Vierte Dimension

2/2007

Das Forth–Magazin

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

In dieser Ausgabe:

Dienstleistungen und Produkte fördernder Mitglieder des Vereins

tematik GmbH
Technische Informatik

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

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

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

LEGO RCX–Verleih

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

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

Inhalt:

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

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

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

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

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

Hier könnte Ihre Anzeige stehen!

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

`Secretary@forth-ev.de`

KIMA Echtzeitsysteme GmbH

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

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

FORTecH Software
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.

Hier könnte Ihre Anzeige stehen!

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

`Secretary@forth-ev.de`

Ingenieurbüro
Klaus Kohl–Schöpe

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

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

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: `4d@forth-ev.de`

Anzeigenverwaltung

Büro der Herausgeberin

Redaktionsschluss

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

Erscheinungsweise

1 Ausgabe / Quartal

Einzelpreis

4,00€ + Porto u. Verpackung

Liebe Leser,

ich begrüße Euch zu einem neuen Heft unseres Forth–Magazins. Die Forth–Jahrestagung im April war ein großer Erfolg. Meine Eindrücke dazu findet Ihr weiter hinten im Heft.

Was könnt Ihr sonst noch erwarten?

Ein Workshop–Thema auf der Tagung war der Vergleich verschiedener Stack–orientierter Programmiersprachen, der Anton Ertl dazu angeregt hat, einmal genauer auf die Gemeinsamkeiten und Unterschiede von Factor, Postscript und Forth einzugehen und Laufzeiten zu messen. Die Ergebnisse findet Ihr in seinem Artikel.

In einem anderen Workshop haben wir uns amForth und das AVR–Butterfly–Board genauer angesehen — das AVR–Sonderheft ist Euch ja schon zusammen mit der letzten Vierten Dimension zugegangen. Michael Kalus und Matthias Trute haben sich des Problems angenommen, einzelnen Bits, etwa in Steuer–Bytes von Peripheriebausteinen, Namen zu geben, um so lesbarere Programme zu bekommen. Rückmeldungen zu ihrem Vorgehen sind ausdrücklich erwünscht.

Das AVR–Butterfly–Board wird klassisch über eine normale Terminal–Schnittstelle programmiert und bedient. Eine andere Vorgehensweise ist die *Nabelschnur–Forth*–Technik: Ein Wirts–Forth–System übernimmt die Kontrolle über einen Forth–Kern im Ziel–System, auf dem statt des klassischen Text–Interpretierers nur ein spezialisierter Token/Adress–Interpretierer arbeitet. Jörg Völker berichtet uns über das T4–System, das ein Vertreter dieser Technik ist, und das er erfolgreich zur Programmierung von Truck–Modellen einsetzt. Interessanterweise hat auf der diesjährigen Tagung Dr. Willi Stricker über seinen ganz ähnlichen Ansatz berichtet. Für mich ist das ein Zeichen, dass dieses Vorgehen viel Sinn macht. Soweit ich mich erinnern kann, habe ich von Hartmut Pfüller das erste Mal von dieser Technik erfahren — in Aachen 1989.

Und sonst noch? Rafael Deliano erläutert uns, welche Bedeutung der Zentrale Grenzwertsatz für die Signalverarbeitung hat. André Elgeti zeigt uns, wie man Datensätze im Forth–Dictionary verwalten kann. Und dann gibt es erstmalig in der Vierten Dimension ein *Centerfold*: Anton Ertl's Forth–Tree.

Ich hoffe, damit ist für jeden etwas Interessantes dabei?

Eine Sorge treibt mich allerdings um. Auf Rückmeldungen, liebe Leser, zu den Artikeln der vergangenen Ausgaben und überhaupt zum AVR–Sonderheft wartet die VD–Redaktion vergeblich. Wir würden uns deutlich mehr Kommentare, Anregungen, Nachrichten wünschen. Leserbriefe, auch aufmunternde :-), bekommen wir gerne! Und — für die kommenden Hefte benötigen wir noch Artikel. Also, geht bitte in Euch und schreibt was für die Vierte Dimension, damit sie weiter existieren kann.

Quasi zeitgleich mit dem Erscheinen dieses Heftes findet die euroForth 2007–Konferenz auf Schloss Dagstuhl bei Saarbrücken statt, die in diesem Jahr unter dem Motto *Stack-orientierte virtuelle Maschinen* steht. Auf dieser internationalen Konferenz treffen sich alljährlich Forth–Interessierte und –Profis, um sich über jüngste Entwicklungen rund um Forth auszutauschen.

Und nun — viel Spaß beim Lesen.

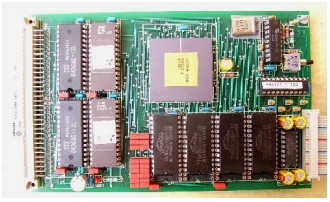
Ulrich Hoffmann

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

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

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

Forth antik: Stackprozessoren



Liebe Forth Freunde,

Rafael Deliano war so freundlich, Photos von Stack–Prozessoren aus der Geschichte des Forth zur Verfügung zu stellen. Sie sind nun auf unserer Website in der Galerie in einem eigens dafür eingerichteten Photoalbum zu sehen: `http://www.forth-ev.de/` und im Menü *Photos* anklicken. Wir sind nun auf der Suche nach weiteren Photos. Wenn ihr solche habt oder machen könntet oder wisst, wen wir ansprechen könnten, um weitere zu bekommen, teilt es bitte der Redaktion mit. Viel Vergnügen beim Stöbern.

Michael

Forth — Wo bin ich?

Wo bin ich denn?

Derjenige, der bei uns im Kollegium diese Frage stellt, weiß meist doch genau, wo er ist: Vor dem Vertretungsplan! Nichts ist gewiss — und schon mal gar nicht, in welcher Klasse man demnächst unterrichten soll. Wo ist unsere Vereinsseite? Wie steht sie da? Auch wer das fragt, weiß meist, wo sie ist: Na klar, unter `www.forth-ev.de`! Aber WO ist das? Wer sind die Nachbarn und Verwandten? Diese Frage beantwortet hochaktuell diese Website: `http://www.touchgraph.com/TGGoogleBrowser.html` (ein Wermutstropfen — s'ist halt von Google). Am 4. 8. bot sich dort das Bild aus Abbildung 1 auf der nächsten Seite.

Gruß Martin

Neues vom Forth-Büro

Als Anlage zu dieser zweiten VD des Jahres finden Sie — wie auch in den letzten vier Jahren — eine aktuelle Mitgliederliste. Nutzen Sie diese bitte.

Wir begrüßen als neue Mitglieder zwei Informatiker.

Der eine, Jens Freimann, noch Student, ist u. a. bei Open Firmware engagiert.

Den anderen, Hans G. Hein, hat Rafael Deliano mit uns bekannt gemacht. Er ist schon „ausgewachsen“ und mit 6800, 6502, ... aufgewachsen.

Beide sympathisieren mit der Linux–Fraktion.

Ebenso begrüßen wir Karl–Heinz Kreis aus Leverkusen, einen gestandenen 50-er. Er bevorzugt Mac und Linux.

Leider werden uns am Jahresende drei Mitglieder (Pütz, Allinger, Kripahle) verlassen. Ihnen wünschen wir für ihren weiteren Berufs– und Lebensweg alles Gute.

Gegenwärtig haben wir 131 Mitglieder. Im Januar 2008 werden es höchstens 128 sein. Also werben Sie bitte für uns im Bekannten–, Freundes–, Kommilitonenkreis. Wir vertreten schließlich eine gute Sache.

Rolf Schöne

eVD — Forth Magazin elektrisch

Begonnen hat das ganze wohl so um 1995. Jedenfalls bezeugt ein erster Beitrag in der Vierten Dimension 1/1997 das: Michael Schröder aus Oschersleben korrespondierte mit Claus Vogt über die neuen Möglichkeiten, die nun per DTP hergestellten Hefte in HTML ins Internet zu bringen. Der Prozess war aber derart mühsam, das es über den Jahrgang 1995 hinaus keine weiteren HTML–Hefte gegeben hat.

Aufgegriffen wurde die Idee dann später von Ullrich Hoffmann. Es zeichnete sich ab, dass PDF wohl das geeignete Format werden würde und Acrobat als Tool fand sich ein. Und heute konnte ich das eVD–Projekt damit abschließen. Immerhin 12 Jahre Entwickklung, bis es im laienhafteren Umfeld handhabbar ist auf diese Weise.

Ulrich hat viele motiviert mitzumachen, und so wurden zunächst alle alten Hefte der Vierten Dimension gescannt, rund 2800 Seiten. All diese Images wurden in schwarz–weiß Bilder komprimiert und mittels Akobat jeweils heftweise zu einem PDF–Dokument gebunden. Die so elektronisch verfügbaren Hefte erhielten Lesezeichen und Links vom Inhaltsverzeichnis ins Heft, und wurden mit paper capture (OCR) durchgearbeitet, so das sie nun auch elektronisch durchsuchbar sind.

Auf diese Weise stellte Ulrich Hoffmann nach und nach in Schüben bis Anfang 2007 schon 40 der insgesamt 83 alten Hefte wieder zusammen. Und lud sie ins Archiv der FG–Website. Erst die Jahrgänge seit 2004 liegen als Originale schon in PDF vor.

Wie bin ich dazu gekommen, in das Projekt einzusteigen? Nun, neulich suchte ich nach einem älteren Beitrag aus der VD — Laufzeitmessung der Forthworte — aber genau dieses gewünschte Heft fehlte, und deshalb, nun auch beflügelt von der Forth–Tagung 2007, beschloss ich, die Lücken zu schließen und auch die restlichen Hefte ins Archiv zu bringen, nicht ahnend, was das für eine Arbeit werden würde!

Nach Einarbeitung ins Acrobat ging die Arbeit dank Ulrichs Vorarbeit gut voran. Alle Scans waren auf dem Server der FG zugänglich. Aber mit dem Zusammenstellen war es ja nicht getan, die Links setzen gehörte dazu und auch das OCR. Und das lief sehr langsam. Aber nach und nach ist es nun doch fertig geworden, und ich kann nun Ulrich gut nachfühlen, wieso irgendwann auch mal wieder was anderes in sein Leben kommen musste als alte Hefte aufzubereiten! Nicht gegangen wäre das ganze ohne so flotte Umgebungen wie OSX auf dem Mac mit Expose und GraficConverter und den flotten Scannern wie HP–Officejet All–in–One. Damit kann sogar ich nun sowas machen.

Etwas flotter als die gescannten Hefte zusammen zu stellen ging es mit den Heften 1994–97, da damals schon

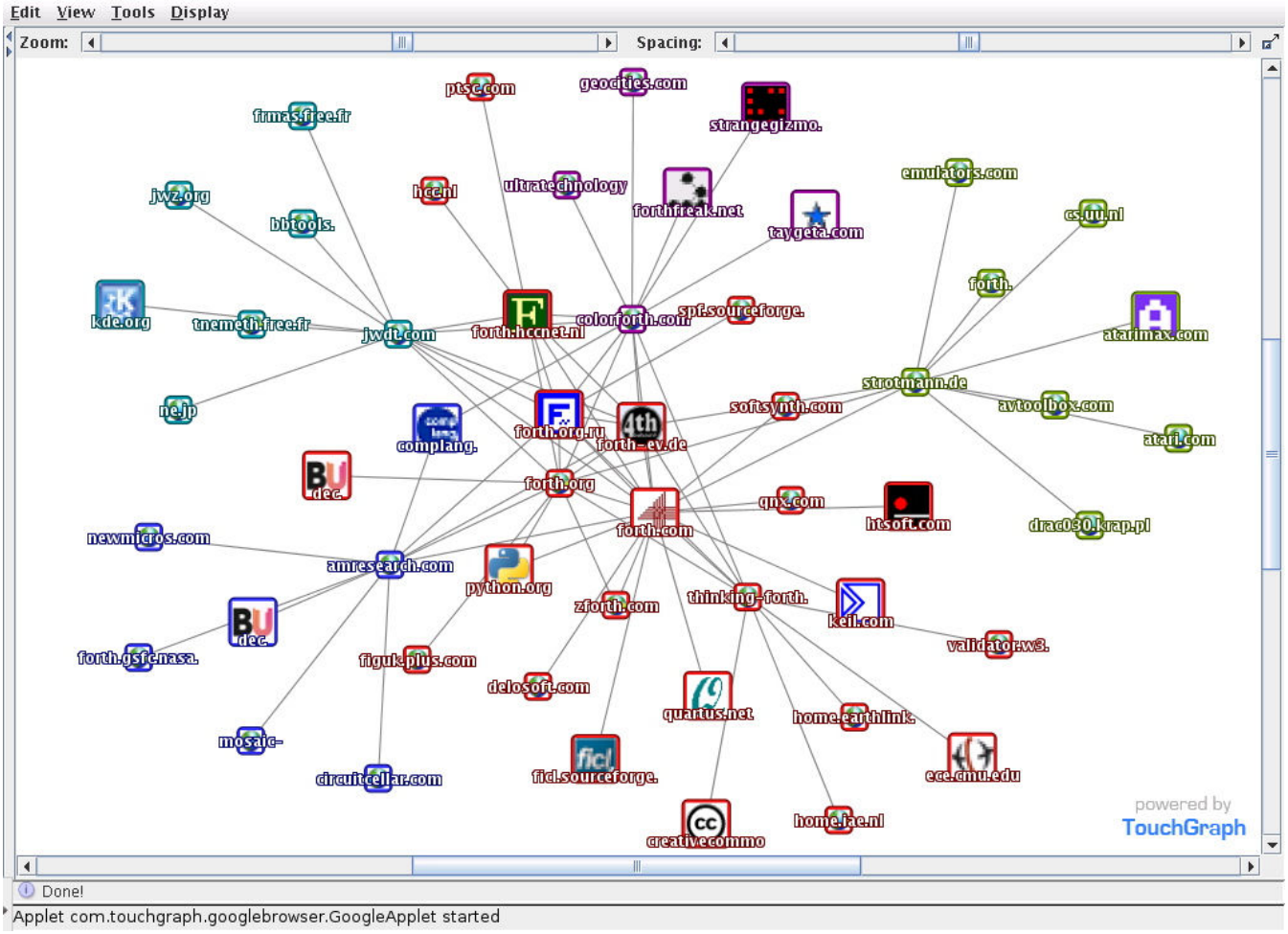


Abbildung 1: Wie ist `www.forth-ev.de` verbunden?

etliche Beiträge in PDF gemacht worden waren. Aber einige Beiträge fehlten dann doch und mussten nachträglich gescannt werden. Die meisten Hefte dafür fand ich in meiner eigenen Sammlung. Und bei den übrigen Heften half eine Anfrage auf de.comp.lang.forth um sogleich Scans zu bekommen — dafür meinen besten Dank noch mal auch an dieser Stelle.

Und so ist es heute soweit. All diesen freundlichen Helfern aus unserem Verein und den übrigen Helferinnen und Helfern sage ich Dank für ihre rasche und bereitwillige Hilfe, ihr habts beflügelt:

Friederich Prinz, Bernd Paysan, Martin Bitter, Klaus Zobawa, Thomas Prinz, Fred Franzkowiak, Janne Hoffmann, Daniel Kalus und Christopher Kalus.

Und mein ganz besonderer Dank geht an Ulrich Hoffman für die Beharrlichkeit und die enorme Vorarbeit von Anbeginn des Projektes an, und seine praktische und technische Unterstützung nun wieder bei der Erledigung des Restes. Sollte in der Aufzählung jemand fehlen, bitte ich um Nachsicht — in den Jahren des Projekt kann die Spur schon mal verloren gehen — und danke auch demjenigen.

Lieber Michael Schröder und lieber Claus Voigt, Euch meinen Dank fürs publik machen. Ideen, die in die Welt gesetzt werden, gehen weiter, so oder so. Ist das nicht schön zu sehen?

Euer M. Kalus

Wiki mit Übersetzungen

Unser Wiki enthält ja Informationen, die nicht nur für den deutschsprachigen Raum interessant sind. Einzelne Übersetzungen haben wir deshalb schon eingestellt (vom R8C–Projekt). Wer Wikipedia kennt, weiß, dass man solche Übersetzungen auch systematisch anzeigen kann.

Inzwischen gibt es auch für Dokuwiki ein entsprechendes Plugin. In jeder Seite erscheint dann oben rechts eine Zeile mit den möglichen Übersetzungen (verfügbare blau, der Rest ausgegraut). Wer eine neue Übersetzung anlegen will, klickt einfach auf den ausgegrauten Kasten, und legt die übersetzte Seite an.

Dabei wird auch die Benutzerführung übersetzt — unser Wiki ist damit internationalisiert. Die Übersetzungen finden sich in eigenen Namespaces wieder — die Top–Level–Ebene ist dabei das Zweibuchstabenkürzel der Sprache.

Euer Bernd Paysan

Zentraler Grenzwertsatz

Rafael Deliano

Er dient gerne als Rechtfertigung für die Annahme, dass natürliche Rauschsignale Gaußverteilung haben. Es ergeben sich aber auch Anwendungen.

Normalverteilte Zufallszahlen

Die Plots für die Verteilung wurden hier über 10k Samples gemacht und die Werte 0–65535 jeweils auf 255 „Eimer“ („bins“) skaliert (Bild 1–5).

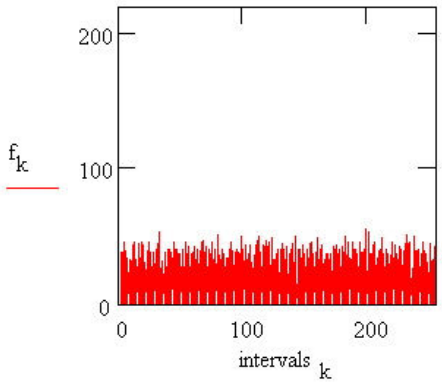


Bild 1: Gleichverteilung LCG

Gängige Generatoren liefern Gleichverteilung (Bild 1). Benötigt wird aber für viele Anwendungen Gauß. In [1] wurde als Generator ein LFSR verwendet und die Umwandlung durch ROM durchgeführt. Das ist wegen Speicherverbrauch für Rechnersimulation unpraktikabel. Hier sind auch LCGs [2] (Listing 1) geeigneter, weil qualitativ meist besser als LFSRs.

Der zentrale Grenzwertsatz („central limit theorem“) beschreibt die Änderung der Verteilung, wenn man den Mittelwert aus mehreren Zufallszahlen bildet. Bei Gleichverteilung ergibt sich mit 2 Samples ein Dreieck, bei 3 Samples bereits zunehmend Gaußform (Bild 2). Als Analogie gäbe es die wiederholte Faltung eines Rechtecksignals mit sich selbst.

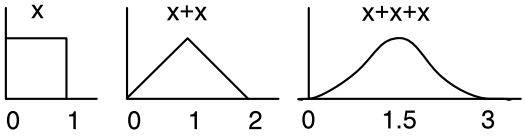


Bild 2: Veränderung der Verteilung

Als ausreichend für Simulation wird meist der Mittelwert aus 12 Samples angenommen (Bild 5, Listing 2). Man hat zwar nun die gewünschte Verteilung, aber die Samples reichen nicht mehr von 0001–FFFF. Abhängig davon, wieviel man mittelt, verändert man nicht nur Form, sondern senkt auch den Pegel.

```
1   2 ZVARIABLE SEED   AAAA SEED !
2
3   : RAND     \ (  -  UN1 )
4      SEED @ E51D U* DROP 3619 + DUP SEED ! ;
```

Listing 1: LCG–Generator

```
5   : RAND'    \ (  -  UN1 )
6      0 0 0B 0
7      DO RAND 0 D+ LOOP 0C U/
8      8000 -     \ to signed
9   ;
```

Listing 2: Umwandlung auf Gauß

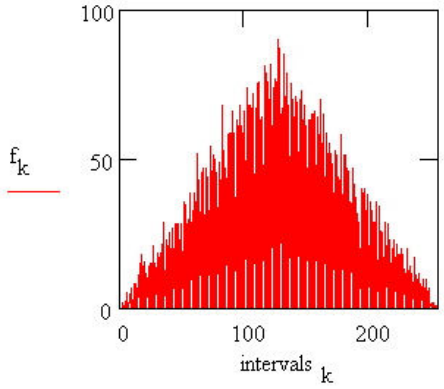


Bild 3: Mittelung 2 Samples

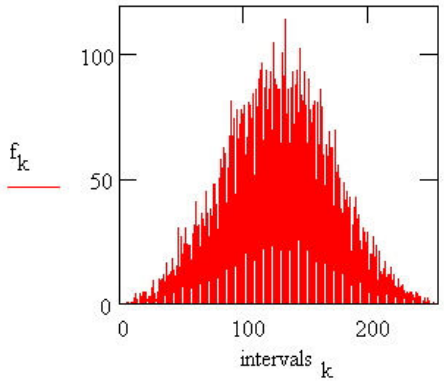


Bild 4: Mittelung 3 Samples

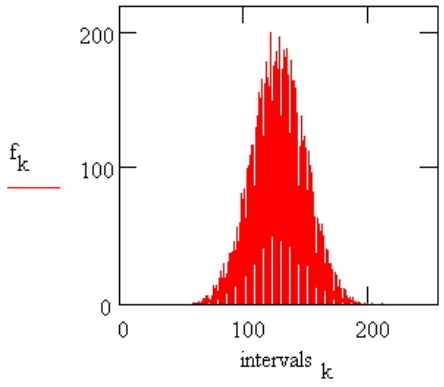


Bild 5: Mittelung 12 Samples

Magic Number

Die Form der Gaußkurven (Bild 6) wird durch die Standardabweichung s („sigma“) bzw. davon abgeleitet der Varianz s^2 beschrieben. Die Fläche unter der Kurve beträgt nominell immer 1,0. Die gängigste Ausführung hat $s^2 = s = 1$.

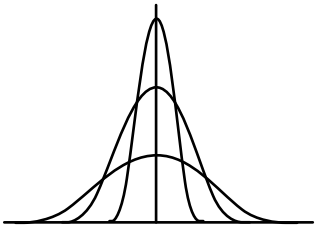


Bild 6: Gaußkurven mit unterschiedlicher Standardabweichung

Für das gleichverteilte Signal in Bild 7 gilt $s^2 = 1/12$ und die Varianz steigt linear mit der Zahl der gemittelten Rauschquellen. Die magische Zahl 12 ergibt also nominell die übliche Gaußkurve $s^2 = s = 1$ (Bild 5).

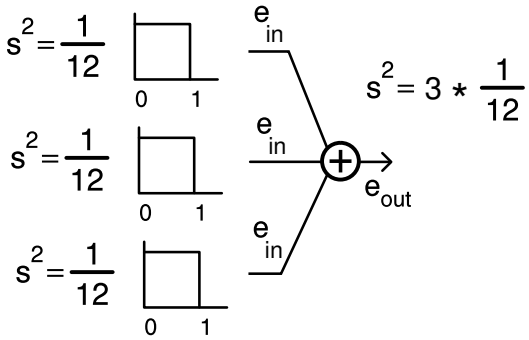


Bild 7: Varianz bei Generatoren mit Gleichverteilung

Das Rauschen hat hier aber noch DC-Offset: nächster Schritt ist die Umwandlung auf ein Signal mit Vorzeichen durch Subtraktion von 8000h.

Die Periode ist natürlich um 1/12 kürzer als die des verwendeten LCGs. Man kann aber 12 verschiedene LCGs parallel zu betreiben, was auch der Qualität förderlich sein dürfte.

Analog

Hier hat das Ausgangssignal bereits Gaußverteilung. Wesentlich ist, dass der Rauschpegel sinkt, wenn man Mittelwert aus mehreren Quellen bildet. Z.B. für rauschärmere Referenzspannungsquellen [3] (Bild 8) oder Verstärker [4] (Bild 9). Wobei hier die Eingangstransistoren durch direkte Verbindung an den Pins für Offsetabgleich parallelgeschaltet wurden. Die konventionellere Lösung ist große diskrete Transistoren vor einen einzelnen OP zu schalten.

Parallelschaltung von 4 Kanälen halbiert das Rauschen nominell

$$e_{\text{out}} = e_{\text{in}} \frac{1}{\sqrt{n}},$$

aber die Formel zeigt auch, dass eine weitere Erhöhung nur noch sehr wenig bringt.

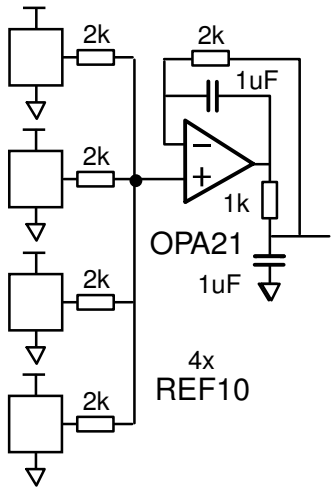


Bild 8: Referenzspannung

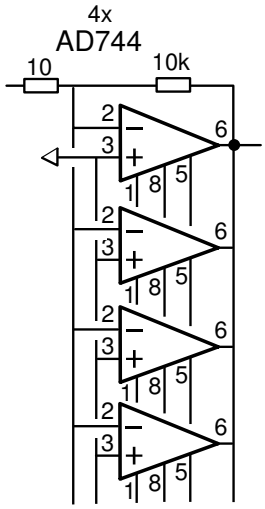


Bild 9: Verstärker

Bonmot

„Everybody believes in it, the experimenters because they think it is a mathematical theorem, the mathematicians because they think it is an experimental fact.“

Die Formeln sollen keine mathematische Exaktheit vortäuschen. Physikalisches Rauschen gehorcht ungern mathematischer Theorie und das Ringen der Mathematiker mit dem Grenzwertsatz dauert schon seit Jahrhunderten [5].

Literatur

[1] emb (11) Rauschgeneratoren
[2] emb (4) Linear kongruente Zufallszahlengeneratoren
[3] Stitt „Paralleled References Cut Noise“ EDN 8.3.1984
[4] Gerstenhaber, Murphy „Stacking Amplifiers Cuts Input Noise“ Electronic Design 5.12.1991
[5] Adams „The Life and Times of the Central Limit Theorem“ Kaedmon Publishing 1974

Lebenszeichen

Bericht aus der FIG Silicon Valley: *Henry Vinerts*

Greetings, gentlemen!

Vielen Dank für die neuen Veröffentlichungen, die ich gestern erhalten habe. Ich beglückwünsche alle, die zu diesen lobenswerten Anstrengungen beigetragen haben, vor allem Fred, dessen geduldige und fachmännische multi–linguale Übersetzungen mehr Seiten füllen, als ich in einer einzigen Sitzung lesen kann. Ich sehe mit Freuden, dass Erich Wälde wieder stark im Rennen liegt, dass die *Oldtimer*, wie Rafael, Michael und Anton, deren Namen mir bekannt sind, Beiträge geliefert haben und dass ein neuer Stern namens André Elgeti am Forth–Himmel erschienen ist.

Zum letzten SVFIG–Treffen traf ich etwas verspätet ein. Dr. Ting war gerade dabei, etwa 15 Forthler mit einer vorbereiteten DVD zu unterhalten, auf welcher Musik von J. S. Bach wiedergegeben wurde. Am Nachmittag berichteten drei Angestellte von *intellaSys* über den Stand der Dinge zu Chucks (Übers.: Charles Moore) neuem Multi–Core–Prozessor–Chip. Als Beispiele brachten sie Programmteile in e–Forth, die sie zu diesem Zweck entwickelt hatten. Nachlesen kann man das unter `http://www.intellasys.net`. Soweit ich gehört habe, wird die Firma bald nach weiteren Forth–Programmierern Ausschau halten. Chuck hat seinen neuen Chip schon mit einer Flasche Champagner begossen und arbeitet bereits am nächsten Chip.

Cogswell College, wo wir unsere SVFIG–Treffen abhalten, hat neue Sicherheitsmaßnahmen eingeführt, die den Zugang zum Gebäude während der Wochenenden unterbinden. Wir werden es dadurch nicht mehr ganz so leicht wie bisher haben, unsere Treffen zu beliebigen Zeiten abzuhalten. Es wurde darüber diskutiert, einen Ausweichversammlungsort zu suchen. Aber ein größeres Problem bleibt trotzdem: Wie es aussieht, wird es immer schwieriger, Leute anzuziehen, die fähig und bereit sind, unsere Treffen mit interessantem forth-bezogenen Inhalt zu füllen.

Ich hoffe, dass ich euch mit diesen Zeilen noch rechtzeitig erreiche, um euch allen eine schaffensfrohe und produktive Zeit auf eurer am Donnerstag beginnenden Forth–Tagung zu wünschen. Nochmals vielen Dank für die Hefte.

Henry Übersetzt von Fred Behringer

Antwort des Übersetzers: Danke, Henry. Ich habe auf der Mitgliederversammlung am Sonntag in meinem Bericht über die Auslandsarbeit auf deine immer sehr informative Mitarbeit und den ständigen E–Mail–Kontakt zwischen uns beiden hingewiesen. Meine Mitteilung wurde mit starkem Klopfen auf den Tisch (akademischer Brauch der Zustimmungsbekundung) quittiert. Danke — und weiter so!

Fred

Factor, Postscript, und Forth: Ein kleiner Vergleich

M. Anton Ertl

Auf der Forth-Tagung präsentierte Ulrich Hoffmann die Programmiersprache Factor von Slava Pestov u.a. (`http://factorcode.org`). Diese Programmiersprache ist in der Syntax recht ähnlich zu Forth, unterscheidet sich aber ansonsten in vielerlei Hinsicht: vor allem hat Factor eine dynamische Typüberprüfung, während Forth keine Typüberprüfung verwendet. Damit einher geht die automatische Speicherverwaltung (garbage collection) in Factor; weiters zeichnet sich Factor noch durch das Vorhandensein von Wörtern aus, die von funktionalen Programmiersprachen inspiriert sind. Es gibt noch weitere interessante Features von Factor, auf die ich in diesem Artikel aber nicht eingehe.

Ich fühlte mich natürlich gleich inspiriert, Factor mit einer älteren stack-basierten Programmiersprache mit dynamischer Typüberprüfung und Speicherverwaltung zu vergleichen: Postscript [Inc99]. Hier gibt es deutlichere Abweichungen in der Syntax, aber von der Semantik sind sich Factor und Postscript näher als Factor und Forth. Als weitere Sprache zum Vergleich wählten wir natürlich Forth.

Wir beschlossen, ein kleines Programm in allen drei Sprachen zu schreiben, um die Unterschiede und Gemeinsamkeiten der drei Sprachen klarer zu sehen; außerdem wollten wir dann noch die Laufzeit der Programme messen. Ich postete die Programme dann in `comp.lang.forth` und `comp.lang.postscript`, und erhielt dabei noch verbesserte Programme (siehe News-Artikel `<2007May3.220119@mips.complang.tuwien.ac.at>` ff.).

Das Programm soll ein Array mit den 1000 Elementen 0...999 (oder 1...1000) erzeugen, und dann die Elemente des Arrays 100.000–mal aufaddieren. Diese Aufgabe ist besonders dazu gedacht, um die funktionalen Elemente von Factor zu zeigen, daher sollte es dabei nicht wundern, wenn Factor besser ausschaut. Die Programme und englischer Text dazu sind auf `complang.tuwien`[1] zu finden.

Forth

Fangen wir zunächst mit einem Forth-Programm (Abb. 1) an, dann versteht man die anderen auch leichter.

Die Definition von `map-array` (aus dem Gforth-Tutorial [CEK+03][2]) erlaubt einen ähnlich funktionalen Programmierstil wie Factor. Bevor `map-array` aber verwendet wird, wird zuerst das Array allokiert und initialisiert, mit einem Forth-üblichen Programmierstil.

In `step` werden die Elemente des Arrays mithilfe von `map-array` aufaddiert: Und zwar wird das Array `array 1000` und das Execution Token (XT) von `+` an `map-array` übergeben; weiters steht noch 0 auf dem Stack. `Map-array` legt jetzt das erste Element des Arrays auf den Stack und führt das XT aus, addiert das Element also zum schon vorhandenen 0 dazu; in der nächsten Iteration legt es das nächste Element auf den Stack, führt nocheinmal `+` aus, und addiert so das zweite Element zur Zwischensumme. Am Ende steht die Summe aller Elemente auf dem Stack (und wird dann mit `drop` weggeworfen).

[1] `http://www.complang.tuwien.ac.at/forth/programs/map-array/`

[2] `http://www.complang.tuwien.ac.at/forth/gforth/Docs-html/Execution-Tokens-Tutorial.html`

```
: map-array ( ... addr u xt -- ... )
    \ executes xt ( ... x -- ... ) for every element of the array starting
    \ at addr and containing u elements
    { xt }
    cells over + swap ?do
        i @ xt execute
    1 cells +loop ;

create array 1000 cells allot
: init  1000 0 do i  array i cells + !  loop ;
init
: step  0 array 1000 ['] + map-array   drop ;
: bench  100000 0 do step loop ;
bench
```

Abbildung 1: Forth-Programm (Anton Ertl)

```
USING: syntax vectors namespaces math kernel sequences tools ;
: step 0 [ + ] reduce drop ; inline
: numbers 1000 [ 1+ ] map ; inline
: bench numbers 100000 [ dup step ] times drop ;
bench
```

Abbildung 2: Factor-Programm (Slava Pestov)

In `bench` wird `step` dann 100.000–mal ausgeführt.

Bemerkenswert bei `map-array` ist noch, dass es mit XTs mit verschiedenen Stack-Effekten umgehen kann, solange dabei immer ein Datenstack-Element verbraucht wird und ansonsten die Anzahl der Elemente gleich bleibt, z.B. kann man es mit `.` zum Drucken und mit `max` zum Finden des Maximums einsetzen.

Factor

Was beim Factor-Programm (Abb. 2) zunächst einmal auffällt, sind die eckigen Klammern. Sie schließen quotations ein, das sind Codestücke, die nicht sofort ausgeführt werden, sondern den nachfolgenden Wörtern übergeben werden, die sie dann mehrmals ausführen. Das ist ähnlich den XTs in Forth, aber etwas bequemer. Auf diese Weise wird Kontrollfluss in Factor implementiert.

In `numbers` wird unser Array erzeugt: `map` führt die quotation `[ 1+ ]` auf den Werten von 0 bis 999 aus, und sammelt das Ergebnis (die Werte 1...1000) in einem Array.

In `step` werden die Werte im Array aufaddiert, genauso wie im Forth-`step`, nur dass statt dem selbstdefinierten `map-array` das vordefinierte `reduce` verwendet wird, das sich von `map-array` durch den Stack-Effect `( array x1 quot - x2 )` unterscheidet; es ist eigentlich nur zur Verwendung auf quotations mit dem Stack-Effekt `( x1 y - x2 )` gedacht (kann aber auch flexibler eingesetzt werden).

Schließlich führt `bench` 100.000–mal `step` aus, wieder über eine Quotation und das Kontrollflusswort `times` gesteuert.

Die `inline`s helfen dem Compiler, schnellen Code zu produzieren, u.a. indem der Compiler Typüberprüfungen wegoptimieren kann.

Postscript

Auch bei der Postscript-Version (Abb. 3) gibt es Codestücke, die nicht sofort ausgeführt werden, diesmal in geschwungenen Klammern und Procedures genannt, aber eigentlich das gleiche wie Quotations. Dafür sind Arrays in eckigen Klammern, die geschwungenen und eckigen Klammern vertauschen also im Vergleich zu Factor die Rolle.

In Postscript gibt es im Prinzip nur ein Definitionswort: `def` nimmt einen Namen (z.B. `step`, als Literal so geschrieben `/step`) und ein Objekt (z.B. eine Prozedur) vom Stack, und bindet das Objekt an den Namen.

Die erste Definition definiert das Array `v`. Bemerkenswert ist hier, wie das Array aufgebaut wird: da wird nämlich zunächst die Marke `[` auf den Stack geschrieben, dann alle Werte des Arrays, und schließlich baut `]` das Array aus allen Elementen auf, die auf dem Stack über der obersten Marke liegen (und nimmt dabei diese Werte alle vom Stack).

Die 1000 Werte in dem Array werden von dem `for` produziert, das alle Werte von 0 bis 999 in 1er-Schritten erzeugt und dann der leeren Prozedur `{}` übergibt, die damit aber nichts macht, sodass die Werte alle auf dem Stack liegen bleiben.

Auf diese Weise lassen sich sehr flexibel Arrays erzeugen und transformieren, mit deutlich weniger verschiedenen Wörtern als in Factor.

Die Verwendung einer Schleife, bei der die Stack-Tiefe mit jeder Iteration zunimmt, ist in Forth unüblich und schlechter Stil, aber in Postscript ist das durchaus üblich, um Arrays aufzubauen. Das hängt auch damit zusammen, dass die dynamische Typinformation in Postscript es erlaubt, die Marke `[` von Werten zu unterscheiden, die als Elemente des Arrays gedacht sind, was in Forth im Allgemeinen nicht möglich ist.

Die Definition von `step` erfolgt wieder nach dem altbekannten Muster, nur dass hier `forall` statt `map-array` bzw. `reduce` verwendet wird.

Zwischen der Prozedur und `def` steht noch `bind`, was die Ausführung noch etwas beschleunigt, indem es dafür sorgt, dass bei eingebauten Operatoren wie `add` der Name gleich aufgelöst wird und nicht erst bei der Ausführung.

Schließlich erfolgt die 100.000–fache Ausführung mittels `repeat`, das Postscript-Äquivalent zum Factor-Wort `times`.

```
/v [ 0 1 999 {} for ] def
/step {0 v { add } forall} bind def
100000 {step pop} bind repeat
```

Abbildung 3: Postscript-Programm (Anton Ertl)

```
: (map) ( a n - a a')   cells over + swap ;
: map[   postpone (map)  postpone ?do postpone i postpone @ ; immediate
: ]map   1 cells postpone literal  postpone +loop ; immediate

create array 1000 cells allot
: init  1000 0 do i  array i cells + !  loop ;
init
: step  0  array 1000 map[ + ]map drop ;
: bench  100000 0 do step loop ;
bench
```

Abbildung 4: Forth-Programm (Andrew Haley)

Forth, die zweite

Andrew Haley wies darauf hin, dass die in Abb. 1 gezeigte Forth-Version nicht idiomatisches Forth ist, und präsentierte eine andere Lösung (Abb. 4).

Statt `map-array`, das ein XT nimmt und ausführt, gibt es hier zwei Wörter `map[` und `]map`, die als Macros eine Schleife um das `+` herum bauen, die pro Element einmal ausgeführt wird, wobei das Element zuerst noch auf den Stack gelegt wird.

Messungen

Die Programme wurden auf einem Dual-Xeon 5160 (3GHz) System unter Linux (Debian Etch) durchgeführt (alle Systeme für 64-bit, ausgenommen die Forth-Systeme (32-bit)), und dabei die Laufzeit (User Time) gemessen:

Zeit	Programm	System
0.64s	map-array.fs	Gforth 0.6.2 (gcc-2.95)
0.48s	map-array.fs	iForth 2.1.2541
0.96s	slava.factor	Factor 0.88
1.99s	bind.ps	ESP Ghostscript 815.03
0.51s	haley2.fs	Gforth 0.6.2 (gcc-2.95)
0.23s	haley2.fs	iForth 2.1.2541

Wie man sieht, kostet die dynamische Typüberprüfung und die anderen mehr oder weniger netten Features von Postscript und Factor schon einiges, aber der Abstand (vor allem bei Ghostscript) ist kleiner, als ich erwartet hätte.

Literatur

[CEK+03] Neal Crook, Anton Ertl, David Kuehling, Bernd Paysan, and Jens Wilke. *Gforth (Manual)*. Free Software Foundation, 0.6.2 edition, 2003.

[Inc99] Adobe Systems Incorporated. *PostScript Language — Reference Manual*. Addison-Wesley, third edition, 1999.

Named Bits

Text: Michael Kalus, Idee und Code: Matthias Trute

Die Handhabung von benannten Bits in Daten–Registern für Ports, Data–Direction–Registern und anderen Registern am Beispiel des ATmega169.

Zusammenfassung:

Mittels `CREATE` und `DOES>` werden auf einfache Weise soviele einzelne Bits benannt, wie man möchte. In den zugehörigen Registern können diese Bits dann einfach gesetzt oder zurückgesetzt werden. So lassen sich Pins in den INPUT– oder OUTPUT–Modus schalten oder andere Registeroperationen durchführen.

Summary:

We use `CREATE` and `DOES>` to give names to individual bits of registers. These may be used to toggle the bits, and for example switch INPUT and OUTPUT modus easily. Or do other operations on registers.

Motiviert durch häufige Versuche mit den Prozessoren Atmega8, Atmega32 und ATmega169 entstanden diese kleinen Test–Helfer. Der Leser mag selbst entscheiden, ob er so etwas auch in Applikationen einbauen möchte. Man kann solchen syntaktischen Code als Ballast sehen. Doch in Phasen intensiver Testung war es ausgesprochen nützlich. Außerdem erhöht sich die Lesbarkeit von Quellcodes damit, ein Kriterium, das nicht gering geschätzt werden sollte. Ich persönlich halte so etwas also nicht für *Ballast*.

Die Aufgabe

Immer wieder mussten Bits in den Daten–Registern der Ports Px und auch in den korrespondierenden Data Direction Registern DDRx des ATmaga169 gesetzt und wieder gelöscht werden, ohne dabei die anderen Bits des Registers zu verändern. Die Routinen dafür sind ja bekannt — Byte aus dem Register lesen, mit einer Bitmaske verknüpfen und das Byte zurückschreiben. Soll dabei das Bit gesetzt werden, wird der geholte Wert mit der Bit–Maske OR–verknüpft und das Ergebnis zurückgeschrieben. Und soll das Bit gelöscht werden, wird der Wert mit der invertierten Bit–Maske AND–verküpft. Bei sieben Ports zu je acht Bits ist es irgendwann einfach lästig, immer wieder nach zu sehen, wie die Adresse des Data–Registers für den Port lautet, und mit welcher Bitmaske denn das Data–Direction–Register gesetzt, oder der Port selbst beschrieben oder gelesen werden muss.

Die Lösung

Diese wiederkehrende Routineaufgabe hat die Registeradresse und die Pin–Nummer als Basis. Jedes einzelne Bit lässt sich durch diese beiden Werte charakterisieren. Und bei den ATmegas liegt das DDRx immer eine Adresse tiefer als sein Port. Was liegt also näher, als für die viel benutzten Bits einen eigenen *intelligenten* Namen zu vergeben? So ein Bit sollte sozusagen selber wissen, wer es ist. Es soll Auskunft geben können, zu welcher Adresse es gehört, und dazu auch mitteilen können, wo in der Bitfolge es sich befindet. Letzteres ist ja unter dem Begriff der Bitmaske bekannt. Im konkreten Anwendungsfall der ATmegas und des amforth passt diese Information gerade ganz gut zusammen in eine einzelne Zelle im internen Flash–Speicher. Die DRx und ihre DDRx liegen alle im Adress–Bereich von 0x20 bis 0x34, da genügt also ein Byte als Speicherplatz für die Adresse. Und die Bitmaske passt in das andere Byte einer Speicherzelle. Im Falle der Ports soll diese Maske allerdings Pin–Maske heißen.

So kann mit Hilfe der shift–Operation aus der Pin–Nummer eine Bit–Maske erzeugt werden, die dann zusammengepackt mit Adresse ins Flash geschrieben wird. Diese Speicherstelle bekommt einen Namen. Und wird dieser aufgerufen, legt er nun die Bit–Maske und die Adresse des Registers auf den Stack. Nun können Operatoren wie `ON` und `OFF` oder `IS_INPUT` und `IS_OUTPUT` gemacht werden. Diese benutzen die Informationen z. B., um einen Portpin an– oder aus zu schalten oder um die Datenrichtung des Pins im Port umzukehren.

Das Listing zeigt wie das gemacht werden kann. Hier noch ein kleiner Test, um die Helfer zu erproben:

```
hex 2B constant PD

PD 0 portpin: PD0
PD0 is_output
PD0 on
PD0 off
```

Hinweis

Die solcherart definierten Bits sind nun nicht auf die PORTx Adressen beschränkt. Sie gelten für alle mit 8 Bit erreichbaren Adressen, und das sind alle IO–Adressn (wers doll treiben will, kann im ATmega auch die CPU–Register damit manipulieren). Man kann also beispielsweise die diversen Steuerregister bis hin zum SREG damit lesbar ansprechen. Lediglich die Worte `is_input`/`is_output` setzen auf der Abhängigkeit DDRx = PORTx − 1 auf, sind aber ohnehin nur für diese sinnvoll.

Viel Vergnügen beim Ausprobieren.

Links:

amforth	http://amforth.sourceforge.net/
Datenblatt des ATmega169	http://www.atmel.com/dyn/resources/prod_documents/doc2514.pdf

```
1   \ Named Bits
2   \ V 1.1 20.05.2007
3
4   \ Code: Matthias Trute
5   \ Text: M.Kalus
6
7   \ A named bit puts a bitmask on stack, so we can operate on the bit that is set.
8   \ And it puts the address of its register on stack, too.
9   \ So we can use this method on data registers for port pins, data
10  \ direction registers for ports and also on other registers.
11
12  \ Example:
13  \ PD 7 portpin: PD7     ( define portD pin #7)
14  \ PD7 is_output         ( set DDRD so that portD pin #7 is output)
15  \ PD7 on                ( turn portD pin #7 on, i.e. set it high-level)
16  \ PD7 off               ( turn portD pin #7 off, i.e. set it low-level)
17
18  hex
19  \ for example:
20  2B constant PD    \ Port D in ATmega169
21  2A constant DDRD  \ Data direction register of Port D
22
23  \ At compiletime:
24  \ Store combination of data register address for a port
25  \ and bit number in one cell and give it a name.
26  \ At runtime:
27  \ Get bitmask and register address on stack.
28
29  \ Now use it for ports:
30  : portpin: create ( C: "ccc" portadr n -- ) ( R: -- pinmask portadr )
31      1 swap lshift
32      8 lshift or ,               \ packed value
33    does> i@                      \ get packed value
34      dup 8 rshift swap ff and    \
35  ;
36
37  \ Turn a port pin on, dont change the others.
38  : on ( pinmask portadr -- )
39      dup  ( -- pinmask portadr portadr )
40      c@   ( -- pinmask portadr value )
41      rot  ( -- portadr value pinmask )
42      or   ( -- portadr new-value)
43      swap ( -- new-value portadr)
44      c!
45  ;
46
```

Fortsetzung des Listings auf Seite 19

Der Forth-Stammbaum

M. Anton Ertl

In der Mitte des Hefts ist ein Stammbaum von Forth-Systemen und Forth-Standards zu sehen. Der Stammbaum enthält zwei Arten von Knoten:

Systeme (normale Schrift) Ausführbare Forth-Systeme.

Standards (fett gedruckt) Spezifikationen der Sprache Forth.

Der Stammbaum enthält drei Arten von Kanten:

Abstammen (schwarz voll) Ein System (das Kind) enthält beträchtliche Mengen Code eines anderen Systems (des Elternsystems), oder zumindest war das am Anfang der Entwicklung des Kindes so. Bei Standards wird Text oder zumindest Forth-Wörter vom Eltern-Standard übernommen.

Erfüllen (schwarz strichliert) Ein System erfüllt einen Standard; man kann auch sagen, das System implementiert den Standard.

Inspiriert (grau voll) Wörter oder Konzepte von einem Forth-System wurden in ein anderes Forth-System übernommen, ohne Code zu übernehmen. Nur die wichtigsten Inspirationskanten werden gezeigt, um eine weitere Überfrachtung der Graphik zu vermeiden.

Aber das Forth-System X fehlt!

Das liegt daran, dass mir noch niemand Daten über X geschickt hat. Nicht aufregen, einfach die Information senden (siehe unten).

Aktualisierung und Online-Version

Dieser Stammbaum ist nicht endgültig, sondern er wird erweitert, wenn neue Daten dazukommen. Die jeweils aktuelle Version findet man auf `http://www.complang.tuwien.ac.at/forth/family-tree/`. Dort findet sich neben dem Stammbaum auch noch eine Timeline: eine Variante des Stammbaums, auf der die Forth-Systeme und Forth-Standards nach dem Zeitpunkt des Erscheinens auf einer Zeitlinie angeordnet sind (die Timeline ist aber leider für dieses Heft zu breit und daher nicht abgedruckt). Die Online-Version hat außerdem auch noch Tooltips mit Zusatzinformationen (z.B. über Autoren oder Features) und Links auf Informationen zu vielen der Systeme oder ihrer Geschichte (einfach auf den Namen des Systems oder des Standards klicken).

Wie schicke ich Information ein?

Am einfachsten schickst Du mir die Information einfach als Textbeschreibung (aber schon so, dass mir klar ist, was Abstammung ist und was Inspiration etc.), und ich schreibe dann den entsprechenden Code. Meine Email-Adresse ist `anton@mips.complang.tuwien.ac.at`.

Wie verstehe ich `tree.fs`?

Die Quelldatei der beiden Graphen ist `tree.fs`. Falls Du sie verstehen oder selbst damit experimentieren willst:

Zunächst muss man die Knoten (Systeme/Standards) definieren, und zwar wie folgt:

```
<Jahr> s" <En>" name-implementation <Kn>
<Jahr> s" <En>" standard <Kn>
```

Das Jahr ist dabei das (erste) Erscheinungsjahr. `<En>` steht für den (Eigen-)Namen der Implementation oder des Standards, der in der Graphik angezeigt wird. `<Kn>` steht für den Knotennamen, der in den Kantendefinitionen verwendet wird.

Für den Eigennamen ist die Syntax sehr frei; dagegen darf ein Knotenname nur so ausschauen wie ein Name in C, er darf also nur aus Buchstaben, Ziffern, und _ bestehen, aber nicht mit einer Ziffer anfangen. Wenn der Eigenname eines Systems schon diesen Kriterien entspricht, kann man die Definition auch abkürzen:

```
<Jahr> implementation <Kn>
```

Beispiele:

```
2002 s" 4IM" name-implementation fourIM
1994 s" ANS Forth" standard Forth94
1996 Implementation Gforth
```

Man kann eine URL und eine Kurzinformation (für den Tooltip) hinzufügen, indem man unmittelbar *vor* der Knotendefinition Folgendes schreibt:

```
\U <URL>
\I <tooltip text>
```

Diese Informationen werden in der Online-Version benutzt. Übrigens braucht man eine URL, damit der Tooltip-Text angezeigt wird.

Wenn man die Knoten definiert hat, kann man sie auch verwenden, um Kanten zu definieren.

```
<Kn1> begot <Kn2>       \ Abstammen
<Kn2> conforms-to <Kn1> \ Erfüllen
<Kn1> inspired <Kn2>    \ inspiriert
```

Beispiel:

```
\U http://www.jwdt.com/~paysan/gforth.html
\I Free, portable and fast
1996 Implementation Gforth
Gforth conforms-to Forth94
BigForth begot Gforth
figForth inspired Gforth
F83 inspired Gforth
```

Weitere Informationen sind, wenn vorhanden, in Kommentaren untergebracht.

Wie funktioniert es?

Die Quelldatei ist `tree.fs`, ein Forth-Programm, das mittels Gforth ausgeführt wird und `tree.dot` ausgibt; Letzteres ist die Eingabe für `dot` (ein Programm aus der Graphviz-Suite zur graphischen Darstellung von gerichteten Graphen); `Dot` kann Graphiken für den Graphen in mehreren Formaten ausgeben, u.a. Postscript. Weiters gibt `dot` Y-Koordinaten für die Knoten aus, die dann in einem weiteren Lauf von `tree.fs` verwendet werden, um `timeline.neato` zu erzeugen; das ist im Prinzip die gleiche Datei wie `tree.dot`, nur sind die Knotenpositionen festgelegt: die X-Koordinate durch das Erscheinungsjahr, und die Y-Koordinate durch die Position im tree-Graphen. `Neato` (ein weiteres Graphviz-Werkzeug) verarbeitet dann `timeline.neato` und produziert die Timeline. Weiters erzeugen `dot` und `neato` noch andere Formate, insbesondere `.gif` und `.map` (client-side image map) für die Web-Version, und eine auf mehrere Seiten aufgeteilte Version der Timeline. Aus den Postscript-Versionen wird PDF erzeugt.

Heimwerken

Wenn Du selbst an dem Stammbaum herumbasteln oder einen eigenen Stammbaum erzeugen willst, kannst Du Dir den Quellcode von der oben angegebenen Adresse holen, `tree.fs` editieren und `make` aufrufen. An Werkzeugen brauchst Du GNU make, Gforth, graphviz, Ghostscript, awk und die netpbm-Werkzeuge, wenn man das aktuelle Makefile verwendet (weniger, wenn man nicht alle Formate erzeugen will).

Danksagung

Viele Leute haben mir Informationen geschickt, die in dem Stammbaum dargestellt sind. Ich habe leider nicht alle notiert, die dazu beigetragen haben, aber darunter sind: Astrobe, John Doty, Marcel Hendrix, Doug Hoffman, George Hubert, Guy Macon, Alex McDonald, Ward McFarland, Krishna Myneni, Stephen Pelc, Friederich Prinz, Elizabeth Rather und Klaus Schleisiek.

Eine weitere Quelle war der Artikel über die Geschichte von Forth [RCM96], der sehr empfehlenswert und online[1] verfügbar ist.

Literatur

[RCM96] Elizabeth D. Rather, Donald R. Colburn, and Charles H. Moore. The evolution of Forth. In *History of Programming Languages*, pages 625–658. ACM Press/Addison-Wesley, 1996.

[1] `http://www.forth.com/resources/evolution/`

Gehaltvolles

zusammengestellt und übertragen von *Fred Behringer*

VIJGEBLAADJE der HCC Forth–gebruikersgroep, Niederlande Nr. 62, Juni 2007

Ervaringen van de erste PiF workshop — Gijsbert Koolen

Die Autoren gehen unserer geliebten Sprache Forth nicht aus. Nicht uns, aber auch nicht den niederländischen Forth–Freunden. Es fühlen sich immer wieder welche angesprochen.

Der Autor geht auf die Ankündigung von Albert Nijhof aus dem Vijgeblaadje 61 zur Einrichtung eines Workshops über „**P**rogrammieren **i**n **F**orth“ anstelle von „Schwatzen (**p**raten) über (**o**ver) **F**orth“ ein. Er bezeichnet sich als *Beinahe-Beginner* (ziemlicher Anfänger) und berichtet (in gut lesbarem Sprachfluss) über den Verlauf des Workshops, auf welchem eine gemeinsame Lösung der im Vijgeblaadje 61 vom *Schoolmeester* Albert Nijhof gestellten Aufgabe herausgearbeitet wurde: Eine `BEGIN...WHILE...REPEAT`–Schleife. Und er berichtet über die Diskussionen, die sich über jedes einzelne der verwendeten Worte ergaben.

IF & WHILE — Albert Nijhof

Albert gibt wieder so'ne Art Rätsel auf: Er formuliert 36 bekannte Sprichwörter und Redensarten unter Verwendung von `IF...ELSE...THEN` und `BEGIN...WHILE...REPEAT` und fragt, welche Formulierungen richtig sind und welche falsch. Beispiel 29: Regnet es im Mai? `IF` April ist vorbei `THEN` . Beispiel 33: Zu spät? `IF` besser `ELSE` gar nicht `THEN` .

Forth voor dummies? — Frans van der Markt

Een goeroe in Forth
schreef veel op het bord
dat ie, als het vol was, weer wiste,
was wel eens verbolgen
en niet meer te volgen,
want kennis, broodnodig,...die miste!

Wieder ein schöner Limerick (DUDEN: Fünfzeiliges Gedicht grotesk-komischen Inhalts), diesmal nicht von Albert, sondern vom neuen Redakteur des Vijgeblaadjes. Leider arbeitet euer Rezensent momentan unter Zeitdruck und muss eine brauchbare Übersetzung (Nachdichtung) schuldig bleiben. Aber Martin Bitter und andere sind ja auch des Holländischen mächtig und können vielleicht einspringen?

Frans gibt einen weiterführenden Hinweis auf `http://www.precies160.nl/`

Fortsezung des Listings von Seite 14

```
47  \ Turn a port pin off, dont change the others.
48  : off ( pinmask portadr -- )
49      dup  ( -- pinmask portadr portadr )
50      c@   ( -- pinmask portadr value )
51      rot  ( -- portadr value pinmask )
52      invert and ( -- portadr new-value)
53      swap ( -- new-value port)
54      c!
55  ;
56
57  \ Only for PORTx bits,
58  \ because address of DDRx is one less than address of PORTx. (ATmega169)
59
60  \ Set DDRx so its corresponding pin is output.
61  : is_output ( pinmask portadr -- )
62      1- on
63  ;
64
65  \ Set DDRx so its corresponding pin is input.
66  : is_input  ( pinmask portadr -- )
67      1- off
68  ;
69
70  \ finis
```

Bericht von der Forth-Tagung 2007 in Wien

Ulrich Hoffmann

Auf der Forth-Tagung 2007 in Baden bei Wien

Ende April fand die Forth–Tagung 2007 statt. Erstmalig befand sich in diesem Jahr der Tagungsort außerhalb Deutschlands, nämlich auf der Krainerhütte in Baden bei Wien. Die Mehrheit der Tagungsteilnehmer der vorjährigen Tagung in Bommerholz hatte sich dafür ausgesprochen, nach Österreich zu gehen — die Forth–Gesellschaft verteht sich ja auch als Interessenverband für den deutschsprachigen Raum — und so übernahm Anton Ertl dankenswerterweise die Tagungsorganisation. Der obligatorische *Vierte Tag* am Donnerstag vor der Tagung diente vielen schon für die Anreise. Ursprünglich sollte er mit Wandern in den Bergen rund um Baden verbracht werden. Durch einen Buchungsfehler jedoch standen keine Übernachtungsmöglichkeiten mehr in der Krainerhütte zur Verfügung und so wurden wir flugs in einem recht exklusiven Hotel nahe der Wiener Innenstadt untergebracht. Wer mit dem Flugzeug oder dem Nachtzug anreiste hatte so einen vollen Tag in der fantastischen Stadt Wien zur Verfügung. Das offizielle Programm des *Vierten Tags* begann dann am Donnertagabend mit dem Abendessen in einem urtypischen Wiener Gaststätten–Keller zwei Stockwerke tief unter der Erde. Nach einer ausgedehnten Exkursion am Freitagmorgen zu Fuß durch die Wiener Innenstadt fuhren wir also auf die Krainerhütte, wo die eigentliche Forth–Tagung am Freitagnachmittag begann.

Die Krainerhütte, als Tagungshotel konzipiert, überzeugte mit guter technischer Ausstattung, hervorragender Küche und einer schönen Anlage (Stichwort Erlebnisgarten). Ideale Voraussetzungen für eine gelungene Tagung. Im Vortragsprogramm sprach Bernd Paysan über *Gforth auf dem NXT*. Gerd Franzkowiak erläuterte uns seine Ideen, *Forth als Roboter–OS* einzusetzen und Anton Ertl berichtete über *Ein portables Forth–C Interface*. Mit welchen Problemen man zu rechnen hat, wenn man Forth auf IBM 370 einsetzen möchte, konnten uns Luca Masini in seinem Vortrag *pforth for MVS (porting pforth to an EBCDIC platform)* nahebringen. Über die neuesten Entwicklungen des *OpenBIOS*–Projektes berichtete uns Carsten Strotmann. Dr. Willi Stricker erläuterte uns seine Herangehensweise, interaktiv eingebettete Systeme zu entwickeln und die *Anbindung eines Forth–Targets an einen Forth–Host* vorzunehmen. Ich selbst habe über *Stack–basierte High–Level–Programmiersprachen* wie Factor und Joy gesprochen.[1] In den abendlichen Workshops vertieften wir die über den Tag aufgeworfenen Fragestellungen (Vergleich Factor/Postscript/Forth, amForth auf dem AVR–Butterfly, usw.) und diskutierten wie üblich bis in den frühen Morgen über Gott und die Welt. Der Ausflug am Samstagnachmittag führte uns zur beeindruckenden Seegrotte nach Hinterbrühl und zum schönen Zisterzienserstift Heiligenkreuz.

Der Sonntagmorgen gehörte dann ganz der Mitgliederversammlung, über deren Verlauf an anderer Stelle in diesem Heft berichtet wird. Für 2008 ist Ulm als Tagungsort ins Auge gefasst.

Insgesamt war die Tagung in Wien eine sehr gelungene Veranstaltung, bei der Vortragsprogramm und Workshops noch genügend Raum zum freien Meinungsaustausch ließen. Das Rahmenprogramm setzte dem ganzen noch das Sahnehäubchen auf. Ich freue mich wirklich, dabei gewesen zu sein.

[1] Die Vortragsfolien finden sich unter `http://www.complang.tuwien.ac.at/anton/forth-tagung07/vortraege/`.

Geburtstage

André Elgeti

Zusammenfassung

Dies soll einmal darstellen, wie eine Lösung in Forth aussehen kann. Ich habe bewusst das Volksforth genutzt, weil es leicht in andere Systeme umgeschrieben werden kann und gewissermaßen eine Basis bildet. Der Artikel ist an den Newcomer–Anfänger gerichtet, und ist deshalb etwas ausführlicher ausgefallen. Ich habe versucht die Gedanken aus dem Buch *Thinking Forth* einfließen zu lassen. Ich hoffe, es ist mir gelungen. Bewusst habe ich auf jeden Vergleich zu anderen Programmiersprachen verzichtet.

Aufgabe

Jeder von uns stand sicher schon einmal vor der Situation, einen Geburtstag vergessen zu haben. Dieses Programm soll nun dabei helfen, dass das nie wieder passieren wird.

1. Interface

Das Wort `Geburtstage` soll anzeigen:

```
Onkel Willi noch 100 Tage,
Tante Laura noch 132 Tage
```

⋮

2. Daten

Da die Anzahl unserer Freunde relativ konstant ist, können wir ihre Daten in einen Screen einlesen. Die Daten werden durch den Spitznamen aufgerufen. z. B.

```
variable willi ," 12.5.1954"
               ," Onkel Willi"
               ," Wilhelm Lange"
               ," Am Brotsteig 5"
               ," 18200 Bratstett"
```

Aus den Daten eine verkettete Liste zu erzeugen, ist ziemlich einfach. Der Aufruf einer Variablen übergibt deren Adresse auf den Stack. Aufgrund der Segmentierung hat sie ebenfalls eine Länge von 16Bit und kann somit auch in einer Variable abgelegt werden.

Wenn beispielsweise `norbert` auf `willi` folgt, kann man eingeben.

```
norbert willi !
```

Der Aufruf von

```
willi @
```

legt dann die Adresse des nächsten Datensatzes auf den Stack.

Die Worte `Geburtstag` und `Anschrift` müssen an die im Wörterbuch gespeicherten Daten zugreifen können. Dazu gibt es das Wort Daten:

```
: daten ( anz addr - addr länge)
    2 + swap
    begin dup 0> while
       1 -  swap  dup c@ + 1 +  swap
    repeat drop ;
```

3. Worte

3.1. Kernfunktion

Die Kernfunktion soll aus zwei Daten, dem aktuellen Datum und dem Geburtsdatum, die Differenz erzeugen.

Nun haben wir die Herausforderung, dass diese Funktion in einigen Programmiersprachen hinzugeladen werden kann.

Die Anlage der Aufgabe zeigt, dass sich das Ergebnis zwischen –365 und 365 bewegen kann, je nach Werten.

Die Aufgabe vereinfacht sich also dahin, die Tagesnummer im Jahr festzustellen.

Vereinfacht könnte man sagen:

Tagesnummer =
$31 * \text{Monatsnummer}(1...12) + \text{Tageszahl}(1...31) - 31$

Was einen Wertebereich von 1 für den 1. 1. bis 372 für den 31. 12. ergäbe.

Das ist nun nicht günstig, deshalb wird das anders gelöst.

Es wird ein Feld angelegt, dessen Inhalte die Nummern der 1. der Monate zeigen.

```
Create monate
      1,   32 ,   60,  91 , 121 , 152 ,
    182 , 213 , 243 , 274 , 304 , 335 ,
```

`Create` legt den Eintrag `Monate` auf dem Wörterbuch an, und mit den Kommas werden dahinter die Zahlen für die Monatsanfänge gespeichert.

Das Wort `Tagesnummer` (tag monat) würde dann so funktionieren:

1. hole von create die Nummer des Monatsersten

   ```
   monate + @
   ```

2. Füge die Tage hinzu und ziehe 1 ab

   ```
   + 1 -
   ```

Das Wort heißt also

```
: Tagesnummer ( tag monat -- tagesnummer)
   monate + @ + 1 - ;
```

Nun benötigen wir noch ein Wort, womit wir aus dem Geburtsdatum die beiden Zahlen herausschneiden können und als solche auf den Stack legen können.

In Forth werden die Zeichenketten abgespeichert:

```
08 33 31 2E 30 35 2E 36 37
```

was dem Geburtsdatum `31.05.67` entspricht. Die `08` ist das *Counter-Byte* und gibt die Länge der Zeichenkette an.

Beim Aufruf der Variablen wird die Adresse des *Counter-Bytes* übergeben.

Das Ziel ist es jetzt, die Zeichenkette `33 31` auszuschneiden und in `1F` für 31 umzuwandeln.

Für den zweiten Teil werden die Zahlen als Zeichen auf den Stack gelegt und eine Funktion erzeugt daraus eine Zahl:

Beim Zehner die oberen vier Bit abschneiden: `15 and`
Die Zahl verzehnfachen: `10 *`
Den Einer hervorholen: `|swap|`
Beim Einer die oberen vier Bit abschneiden: `15 and`
Zusammenaddieren: `+`

```
: Zahlbilden( char1 char10 --- zahl)
    15 and 10 * swap 15 and + ;
```

Dieses Wort kann nach dem Horner-Schema erweitert werden, genügt hier aber so vollauf.

```
: Tagesdatum ( addresse --- Tagesnummer)
    dup 1+ c@
    swap dup 2 + c@
    rot zahlbilden
    swap dup 4 + c@
    swap 5 + c@
    zahlbilden
    tagesnummer ;
```

Das Systemdatum wird mit `date@ (--- d m y)` auf dem Stack abgelegt

```
: Tagesdifferenz(adresse --- Differenz)
    Tagesdatum Date@ drop ( Jahreszahl löschen)
    Tagesnummer -         (Differenz bilden)
;
```

	TOS	2nd	3rd
Adr			
Dup	Adr	Adr	
1 + c@	Char1	Adr	
Swap	Adr	Char1	
Dup	Adr	Adr	Char1
2 + c@	Char2	Adr	Char1
Rot	Char1	Char2	Adr
Zahlb.	Zahl1	Adr	
Swap	Adr	Zahl1	
Dup	Adr	Adr	Zahl1
4 + c@	Char3	Adr	Zahl1
Swap	Adr	Char3	Zahl1
5 + c@	Char4	Char3	Zahl1
swap	Char3	Char4	Zahl1
Zahlb	Zahl2	Zahl1	

Tabelle 1: Stackdiagramm zu `Tagesdatum`

Nun fassen wir dies in einer Funktion zusammen, die uns angibt, wie lange es noch Zeit ist bis zum Geburtstag der betreffenden Person:

```
: Geburtstag ( Addresse --- )
   dup 2 swap daten count type
   space ." noch"
   1 daten tagesdifferenz space . ." Tage" cr ;
```

Die Eingabe

```
Willi Geburtstag
```

ergibt dann

```
Onkel Willi noch 126  Tage.
```

Daraus ergibt sich dann

```
: Geburtstage
   willi        ( 1. Eintrag)
   begin        ( Schleifenbeginn)
    dup         ( Adresse doppeln)
    geburtstag  ( Ausgabe )
    @ dup =0    ( ist die nächste Adresse NULL ?)
   until        ( wenn nicht, neue Schleife)
;
```

Zusammenfassung:

Das System zeigt nun die Zeitdifferenzen zu den Geburtstagen an, allerdings nur in einer Richtung. Als Nächstes könnte die Anzeige der Adressen und das Schreiben der Glückwunschkarten erfolgen.

Links:

Thinking Forth, Leo Brodie, `http://thinking-forth.sourceforge.net/`

Listing

```
1   variable willi              ( Personendaten )
2       ," 12.5.1954"
3       ," Onkel Willi"
4       ," Wilhelm Lange"
5       ," Am Brotsteig 5"
6       ," 18200 Bratstett"
7
8   : daten ( anz addr --- addr )
9       2 +                     ( Variable überspringen)
10      swap
11      begin dup 0>            ( Anzahl >0? )
12          while               ( wenn ja )
13               1 -            ( Anzahl = Anzahl -1 )
14               swap dup c@    ( Länge des aktuellen Strings holen )
15               + 1 + swap     ( Auf nächstes Zählbyte setzen )
16      repeat
17      drop ;                  ( Anzahl löschen )
18
19  create monate 1 , 32 , 60, 91 , 121 , 152 , 182 , 213 , 243 , 274 , 304 ,
20  335 ,
21
22  : tagesnummer ( tag monat --- tagesnummer)
23      1 -                     ( Korrektur, weil mit 0 beginnend )
24      2 * monate + @          ( Tag des Monatsersten holen )
25      +                       ( Summe bilden )
26      1 - ;                   ( Korrektur, weil mit 0 beginnend )
27
28  : zahlbilden( char1 char10 --- zahl)
29      15 and                  ( 1. Zahl holen )
30      10 *                    ( Zehner bilden )
31      swap
32      15 and                  ( 2. Zahl holen )
33      + ;
34
35  : tagesdatum ( addresse --- tagesnummer)
36      dup 1+ c@               ( Zehner der Tage )
37      swap
38      dup 2 + c@              ( Einer der Tage )
39      rot
40      zahlbilden              ( Tage bilden)
41      swap
42      dup 4 + c@              ( Zehner der Monate )
43      swap
44      5 + c@                  ( Einer der Monate )
45      swap
46      zahlbilden              ( Monat bilden )
47      tagesnummer ;
48
49  : tagesdifferenz ( adresse --- differenz)
50      Tagesdatum
51      Date@                   ( Systemdatum holen )
52      drop                    ( Jahreszahl löschen )
53      Tagesnummer
54      -   ;                   ( Differenz bilden )
55
56
57  : geburtstag ( addresse --- )
58      dup
59      2 swap daten            ( Spitznamen holen )
```

```
60      count type              ( Spitznamen ausgeben )
61      space ." noch"
62      1 daten                 ( Geburtsdatum holen )
63      tagesdifferenz          ( Tagesdifferenz ausrechnen )
64      space ." Tage" cr ;
65
66  : geburtstage( --- )
67      willi                   ( 1. Eintrag)
68          begin               ( Schleifenbeginn )
69              dup             ( Adresse doppeln )
70              geburtstag      ( Ausgabe )
71              @ dup 0=        ( ist die nächste Adresse NULL ?)
72          until ;             ( wenn nicht, neue Schleife )
73
74
```

Protokoll der Mitgliederversammlung der Forth–Gesellschaft e. V. vom 29. April 2007

Gerd Franzkowiak

Tagesordnung

1. Begrüßung
2. Bericht des Direktoriums und Kassenbericht
3. *Vierte Dimension*, Organ der Forth–Gesellschaft
4. Entlastung des Direktoriums
5. Wahl des neuen Direktoriums
6. Vorstellung neuer Projekte
7. Verschiedenes

1. Eröffnung der Mitgliederversammlung

9:07 Uhr

- Begrüßung und Eröffnung der Tagung durch Direktor Ulrich Hoffmann
- Festlegung des Versammlungsleiters: Martin Bitter
- 22 Mitglieder (von 128) anwesend
- Verleihung des diesjährigen SWAP an: Anton Ertl
- ergänzende Punkte zur Tagesordnung:
  - Vorschlag zum Tagungsort 2008
  - Fortgeschrittenenschulung FORTH
  - Werbung EuroFORTH 2007

2. Bericht des Direktoriums/Kassenbericht

- Vermögensbericht etc. durch: Rolf Schöne
  Mitgliederentwicklung: derzeit 128 Mitglieder (2006: -7, +8; 2007: -1, +1). (ed.)
- Bericht des Kassenprüfers: Egmont Woitzel
  die Ausgaben des Vereins sind alle belegbar.
  die Mehrwertsteuer wurde nicht korrekt verrechnet.
  Empfehlung: Korrektur gegenüber dem Finanzamt innerhalb der nächsten 4 Wochen, Jahrestagungen sollten finanztechnisch ausgeglichen sein.

3. *Vierte Dimension*

- Ulrich Hoffmann spricht zur Gestaltung und Entstehung des Sonderheftes
  Dank an Fred Behringer für die tatkräftige Unterstützung bei der Erstellung
  Bitte um mehr Aktivitäten/Mitarbeit in Form von Leserbriefen, Artikeln u. ä. gute Auswirkung der VD–Serie und der PDF–Files auf der Website
  Satzsystem LaTeX empfohlen für Artikel
  Aufarbeitung alter VD–Hefte und Konvertierung in PDF–Format
  Dank an Ulrich Hoffmann für die geleistete Arbeit zur Aufarbeitung alter VD–Hefte in elektronischer Form
- Internet-Präsenz
  es muss Einfluss darauf genommen werden, dass über Google-Suche die Forth-ev.de-Seite problemlos gefunden wird Google-Suche
  ist durch redaktionellen Eintrag beeinflusst und verhindert Ergebnis „FORTH-ev“
- Auslandsarbeit
  - USA: Forth Interest Group existiert in der öffentlichen Präsenz eigentlich nicht mehr
  - Niederlande : beste Kontakte!
  - England: keinen Redakteur und damit keine Zeitschrift
  - Italien: es liegen keine Informationen vor

- Arbeitsbericht von Bernd Paysan
  Kontakte auf dem Linux-Tag 2006 in Wiesbaden geknüpft
  Linux–Tag in Berlin wird in diesem Jahr nicht mit einem Stand besetzt 1x aussetzen
  Projekt der FORTH–Gesellschaft: Lego NXT
  große Resonanz auf das elektronische Buch *Thinking FORTH* (über 25.000 Downloads)

10:30 – 10:45 Uhr — Pause

4. Entlastung des Direktoriums

- Antrag von Heinz Schnitter und Egmont Woitzel „Entlastung mit Auflage zur Korrektur des Abweichens der Umsatzsteuer für die Jahre 2003–2006 und Klärung aller Anfragen des Finanzamtes“
  Formulierung einstimmig angenommen
- einstimmige Entlastung des Direktoriums

5. Wahl des neuen Direktoriums

- Antrag Michael Kalus: Bestätigung des gegenwärtigen Direktoriums
  einstimmige Annahme des vereinfachten Verfahrens
- Abstimmungsergebnis zum neuen Direktorium

Zustimmung	20
Gegenstimmen	0
Enthaltungen	2

6. Projekte

- R8C
  Anfrage Martin Bitter : „Wo bekommt man technische Infos“
  hier Verweis auf Datenblatt
  Projekt ist abgeschlossen.
- NXT
  FORTH lauffähig bekommen
  Ansteuerung der Hardware
  Ansprechpartner: Bernd Paysan
- amFORTH
  Angebot zur Mitarbeit
  Ziel: Haus-Automatisierung
- VolksFORTH
  Entwicklung für Prozessor 65256

Diskussion über neue Projekt–Tendenzen

- FORTH soll gegenüber Massenartikeln höhere Ingenieur–Leistung bieten
- FORTH auf Handys z. B. für Bar–Code–Erkennung
- FORTH für Nutzung von Komponententechnologien

7. Verschiedenes

- Tagungsort 2008
  Ulm/Roggenburg
  Terminvorschlag: 25.4.2008
- FORTH für Einsteiger soll als Vorstellung und Lehrgang im terminlichen Zusammenhang mit der FORTH–Tagung angeboten werden.
- EuroFORTH 14.–16.09.2007 Saarland Schloss Dagstuhl
  Klaus Schleisiek als Organisator
- Gestaltung der Web–Präsenz der FORTH–Gesellschaft auf http://www.forth-ev.de/
  Über–/Bearbeitung der Wiki–Gestaltung

12:00 Uhr — Ende der Mitgliederversammlung

Vermögen	31.12.2006 € (gerundet)
Bargeld und Bankkonten	
Girokonto	2.888
GESAMT Bargeld und Bankkonten	2.888
Andere Vermögen	
Drucker VD Redaktion	0
Notebook Forth-Büro	365
Postbank Business Festgeld	17.052
GESAMT Andere Vermögen	17.417
GESAMT VERMÖGEN	20.305
VERBINDLICHKEITEN	
Andere Verbindlichkeiten	
Mehrwertsteuerkonto	-4.592
GESAMT Andere Verbindlichkeiten	-4.592
GESAMT VERBINDLICHKEITEN	-4.592
GESAMTSUMME	24.896

Übersicht über das Vereinsvermögen

	08.05.06	24.04.07	Differenz
PB Festgeld	16.806,52	17.174,26	367,74
PB Girokonto	8.139,03	7.210,84	-928,19
Summe	24.945,55	24.385,10	-560,45

Vergleich der Kontostände von 2006 und 2007

T4 — ein Embedded–Forth–Experiment

Jörg Völker

Seit nunmehr fast 6 Jahren bringt die tematik GmbH unter dem Markennamen *Servonaut* (`www.servonaut.de`) Module für den Truckmodellbau bzw. Funktionsmodellbau im Maßstab 1:16 bis 1:8 auf den Markt. Die Fahrtregler, Lichtanlagen und Truck–Soundmodule sind bis auf wenige Ausnahmen in Forth programmiert.

Unser erstes Modellbau–Produkt, eine Kombination aus Fahrtregler, Lichtanlage und einem kleinen Bordcomputer, wurde 2001 noch auf Basis eines 68HC11 und unter Holon11 entwickelt. Nach dem Erfolg dieses ersten Moduls musste dann allerdings leider gleich ein Wechsel der Prozessorfamilie stattfinden, denn der 68HC11 wird von Freescale (ehemals Motorola) nicht weitergeführt und ist für einfache Anwendungen zu teuer. Schade — Holon11 hatte sich bestens bewährt.

Die Wahl fiel auf die kleinere 68HC08–Familie. Zwar ist dieser Prozessor etwas einfacher aufgebaut als ein 68HC11, dafür sind zahllose auch sehr kleine Varianten mit Flash–Speicher und leistungsstarker Peripherie zu fairen Preisen verfügbar. Alle Typen sind zudem in–circuit–programmierbar, und durch die höheren Taktfrequenzen im Vergleich zum 68HC11 wird der weniger leistungsfähige Befehlssatz mehr als kompensiert.

Für den 68HC08 konnten wir allerdings weit und breit nur ein Forth–System finden, und zwar das nanoFORTH von Rafael Deliano. Da diese Forth–Variante auf dem Zielsystem selbst läuft, benötigt sie jedoch für unsere Zwecke zu viel Speicher und ist auf Chips mit weniger als 24kByte nicht sinnvoll einzusetzen.

In dieser Not und unter Zeitdruck wurde deshalb zunächst ein Zwischenschritt unternommen und lediglich ein Forth–Kern und ein innerer Interpreter für Direct–Threaded–Code erstellt. Die Programmierung erfolgte dann jedoch weiterhin in Assembler, die Forth–Programmteile wurden in Form von Konstanten–Tabellen eingegeben — ein recht mühsames und fehlerträchtiges Unterfangen. Immerhin konnten so bewährte Programmteile von Holon11 übernommen werden.

Auf die Dauer konnte dies kein befriedigender Zustand sein. Aber vor dem letzten Schritt hin zu einem eigenen kompletten Forth–System wurde lange gezögert. Da ich noch nie ein vollständiges Forth implementiert hatte, war der Aufwand schwer abzuschätzen. Es geht dabei schließlich um kommerzielle Produktentwicklung. Zeit ist hier nun einmal Geld. Ich hatte offen gesagt große Sorge, mich zu verzetteln. Nach jahrelanger Software–Entwicklung für Industrie–Waagen in C stand ein Umstieg auf ein kommerzielles C für den 68HC08 als Alternative allerdings auch nie zur Debatte.

Abbildung 1: Mit T4 programmierter Modellbau-Truck

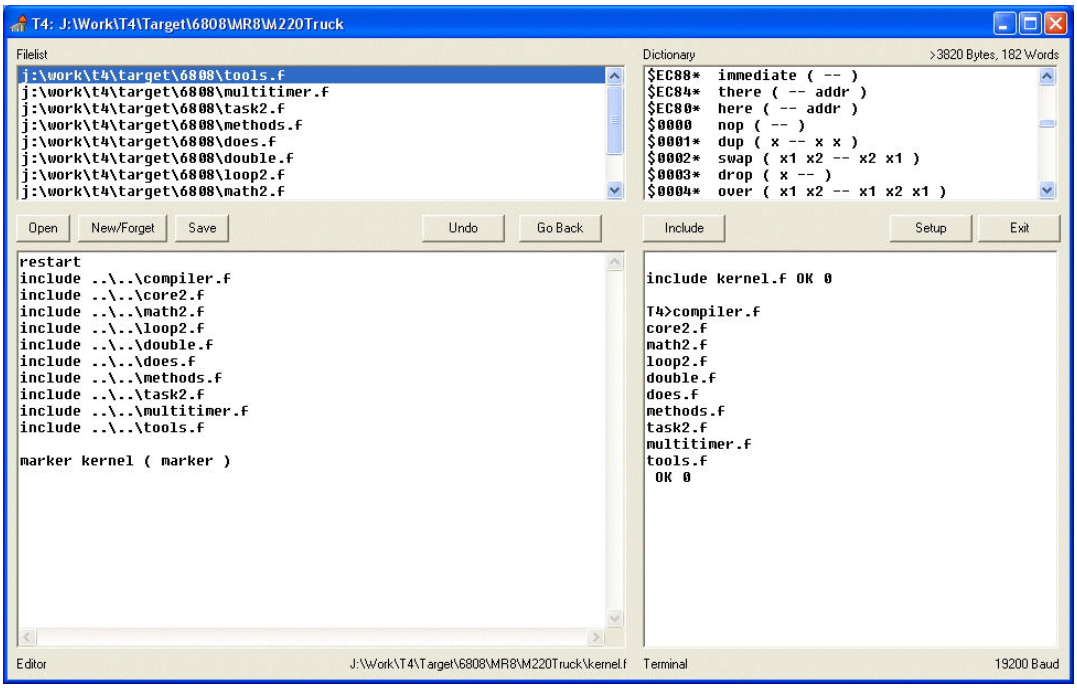


Abbildung 2: T4-Oberfläche orientiert sich entfernt an Holon11.

Wie kann man nun aber bei der Implementierung von Forth das Kosten/Nutzen–Verhältnis optimieren, wie kommt man mit wenig Aufwand zu einem einigermaßen komfortablen System?

Alle Vorgehensweisen haben ihre Vor– und Nachteile. Wie man ein Stand–Alone–Forth auf einem Zielsystem erstellt ist relativ gut dokumentiert, und es gibt viele Vorlagen, an denen man sich orientieren kann. Außerdem entspricht das Ergebnis noch am ehesten einem klassischen Forth. Leider ist der Speicherbedarf hoch. Auch der Code, der nur zur Übersetzungszeit benötigt wird, (und das Dictionary) belegt Speicherplatz im Zielsystem. Der Komfort bei der Entwicklung hält sich in Grenzen, da ein Embedded System oft keinen Massenspeicher besitzt und nur über das Nadelöhr einer seriellen Schnittstelle und einen PC auf Quellen zugegriffen werden kann.

Ein typischer Cross–Compiler dagegen, der vollständig auf einem PC als Entwicklungsrechner läuft, könnte einigen Luxus heute üblicher integrierter Oberflächen bieten. Nur eben zwei entscheidende Forth–Eigenarten nicht: Die Erweiterbarkeit der Sprache und das interaktive Entwickeln und Debuggen werden nicht unterstützt.

Bleiben noch die *großen* Lösungen, darunter die kommerziellen Systeme wie SwiftX von Forth,Inc., VFX Forth von MPE, und in gewisser Weise auch Holon11. Hier wird ein Cross–Compiler mit einem Forth auf dem Entwicklungsrechner kombiniert, und ein Interface zum Zielsystem erlaubt zudem interaktives Arbeiten. Optimal — nur leider nicht gerade an einem Wochenende implementiert. Allein die Notwendigkeit, mehrere Adressräume im PC zu verwalten, erhöht die Komplexität wesentlich, und fast alle Forth–Worte müssen zudem in zwei Versionen vorliegen, für den Host und für den jeweiligen Targetprozessor.

Die ersten beiden Ansätze kommen auf Grund ihrer Einschränkungen für mich nicht in Frage, die dritte Variante ist zu aufwändig — Was also tun? Not macht zum Glück erfinderisch — so auch in diesem Fall.

Schaut man sich einmal die Quellen für ein Stand–Alone–Forth, z. B. ein altes FIG–Forth genauer an, dann fällt auf, das die längsten Worte, die gleichzeitig am meisten Speicher belegen und beim Übersetzen den größten Teil der Rechenzeit benötigen, natürlich mit dem interaktiven Interpreter, der Compilierung und dem Dictionary zu tun haben: Accept, Parse, Find, Create usw. Dazu kommt noch die Zahlenkonvertierung mit Number, <#, #, #>, usw. Diese Worte werden in einem Embedded–System (typischerweise) wenn überhaupt nur in der Entwicklungsphase benötigt. Die Grundidee besteht nun darin, nur diese Worte zusammen mit dem Dictionary auf einen Entwicklungsrecher auszulagern — den eigentlichen Interpreter– und Compilerkern aber im Zielsystem zu belassen. Die Übersetzung wird also durch das Zielsystem gesteuert, der Entwicklungsrechner ist nur ein Slave, eine Art intelligentes Terminal mit Forth–spezifischen Zusatzfunktionen. Dieser Ansatz ist ein Kompromiss. Es werden zwar nicht alle nur für die Übersetzung erforderlichen Worte aus dem Zielsystem entfernt, aber der weitaus überwiegende Teil. Der Charakter eines traditionellen interaktiven Forth–Systems bleibt vollständig erhalten. Die Komplexität ist dabei nicht nennenswert höher wie bei einem Stand–Alone Forth. Da der PC sowohl das Dictionary als auch alle Quellen verwalten kann, sind komfortable Hilfe– und Suchfunktionen einfach umzusetzen.

Das Projekt wurde nach einigem Zögern dann im Sommer 2005 unter dem Namen *T4* in Angriff genommen. Für die PC–Software fiel die Wahl auf Delphi, einfach

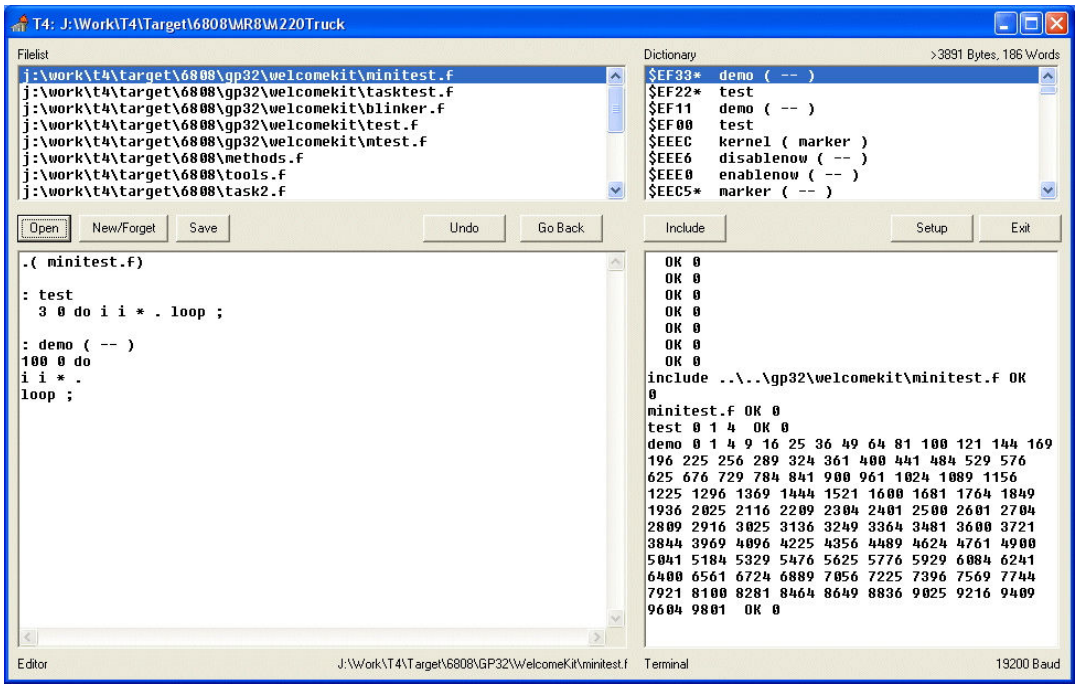


Abbildung 3: Ein einfaches Demo wird übersetzt und ausgeführt.

weil bei uns Erfahrung mit diesem System vorhanden ist. Die Oberfläche orientiert sich entfernt an Holon11 und ist aufgeteilt in vier fest zugeordnete Bereiche. Je ein großes Fenster stehen für den Editor und das Terminal, d.h. die direkte Verbindung zum Forth–Interpreter im Zielsystem zur Verfügung. Zwei kleine Fenster dienen der Navigation in den Quelldateien und dem direkten Einblick in das Dictionary. Um schnell voranzukommen, wurden wo immer möglich fertige Delphi–Komponenten eingesetzt, auf Kosten der Effizienz und Geschwindigkeit. Da Speicherplatz im PC im Überfluss vorhanden ist, können im Dictionary beliebige Zusatzinformationen abgelegt werden. T4 merkt sich für jedes neu definierte Wort die Quelldatei und Zeilennummer, außerdem wird die vollständige Zeile, in der die Definition erfolgt ist, mit abgespeichert. Ist die Quelle entsprechend aufgebaut, hat jeder Dictionary–Eintrag somit automatisch auch einen hilfreichen Kommentar, z. B. in der Form `rot (n1 n2 n3 -- n2 n3 n1)` Ein Doppelklick auf ein Wort im Editor holt den entsprechenden Eintrag in das Dictionary–Fenster — ein Doppelklick in das Dictionary lädt wiederum den Quelltext eines Worts direkt in den Editor.

Auf dem Zielsystem wird ein Forth–Kernel benötigt, der notwendigerweise in Assembler erstellt werden muss. Der Kernel umfasst zunächst einmal alle grundlegenden Worte. Compiler und Interpreter sind weitgehend bereits in Forth erstellt, allerdings notgedrungen im Kernel noch von Hand codiert. Viele Worte des Compilers, wie if, else, then usw., können dagegen schon aus einer Forth–Quelle nachgeladen werden.

Die Kommunikation zwischen PC und dem Zielsystem erfolgt seriell, dabei hat die Schnittstelle eine Doppelfunktion. Neben der ganz normalen ASCII–Ein– und –Ausgabe bei einem laufenden Anwender–Programm werden vom Interpreter und Compiler dagegen ASCII–Steuerzeichen übertragen. Der eigentliche Ablauf ist verblüffend einfach, ein Beispiel: Im Zielsystem ist Accept wie folgt definiert:

```
: accept ( -- u ) 12 emit key ;
```

Das Steuerzeichen 12 startet im PC die eigentliche Accept–Funktionalität. Es wird eine Zeile gelesen und im TIB, ebenfalls im PC, abgelegt. Danach sendet der PC die Anzahl der gelesenen Zeichen gleichsam als Quittung an das Zielsystem zurück. Der Ablauf ist immer gleich: Das Zielsystem sendet ein Kommando, der PC antwortet mit bis zu drei Byte. Ein zusätzliches Handshake ist so nicht erforderlich. An die Schnittstelle werden minimale Anforderungen gestellt, noch nicht einmal ein Buffer (im Target) ist erforderlich. Die Verarbeitung im Zielsystem ist nämlich normalerweise schnell genug, um beim Empfang der Daten mit der Übertragungszeit zweier Stopp–Bits auszukommen. Die Schnittstelle kann so notfalls auch vollständig in Software erstellt werden, ohne Timer, Interrupts oder UART–Hardware zu benötigen oder zu blockieren. Bei unseren Modulen werden in der Entwicklungsphase Port–Pins zur Kommunikation genutzt, die später dann die Funktion von Konfigurations–Jumpern übernehmen.

Das Konzept von T4 ist von der Art der Implementierung des Forth auf dem Target zunächst einmal unabhängig. Welche Implementierung *optimal* ist, hängt völlig vom Prozessor und den Randbedingungen ab, pauschale Aussagen kann man da meiner Meinung nach nicht machen. Für unsere Anwendungen soll möglichst viel Funktion

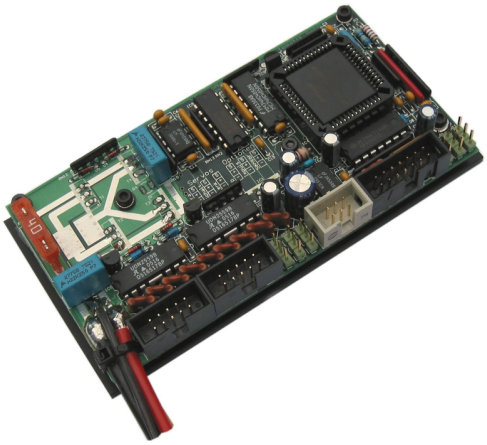


Abbildung 4: Truck-Fahrtregler mit Lichtanlage, noch in bedrahteter Technik mit dem 68HC11

Abbildung 5: Aktueller 20A Fahrtregler für Funktionsmodelle mit Lichtanlage auf der Basis eines 68HC08

in möglichst kleinen (preiswerten) Chips untergebracht werden. Die Rechenleistung ist dabei eher zweitrangig. Immerhin hat ein 68HC08 je nach Taktfrequenz schon den vier– bis achtfachen Durchsatz eines Commodore C64, für viele Anwendungen ist das völlig ausreichend. Und zeitkritische Funktionen sowie Interrupt–Services werden eh bevorzugt in Assembler erstellt. Die Wahl fiel nach einigen Voruntersuchungen auf eine Kombination von direkt–threaded und token–threaded Forth. Der innere Interpreter kann bis zu 128 7–Bit Token erkennen und einen Adressraum bis zu 32 kByte über 15–Bit Adresstoken abdecken. Ein Bit unterscheidet die beiden Fälle. Die in Assembler codierten Worte werden dabei grundsätzlich mit Byte–Token angesprochen, die über Forth compilierten Worte dagegen mit 15–Bit–Adressen. Einige weitere Optimierungen führen dazu, dass z. B. das Wort `2DUP` in nur vier Byte codiert werden kann:

```
$FD jsr  ,x
$44 dc.b over
```

```
$44 dc.b over
$4B dc.b next
```

Der Einsatz eines *klassischen* inneren Interpreters hat zudem den Vorteil, das sich auch ein Single–Step Debugging schön einfach implementieren läßt. Für ein Multitasking wird ebenfalls das typische kooperative Konzept mit `pause` angewendet, bei unseren Modellbau–Projekten werden meistens zwei bis vier Tasks eingesetzt. `Create – Does>` darf natürlich auch nicht fehlen. Da T4 aber direkt in einen Flash–Speicher kompiliert, wurde aus *technischen* Gründen auf das ältere `<Builds Does>` Konstrukt zurückgegriffen. Die Übersetzung in den Flash–Speicher hat zusammen mit einem Marker–Mechanismus (ähnlich Forget) den unschätzbaren Vorteil, dass nach einem Absturz des Zielsystems in der Entwicklungsphase oft nur einige Sekunden vergehen, bis nach dem Reset die Arbeit (und das heißt in diesem Fall die Fehlersuche) fortgesetzt werden kann.

```
 1     600   84i
 2     601   85i a00E1DC 8A          swap          pulh           ; 1/2
 3     602   86i a00E1DD 88                        pulx           ; 1/2
 4     603   87i a00E1DE 9EE6 02                   lda    2,sp    ; 3/4
 5     604   88i a00E1E1 87                        psha           ; 1/2
 6     605   89i a00E1E2 9EE6 02                   lda    2,sp    ; 3/4
 7     606   90i a00E1E5 87                        psha           ; 1/2
 8     607   91i a00E1E6 9EEF 04                   stx    4,sp    ; 3/4
 9     608   92i a00E1E9 8B                        pshh           ; 1/2
10     609   93i a00E1EA 88                        pulx           ; 1/2
11     610   94i a00E1EB 9EEF 03                   stx    3,sp    ; 3/4
12     611   95i                                   next           ; summe :18/28
13     612   92m a00E1EE CC E114    +              jmp    doNext
14     613   96i
15     614   97i a00E1F1 FD          rot           jsr    ,x
16     615   98i a00E1F2 1102                      dc.b   toRToken,swapToken
17     616   99i a00E1F4 1302                      dc.b   rFromToken,swapToken
18     617  100i a00E1F6 3C                        dc.b   exitToken
19     618  101i
```

Abbildung 6: `swap` ist in Assembler codiert. `rot` belegt in handcodiertem Forth bereits nur noch 6 Byte.

Verzichtet man auf Zahlenkonvertierung (im Target) und doppelt–genaue Arithmetik so benötigt ein Minimal–System mit Multitasking nur etwas über 2kByte Speicher. Der äußere Interpreter belegt davon ca. 230 Byte, der Compiler rund 260 Byte und die nötigen Flash–Routinen etwa 220 Byte. Single–Step und ein einfaches Dump belegen nochmals ca. 170 Byte. Zusammen sind das weniger als 1kByte, die nur in der Enwicklungsphase benötigt werden und später *nutzlos* in der Seriensoftware verbleiben.

Bereits ab 4k Flash ist so ein sinnvoller Einsatz möglich, mit Platz für bis zu 1000 Forth–Token für die Applikation. Die Schnittstelle kann noch auf Halb–Duplex minimiert werden, dann ist der kleinste sinnvoll einsetzbare Microcontroller ein 4k–Chip z. B. ein 68HC908QT4 im DIL8– oder SO8–Gehäuse mit noch 5 freien Ein–/Ausgängen (Zwei werden für die Versorgungsleitung benötigt, ein weiterer Pin für die Kommunikation mit T4 in der Entwicklungsphase.)

T4 wurde, kaum dass es einigermaßen einsetzbar war, sofort für einem Software–Redesign eines Fahrtreglers mit Lichtanlage verwendet. Der Funktionsumfang der Baugruppe konnte ein gutes Stück erweitert werden, denn durch den nun sehr kompakten Code wurde wieder Speicherplatz im Controller frei. Bis heute wurden fünf Projekte unterschiedlicher Komplexität mit T4 realisiert, und das System soll auch für zukünftige Entwicklungen, soweit irgendwie möglich, zum Einsatz kommen. Leider gab es bisher keine Zeit, den PC–Teil von T4 weiter zu optimieren. Es gibt noch einige Mängel, allem voran in der Fehlerbehandlung, die nur rudimentär existiert. Die Arbeit an einer AVR–Variante wurde bereits aufgenommen, ist aber noch nicht abgeschlossen.

Fazit: Mit dem beschriebenen Ansatz kann ein Forth für die Cross–Softwareentwicklung vergleichsweise einfach und in kurzer Zeit implementiert werden. Alle wesentlichen Merkmale von Forth, wie interaktives Arbeiten und Debuggen, inkrementelles Compilieren und die Erweiterbarkeit der Sprache, bleiben erhalten. Zusätzlich kann einiges an Komfort in der Entwicklungsumgebung realisiert werden. Die Software im PC ist weitgehend unabhängig vom Zielprozessor. In der Praxis hat sich T4 gut bewährt und die Entwicklungszeiten deutlich reduziert. Das Konzept eignet sich allerdings nicht für alle Prozessor–Typen und Anwendungen gleich gut. Ein Flash–Speicher mit geringer Blockgröße und die Fähigkeit des Prozessors, auf den eigenen Code–Speicher zuzugreifen, sind Voraussetzung. Im Forth–Sinn nicht optimal gelöst ist die Entwicklung auf Assembler–Ebene und die geringe Flexibilität des Dictionaries, dessen Struktur nur im PC selbst geändert oder erweitert werden kann. Die Interaktion zwischen PC und Zielsystem setzt außerdem der Übersetzungsgeschwindigkeit Grenzen. Der fertige Code für die Serie muss zudem mit einem zusätzlichen Tool aus dem Target kopiert werden. ■

Abbildung 7: Die Servonaut–Werbe–Trucks — Forth driven

Forth–Gruppen regional

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

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

München
Bernd Paysan
Tel.: (0 89) – 79 85 57
bernd.paysan@gmx.de
Treffen: jeden 4. Mittwoch im Monat ab 19:00. Treffpunkt ändert sich im Moment gerade, bitte auf der Web–Site nachsehen oder nachfragen!

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

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

Gruppengründungen, Kontakte

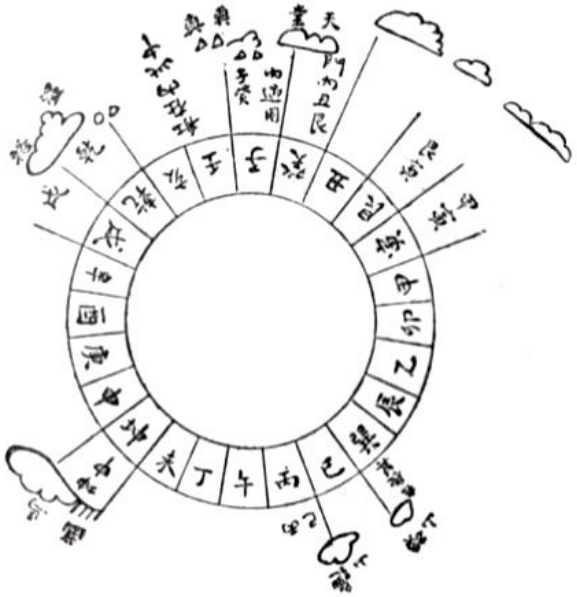
Hier könnte Ihre Adresse oder Ihre Rufnummer stehen — wenn Sie eine Forthgruppe gründen wollen.

µP–Controller Verleih

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

Spezielle Fachgebiete

FORTHchips (FRP 1600, RTX, Novix)	**Klaus Schleisiek–Kern** Tel.: **(0 40) – 37 50 08 03** (g)
KI, Object Oriented Forth, Sicherheitskritische Systeme	**Ulrich Hoffmann** Tel.: **(0 43 51) – 71 22 17** (p) Fax: **– 71 22 16**
Forth–Vertrieb volksFORTH ultraFORTH RTX / FG / Super8 KK–FORTH	Ingenieurbüro **Klaus Kohl–Schöpe** Tel.: **(0 70 44)–90 87 89** (p)



Möchten Sie gerne in Ihrer Umgebung eine lokale Forthgruppe gründen, oder einfach nur regelmäßige Treffen initiieren? Oder können Sie sich vorstellen, ratsuchenden Forthern zu Forth (oder anderen Themen) Hilfestellung zu leisten? Möchten Sie gerne Kontakte knüpfen, die über die VD und das jährliche Mitgliedertreffen hinausgehen? Schreiben Sie einfach der VD — oder rufen Sie an — oder schicken Sie uns eine E–Mail!

Hinweise zu den Angaben nach den Telefonnummern:
Q = Anrufbeantworter
p = privat, außerhalb typischer Arbeitszeiten
g = geschäftlich
Die Adressen des Büros der Forth–Gesellschaft e.V. und der VD finden Sie im Impressum des Heftes.

euroForth 2007 — at Dagstuhl Castle
Invitation

www.dagstuhl.de

Dear Forth enthusiast,

I would like to invite you to attend the **23rd euroForth conference** on the Forth Programming Language and Forth Processors, the annual get-together of the European Forth Community from

Friday, September 14th until Sunday, September 16th at Dagstuhl Castle, Germany

with the traditional 4th day as an option until Monday

and a preceeding "Forth 200X" standards day on Thursday.

This years motto: **Stack Orientated Virtual Machines**

As always, the conference is open to any Forth related topic:

Forth compilation techniques, advances in platform independence, stack based architectures, compilation techniques for stack based architectures, real-time and embedded systems solutions ...

You can contribute by presenting a Paper (20 minutes), by presenting a Poster, or by hosting a Workshop.

If you mail your paper (< 15 pages) by September 3rd **in PDF format** to **euro4th@send.de** it will be included in the conference handouts. This is also the deadline for workshop requests. Your registration is needed by August 24th.

See: http://www.complang.tuwien.ac.at/anton/euroforth2007/ for the call for papers.

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

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

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

Latest conferenc info: http://dec.bournemouth.ac.uk/forth/euro/index.html

I hope to see your there :)

Klaus Schleisiek

This years euroForth office:

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

Schloss Dagstuhl (Photo © 2004 David Monniaux — Wikipedia)