresse der Ebene n zurueck
  10 10 * * spielfeld + ;

: zse>adr ( zeile spalte ebene -- adr ) \ gibt die Adresse der gewaehlten Speicherstelle
  10 10 * * -rot swap
  10 * + + spielfeld + ;

: leer? ( zeile spalte ebene -- flag )  \ ist die gewaehlte Zelle leer?
  zse>adr c@ FF = ;

: neu ( -- )                            \ Aufgabe ins Spielfeld uebertragen
  Aufgabe 10 0
     DO count 0
        DO count bl = IF FF ELSE dup 1- 1 s>number drop THEN
           0 ebene J 10 * i + + c!
        LOOP
     LOOP drop ;

: vorbesetzen ( -- )    \ Jede Ebene wird mit 'ihrem' Wert gefuellt
  10 0 DO
       10 0 DO
          10 0 DO K J I 1+ zse>adr I swap c!
               LOOP
            LOOP
```

```
60          LOOP
61    11 ebene 10 10 * erase ;
62
63    : fuelle_zeile ( n zeile ebene -- )     \ eine ganze Zeile mit einem Wert fuellen
64    ebene swap 10 * + 10 rot fill ;
65
66    : fuelle_spalte ( n spalte ebene -- )   \ eine ganze Spalte mit einem Wert fuellen
67    ebene + 10 0 DO 2dup c! 10 + LOOP 2drop ;
68
69    : fuelle_quadrat ( n Index ebene -- )   \ Ein Spielquadrat mit einem Wert fuellen (Index 0-F)
70    ebene swap 4 /mod 40 * swap 4 * + +
71    4 0 DO 2dup swap 4 swap fill 10 + LOOP 2drop ;
72
73    : loesche_saeule? ( zeile spalte -- )   \ bei besetzter Zelle alle Moeglichkeiten stornieren
74    2dup 0 leer?  \ wenn Spielfeld (noch) leer ist ...
75    IF   2drop            \ tu nichts
76    ELSE swap 10 * + 10 0         \ ansonsten:
77          DO FF over I 1+ ebene + c!      \ trage in jeder Ebene den Leermarker (FF) ein
78          LOOP drop
79    THEN ;
80
81    : zaehle_saeule ( zeile spalte -- )     \ Eintraege einer Saeule ( moegliche Werte fuer die
82    swap 10 * +   \ entsprechende Zelle) zaehlen, Ergebnis in Ebene 11
83    0 pad !               \ speichern
84    10 0 DO dup I 1+ ebene + c@ FF <>
85            IF 1 pad +! THEN
86            LOOP
87     11 ebene + pad @
88     swap c! ;
89
90    : zaehle_saeulen ( -- )         \ fuer jede Zelle die Anzahl der Moeglichkeiten ermitteln
91    10 0 DO
92          10 0 DO I J loesche_saeule? I J zaehle_saeule
93                      LOOP
94             LOOP ;
95
96    Variable gueltige_Zahl  \ hier wird der gueltige Wert gespeichert
97
98    : notiere_wert ( zeile spalte -- )      \ falls es nur einen Wert gibt, ihn notieren
99    0 gueltige_Zahl !
100   swap 10 * +
101   dup 11 ebene + c@
102   1 = IF 10 0 DO dup I 1+ ebene + c@ dup
103                                FF <> IF gueltige_Zahl +! ELSE drop THEN
104                         LOOP
105                  dup 11 ebene + 0 swap c!
106                  gueltige_Zahl @ swap 0 ebene + c!
107     ELSE drop
108     THEN ;
109
110   : schreibe_zahlen ( -- )        \ fuer jede Zelle den 'einen' Wert notieren
111   10 0 DO
112         10 0 DO J I notiere_wert
113                     LOOP
114            LOOP ;
115
116   : wert! ( n zeile spalte -- )   \ einen Wert in eine Zelle schreiben
117   swap 10 * + 0 ebene + c! ;
118
119   \ ---------------------------- Erlaubte Zahlen pruefen -------------------------
120
121   : markiere_zeile ( zeile -- )   \ benutzte Zahlen als besetzt markieren (Zeilen/Ebene)
122   dup 10 * spielfeld +
123   10 0 DO dup I + c@ dup FF <>                  \ zeile adr n flag
124           IF   FF 3 pick rot 1+ fuelle_zeile
125           ELSE drop
```

```
126             THEN
127           LOOP 2drop ;
128
129   : markiere_zeilen ( -- )        \ dies fuer alle Zeilen
130     10 0 DO I markiere_zeile LOOP ;
131
132   : markiere_spalte ( spalte -- )         \ benutzte Zahlen als besetzt markieren (Spalten/Ebene)
133     dup spielfeld +
134     10 0 DO dup I 10 * + c@ dup FF <>
135             IF   FF 3 pick rot 1+ fuelle_spalte
136             ELSE drop
137             THEN
138           LOOP 2drop ;
139
140
141   : markiere_spalten ( -- )       \ ... fuer alle Spalten
142     10 0 DO I markiere_spalte LOOP ;
143
144   : sz>index ( spalte zeile -- index )    \ errechne aus Koordinaten den Index eines Quadrates
145     4 / swap 4 / 4 * + ;
146
147   : markiere_quadrat ( Index -- )         \ benutzte Zahlen als besetzt markieren (Quadrate/Ebene)
148     dup
149     4 /mod 40 * swap 4 * + 0 ebene +
150     4 0 DO
151          4 0 DO count dup FF <>
152                  IF 1+ 2 pick swap FF -rot fuelle_quadrat
153                  ELSE drop
154                  THEN
155              LOOP
156          10 4 - +
157         LOOP 2drop ;
158
159   : markiere_quadrate ( -- )      \ fuer alle Quadrate
160     10 0 DO I markiere_quadrat LOOP ;
161
162   : markiere_spiel        \ alle Felder des Spieles pruefen und markieren
163     markiere_quadrate
164     markiere_zeilen
165     markiere_spalten ;
166
167   Variable minimum        \ Speicherstelle
168
169   : finde_minima ( -- )   \ welche 'leere' Zelle hat die wenigsten Loesungen
170     10 minimum !
171     11 ebene
172     10 10 * 0 DO count dup 0<> IF minimum @ min minimum ! ELSE drop THEN
173               LOOP drop ;
174
175   : loese_1 ( -- flag )   \ alle Zellen, die eindeutig (1) sind, loesen
176     vorbesetzen
177     markiere_spiel
178     zaehle_saeulen
179     finde_minima
180     minimum @ 1 = IF schreibe_zahlen true ELSE false THEN ;
181
182   \ ----------------------------- Spielstaende merken ----------------------------
183
184   Create zug_stack 3 10 10 * * allot      \ einen Stack fuer die Spielzuege einrichten
185
186   Variable zug# 0 zug#    \ sozusagen der Zug-Stackpointer
187
188   : merke_zug ( n n n -- )        \ Wertetripel in den Zugstack schreiben
189     2dup  zug# @ 3 * zug_stack + dup >R 2 + c!
190                                     R@ 1 + c!
191     2 pick                          R>     c!
```

```
192   zug# @ 1+ 10 10 * mod zug# ! ;        \ 'ueberfluessige' ? Ueberlaufsicherung
193
194   : hole_zug  ( n -- n n n )      \ Wertetripel vom Zugstack holen
195     3 * zug_stack + count swap count swap c@ ;
196
197   : .zuege ( -- ) \ den Zugstack ausgeben (jeweils letzte Eintraege)
198     0 50 at ." Wert Zeile Spalte"
199     zug# @ dup 18 - dup 0 <=
200     IF drop 0 0 ELSE dup THEN
201     -rot
202     ?DO
203         I over - 2+ 50 at
204         I hole_zug rot 4 .r swap 6 .r 6 .r
205     LOOP drop ;
206
207   : zeige_dateien ( -- ) s" dir zuege*.f " evaluate ;     \ der Name sagt's
208
209   Variable datei_ID       \ Platzhalter
210
211   : save_zuege ( -- )     \ Zugstack (human readable) in Datei speichern
212     s" zuege_" pad place
213     base @
214     &10 base !
215     time&date rot 0 <# [char] _ hold # # #> pad +place
216              swap 0 <# [char] _ hold # # #> pad +place
217                   0 <# [char] _ hold # # # # #> pad +place
218                   0 <# [char] : hold # # #> pad +place
219                   0 <# [char] : hold # # #> pad +place
220                   0 <# # # #>               pad +place
221     s" .f" pad +place
222     base !
223     pad count r/w create-file cr drop
224     datei_ID !
225     zug# @ 0 DO
226              3 0 DO J 3 * I + zug_stack + c@ 0 <# bl hold  # # #> datei_ID @ write-file drop
227                  LOOP 0A pad c! pad 1 datei_ID @ write-file drop
228              LOOP
229      datei_ID @ close-file drop
230      ." Getan!" ;
231
232   : lade_zuege ( c-addr count -- )        \ Zugstack aus Datei fuellen
233     0 zug# !
234     r/o open-file
235     IF datei_ID !
236        BEGIN pad &10 datei_ID @ read-file nip nip
237        WHILE zug# @ 3 * zug_stack +
238           3 0 DO I over + pad I + 2 s>number drop swap c! LOOP drop
239           zug# @ 1+ zug# !
240        REPEAT
241        datei_ID @ close-file
242     THEN ;
243
244
245   \ ----------------------------- Statistik -------------------------------------
246
247   Create Doppelte 16 cells allot  \ Speicherbereich fuer benutzte Werte
248   Variable Fehler Fehler off      \ flag
249
250   : frei_vorbereiten ( -- )       \ Fehlerabfrage initialisieren
251     0 pad !
252     Fehler off
253     Doppelte 16 FF fill ;
254
255   : gueltig_zaehlen ( n -- )      \ sind Eintraege doppelt vorhanden?
256     dup FF = IF   drop 1 pad +!
257              ELSE dup doppelte + dup
```

```
258                 c@ ff <> IF 2drop Fehler on bell ELSE c! THEN
259             THEN ;
260
261   : frei_quadrat ( index -- n )   \ alle Werte eines Spielquadrates miteinander vergleichen
262     frei_vorbereiten
263     4 /mod 40 * swap 4 * + 0 ebene +
264     4 0 DO
265         4 0 DO count gueltig_zaehlen
266             LOOP 10 4 - +
267         LOOP drop
268     pad @ ;
269
270   : frei_zeile ( zeile -- n )     \ alle Werte einer Zeile miteinander vergleichen
271     frei_vorbereiten
272     10 * 0 ebene + 10 0 DO count gueltig_zaehlen LOOP drop
273     pad @ ;
274
275   : frei_spalte ( spalte -- n )   \  alle Werte einer Spalte miteinander vergleichen
276     frei_vorbereiten
277     0 ebene +
278     10 0 DO dup c@ gueltig_zaehlen 10 + LOOP drop
279     pad @ ;
280
281   : Moeglichkeiten ( -- d )       \ Moeglichkeiten errechnen (Ueberlauf!!!!!)
282     1 s>d
283     10 0 do 10 0 do J I 11 zse>adr c@ dup 0<> IF s>d d* ELSE drop THEN LOOP LOOP ;
284
285   : .Moeglichkeiten ( -- )        \ Ausgeben
286    Moeglichkeiten ."    Moegl.: " ud. ;
287
288   \ ----------------------------- Anzeige ---------------------------------------
289
290   : .wert ( c -- )        \ Spielfeldwert ausgeben, wenn gueltig!
291      dup FF = IF drop ."   " ELSE 2 .r THEN ;
292
293   : .trenner ( -- ) ."   +-----------+-----------+-----------+-----------+" ;
294   : .spalten ( -- ) ."     0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F" ;
295
296   : .zeile  ( adr -- adr )
297      ." |"
298      4 0 DO
299          3 0 DO count .wert ."  " LOOP count .wert ." |"
300      LOOP ;
301
302   : .saeule ( zeile spalte -- )   \ alle Moeglichkeiten einer Zelle anzeigen
303      1 zse>adr 10 0 DO dup c@ dup FF <>
304                                          IF . ELSE drop THEN 10 10 * +
305                                  LOOP drop ;
306
307   Variable Statistik? Statistik? on       \ flag
308
309   : .Fehler ( -- )        \ gibt ein X aus, falls ein Fehler bemerkt wurde
310      Fehler @ IF ." X" ELSE ."  " THEN ;
311
312   : .ebene ( n -- )       \ Zeigt eine Spielfeldebene formatiert an
313     page
314     ebene
315     cr .spalten
316     cr .trenner
317     4 0 DO
318         4 0 DO cr j 4 * I + . .zeile j 4 * I + 2 .r
319             Statistik? @ IF ."   --> " J 4 * I + frei_zeile . .Fehler THEN
320             LOOP
321         cr .trenner
322         Statistik? @ IF ."    " 4 0 DO ." - " J 4 * I + frei_quadrat . .Fehler LOOP THEN
323         LOOP drop
```

```
324     cr .spalten
325         Statistik? @ IF cr ."     " 10 0 DO I frei_spalte . .Fehler LOOP THEN
326         .zuege 1A 2 at ;
327
328   : .spielfeld            \ Zeigt das Spielfeld
329       0 .ebene ;
330   \
331   \ : .wuerfel \ Tasteneingabe bei eingeschaltetem Numpad dort ... schei... 'key'
332   \  cr 0 .ebene 1
333   \   BEGIN cr dup ." Ebene: " .
334   \         key
335   \         dup   1b = IF 2drop true THEN
336   \         dup 38 ( FF52 ) = IF drop 1+ 11 min dup .ebene false then
337   \         dup 32 ( FF54 ) = IF drop 1-  0 max dup .ebene false then
338   \   UNTIL ;
339
340    \ ------------------------ Kuerzel -----------------
341
342   : start ( -- )
343     neu vorbesetzen 0 .ebene ;
344
345   : automatik ( -- n )    \ loest alle eindeutigen Zellen
346     0 >R
347     Begin r> 1+ >R loese_1 false =
348     UNTIL r> ;
349
350   : au ( -- ) automatik drop      \ zeigt das Spielfeld
351       0 .ebene ."             " .moeglichkeiten ;
352
353   : z! ( wert zeile spalte -- )   \ einen Wert bei den Koordinaten eintragen
354     merke_zug
355     wert! automatik  0 .ebene ."   " . ." Zuege!" .moeglichkeiten  ;
356
357   : wiederhole ( -- )     \ alle Eintraege des Zugstacks nochmal abspielen
358     neu
359     zug# @ 0 ?DO I hole_zug wert! automatik drop LOOP
360     0 .ebene ."             " .moeglichkeiten   ;
361
362   : ? ( zeile spalte -- werte )   \ Moegliche Werte fuer die Koordinaten anzeigen
363     ." :" .saeule ;
364
365   : .v 11 .ebene .moeglichkeiten ;        \ Anzahl der Moeglichkeiten pro Zelle anzeigen
366
367   : z  ( -- )  zug# @ 1- 0 max zug# ! wiederhole ;        \ einen Zug zurueckgehen
368   : n  ( -- )  zug# @ 1+ 100 min zug# ! wiederhole ;      \ einen Zug wiederherstellen (nochmal)
369
370   \ ----------------------- Loslegen -----------------
371
372   start
373
374   2 f 1 z!                \ drei
375   6 8 0 z!                \       moegliche
376   E 5 E z!                \               Eingaben
377
378   \ 10 0 [do] [i] dup 6 swap z! [loop]
379
380   \ 10 0 [do] [i] dup 7 swap z! [loop]
```

Forth–Gruppen regional

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

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

München
Jens Wilke
Tel.: (0 89) – 8 97 68 90
Treffen: jeden 4. Mittwoch im Monat 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 82 33)–3 05 24** (p)
Fax: **(0 82 33)–99 71**

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

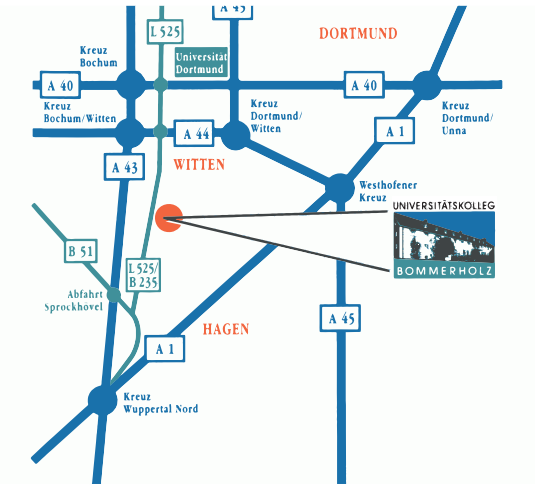
Die nächste Jahrestagung der Forthgesellschaft wird in der Nähe von Witten stattfinden. Witten, am Südrand des Ruhrgebietes zwischen Bochum und Dortmund gelegen, bewirbt sich wie die ganze Region um den Titel der Kulturhauptstadt 2010. Menschen einander näher zu bringen durch Kultur, mit diesem Ziel 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 Perspektive des Forth. 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 gefertigten Chips. Ich hoffe, es gibt eine Ausstellung dazu. Sodann gebührt den Applikationen in Forth unsere Aufmerksamkeit und — last 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`