Vierte Dimension

2/2008

Das Forth–Magazin

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

In dieser Ausgabe:

Dienstleistungen und Produkte fördernder Mitglieder des Vereins

tematik GmbH
Technische Informatik

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

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

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

LEGO RCX–Verleih

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

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

Inhalt:

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

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

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

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

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

Hier könnte Ihre Anzeige stehen!

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

`Secretary@forth-ev.de`

KIMA Echtzeitsysteme GmbH

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

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

FORTecH Software GmbH
Entwicklungsbüro Dr.–Ing. Egmont Woitzel

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

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

Hier könnte Ihre Anzeige stehen!

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

`Secretary@forth-ev.de`

Ingenieurbüro
Klaus Kohl–Schöpe

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

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

Impressum

Name der Zeitschrift

Vierte Dimension

Herausgeberin

Forth–Gesellschaft e. V.
Postfach 32 01 24
68273 Mannheim
Tel: ++49(0)6239 9201-85, Fax: -86
E–Mail: `Secretary@forth-ev.de`
`Direktorium@forth-ev.de`
Bankverbindung: Postbank Hamburg
BLZ 200 100 20
Kto 563 211 208
IBAN: DE60 2001 0020 0563 2112 08
BIC: PBNKDEFF

Redaktion & Layout

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

Anzeigenverwaltung

Büro der Herausgeberin

Redaktionsschluss

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

Erscheinungsweise

1 Ausgabe / Quartal

Einzelpreis

4,00€ + Porto u. Verpackung

Manuskripte und Rechte

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

Liebe Leser,

ich begrüße Euch zur zweiten Ausgabe dieses 24. Jahrgangs der Vierten Dimension.

Anfang Juni fand in der Hamburger Niederlassung von Lehmanns Fachbuchhandlung das *Community–Treffen Hamburger Computergruppen* statt, in der gut zwei Dutzend Formationen von Computer–Interessierten sich vorstellten und über ihre Arbeit berichteten. Man erfuhr also unter anderem etwas über die German–Unix–User–Group (GUUG), openStreetmap, Barmbeks Linux User Group (BALISTA), den Lisp–HH–Stammtisch und eben auch über die lokale Gruppe *KüstenForth* der Forth–Gesellschaft. Neben informellen Gesprächen gab es auch formale Vorstellungen der einzelnen Gruppen. Da wie beim Programmieren die Einschätzungen differieren, was eine knappe Darstellung einer Computer–Gruppe ist, zogen sich die Vorstellungen gut drei Stunden hin: interessant aber ermüdend. „Forth? Gibt's das denn noch?“ war natürlich auch wieder zu hören. Nun — mit einem Alter von 24 Jahren gehört die Forth–Gesellschaft mit zu einer des ältesten Vereine im Computer–Bereich. Sicherlich ist Forth heute nicht mehr im Mainstream, sondern mittlerweile wohl eher in Nischenbereichen im Einsatz. Aber, die Forth–Gemeinde ist klein und aktiv und Forth–Anwendungen verrichten ihre Arbeiten eher im Verborgenen: in Geräten, die brav und zuverlässig ihre Arbeit erledigen und kein weiteres Aufsehen erregen. Man muss eben genau hinsehen, um zu beobachten, wo Forth überall werkelt. Auf dem Community–Treffen kannten mehr als zwei Drittel der Anwesenden Forth, aber über aktuelle Entwicklungen war ihnen wenig bekannt. Wie kann Forth wieder in den Wahrnehmungsbereich der Computer–Interessierten treten?

Forth ist eine Breitband–Sprache, die sowohl für eingebettete Systeme und die direkte Programmierung der Hardware als auch für aufwändige Algorithmen eingesetzt werden kann und dabei eine homogene Entwicklungsumgebung bietet. Dass Forth in all diesen Bereichen quicklebendig ist und auch in aktuellen Entwicklungen seinen Platz hat, beweist dieses Heft eindrücklich:

Carsten Strotmann berichtet uns, wie Forth beim Booten eines Computers hilfreich ist und auch ohne extra Betriebssystem direkt auf PC–Hardware laufen kann.

Manfred Mahlow zeigt, wie graphische Benutzeroberflächen mit Forth und GTK+ realisiert werden können.

Michael Kalus erleichtert den Büro–Alltag mit einem einfachen Textgenerator, der konfigurierbar Textentwürfe erzeugt, führt uns in die Geheimnisse der doppeltgenauen Multiplikation ein und löst ein weiteres Problem des Euler–Projekts.

Im 10. Teil seiner Serie über Forth auf dem Atmel–AVR führt uns Ron Minke vor Augen, welche Auswirkungen veränderte Designentscheidungen auf die Implementierung des AVR–Forths haben.

Mein Artikel beschreibt ein Konzept für Programm–Bibliotheken auf Quellcode–Ebene. Außerdem berichtet Fred Behringer über die aktuelle Ausgabe des Vijgeblaadje, des niederländischen Forth–Magazins.

Ich wünsche uns allen einen schönen Sommer und 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

Lebenszeichen

Bericht aus der FIG Silicon Valley: *Henry Vinerts*

Dear Fred,

ich wollte dir nur mal schnell Hallo sagen und dich wissen lassen, dass es SVFIG in Kalifornien immer noch gibt. Zu verdanken ist das einer Handvoll *Old–Timer* wie George Perry, C.H. Ting, Kevin Appert, and David Jaffe. Am 26. April 2008 schaffte ich es wieder mal, am Treffen am Cogswell College teilzunehmen. Dort reichte ich mein kürzlich empfangenes VD–Heft 1/2008 unter dem guten Dutzend Tagungsteilnehmern herum. Dr. Glen Haydon sagte mir, dass er sein Exemplar bereits durchgelesen habe, und er bat mich um deine E–Mail–Adresse. Wir sind beide der Forth–Gesellschaft dafür dankbar, dass sie sich unser mit den Heften erinnert.

Wie üblich, hielt Dr. Ting während der beiden Morgenstunden einen langen Vortrag — diesmal über Fortschritte bei seinem DSO–(Digital–Signal–Oscilloscope)–Projekt, für das er sich mit der Verwendung der *FORTHdrive*–Platine von IntellaSys (http://www.intellasys.net/) befasst hat. Er empfand das erste Programmieren der Platine als nicht gerade leicht. Ting erwähnte drei andere Platinen, die seinem Ziel dienen könnten: Er möchte ein *hand–held* DSO im Hosentaschenformat produzieren, das bis zu 200MHz verarbeiten kann.

Eine kleinere Gruppe von Gewichtsbewussten blieb im Raum, während andere zum Lunch gingen. Währenddessen lauschten wir unseren *Hausastronomen*, die sich über Nicht–Forth–Themen ausließen, so da wären *Dunkle Materie, Rotverschiebung interstellarer Materie und 'MACHOs' (Massively Compact Halo Objects),* alles Themen, die mein Gefühl der Vergeblichkeit und der Selbstzufriedenheit beim Versuch, mit der Bildung und Ausbildung mitzuhalten, die für ein aktives Mitarbeiten bei den SVFIG–Treffen nötig wären, nur verstärkten.

Während wir die Geheimnisse des Kosmos sondierten, wurde uns bewusst, dass wahrscheinlich zur selben Zeit im Kloster Roggenburg, wohl abends, ähnliche Diskussionen unter den Teilnehmern der Forth–Tagung 2008 stattfanden.

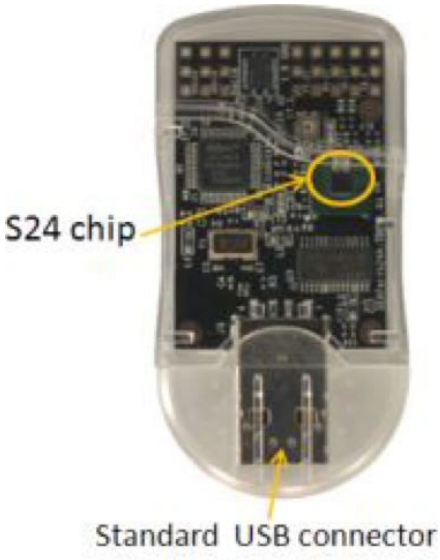
Nach dem Lunch zeigte uns Dave Jaffe mitgebrachte Bilder von der Embedded Systems Conference, die er kürzlich besucht hatte, und da keine weiteren Vorführungen oder Anlässe für weitere Diskussionen mehr auf dem Plan standen, beendeten wir das Treffen eine Stunde früher als gewöhnlich. Ja, ich denke, dass das Durchschnittsalter der SVFIG–Zugehörigen jedes Jahr immer noch um genau eine Einheit wächst, und wenn auch die kollektive Weisheit proportional dazu ansteigt, so gibt es doch Anzeichen dafür, dass die Fähigkeit, stillzusitzen und anspruchsvollen Vorträgen längere Zeit zuzuhören, entsprechend abnimmt.

Fred, ich möchte dir einen schönen Sommer wünschen, und meine besten Wünsche gehen an die gesamte Forth–Gesellschaft in ihrer Arbeit mit Forth und ihrem Spaß daran.

Wie immer,

Henry

Übersetzt von Fred Behringer



Das Forth–Drive mit S24–Prozessor von Intellasys (Quelle: www.intellasys.net)

Doppeltgenaue Multiplikation

Michael Kalus

Am Beispiel der doppeltgenauen Multiplikation d* wird gezeigt wie der Algorithmus der Multiplikation auf einer Maschine abgearbeitet werden kann.

Grundlagen

Wir lernen in der Schule multiplizieren in der "langen Formünd die Frage ist nun, können wir damit einen Algorithmus finden, um doppeltgenaue Zahlen zu multiplizieren?

Die lange Multiplikation geht so:

```
12 * 34
-------
      8
+    40
+    60
+   300
     ---
+   100 Ü
     ---
     408 Summe
     ===
```

Man multipliziere die beiden Einer (2∗4), dann den zweiten Einer mit dem ersten Zehner $(4 * 10)$, den zweiten Zehner mit dem ersten Einer $(30 * 2)$ und nun noch die beiden Zehner (10∗30). Diese Zwischenergebnisse werden schließlich aufsummiert und die Überträge hinzugerechnet.

Der Algorithmus

```
  A B  *  C D
---------------
            BD
+        AD 00
+        BC 00
+     AC 00 00
   -----------
+  ZZ YY XX 00 Ü
   -----------
   GG KK MM NN Summe
```

Bild 1: Doppeltgenaue Multiplikation

Die beiden Zahlen AB und CD stellen schon unsere doppeltgenauen Zahlen dar. Der Wert in der oberen Zelle A entspricht der höheren Stelle, der Wert B in der unteren Zelle der unteren Stelle, ebenso bei CD. Jede Stelle füllt also eine Zelle auf dem Stack. In der Stacknotation mit lokalen Variablen sieht das dann so aus:

```
: d* { B A D C -- NN MM }
  B D um* ( NN XX )
  A D * +
  B C * + ;
```

Und wo sind nun der Überlauf YY und die Zelle KK geblieben, also quasie die dritte Stelle? Da das Ergebnis der doppeltgenauen Multiplikation auch nur höchstens doppeltgenau sein soll, entfällt die Positionen AC samt dem Übertrag YY einfach, und damit gibt es auch die Summe KK nicht. Werden Zahlen multipliziert, die dahin überlaufen, haben wir einen Fehler gemacht. Das ist bei der einfach genauen Multiplikation auch so. Auch dort dürfen mit `*` nur Zahlen multipliziert werden, die keinen Übertrag in die nächste Zelle ergeben.

```
hex
2 FF * u.   <ret>          1FE  ok
2 true * u. <ret>     FFFFFFFE  ok
\ sollte aber sein: 1FFFFFFFE
\ Der Übertrag entfällt einfach!

2 true um* ud. <ret> 1FFFFFFFE  ok
```

Das Wort `um*` liefert das richtige Ergebnis, weil der Übertrag mit berechnet wird. Die 'richtige' einfachgenaue Multiplikation muss also in einem doppeltgenauem Ergebnis münden, sonst stimmt sie ab einer gewissen Größe nicht mehr.

Im Forth wird dieser Überlauf im `*` aber nicht abgefangen und auch nicht als Fehler angezeigt. Der Programmierer muss selbst darauf achten, dass seine Produkte in der Grenze einer Zelle bleiben, die Faktoren also nicht zu groß werden. Oder er nimmt gleich `um*` statt `*` für seine Multiplikation.

Für `d*` gilt das ebenso. Die doppeltgenaue Multiplikation kann auch überlaufen in die nächst höhere Zelle GG. Um das darzustellen muss `ut*` benutzt werden. Damit wird eine weitere Zelle bereitgestellt (t steht für „trippel“).

So kann man sich seine eigenen Multiplikationsworte zusammenstellen, so genau wie man möchte, indem immer weitere Zellen hinzu genommen werden. Um die Verwirrung auf dem Stack nicht zu groß werden zu lassen, können lokale Variablen benutzt werden. Die Ausführungsgeschwindigkeit sinkt dadurch aber. Ein Beispiel dazu ist weiter unten angegeben.

Die Stack–Akrobatik im `d*` soll auch auf 64Bit–Maschinen laufen.

Aus Bild 1 entnehmen wir: In die unterste Zelle kommt das Produkt BD und sein Überlauf nach XX muss mitgenommen werden. Die Multiplikation muss also mit `um*` erfolgen, und in der höheren Zelle finden wir den Überlauf XX wieder: BD `um* ( -- NN XX )`. In die höhere Zelle kommt die Summe $AD + BC + XX$ und es kann einfach gerechnet werden. Überläufe nach YY werden nicht mehr berücksichtigt, denn die Summe KK wird nicht mehr gebildet.

Um das ohne lokale Variablen zu formulieren muss man etwas tüfteln. Auch um dafür zu sorgen das die Version auf 64Bit Maschinen korrekt arbeitet.

```
: d*      ( ud1 ud2 -- udprod )
  >r swap >r 2dup um*
  2swap r> * swap r> * + + ;
```

Was passiert da eigentlich?

```
( B A D C -- NN MM ) \ mit ud1 = B A und ud2 = D C
 >r          ( -- B A D ) ( R: -- C )
 swap        ( -- B D A ) ( R: -- C )
 >r          ( -- B D   ) ( R: -- C A )
 2dup um*    ( -- B D NN XX ) ( R: -- C A )
 2swap r>    ( -- NN XX B D A ) ( R: -- C )
 *           ( -- NN XX B AD ) ( R: -- C )
 swap r>     ( -- NN XX AD B C ) ( R: -- )
 *           ( -- NN XX AD BC )
 + +         ( -- NN MM )
```

Wer nun noch weitere Genauigkeit haben will, braucht auch die dritte Zelle mit dem Wert KK darin.

```
\ tripple genaues Ergebnis
: ut* { B A D C -- NN MM KK }
  B D um*     ( -- NN XX )
  A D um* >r  ( -- NN XX AD  )  ( R: -- YY1 )
  +           ( -- NN MM1    )  ( R: -- YY1 )
  B C um* >r  ( -- NN MM1 BC )  ( R: -- YY1 YY2 )
  +           ( -- NN MM     )  ( R: -- YY1 YY2 )
  A C *       ( -- NN MM AC  )  ( R: -- YY1 YY2 )
  r> +        ( -- NN MM KK2 )  ( R: -- YY1 )
  r> +        ( -- NN MM KK ) ;
```

Und so kann man das weiter steigern, soweit wie man möchte.

Die Olympia RAE 4/30-3 Rechenmaschine
von Olympia in Wilhemshaven
realisierte schon 1966 doppeltgenaue Multiplikation
(Quelle: oldcalculatormuseum.com, Photo: Serge Devidts)

Gehaltvolles

zusammengestellt und übertragen von *Fred Behringer*

VIJGEBLAADJE der HCC Forth–gebruikersgroep, Niederlande Nr. 67, Februar 2008

Forth en schakelende voedingen — Jan van Kleef

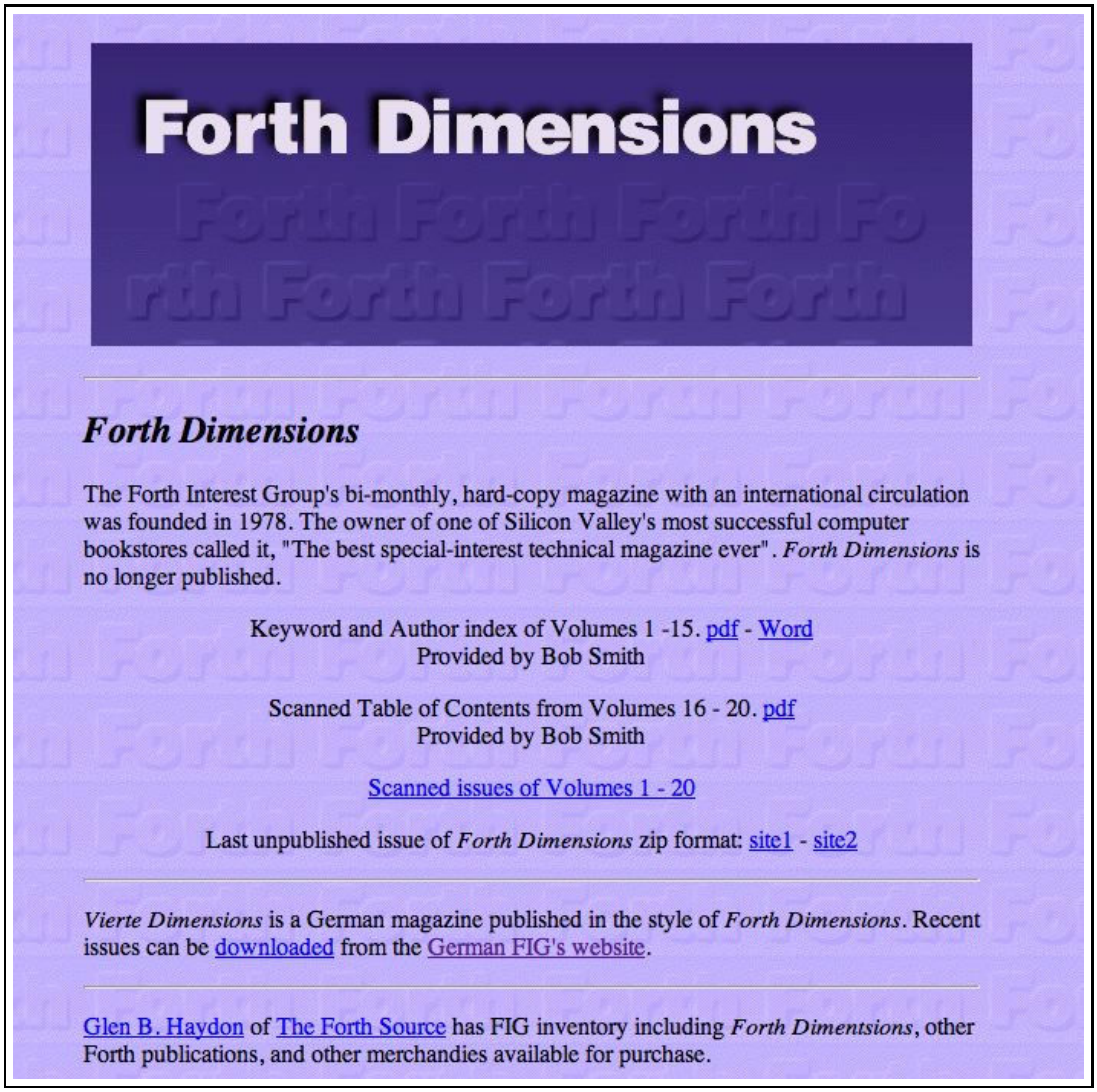
Halbleiter–Schalter kosten fast nichts und sind in allen möglichen Variationen zu haben. Zwei Grundversionen von Schalt–Speisespannungsversorgern: Von niedrigerer Spannung zu höherer, und von höherer zu niedrigerer. Spule und genügend großer Elektrolyt–Kondensator dienen der Energiespeicherung zwischen den Durchschaltzeiten. Die Spannungshöhe wird über die Impulslängen gesteuert und geregelt. Zur Steuerung bietet sich ein preisgünstiger Mikro–Controller an. Programmiert hat der Autor dieses einführenden Artikels seine Versuchsaufbauten in Forth

Gevonden op het internet — Ron Minke

Ron hat bei seiner Suche nach einer bestimmten Passage aus den (seit geraumer Zeit nicht mehr erscheinenden) Forth Dimensions herausgefunden, dass die amerikanischen Forth-Freunde inzwischen 20 (!) (statt der bisher nur 6) Jahrgänge der FDs eingescannt haben:

`http://www.forth.org/fd/contents.html`

Beim dort Nachlesen habe ich (der Rezensent) die Bemerkung gefunden, dass das Einscannen (naturgemäß) nicht hundertprozentig geglückt ist. Insbesondere wird empfohlen, sich auf die Listings nicht unbedingt Zeichen für Zeichen zu verlassen.



Webseite des Forth–Dimension–Online–Archivs unter `http://www.forth.org/fd/FDcover.html`

Forth ohne/als Betriebssystem

Carsten Strotmann[1]

In der VD 3+4/2007 fragt Fred Behringer als Notiz zur Besprechung des XO Laptops (One Laptop per Child): „... Gibt es eine Möglichkeit, dieses BIOS (die OpenFirmware des XO Laptops) zeitweilig (so wie bei Zeichensätzen), vielleicht anstelle eines Schatten–BIOS, in meinen Rechner zu schalten? Am liebsten über einen Bootmanager (bei mir XFDisk)".

Nachdem ich für die Forth–Gesellschaft im Jahre 2007 zusammen mit dem LinuxBIOS–Team auf dem LinuxTag anwesend war und wir dort die *Forth–Flagge* hochgehalten haben, lag die Idee, einen Artikel über den Status von OpenFirmware zu schreiben, schon recht lange auf meinem Tisch. Freds Anfrage hat mir den notwendigen Impuls gegeben, diesen Artikel fertigzustellen.

PC Real Mode

Um die Kompatibilität mit alten 16–Bit–Betriebssystemen wie MS–DOS oder CP/M aufrechtzuerhalten, starten selbst moderne PCs immer noch im 16–Bit–*Real–Mode*. In diesem Modus wird selbst bei neuesten PCs ein (sehr schneller) Original–IBM–PC mit 1 MB Hauptspeicher und VGA Karte eingerichtet. Neben dem werbewirksamen Einblenden von Markennamen des Rechner–, BIOS–, Motherboard–, CPU– und Grafikkarten–Herstellers ist diese Einrichtung des 16–Bit–Real–Modes verantwortlich, warum der BIOS–Startvorgang auf modernen Rechnern immer noch so langsam ist wie beim Original–IBM–PC von 1981 (zum Teil, man mag es nicht glauben, sogar noch langsamer).

Moderne BIOS–Ersatz–Technologien wie EFI oder CoreBoot starten den Rechner im 32–Bit–Modus erheblich schneller. Doch dazu später.

Zuerst schauen wir uns an, welche Optionen wir heutzutage haben, um ein Forth–System ohne Betriebssystem zu starten. Hierbei beziehe ich mich auf x86–kompatible PCs, welche in den letzten 4–5 Jahren hergestellt und verkauft wurden.

Forth vom PC–BIOS gebootet

Die Idee, Forth direkt, d. h. ohne Umwege über MS–DOS oder ein anderes Betriebssystem, vom BIOS aus zu starten, ist so alt wie die ersten Forth–Systeme für den IBM–PC. Damals, in der ersten Hälfte der 1980er Jahre, war diese Technik eher die Norm als denn die Ausnahme. Ein Forth wurde direkt von einer Boot–Diskette gestartet.

Das war die Zeit der Disketten, vor der Zeit der Festplatten. Nach dem Anschalten des PCs wird das BIOS aktiv, initialisiert die Hardware (Tastatur, Floppykontroller, Grafikkarte, Schnittstellen) und lädt den ersten Sektor (Master Boot Sektor). Das Programm in diesem Sektor übernimmt die Kontrolle über den Rechner und lädt das Forth–Programm, direkt vom Datenträger (meist Diskette).

Auf der Diskette befindet sich meist ein Forth–eigenes (Block–) Dateisystem. Ein– und Ausgabe (Bildschirm, Datenträger) erfolgt über die vom BIOS bereitgestellten Systemroutinen. Insofern läuft ein solches Forth nicht vollständig unabhängig von einem Betriebssystem, da das Rechner–BIOS ein sehr simples Betriebssystem bereitstellt (genannt Basic–Input–Output–System = BIOS).

Auch heute kann man Forth–Systeme über dieses System starten, doch finden sich auch immer mehr Rechner, bei denen dieser Weg nicht mehr funktioniert:

- keine Diskettenlaufwerke
- ein versehentlicher Zugriff des Forth–Systems auf die Festplatte kann Gigabytes an Daten zerstören

Syslinux COMBOOT 16–Bit

Das Syslinux Programm[2] erlaubt das Starten von Betriebssystemen (nicht nur Linux) von FAT–formatierten Datenträgern (Disketten, Festplatten oder USB–Speicher), von CD–ROM oder über PXE–Netzwerkboot. Dabei unterstützt Syslinux ein spezielles Format, das so genannte COMBOOT–Format. COMBOOT–Dateien sind Standard–MS–DOS–COM–Programm–Dateien [3] (64–KB–Speicher, Start bei $100).

Über das Syslinux–COMBOOT–Verfahren lassen sich MS–DOS–kompatible Forth–Systeme, welche als COM–Dateien vorliegen, direkt starten. Im Vergleich zum direkten Start über das BIOS unterstützt Syslinux den Start von beliebigen, FAT–formatieren Datenträgern. Voraussetzung allerdings ist, dass die Forth–Systeme keine MS–DOS–Systemaufrufe benutzen, sondern ausschließlich direkt mit der Hardware sprechen oder das BIOS benutzen. Ein Forth–System, welches auf diesem Wege gestartet werden kann, ist die VolksForth–8086–Version ohne MS–DOS–Abhängigkeiten (VFORTHN.COM).

[1] mit Hilfe von Stefan Reinauer

[2] http://syslinux.zytor.com

[3] http://en.wikipedia.org/wiki/COM_file

Über die MEMDISK[4]–Erweiterung ist es auch möglich, Forth–Systeme auf Disketten–Abbilddateien von CD–ROM oder sogar über das Netzwerk (PXE–Boot) zu starten. Mit MEMDISK kann man z. B. FreeDOS[5] von einem kleinen (230K) Diskettenabbild vom USB–Stick laden. Das FreeDOS erkennt den FAT–formatierten USB–Stick als Laufwerk C: (das MEMDISK Diskettenabbild ist Laufwerk A:). Von dort aus kann jedes MS–DOS–basierte Forth (z. B. VolksForth) geladen werden.

PC–Protected–Mode

Um heutige 32–Bit oder 64–Bit–Hardware von Forth ausnutzen zu können, sollte das Forth–System im so genannten *Protected–Mode*[6] gestartet werden. Im Gegensatz zum *Real–Mode*, welcher einen schnellen 8086 Prozessor mit 1 MB Adressraum zur Verfügung stellt, stehen dem Entwickler im *Protected–Mode* alle erweiterten Funktionen moderner Prozessoren zur Verfügung.

Da nicht jeder Entwickler das Rad neu erfinden und das Umschalten in den *Protected–Mode* sowie das Laden des Programmes von einem Speichermedium selbst neu programmieren möchte, gibt es eine Reihe von Boot–Loadern, Ladeprogramme, welche den Prozessor in den *Protected–Mode* setzen, das Programm (z. B. das Forth–System) laden und dann die Ausführung an das geladene Programm übergeben.

Syslinux COM32

Syslinux unterstützt das Laden von speziellen 32–Bit–Programmen mit einem sehr einfachen Programmaufbau, ähnlich dem von MS–DOS–*COM*–Dateien. Programme im Dateiformat COM32[7] müssen ab Speicherstelle 0x101000 lauffähig sein und dürfen keine Segmente ansprechen, stattdessen muss das *Flat–Memory*[8] Modell des i386–Prozessors benutzt werden.

Syslinux stellt für COM32 Programme eine simple, BIOS–ähnliche API für Ein– und Ausgabe zu Verfügung, so dass Programme im 32–Bit–Protected–Mode diese Funktionen anstatt BIOS–Funktionen benutzen können.

Es sollte möglich sein, mit einwenig Programmieraufwand bestehende Linux– oder Windows–32 oder –64–Bit–Forth–Systeme umzuschreiben, um im COM32–Modus zu laufen. Erweiterte I/O–Funktionen wie Zugriff auf die Grafikkartenfunktionen oder Datenspeicher (USB etc), müssen über Treiber im Forth–System realisiert werden. Dies kann für den einen oder anderen eine interessante Herausforderung darstellen.

GRUB

GRUB (ein Akronym für GRand Unified Bootloader) ist ein sehr flexibler Bootloader für PC–Systeme, welcher für eine Vielzahl von Betriebssystemen eingesetzt wird (z. B. Linux und OpenSolaris x86). Grub kann von verschiedenen Medien gestartet werden (Festplatten, Floppy–Disk, CD– und DVD–Laufwerken sowie Flash–Disks) und ist in der Lage, von einer großen Anzahl von Dateisystemen[9] Betriebssystemkerne zu starten. Grub wird in der Regel vom PC–BIOS gestartet und bietet dem Benutzer ein Auswahlmenü, über das Betriebssysteme gestartet werden können.

Grub kann Betriebssysteme nach dem Multiboot–Standard[10], Linux, FreeBSD, NetBSD und OpenBSD direkt starten. Andere Betriebssysteme (z. B. OS/2 oder Windows) werden über das so genannte Chain–Load geladen, bei dem Grub einen Original–Bootsektor für das gegebene Betriebssystem aus einer Datei lädt und startet.

Um ein 32–Bit–Forth–System über Grub zu starten, sollte dieses Forth System dem Multiboot–Standard entsprechen.

Ein separates Projekt, Grub4Dos[11], bietet Grub und die Grub–Installations–Werkzeuge für DOS–basierte Systeme an.

CoreBoot

CoreBoot ist ein OpenSource–Projekt, welches an einem quelloffenen BIOS–Ersatz arbeitet. Das Projekt firmierte lange unter dem Namen *LinuxBIOS*, da es sich aber nicht primär um ein Linux–Projekt handelt, und das Ergebnis kein *BIOS* ist, hat sich das Projekt Anfang 2008 in *CoreBoot* umbenannt. Im Folgenden sind daher die Begriffe LinuxBIOS und CoreBoot synonym.

CoreBoot bietet ein modulares System, um Betriebssysteme von Datenspeichern (Festplatten, USB–Sticks etc) oder direkt aus dem Flash–ROM zu starten. CoreBoot selbst ist nur ein kleiner, in C geschriebener Programmteil, welcher den PC soweit vorbereitet, dass ein Betriebssystem geladen und gestartet werden kann. CoreBoot initialisiert hierzu:

- die CPU(s)
- den Speicher
- den Bus (PCI)
- Controller (IDE, ATA, SATA, USB ...)
- Grafikkarte

[4] http://syslinux.zytor.com/memdisk.php
[5] http://www.freedos.org
[6] http://en.wikipedia.org/wiki/Protected_mode
[7] http://syslinux.zytor.com/comboot.php
[8] http://en.wikipedia.org/wiki/Flat_memory_model
[9] ext2, ext3, UFS, UFS2, ReiserFS, FAT, NTFS, ISO9660, JFS, Minix, FFS, XFS
[10] http://www.gnu.org/software/grub/manual/multiboot/multiboot.html
[11] http://grub4dos.sourceforge.net/

Um CoreBoot benutzen zu können, muss man ein von CoreBoot/LinuxBIOS unterstütztes PC–Board besitzen. Leider halten die meisten PC–Hersteller wichtige Informationen über die verwendete Hardware und deren Schnittstellen unter Verschluss, so dass nicht alle PC–Boards von CoreBoot unterstützt sind. Eine Liste der Boards und Systeme, welche mit CoreBoot gestartet werden könnnen, findet sich auf der CoreBoot–Webseite[12].

Payloads

CoreBoot kann verschiedene Betriebssysteme oder Bootloader starten (die so genannten Payloads):

- Linux — direkter Start eines Linux–Kerns
- FILO — ein einfacher Bootloader mit Dateisystem–Unterstützung. Dies ist ein GRUB ohne PC–BIOS Unterstützung.
- GRUB2 — wird FILO ablösen, ist aber noch in der Entwicklung und noch nicht einsatzbereit
- Mitch Bradley's Open Firmware
- CodeGen's SmartFirmware
- OpenBIOS
- GNUFI[13] (UEFI)
- Etherboot[14] — beinhaltet FILO und unterstützt neben Netzwerkboot auch SATA und USB–Speicher
- ADLO — Verbindungsschicht, um ein 16–bit–Bochs–BIOS zu laden. Dies erlaubt den Start von Betriebssystemen, welche PC–BIOS–Aufrufe machen, wie z. B. Windows und OpenBSD.
- Plan 9[15] — ein verteiltes Betriebssystem

Motivation für CoreBoot

Es gibt verschiedene Motivationen, um CoreBoot anstatt eines Standard–PC–BIOS einsetzen zu wollen:

- um ein komplettes Rechner–System aus freier Software zu erstellen
- um volle Kontrolle über den Rechner (Persönlichen Computer = PC) zu behalten, da moderne kommerzielle BIOS–Systeme und –Erweiterungen über DRM (Digital–Rights–Management) den Benutzer zu kontrollieren versuchen
- um Ansatzpunkte für Behörden–Schnüffelsoftware (aka Bundestrojaner) zu verhindern
- um den Rechner–Startvorgang zu verkürzen
- um Lizenzgebühren für ein kommerzielles BIOS zu sparen

Auf Youtube[16] gibt es ein Video, welches zeigt, wie schnell ein LinuxBIOS–Start bis in eine grafische Unix–X11–Oberfläche sein kann.

Auf der FOSDEM–2007–Konferenz hat Projekt–Mitbegründer Ron Minnich[17] einen interessanten Vortrag über CoreBoot (damals noch unter dem Titel LinuxBIOS) gehalten, in welchem er auch über die Motivation hinter LinuxBIOS spricht. Der Vortrag ist als Video im Medienarchiv[18] der FOSDEM–Seite abrufbar.

CoreBoot und Forth

CoreBoot kann benutzt werden, um ein (oder mehrere) Forth–Systeme direkt aus dem Flash–Speicher zu laden. Was mit großen Betriebssystem–Kernen wie Linux bei geringem Flash–Speicher auf dem Motherboard (512 KB, 1 MB) ein Problem sein kann, ist für Forth eine Tugend. In 256 KB bekommt man viel Forth hinein.

Mit OpenBIOS und Mitch Bradleys OFW sind schon zwei OpenFirmware–kompatible Forth–Systeme für CoreBoot verfügbar.

EFI

EFI, das Extensible Firmware Interface, ist ein von Intel entwickelter PC–BIOS Ersatz. EFI wurde im Jahre 1989–2000 zusammen mit den ersten Itanium–Prozessoren veröffentlicht. Vom Konzept her gleicht EFI dem schon älteren OpenFirmware, so lässt EFI z. B. auch plattformneutrale Gerätetreiber in ByteCode zu. Allerdings ist im EFI kein Forth–Interpreter eingebettet. In der Realität sind aber die meisten EFI–Treiber in EFI–kompatiblen (PCI–) Geräten in x86–Programmcode und nicht ByteCode. Die ByteCode–Möglichkeit wird bisher von den Herstellern nicht benutzt (da es EFI bisher auch hauptsächlich auf der x86–Hardwareplatform gibt).

Im Jahre 2005 wurde die Weiterentwicklung von EFI in die Hände von UEFI, dem *Unified EFI Forum*, gelegt. Einige Grundkomponenten von EFI und die Grund–Spezifikationen[19] sind seitdem öffentlicht zugänglich. Während sich EFI bei Standard–PCs noch nicht durchgesetzt hat, ist EFI bei auf Intel–CPU basierten Rechnern der Firma Apple die Firmware.

Das EFI ist bei Apple–Rechnern für den Benutzer *unsichtbar*. Um bei Apple–Rechnern das EFI–Interface für

[12] http://www.coreboot.org/Supported_Motherboards
[13] http://www.gnu.org/software/gnufi/
[14] http://www.coreboot.org/Etherboot
[15] http://www.coreboot.org/Plan_9
[16] http://www.youtube.com/watch?v=nuzRsXKm_NQ
[17] Interview: http://archive.fosdem.org/2007/interview/ronald+g+minnich
[18] http://archive.fosdem.org/2007/media/video
[19] http://developer.intel.com/technology/efi/
[20] http://refit.sourceforge.net/

Abbildung 1: rEFIt–Bootmenü

den Anwender *nutzbar* zu machen, kann das EFI–Toolkit des *rEFIt*–Projektes installiert werden[20]. rEFIt bietet dem Apple–(x86–CPU)–Benutzer ein grafisches Boot–Menü, Zugriff auf den Apple–Hardwaretest und die EFI–Shell.

EFI–Programme

EFI kann neben Betriebssystemkernen auch direkt Programme starten. Der Apple–Hardwaretest und die EFI–Shell sind zwei Beispiele hierfür. Für diese Programme bietet EFI dem Programm eine simple I/O–Schnittstelle, ähnlich dem PC–BIOS.

Amit Singh, Autor des Buches *MacOS X Internals — A Systems approach*[21] (viele, zusätzliche Kapitel als PDF auf der Webseite erhältlich), beschreibt im Kapitel *EFI Programming on Mac OS X*[22], wie Programme erstellt werden, welche dann direkt unter EFI ausführbar sind. Mit dieser Anleitung sollte es möglich sein, ein bestehendes 32–Bit– oder 64–Bit–Forth mit einem EFI–Wrapper zu versehen und dann direkt von der EFI–Shell oder dem rEFIt–Bootmenü aus zu starten.

Die TianoCore–Webseite[23] hat weitergehende Informationen über EFI–Programmierung, inklusive des EFI–Toolkits. Das EFI–Toolkit bietet einfachen Zugriff auf EFI–Ressourcen wie

- Standard–C–Library
- die Libz–Kompressions–Library
- die BSD–Datenbank–Routinen, libdb
- die Serial–Port–Library, libtty
- TCP/IP–v4– und PPP–Netzwerkzugriff
- Python–Interpreter für EFI
- ein Text–Editor
- RAM–Disk für EFI

An dieser Auflistung sieht man, dass EFI mehr ein embedded Betriebsystem denn ein Bootloader ist. Auch ist nicht jeder von EFI begeistert, wie E–Mails[24] vom Linux–Oberentwickler Linus Torvals zeigen („EFI is this other Intel brain–damage (the first one being ACPI).“).

[21] http://www.osxbook.com/
[22] http://www.osxbook.com/book/bonus/chapter4/efiprogramming/
[23] https://www.tianocore.org/
[24] http://kerneltrap.org/node/6884
[25] http://www.openbios.info

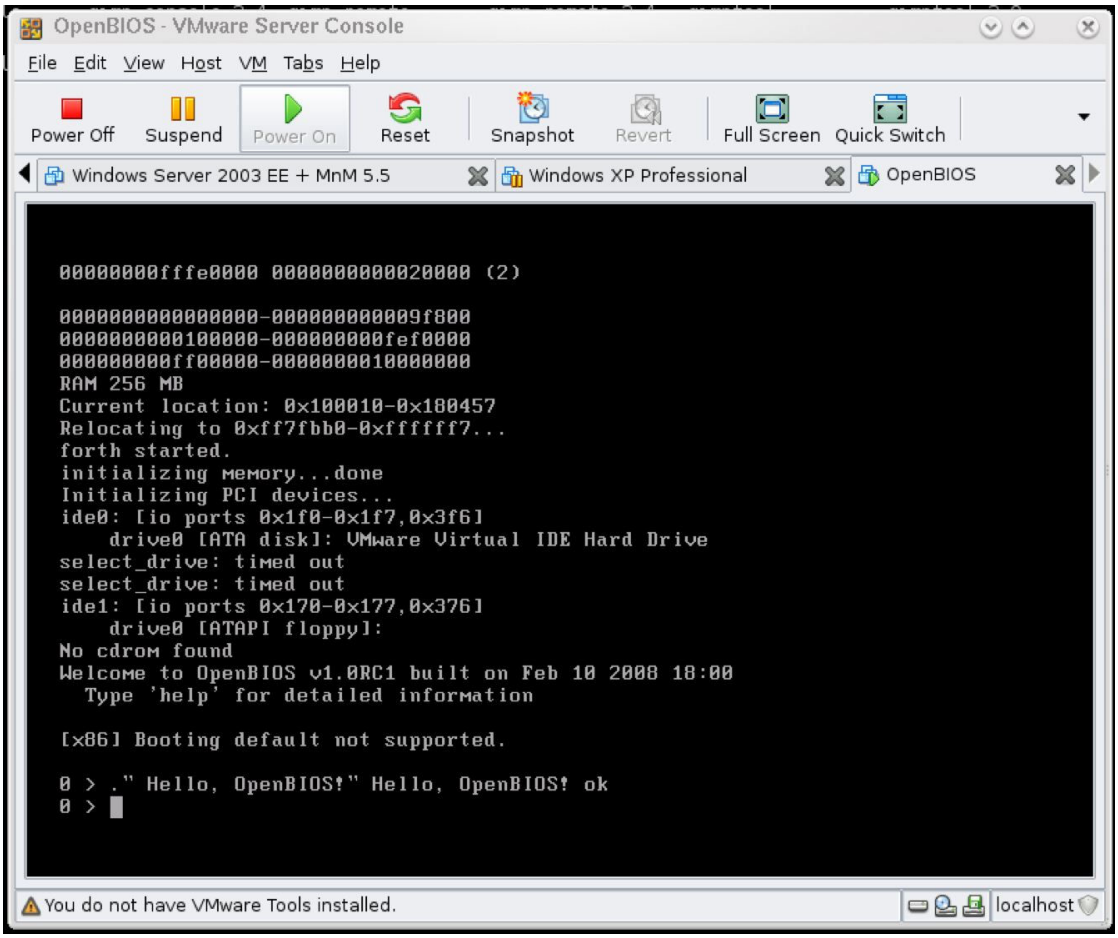


Abbildung 2: OpenBIOS in VMWare

Open–Firmware–Implementierungen

Derzeit gibt es fünf IEEE–1275–(aka OpenFirmware)–kompatible Implementierungen, welche unter verschiedenen OpenSource–Lizenzen im Quellcode verfügbar sind. Zentrale Anlaufstelle für alle diese Implementierung ist die Seite des OpenBIOS–Projektes[25]. Dort kann man den Quellcode der meisten dieser OpenFirmware–Versionen einsehen und laden.

Die verschiedenen OpenFirmware–Implementierung sind eine Schatzgrube für Forth–Programmierer, enthalten sie doch Treiber für Dateisysteme (FAT, ext2, HFS...) und Hardwarekomponenten (USB, IDE/ATA, Grafik ...), sowie einen Webserver in Forth.

OpenBIOS (OpenBIOS–Projekt)

OpenBIOS wurde als Projekt gegründet mit dem Ziel, eine quelloffene (OpenSource) Implementierung von IEEE–1275 für x86–Architekturen zur Verfügung zu stellen. Als das Projekt startete (2000), gab es am Markt nur proprietäre Implementierungen von IEEE–1275. Im Jahre 2006 wurde OpenBIOS durch die in kurzen Zeitabständen erfolgte Freigabe fast aller dieser proprietären Versionen unter einer OpenSource–Lizenz überholt. Derzeit aktuell ist die Version 1.0rc2 vom April 2007.

Die Weiterentwicklung des OpenBIOS ist nach der Freigabe der anderen OpenFirmware–Versionen verlangsamt worden. Hier können Forth–affine Entwickler ein interessantes Projekt zur Mitarbeit finden, im OpenBIOS über die letzten Meter zur Versionsnummer 1.0 zu bringen. Das QEmu–System–Emulator Projekt[26] benutzt OpenBIOS, um für die SUN–Sparc–Emulation eine IEEE–1275–Umgebung bereitzustellen.

OpenBIOS war seit der Gründung als Projekt eng mit dem LinuxBIOS–(nun CoreBoot–)Projekt verbunden. Daher wundert es nicht, dass OpenBIOS als eines der möglichen *Payloads* mit dem CoreBoot in einen BIOS–Flashbaustein geschrieben werden kann. Zusammen mit CoreBoot ermöglicht OpenBIOS einen direkten Start des IEEE–1275–Forth–Systems aus dem BIOS–Flash–ROM.

Alternativ kann OpenBIOS auch über einen Bootloader (GRUB, Syslinux, FILO), oder als normales Programm unter Linux oder anderen Unix–Systemen gestartet werden. Mit OpenBIOS kann ein ISO–CD–ROM–Image mit OpenBIOS erstellt werden, welches dann zum Testen von OpenBIOS in Emulatoren wie QEmu oder VMWare benutzt werden kann.

[26] http://fabrice.bellard.free.fr/qemu/

OpenBIOS kann für die Architekturen x86, amd64, Xbox(1), SPARC und PPC erstellt werden. Als Zusatzprogramme beinhaltet OpenBIOS einen IEEE–1275–Tokenizer und De–Tokenizer, um Forth zu IEEE–1275–Tokencode (FCODE) umzuwandeln und Tokencode auch wieder zurück in Forth–Quellcode.

OpenFirmware (Firmworks)

OpenFirmware[27] ist die IEEE–1275–Version von Mitch Bradley (Firmworks[28]). OpenFirmware wird im OLPC XO (One Laptop per Child) als Start–Firmware benutzt und steht unter einer BSD–ähnlichen OpenSource–Lizenz zur Verfügung.

Die OpenFirmware–Quellen stellen mehrere Zielarchitekturen (ARM, x86, PPC ..) zur Verfügung und beinhalten eine Anzahl von Hardware– und Dateisystemtreibern (in Forth geschrieben).

OpenFirmware kann über verschiedene Wege gestartet werden:

- direkt im (Flash–)ROM
- über CoreBoot/LinuxBIOS
- über einen Booloader wie Grub oder Syslinux
- im Unix–Betriebssystem als normaler Prozess

Der derzeitige Fokus der OpenSource–Version ist der OLPC–Laptop. Alle anderen Architekturen und Ladevarianten (im OpenSource–Quellcode–Tree) sind derzeit nicht vollständig getestet und benötigen ggf. manuelle Änderungen und Aktualisierungen, bevor diese übersetzt und benutzt werden können.

OLPC–Laptops sind sehr schwierig zu bekommen, aber der kleine Linutop–Rechner[29] verfügt über eine recht ähnliche Hardwareausstattung (AMD–Geode–Chipsatz und –CPU) im Vergleich zum OLPC XO und ist ein gutes Testgerät für CoreBoot– und OpenFirmware–Experimente.

Smart Firmware (CodeGen)

Smart Firmware[30] der Firma CodeGen[31] ist eine in C geschriebene Implementierung von IEEE–1275, welche auch im Jahr 2006 unter einer OpenSource–Lizenz freigegeben wurde. Smart Firmware ist die Firmware–Basis des EFIKA–PPC–Boards, welches das Referenz–Board des PowerPC–Development–Forums ist.

Das EFIKA–Board wurde von der deutschen Firma bplan[32] für die US–Firma Genisi entwickelt. Das EFIKA–Board kann unter dem Namen *OpenClient* zusammen mit 2.5"–Harddisk, Grafikkarte und kleinem Gehäuse erworben werden[33][34]. Der *OpenClient* ist eine interessante Plattform, um Smart–Firmware auf PPC–CPU auszutesten.

In den Smart–Firmware–Quellen findet man als Besonderheit einen *C–zu–Forth*–Compiler[35], welcher aus C–Quellcode IEEE–1275–kompatiblen Forth–Quellcode erzeugt, welcher dann mit einem Tokenizer in FCODE (ByteCode) übersetzt werden kann.

OpenBOOT (SUN Microsystems)

OpenBOOT[36] ist die IEEE–1275–Implementierung von SUN Microsystems, welche sich in allen SPARC–SUN–Rechnern findet. Im Jahre 2006 hat SUN Microsystems die SUN4V–Architektur, besser bekannt unter dem Namen *SPARC–T1 Niagara*, unter eine OpenSource–Lizenz gestellt. Teil dieser Freigabe war nicht nur die Prozessorarchitektur, sondern auch die dazugehörige IEEE–1275–Implementierung für diese Prozessoren.

Die wenigsten Mitglieder der Forth–Gesellschaft werden eine T1–Sparc–Prozessor–Maschine besitzen, aber vielleicht gibt es den Prozessor in Kürze als FPGA, da Open Source. Die OpenBOOT–Quelldateien sind auf jeden Fall eine Besichtigung wert.

SLOF (SlimLine Open Firmware)

SlimLine Open Firmware, oder kurz SLOF[37], ist die jüngste OpenSource–IEEE–1275–Implementierung. SLOF wurde von der IBM für Cell–Prozessor–Maschinen erstellt.

Die Quellen von SLOF stehen auf den IBM–Seiten zum Download zur Verfügung, jedoch muss man sich für den Zugriff registrieren.

Forth auf alternativen, *mini*–32/64–Bit–Betriebssystemen

Auch Forth–Programmierer möchten nicht immer das Rad wieder neu erfinden (ok, jedenfalls nicht *jeder* Forth–Programmierer möchte das). Anstatt Forth auf großen, Mega– oder Gigabytes an Haupt– und Plattenspeicher verbrauchenden Betriebssystemen zu starten, kann Forth auch auf kleinen, sehr kompakten Alternativ–Betriebsystemen ausgeführt werden.

[27] http://openbios.org/Open_Firmware
[28] http://firmworks.com/
[29] http://www.linutop.com, http://www.linuxkistchen.de
[30] http://openbios.org/SmartFirmware
[31] http://codegen.com/
[32] http://www.bplan.de
[33] http://www.vesalia.de
[34] http://www.genisi.com
[35] http://www.codegen.com/SmartFirmware/ccfcode.html
[36] http://openbios.org/OpenBOOT
[37] http://www-128.ibm.com/developerworks/power/pa-slof/

Diese alternativen Betriebsysteme sind meist direkt in Assembler geschrieben und werden von einer kleinen, engagierten Gruppe von Entwicklern gepflegt.

Anbei eine Auswahl dieser alternativen Systeme:

- DexOS[38] — ein 100 KB Betriebssystem für i386+–Rechner. Es gibt schon eine Retro–Forth–Portierung
- KolibriOS[39] — ein kompaktes Betriebssystem aus Russland
- Solar_OS[40] — ein kompaktes Echtzeit–Betriebsystem mit GUI. Benötigt nur 8 MB RAM, für 32– or 64–Bit–x86–CPU.
- OcatOS[41] — ein kleines OS für 486 (und besser) mit 4MB Hauptspeicher, GUI.
- MenuetOS[42] — OS in Maschinensprache mit Ethernet und TCP/IP

Alle diese Mini–Betriebssysteme lassen sich von einer 3.5"–Diskette starten oder schnell in einer Virtualisierungslösung wie VMWare, VirtualBox oder QEmu testen.

Fazit

Eine ausführliche Anwort auf die kurze Frage von Fred aus der vorherigen VD. Leider gibt es sie nicht, die ganz einfache Lösung: „Nimm diese Software, klicke dort und hier, und in zwei Minuten bist Du am Ergebnis...".

Stattdessen gibt es eine reichhaltige Auswahl von möglichen Lösungen, in den verschiedensten Schwierigkeitsstufen. Eine Einarbeitung in das Thema ist immer notwendig.

Aber, hey, wir arbeiten mit Forth, *klicki–bunti* ist woanders. Wir wollen *verstehen*, was das System im Innersten zusammenhält. Oder nicht?

Auch Baron Münchhausen verstand sich auf den Bootstrap (Quelle: Wikimedia.org)

[38] http://www.dex4u.com/index.htm
[39] http://kolibrios.org/
[40] http://www.oby.ro/os/
[41] http://octavio.vega.fernandez.googlepages.com/octaos
[42] http://www.menuetos.net/

Widgets zum Anfassen - GUI-Skripting mit Forth und GTK+

Manfred Mahlow

PC, Laptop, Notebook, PDA - grafische Oberflächen überall. Da kommt früher oder später der Wunsch auf, schnell mal eine (einfache) grafische Benutzerschnittstelle (GUI) für ein kleines Tool oder für die Anzeige von Daten schreiben zu können, wie das z.B. mit der Skriptspache TCL/TK möglich ist. Aber, es sollte eben mit Forth möglich sein, mit möglichst kompakter *forthiger* Syntax.

Das war der Ausgangspunkt als ich begann, mir Gedanken über GUI-Programmierung mit Forth zu machen. Ein erstes Ergebnis, eine Programmierschnittstelle (API) für das GTK+-Toolkit, habe ich auf der Jahrestagung 2008 der Forth-Gesellschaft vorgestellt.

Warum GTK+ ?

GTK+ ist ein weit verbreitetes GUI-Toolkit, das für Linux und Windows verfügbar ist. Es ist die Basis für den GNOME- und den XFCE-Desktop, wird für Fenstermanager und viele Applikationen verwendet und ist auch auf PDAs zu finden.

GTK+ hat ein objektorientiertes Design, ist aber in C geschrieben und kann relativ leicht in andere Programmiersprachen eingebunden werden, so auch in Forth.

Eine GTK+-API für Forth

Die Implementierung der GTK+-API orientiert sich an folgenden Leitlinien:

1. Für eine erste Implementierung sollte ein möglichst kleines 32 Bit Forth System für Linux verwendet werden, leicht zu verstehen und leicht zu ändern.
2. Die Implementierung sollte objektorientiert erfolgen, mit möglichst kompakter *forthiger* Syntax.
3. Da das GTK+-Toolkit sehr umfassend ist, wird eine modulare Struktur der API angestrebt.
4. GTK+-Basisfunktionen und GTK+-Klassen werden als Quelltextmodule implementiert, die bei Bedarf geladen werden können.
5. Pro GTK+-Klasse wird nur eine kleine Teilmenge der Instanzvariablen und Methoden implementiert, die für einfache Anwendungen ausreicht. Weitere Instanzvariablen und Methoden können später hinzugefügt werden, wenn sie für eine Applikation tatsächlich benötigt werden.
6. Klassen, Instanzvariablen und Methoden erhalten in Forth, soweit sinnvoll, die gleichen Namen, wie in C.
7. Das Klassen-Präfix von Methodennamen wird in Forth wegen der objektorientierten Implementierung weggelassen.
8. Die GTK+ Methoden _get und _set werden in Forth auf fetch (@) und store (!) abgebildet.

Ein Forth für die GTK+-API

Auf der Suche nach einem möglichst kleinen leicht zu verstehenden und leicht zu ändernden 32 Bit Forth System für Linux bin ich auf Reva Forth von Ron Aron gestoßen, das weitgehend meinen Vorstellungen entsprach.

Aus Reva 6.0 habe ich dann cspForth abgeleitet, ein Testsystem für OOP mit Forth. Es unterstützt - wie Reva 6.0 - das bedarfsabhängige Laden von Quelltextmodulen, das Importieren von Funktionen aus Shared Libraries sowie das Speichern des jeweiligen Systemzustands als ausführbare Datei.

Darüber hinaus hat cspForth zwei besondere Eigenschaften, durch die es sich von den meisten anderen Forth Systemen unterscheidet:

1. cspForth unterstützt zwei Suchreihenfolgen, die bekannte Suchreihenfolge für Vokabulare und Wortlisten und eine zusätzliche Suchreihenfolge für Klassen, Objekte und Schnittstellen.
2. cspForth unterstützt Preludes. Ein Prelude ist ein Forth-Wort, das einem anderen Forth-Wort so zugeordnet ist, dass es im Ausführungsmodus oder im Compilermodus ausgeführt wird, bevor das Wort, dem es zugeordnet ist, selbst ausgeführt oder compiliert wird.

Diese beiden Eigenschaften sind die Basis für eine als Quelltextmodul ladbare OOP-Erweiterung, die *Context Switching Preludes* verwendet, um Objekte mit den Instanzvariablen und Methoden ihrer Klasse zu verbinden:

- Einem Objekt wird seine Klasse als Prelude zugewiesen. Wenn diese ausgeführt wird, erzeugt und aktiviert sie eine klassenspezifische Suchreihenfolge, die den Zugriff auf die Instanzvariablen und Methoden freigibt.
- Die Instanzvariablen einer Klasse sind selbst wieder Objekte einer Klasse, mit dem gerade beschriebenen Verhalten.
- Die Methoden einer Klasse haben ein anderes Verhalten. Einer Methode wird ein Prelude zugewiesen, das die klassenspezifische Suchreihenfolge, in der die Methode gefunden wird, wieder deaktiviert (oder ändert).
- Standard-Suchreihenfolge ist die Suchreihenfolge für Vokabulare und Wortlisten.

Stand der Implementierung

Die meisten einfachen Widget-Klassen des GTK+-Toolkits sind bereits implementiert und zu fast jeder Klasse gibt es mindestens ein Anwendungsbeispiel (siehe Abbildung 4). Weitere Widget-Klassen werden nach und nach implementiert. Ein Tutorium und Hilfetexte für die bereits implementierten Klassen sollen erstmal Vorrang haben.

Von C nach Forth - Ein Beispiel

Für das *Hello World Beispiel* aus dem GTK+-Tutorium ist im Listing 1 der Forth-Code dem C-Code gegenüber gestellt. Der Forth-Code ist mit cspForth ausführbar und erzeugt das GUI der Abbildung 1. Der C-Code ist als Kommentar enthalten.

Der C-Code beginnt in Zeile 3 mit einer include-Anweisung, welche die GTK+-Bibliotheken einbindet.

Auch in Forth werden die GTK+-Bibliotheken verwendet. Ihre Funktionen werden durch Laden der benötigten GTK+-Klassen importiert (Zeilen 5,6), die als Quelltextmodule implementiert sind.

Für das Beispiel werden die Klassen *GtkWindow*, für das Hauptfenster der Anwendung, und *GtkButton*, für den Knopf, benötigt.

Zeile 7 legt fest, dass nachfolgende Definitionen zum Vokabular *oop* hinzugefügt werden. Das Vokabular *gtk api* - es enthält die Wörter der GTK+-Schnittstelle - wird in die Suchreihenfolge aufgenommen und es wird die Zahlenbasis festgelegt.

GUI-Programme sind durch Ereignisse gesteuerte Programme. Wenn ein (bekanntes) Ereignis eintritt, wird ein bestimmter Programmcode ausgeführt.

In GTK+ ist dieser Mechanismus durch Signale und Signal-Handler (ausführbarer Code) implementiert, die an GtkWidgets gebunden werden. Trifft ein Signal ein, das mit einem Signal-Handler verknüpft wurde, wird dieser von GTK+ mit definierten Parametern aufgerufen.

Im betrachteten Beispiel soll auf die Ereignisse *Anwendungsfenster schließen* und *Hello World Knopf gedrückt* reagiert werden. Dazu müssen zwei Signal-Handler definiert werden.

In C werden sie als Funktionen des Typs static void definiert, in Forth als normale Forth Wörter, die dann einem Callback-Handler zugewiesen werden, der sie im Hintergrund (mit eigenem Stack) ausführt.

In Zeile 16 wird der Signal-Handler für den Knopf definiert und dem Callback-Handler *cb.hello* zugewiesen, der ihn mit zwei Parametern und einer Stacktiefe von 20 im Hintergrund ausführen wird.

Ganz analog wird in Zeile 25 ein Callback-Handler *cb.destroy* definiert, der beim Schließen des Anwendungsfensters ausgeführt werden soll. Er ruft dann die Funktion *gtk quit* auf (gtk_main_quit in C).

In Zeile 29 beginnt die Hauptfunktion des C-Programms. Bevor das GTK+-Toolkit verwendet werden kann, muss es initialisiert werden. In C erledigt das die Funktion gtk_init (Zeile 31). In Forth erfolgt die Initialisierung im Hintergrund, wenn die GTK+-API (*gtk api*) geladen wird.

In C wird auf GtkWidgets über Zeiger zugegriffen (Zeile 32,33). In Forth treten an die Stelle der Zeiger Objekte der jeweiligen Widget-Klasse. Für das Hauptfenster des Beispiels wird ein Objekt der Klasse GtkWindow erzeugt (Zeile 35) und für den Knopf ein Objekt der Klasse GtkButton (Zeile 36).

In Analogie zur Struktur des C-Codes fassen wir auch den Forth-Code zu einer Funktion (zu einem Wort) zusammen (Zeile 39).

In Zeile 47 wird das in Zeile 35 erzeugte Objekt *window* initialisiert. Die *init*-Methode erzeugt ein *GtkWindow* vom Typ GTK_WINDOW_TOPLEVEL als Hauptfenster des Programms und weist dessen Adresse - in Forth *widget identifier (wid)* genannt - dem Objekt zu.

In Zeile 48 werden das *destroy*-Signal und der Callback-Handler *cb.destroy* an das Hauptfenster *window* gebunden. Damit wird festgelegt, das der Callback-Handler *cb.destroy* (Zeile 25) ausgeführt wird, wenn das Fenster ein destroy-Signal erhält. GTK+ legt dann den *widget identifier (wid)* des Fensters und einen Datenzeiger (hier 0) auf den Stack des Callback-Handlers. Beide werden hier vom Signal-Handler aber nicht verwendet. Die *signal connect*-Methode übergibt einen Identifier auf dem Datenstack, der hier nicht gebraucht wird.

Zeile 54 ist ein Beispiel für den Zugriff auf eine Instanzvariable eines GtkWidgets. Die Breite des inneren Rands des Hauptfensters wird auf 10 Pixel gesetzt.

Der entsprechende C-Code in Zeile 52 mag im Vergleich etwas verwirrend sein, da auf das Hauptfenster vom Typ *GtkWindow* mit einer Methode der Klasse *GtkContainer* zugegriffen wird. Das liegt daran, dass die Instanzvariable *border-width* nicht in der Klasse GtkWindow sondern in der Klasse GtkContainer definiert ist.

Logisch erbt die Klasse GtkWindow von der Klasse GtkContainer. Da sich diese Vererbung aber in C nicht direkt abbilden lässt, muss eine Typumwandlung erfolgen und die Methodennamen müssen mit einem Klassen-Präfix versehen werden.

In C muss ein Programmierer also immer die GTK+-Klassenhierarchie beachten und die notwendigen Typumwandlungen und Methodenzuweisungen selbst vornehmen.

In Forth ist das einfacher. Die GTK+-Klassenhierarchie wurde hier bereits bei der objektorientierten Implementierung der API berücksichtigt. So erbt z.B. die Klasse GtkWindow auch alle Instanzvariablen und Methoden der Klasse GtkContainer.

In Zeile 63 wird das in Zeile 36 erzeugte Objekt *button* initialisiert. Die *init*-Methode erzeugt einen *GtkButton* mit der Beschriftung *Hallo World*, weist dessen Adresse

Abbildung 1: Der Hello-World–Dialog

- in Forth *widget identifier (wid)* genannt - dem Objekt zu und übergibt sie außerdem auf dem Stack. Die Phrase *window add* findet die Adresse auf dem Stack und packt den adressierten GtkButton in das Hauptfenster.

In der Zeile 64 werden das *clicked*-Signal und der Callback-Handler *cb.hello* an den GtkButton gebunden. So wird festgelegt, das der Callback-Handler *cb.hello* (Zeile 16) ausgeführt wird, wenn der GtkButton mit der Maus oder mit der Eingabetaste aktviert wird. GTK+ übergibt dann den *widget identifier (wid)* des GtkButton und einen Datenzeiger (hier 0) an den Callback-Handler. Beide werden hier vom Signal-Handler aber nicht verwendet.

Alle GtkWidgets sind nun erzeugt aber noch nicht zu sehen. Sie werden durch den Code in Zeile 70 sichtbar gemacht.

Die Methode *show all* ist in der Klasse *GtkWidget* definiert. Sie macht das angegebene Widget und alle darin enthaltenen Widgets sichtbar. In Forth erbt die Klasse GtkWindow diese Methode von der Klasse GtkWidget. In C muss wieder eine Typumwandlung erfolgen und dem Methodennamen muss das Klassen-Präfix vorangestellt werden(Zeile 68).

Mit den Zeilen 74 bis 76 wird der C-Code abgeschlossen. gtk_main ist die GTK+-Hauptschleife, die auf Ereignisse wartet und dann den zugeordneten Code ausführt. Sie wird erst dann verlassen, wenn die Funktion gtk_main_quit ausgeführt wird. Das geschieht durch den in Zeile 20 definierten Signal-Handler destroy, der ausgeführt wird, wenn das Hauptfenster geschlossen wird. Das C-Programm gibt dann die Kontrolle an die aufrufende Ebene zurück.

Der entsprechend Forth-Code folgt in den Zeilen 78 u. 79, allerdings mit abweichender Funktion:

Wenn der cspForth-Prozess, der den Forth-Code ausführt, in einem Terminal gestartet wurde, muss keine GTK+-Hauptschleife aufgerufen werden, weil die Ereignisverarbeitung dann im Hintergrund erfolgt, während auf die Eingabe einer Kommandozeile gewartet wird.

Wird der cspForth-Prozess nicht in einem Terminal gestartet, z.B. weil der Forth-Code per *Drag and Drop* direkt an cspForth übergeben wird, erfolgt keine Ereignisverarbeitung im Hintergrund. Dann muss auch in Forth explizit eine GTK+-Hauptschleife gestartet werden. Der Code in Zeile 78 stellt das sicher.

In Zeile 81 wird nun das gerade definierte Forth-Wort main ausgeführt. Es erzeugt und initialisiert die Widgets und stellt, wie gerade beschrieben, die Ereignisverarbeitung sicher.

Die nachfolgende Phrase *oop ??* ist nur relevant, wenn der cspForth-Prozess in einem Terminal gestartet wurde. Sie zeigt dann das Vokabular *oop* und den aktuellen Systemzustand im Terminal an (siehe Abbildung 2) und wir können interaktiv auf die GtkWidgets zugreifen - Widgets zum Anfassen.

In Forth ist ein GtkWidget ein Objekt. Im interaktiven Modus aktiviert ein Objekt seine klassenspezifische Suchreihenfolge (seine Klasenhierarchie) und übergibt seinen Identifier (oid) auf dem Datenstack. Mit der Methode ??? kann man sich beide anzeigen lassen (Abbildung 3).

Man kann also sehen, über welche Instanzvariablen und Methoden auf ein Objekt zugegriffen werden kann und man kann das auch tun!

Hinweis: Mit dem Wort .. kann man einen Klassenkontext wieder verlassen.

Also fröhliches Experimentieren, aber vorher besser die cspForth README-Datei lesen.

Es gibt auch eine kontextsensitive Hilfefunktion. Mit h <name> kann man sich den Glossary-Eintrag für das Worte <name> anzeigen lassen, vorausgesetzt, ein Glossary-Eintrag wurde bereits geschrieben, was noch nicht für alle Wörter der GTK+-API der Fall ist.

Zur Jahrestagung 2008 der Forth-Gesellschaft hatte ich einen Entwicklungsschnappschuss zusammengestellt, mit dem man in die GTK+-Programmierung mit Forth einsteigen kann. Das cspForth System mit der GTK+-API kann vom Wiki der Forth-Gesellschaft heruntergeladen werden. Kommentare, Anregungen, Kritik aber auch Mitarbeit sind sehr willkommen.

Das Beispiel mag den Eindruck vermittelt haben, dass man für die GUI-Programmierung mit Forth auch mit C vertraut sein muss. Das ist aber nicht generell der Fall. Soweit alle benötigten GTK+-Klassen in Forth bereits implementiert sind, muss man sich mit C nicht befassen. Nur wenn man die API erweitern will oder C-Beispiele übernehmen möchte, muss man sich mit dem Zusammenhang von C und Forth befassen.

Entfernt man den C-Code aus Listing 1 bleibt das sehr überschaubare Listing 2 über.

Wie soll es weitergehen?

Im Wiki der Forthgesellschaft werde ich ein Projekt für die GTK+-Programmierung mit Forth einrichten. Dort werde ich ein Tutorium, weitere Beispiele und dann und wann einen neuen Entwicklungsschnappschuss veröffentlichen.

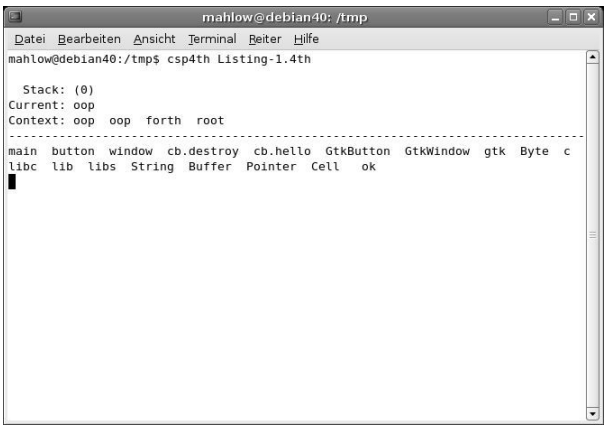


Abbildung 2: Status nach dem Laden von Listing 1

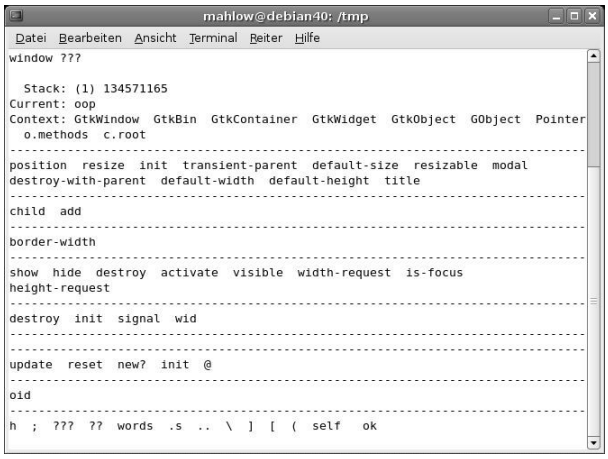


Abbildung 3: Klassenhierarchie des Hauptfensters

Weiter wäre es interessant, die GTK+-API auch für andere Forth Systeme zu implementieren, z.B. für ein in C implementiertes Forth, das nicht nur auf PC-Systemen läuft (der cspForth-Kern ist in Assembler geschrieben und damit nicht so einfach auf andere Prozessoren zu portieren). Ich denke dabei an PDAs, wie z.B. das Nokia N810. Wer hat daran Interesse?

Literatur

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

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

Listing 1:

```
1   \ Listing 1: GTK+-Beispiel "Hello World" in C und Forth
2
3   \ #include <gtk/gtkh>
4
5   needs GtkWindow
6   needs GtkButton
7   oop definitions gtk api decimal
8
9
10
11  \ static void hello( GtkWidget *widget, gpointer data )
12  \ {
13  \ g_print ("Hello World\n");
14  \ }
15
16  :: ( wid data -- ) ." Hello World" cr ;  2 20 cb cb.hello
17
18
19
20  \ static void destroy( GtkWidget *widget, gpointer data )
21  \ {
22  \ gtk_main_quit ();
23  \ }
24
25  :: ( wid data -- ) gtk quit ;  2 20 cb cb.destroy
26
27
28
29  \ int main( int argc, char *argv[] )
30  \ {
31  \ gtk_init (&argc, &argv);
32  \ GtkWindow *window;
```

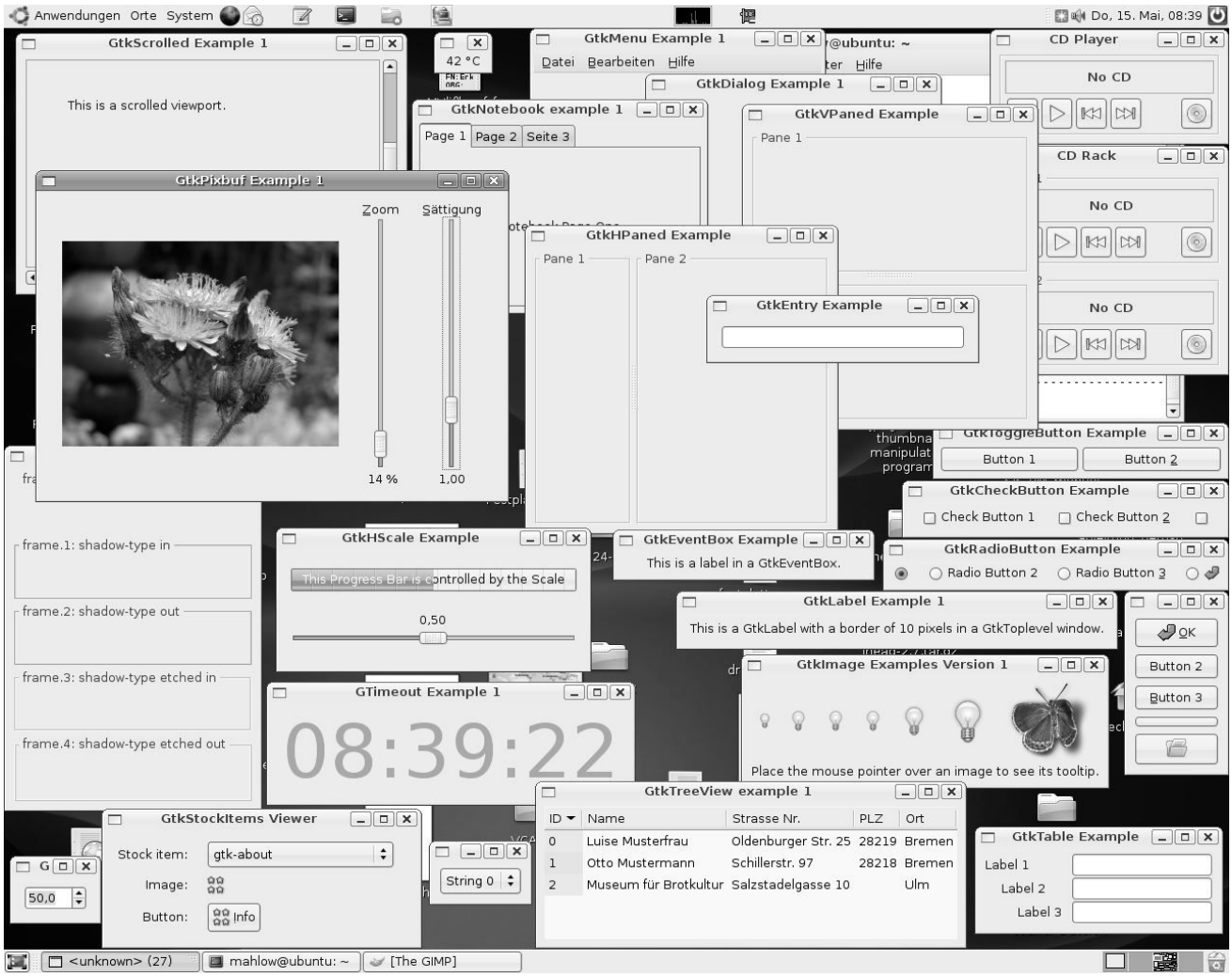


Abbildung 4: Beispiele implementierter GtkWidgets

```
33  \ GtkButton *button;
34
35  GtkWindow new window
36  GtkButton new button
38
39  : main ( -- )
40
41
42
43  \ window = gtk_window_new (GTK_WINDOW_TOPLEVEL);
44  \ g_signal_connect (G_OBJECT (window), "destroy",
45  \ G_CALLBACK (destroy), NULL);
46
47    GTK_WINDOW_TOPLEVEL window init
48    " destroy" cb.destroy 0 window signal connect drop
49
50
51
52  \ gtk_container_set_border_width (GTK_CONTAINER (window), 10);
53
54    10 window border-width !
55
56
57
```

```
58  \ button = gtk_button_new_with_label ("Hello World");
59  \ gtk_container_add (GTK_CONTAINER (window), button);
60  \ g_signal_connect (G_OBJECT (button), "clicked",
61  \ G_CALLBACK (hello), NULL);
62
63    " Hello World " button init window add
64    " clicked" cb.hello 0 button signal connect drop
65
66
67
68  \ gtk_widget_show_all (window);
69
70    window show all
71
72
73
74  \ gtk_main ();
75  \ return 0;
76  \ }
77
78    term? 0if gtk main bye then
79  ;
80
81  main  oop ??
```

Listing 2:

```
 1  \ Listing 2:  Gtk+-Beispiel "Hello World" in Forth
 2
 3  needs GtkWindow
 4  needs GtkButton
 5
 6  oop definitions  gtk api  decimal
 7
 8  :: ( wid data -- ) ." Hello World" cr ;  2 20 cb cb.hello
 9
10  :: ( wid data -- ) gtk quit ;  2 20 cb cb.destroy
11
12  GtkWindow new window
13  GtkButton new button
14
15  : main ( -- )
16      GTK_WINDOW_TOPLEVEL window init
17      " destroy" cb.destroy 0 window signal connect drop
18      10 window border-width !
19      "  Hello World " button init  window add
20      " clicked" cb.hello 0 button signal connect drop
21      window show all
22      term? 0if  gtk main bye  then
23  ;
24
25  main  oop ??
```

Protokoll der Mitgliederversammlung der Forth–Gesellschaft e. V. vom 27. April 2008

Thomas Prinz

Tagesordnung

1. Begrüßung
2. Wahl des Protokollführers
3. Wahl des Versammlungsleiters
4. Übergabe des Drachens
5. Bericht des Direktoriums und Kassenbericht
6. Wahl des neuen Direktoriums
7. Verschiedenes

Begrüßung

Ulrich Hoffmann eröffnet die Versammlung um 9 Uhr 20 und begrüßt die Mitglieder.

Wahl des Protokollführers

- Thomas Prinz erklärte sich bereit, die Protokollführung zu übernehmen. Er wurde ohne Gegenstimmen von der Versammlung gewählt.

Wahl des Versammlungsleiters

- Als Versammlungsleiter wird Winfried Clemens vorgeschlagen und ohne Gegenstimmen gewählt.
- Ulrich Hoffmann übergibt die Leitung der Versammlung an Winfried Clemens.
- Winfried Clemens stellt die Beschlussfähigkeit der Versammlung fest. Es sind 21 wahlberechtigte Vereinsmitglieder von 126 Mitgliedern anwesend.

Übergabe des Drachens

Anton Ertl, der amtierende Drachenträger, hält eine Laudatio für den vom Drachenrat gewählten nächsten Drachenträger. Manfred Mahlow erhält dieses Jahr den Drachen für seine unermüdliche Entwicklungsarbeit in objektorientierter Programmierung und Einbindung der GTK-Bibliothek in Forth-Systeme.

Bericht des Direktoriums

Kassenbericht

Rolf Schöne stellt den Jahresabschluss 2007 (Einnahmen und Ausgaben des Vereins) vor. Das Vereinsvermögen betrug zum 31.12.2007 18.276,87 Euro. Der Kassenbericht wird als Anlage dem Protokoll beigefügt.

Kassenprüfung

Die Kasse wurde von Egmont Woitzel geprüft. Er bestätigte eine ordentlich geführte Kasse, in der alle Buchungen nachvollziehbar sind und der Kassenabschluss korrekt ist.

Mitgliederstand

Rolf Schöne berichtet über die Mitgliederentwicklung: Im Laufe des letzten Jahres sind bis zu Beginn der Mitgliederversammlung 5 Mitglieder ausgetreten oder haben ihren Beitrag nicht bezahlt, 1 Mitglied ist verstorben und 4 neue Mitglieder sind beigetreten. Zum 27.4.2008 zählt die Forth–Gesellschaft 126 Mitglieder.

Auslandskontakte

Fred Behringer berichtet über Auslandskontakte: Es bestehen weiterhin Kontakte zur Gruppe um Henry Vinerts in Silicon Valley und zu den Holländern.

Forth-Magazin VD und Internet-Präsenz (Ulrich Hoffmann)

Ulrich Hoffmann berichtet über das letzte Jahr der Redaktionsarbeit für die Vierte Dimension und ruft zur Einreichung von Beiträgen für die kommenden Ausgaben auf. Die Vierte Dimension wird federführend von Ulrich Hoffmann redigiert und gesetzt. Tatkräftige Unterstützung erhält er von Fred Behringer und Michael Kalus, den Druck und Versand organisiert Rolf Schöne im Forth–Büro. Ulrich spricht ihnen seinen ausdrücklichen Dank aus, betont aber, dass eine breitere Unterstützung wünschenswert wäre. In 2007 hat es drei reguläre Ausgaben der Vierten Dimension gegeben (1, 2, 3+4) und ein AVR–Sonderheft. Rückmeldungen zu den Ausgaben gibt es so gut wie gar nicht. Insgesamt wäre eine intensivere Kommunikation und auch die Mitarbeit an den kommenden Ausgaben der Vierten Dimension sehr zu begrüßen. Die technischen Voraussetzungen würden bestehen. Druck und Versand der Vierten Dimension wird ab Ausgabe 2/2008 durch das neue Forth–Büro bei Andrea Rieger erfolgen.

Ulrich Hoffmann berichtet weiterhin über die Internet-Präsenz www.forth-ev.de, die im Jahr 2007 stabil und zuverlässig zur Verfügung stand. Auch hier stellt er fest, dass nur ein kleiner Kreis von Personen aktiv an der Gestaltung der Web–Seite teilnimmt und eine umfangreichere Beteiligung wünschenswert sei. Ein weiteres Thema des Jahrs 2007 sei die Bewertung von www.forth-ev.de durch das Google–Ranking gewesen, das extrem

Bericht - Einnahmen und Überschussrechnung	St.-Nr. 143/215/10249 FORTH-Gesellschaft e.V. 01.01.2007 bis 31.12.2007
Kategoriebeschreibung	
BETRIEBSEINNAHMEN	
Einnahmen VD-Verkauf Vierte Dimension	2.014,79
Mitgliedsbeitrag-Beitrag für lfd. Jahr	2.434,00
Porto Einnahmen-Porto aus Abos	5,00
Spende-Spenden, Wohltätigkeit	54,36
Steuererst.-Steuererstattungen	746,44
Zinseinkünfte-Zinseinkünfte:	
Zinsabschlag-Freigestellte Zinsen	0,00
Zinseinkünfte-Zinseinkünfte-Andere	483,58
GESAMT Zinseinkünfte-Zinseinkünfte	**483,58**
VON Mehrwertsteuer	141,93
GESAMT BETRIEBSEINNAHMEN	**5.880,10**
BETRIEBSAUSGABEN	
Abschreibungen-Abschreibungen auf Investitionsgüter	364,75
Bürobedarf B-Büromaterial ohne MwSt.	2,00
Bankgebühren-Kontoführungsgebühren	416,66
Druckkosten-Druckkosten VD	1.085,10
Fracht VD-Frachtkosten für VD	146,97
Internet-Kosten der Internetpräsenz	379,90
Jahrestagung (A)-Kosten für Ausrichtung Jahrestagung	664,30
Porto-Porto, Postgebühren:	
Porto VD-Portokosten für Versand VD	477,80
Porto Verwaltung-Portokosten Verwaltung	15,40
GESAMT Porto-Porto, Postgebühren	**493,20**
Projekte-Ausgaben zur Unterstützung des Vereinsziels	322,49
Steuer-Steuern	2.757,09
Versand VD-Versand VD	64,74
Verwaltung-Kosten für Vereinsverwaltung	892,00
Kategorielos: Ausgaben	0,00
AUF Mehrwertsteuer	318,68
GESAMT BETRIEBSAUSGABEN	**7.907,88**
GESAMTSUMME	**-2.027,78**

Einnahmen und Überschuss 2007

Bericht - Vermögen	St.-Nr. 143/215/10249 FORTH-Gesellschaft e.V.
Konto	Stand 31.12.2007
VERMÖGEN	
Bargeld und Bankkonten	
Postgiro	741,53
GESAMT Bargeld und Bankkonten	741,53
Andere Vermögen	
Drucker FS1900	0
Notebook MD 41300	0
Postbank Business Festgeld	17.535,34
GESAMT Andere Vermögen	17.535,34
GESAMT VERMÖGEN	18.276,87
VERBINDLICHKEITEN	
Andere Verbindlichkeiten	
Mehrwertsteuer	0
GESAMT Andere Verbindlichkeiten	0
GESAMT VERBINDLICHKEITEN	0
GESAMTSUMME	**18.276,87**

Übersicht über das Vereinsvermögen

schlecht war. Durch Umstrukturierung der Inhalte versuche man der Bewertung entgegenzuwirken, auf die Platzierung in Suchergebnissen habe man aber nur indirekten Einfluss[1]. Neben der Internet–Präsenz wird der bei Strato gemietete Server auch als Repository für die Vierte Dimension und die Protokolle der wöchentlich stattfindenden IRC–Fernkonferenzen genutzt. Für die Produktion der Vierten Dimension ist der Server unabdingbar.

Vereinsinterne Entwicklungen

Ewald Rieger berichtet, dass Rolf Schöne zum Ende dieser Tagung das Forth-Büro nach 5–jähriger Betreuung an Frau Andrea Rieger übergibt. Damit wird ein Umzug des Forth–Büros von München nach Bobenheim-Roxheim notwendig. Die laufenden Geschäfte wurden während der Tagung an Frau Andrea Rieger zu gleichen vertraglichen Bedingungen übergeben. Die neue postalische Anschrift des Vereins lautet: Forth-Gesellschaft-e.V. Postfach 32 01 24 68273 Mannheim.

Außendarstellung und Projekte (Bernd Paysan)

Bernd Paysan berichtet über die Buchprojekte:

- Thinking Forth: Ca. 40000 Downloads, (nach Rückfrage mit Leo) ca. 400 verkaufte Bücher
- Starting Forth: Marcel Hendrix' HTML–Version mit den original–Bildern auf der Forth–Inc.–Seite

Für laufende Projekte wird die Zuständigkeit geklärt:

- E. Wälde — AVR AM–Forth
- B. Paysan / U. Hoffmann / M. Kalus / W. Clemens — Stand LEGO Roboter
- A. Ertl / B. Paysan Was machen die IT–Studenten? Weitere Diskussion hierzu auf *de.comp.lang.forth*
- U. Hoffmann, B. Paysan — FPGA b16/microcore
- A. Ertl — Forth Repository

Entlastung des Direktoriums

Das Direktorium wurde ohne Gegenstimme entlastet.

Wahl des Direktoriums

Das derzeitige Direktorium stellte sich zur Wiederwahl und wurde einstimmig von der Versammlung gewählt. Die Mitglieder Ulrich Hoffmann, Bernd Paysan und Ewald Rieger nehmen die Wahl an.

[1] Im Moment Platz 3 hinter den Wikipedia–Einträgen von Forth und der Forth–Bridge

Verschiedenes

- In den vergangenen Jahren gab es häufig Terminüberschneidungen der Forth-Tagungen mit anderen wichtigen Veranstaltungen (HMI/Vintage–Computer) oder mit beruflichen Belangen einiger Mitglieder. Es wurde vorgeschlagen, den Zeitpunkt der Tagungen in einem größeren Zeitfenster zu streuen.

- Das hohe Vereinsvermögens soll durch den Erwerb von Lizenzen für Forth-Bücher — vorzugsweise deutschsprachig — und den Kauf von Mac-Mini zum Portieren von G-FORTH auf Macintosh gemäß der Satzung abgebaut werden.

- Die unter Projekte genannten Aktionen sollen zur Mitgliederwerbung durch den Verein finanziell gefördert werden.

- Die nächste Tagung wird von Carsten Strotmann in der Nähe seines Heimatortes Neuenkirchen durchgeführt.

- Im nächsten Jahr feiert die Forth–Gesellschaft ihr 25–jähriges Bestehen. Die anwesenden Mitglieder schlugen vor, einen Festausschuss zur Gestaltung der Feier zu gründen.

Um 12:11 Uhr wird die Versammlung von Winfried Clemens geschlossen.

gez. Thomas Prinz (Protokollführer)

Auf der Forth–Tagung 2008 (Photo: Bernd Paysan)

Projekt Euler - Problem 9

Michael Kalus

Vorsicht, die mathematischen Knobeleien in dem Projekt Euler machen süchtig. Hier eine weitere Lösung. Mögen noch viele in Forth folgen.

Die Aufgabe

Ein pythagoreisches Triplet ist ein Satz natürlicher Zahlen, $a < b < c$, für den $a^2 + b^2 = c^2$ gilt.

Zum Beispiel $3^2 + 4^2 = 9 + 16 = 25 = 5^2$.

Es existiert genau ein pythagoreisches Triplet, für das gilt: $a + b + c = 1000$

Finde das Produkt *abc*.

Quelle: `http://projecteuler.net/`

Eine Lösung

```
1   decimal
2   vocabulary euler
3   euler definitions
4
5   \ Satz des Pythagoras: a^2 + b^2 = c^2
6   : pyt ( a b -- c^2 )
7     dup * swap dup * + ;
8
9   \ Quadratwurzel ziehen: Heron-Verfahren (Heron von Alexandria)
10  \ Die Iterationsvorschrift lautet: xn+1 = ( xn + a/xn ) / 2
11  \ In RPN:  a xn / xn + 2 / --> xn+1
12  \ Weil nur die natürlichen Werte der Wurzel aus c^2 gebraucht werden,
13  \ wird ganzzahlig kalkuliert.
14  \ Quelle: forth-ev wiki
15  \ http://www.forth-ev.de/wiki/doku.php
16  : sqrt ( a -- s )
17          DUP 0= ?EXIT
18          DUP >R
19          BEGIN
20           R@ OVER / OVER + 2/
21       DUP ROT - ABS 1 <=
22      UNTIL R> DROP ;
23
24  \ Wir brauchen nur die Werte von c, für die c eine natürliche Zahl ist.
25  : nsqrt? ( a b -- f )
26    pyt
27    dup sqrt         ( -- n s )
28    dup *            ( -- n s*s )
29    - 0= ;           ( -- f )  \ if zero, s is natural number.
30
31  \ Ist a+b+c=1000?
32  : 1000?  ( a b  -- f )
33    2dup pyt sqrt
34    + + 1000 = ;
35
36  \ Hübsche Ausgabe der gefundenen Lösungen (!)
37  : .solution ( a b -- )
38    cr ." -------------------"
39    cr 2dup swap ." a=" . ." b=" .
40       2dup pyt sqrt ." c=" .
41    cr 2dup 2dup pyt sqrt + + ." a+b+c=" .
```

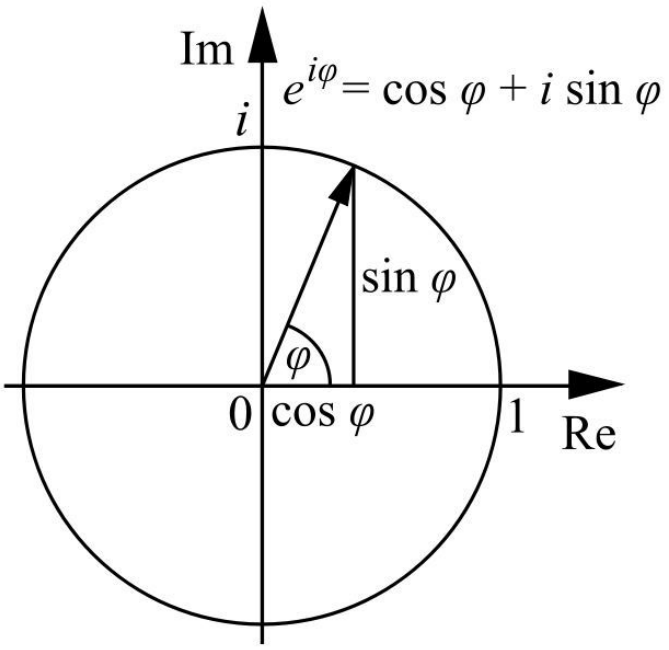
```
42    cr 2dup pyt sqrt * * ." abc=" . ;
43
44  \ Teste a,b bis zum Wert a=b=xx
45  variable xx  1000 xx !
46  : p9  ( -- )
47    xx @  0 DO
48      xx @  0 DO
49        i j nsqrt? IF
50          i j 1000? IF i j .solution THEN
51        THEN
52      LOOP
53    LOOP
54    cr ." --------------------" ;
55
56  words cr cr
```

Und zu diesem Thema schreibt Michael auch noch in einem Leserbrief:

Projekt Euler

Im Projekt Euler ist unter den Top 1000 Scorers niemand, der Forth benutzt. Jungs, das können wir nicht auf uns sitzen lassen! Wie könnten wir das organisieren?

Grüße, Michael



Ein einfaches Quellcode–Bibiliothekssystem

Ulrich Hoffmann

Zusammenfassung: Der ANS–Forth–Standard ermutigt uns dazu, portablen Code zu schreiben. Zum ersten Mal in der Forth–Geschichte ist es möglich, substantielle Mengen Software zu entwickeln, die nicht nur auf einem einzigen Forth–System laufen kann. Wiederverwendbare Teile von Applikationen können in portablen Bibliotheken zusammengefasst werden.

Diese Möglichkeit, portable Bibliotheken schreiben zu können, wirft die Frage auf, wie sie auf einfache und dennoch effiziente Weise verwaltet werden können.

Dieser Artikel beschreibt ein einfaches System, Bibliotheken auf Quellcode–Ebene zu behandeln. Seine Einfachheit macht es leicht verständlich und gut handhabbar. Dieses Bibiliothekssystem ist seit 1995 auf unterschiedlichen ANS–Forth–Systemen im Einsatz und dient als Grundlage einer kontinuierlich wachsenden Bibliothek von Standard–Forth–Definitionen.

Schlüsselwörter: Quellcode, Bibliothek, Interpretierer, bedingte Übersetzung

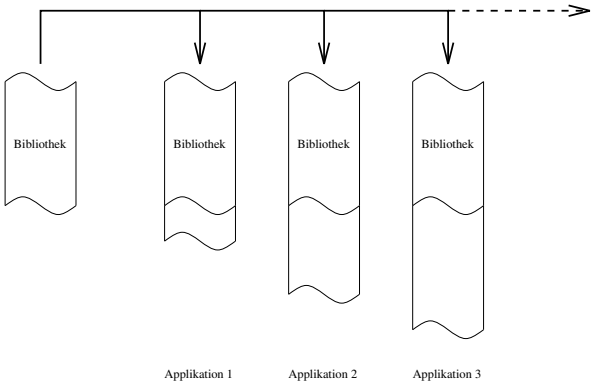
Einleitung

Ein Vorteil den der ANS–Forth–Standard der Forth–Gemeinde bringt, ist die Möglichkeit, Algorithmen in einer wohldefinierten und portablen Notation [1] formulieren zu können. Dies ermöglicht, Standard–Bibliotheken anzulegen, die auf einem breiten Spektrum Standard–konformer Forth–Systeme benutzt werden können. Solche Bibliotheken können die unterschiedlichsten Arten von Funktionalität bereitstellen. Sie können numerische Algorithmen enthalten, wie etwa im *Scientific Forth Library*–Projekt [2], häufig verwendete Datenstrukturen, wie beispielsweise in der *FFL* [7] oder anderes. Allerdings ist zu beachten, dass der ANS–Forth–Begriff des *Standard–Programms* Portabilität nur auf Quellcode–Ebene definiert. Derzeit gibt es keine Ansätze, die Portabilität auf Zwischencode– oder Maschinencode–Ebene zu behandeln. Auch dieser Artikel konzentriert sich auf die Verwaltung von Quellcode–Bibliotheken.

Der Kopier–Ansatz

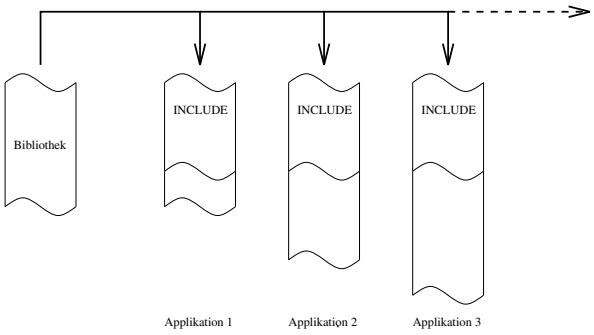
Der einfachste Ansatz, Applikationen auf Basis von Quellcode–Bibliotheken zu bauen, besteht darin, den Quellcode der Bibliothek zur Konstruktionszeit der Applikation vor dem Applikations–Quellcode textuell einzufügen. (Man nennt ihn auch *Call–by–Editor*–Ansatz.)

Dieser Ansatz hat jedoch einen Nachteil. Der Bibliotheks–Quellcode wird für jede Applikation kopiert. Aus Wartungsgesichtspunkten ist das nicht wünschenswert, weil Verbesserungen im Bibliothekscode in jeder Kopie vorgenommen werden müssen. Höchstwahrscheinlich werden die kopierten Bibliotheken auseinanderlaufen und die Gesamtqualität der Bibliothek wird sich nicht wirklich verbessern.[1]



Die Bibliothek zur Konstruktionszeit der Applikation ganz oder teilweise kopieren

Dieser Nachteil kann gelöst werden, indem man den Bibliotheks–Quellcode eben nicht zur Konstruktionszeit der Applikation einkopiert, sondern ihn mit `INCLUDE` einfügt, also den Kopierprozess effektiv auf die Übersetzungszeit verschiebt. Dies zentralisiert den Bibliotheks–Code an einer Stelle und ermöglicht eine angemessene Wartung (vermutlich mit Hilfe einer Versionsverwaltung).



Auf die Bibliothek mit `INCLUDE` zugreifen

[1] Es kann durchaus sinnvoll sein, diesen Ansatz zu wählen, etwa wenn die verschiedenen Konfigurationen für individuell angepasste Geräte verwaltet werden sollen. Jedes Gerät wird dann mit dem vollständigen Quellcode der auf ihm laufenden Software — inklusive der Bibliotheken im jeweiligen Wartungszustand — ausgestattet.

Format für die gesamte Bibliothek:

```
<Bibliotheks-Kopf-Kommentar>
<Verarbeitung des Identifikations-Strings>
 CR .( looking for ) 2DUP TYPE SPACE
<Bibliotheks-Fragmente>
CR TYPE .(  NOT provided! )  QUIT
```

Format für jedes einzelne Fragment:

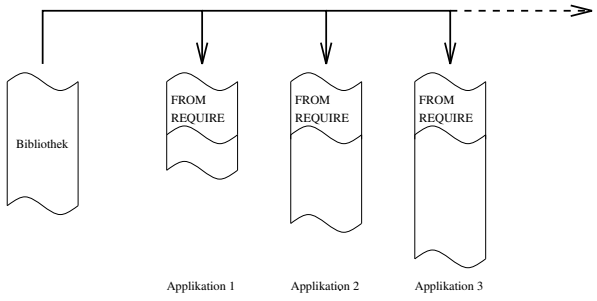
```
desires <Identifications-String> [IF] \\ -----------
      <Definitionen um, Identifikations-String bereitzustellen>
\\ [THEN]
```

Abbildung 1: Vorschlag für die Formatierung von Quellcode–Bibliotheken mit sequentieller Suche

Der `INCLUDE`–Ansatz hat allerdings einen anderen Nachteil. Er fügt immer den gesamten Bibliotheks–Code in den Applikations–Code ein, obwohl die meisten Applikationen nur einen Teil der gesamten Bibliothek benötigen. Wir könnten diesen Nachteil überwinden, indem wir ausgewählte Fragmente der Bibliothek in unseren Applikations–Code kopieren — aber dann sind wir wieder beim obigen Kopier–Ansatz mit seinem Nachteil.

Der selektive `INCLUDE`–Ansatz

Die Schlüsselidee, dieses Bibliothekverwaltungs–Problem zu überwinden, ist, selektiv nur Fragmente der Bibliothek einzufügen. Dann enthält jede Applikation nur die wirklich von ihr benötigten Teile. Und — der Bibliotheks–Code selbst steht an einer zentralen Stelle und kann einfach gewartet werden.



Selektiv auf die Bibliothek zugreifen

Obwohl ein solcher Mechanismus in ANS–Forth nicht standardisiert ist, kann er leicht hinzugefügt werden.

Eine Forth–Eigenschaft hilft hier: die Möglichkeit, benutzerdefinierte Funktionen auszuführen, während man Quellcode lädt. Das erlaubt uns, Text benutzerdefiniert zu verarbeiten — und das neben dem eigentlichen Laden des Quellcodes. Selektiv Fragmente einer Quellcode–Bibliothek zu laden, ist lediglich eine spezielle Verarbeitung von Text.

Wir könnten nun ausgefeilte Bibliotheks–Suchfunktionen definieren, die, im Applikations–Code ausgeführt, die gewünschten Fragmente der Bibliothek finden und sie laden. Das werden wir hier aber nicht tun! Statt dessen machen wir die Bibliothek selbst für die Bibliothekssuche verantwortlich, ähnlich wie bei *Bedingter Übersetzung*. Wir definieren zunächst nur die Schnittstelle, über die Applikationsprogramme eine solche intelligente Bibliothek verwenden. Das hat den Vorteil, dass die Bibliothekssuche für jede einzelne Bibliothek individuell optimiert werden kann.

Die Bibliotheks–Schnittstelle

Um selektiv ein Fragment einer Quellcode–Bibliothek zu laden, hinterlegt die Applikation einen *Identifikations–String* auf dem Datenstack und lädt dann den gesamten Bibliotheks–Quellcode. Anhand des Identifikations–Strings kann die Bibliothek dann entscheiden, welches Fragment die Applikation benötigt, und es selektiv laden.

```
( c-addr u -- )
```

Da die Bibliothek die Bibliothekssuche selbst vornimmt, können wir speziell zugeschnittene Such–Algorithmen je nach Bibliothek einsetzen, die so einfach oder so effizient sein können, wie benötigt (sequentielle, binäre, hash–Suche, usw.). Die einzige Anforderung an die Bibliothek ist, die Schnittstelle zu befriedigen und den übergebenen Identifikations–String zu konsumieren.

Weil Teile der Bibliothek abhängig von anderen Bibliotheken (oder anderen Teilen derselben Bibliothek) sein werden, muss es möglich sein, Bibliotheken rekursiv und verschachtelt zu verwenden. Das ermöglicht die Konstruktion von Bibliotheks–Hierarchien.

Eine Beispiel–Implementierung

Die hier präsentierte Idee nimmt an, dass Quellcode in sequentiellen Files gehalten wird. Obwohl das Bibliotheks–System nicht mit Quellcode–Blöcken getestet wurde, sollte eine entsprechende Anpassung problemlos möglich sein.

Die einfachste Implementierung wäre wohl, den Identifikations–String direkt auf den Datenstack zu legen und dann mit `INCLUDE` die Bibliothek zu laden:

```
<Identifikations-String> S" <filename>" INCLUDED
```

Aber — hier gibt es ein Problem, denn `S"` legt den String, den es parsed, an einer temporären Stelle im Speicher ab[2] und ein Standard–Forth–System darf mit einem zweiten `S"` diesen temporären Puffer überschreiben.

Es ist also nötig, den Identifikations–Stings in einem sicheren Bereich zu speichern. Wir versehen das mit ein wenig syntaktischem Zucker, der uns eine einfachere Handhabung erlaubt, und schlagen folgende Syntax vor, mit der eine Applikation auf eine Quellcode–Bibliothek zugreift:[3]

```
{ FROM <Bibliotheksname>
  { REQUIRE <Identifikations-String> }*
}*
```

Es gibt einen gewissen *Overhead* durch die Definition dieser Worte, die ja jede Applikation enthalten muss. Auf Systemen, die transiente Definitionen erlauben, kann dieser Overhead vermieden werden.

Der Quellcode in Listing 3 auf Seite 32 definiert das einfache Bibliotheks–System.

Bibliotheks–Formatierungs–Vorschlag

Der Bibliotheks–Quellcode führt selbst die Suche der zu ladenden Fragmenten durch. Es ist sinnvoll, ihn je nach Suchverfahren in geeigneter, spezieller Weise zu formatieren.

Der Formatierungs–Vorschlag in Abbildung 1 auf der gegenüberliegenden Seite zeigt, wie man den Bibliotheks–Quellcode für eine sequentielle Suche formatieren kann.

Zunächst gibt es den Bibliotheks–Kopf, der den Namen und die Version sowie das Veröffentlichungsdatum der Bibliothek enthält. Dann folgt ein wenig Code, um den Identifikations–String zu verarbeiten. Typischerweise wird dieser Teil ermitteln, ob die betreffenden Definitionen bereits geladen wurden, und in diesem Fall die Bibliotheks–Suche vermeiden. Anschließend wird für jedes Fragment festgestellt, ob es geladen werden soll. Das erste zutreffende Fragment wird geladen und dann das Laden des restlichen Files mit \\ abgebrochen. Kann schließlich kein geeignetes Fragment gefunden werden, wird eine Meldung, z. B. `not provided`, ausgegeben und das selektive Laden bricht mit einem Fehler ab.

Verwandte Arbeiten

Der erste bekannte Versuch zur modularen Programmierung in Forth wurde 1980 von D. Val Schorre [6] unternommen. Sein Ansatz entfernt die Namen interner Worte durch *Auslinken* aus dem Forth–Dictionary und beschränkt auf diese Weise ihre Benutzung außerhalb des Moduls. Diese Idee wurde von mir in [4] verfeinert und entfernt — im Kontext moderner Forth–Systeme, die temporäre Namen erlauben — die Namen der internen Worte vollständig.

Die von S. Pelc und N. Smith beschriebene DOS–basierte Forth–Implementierung *Modular Forth* erlaubt das Linken von Modulen auf Objektcode–Ebene [3].

Im Gegensatz zu dieser Arbeit, bei der die Applikation selbst ein (spezielles) Modul ist und also Applikations–Programme und Systemmodule ähnlich strukturiert sind, berücksichtigt der hier vorgestellte Ansatz die unterschiedlichen Rollen, die Systembibliotheken und Anwendungsprogramme im Software–Konstruktionsprozess haben. Als Konsequenz sind hier Applikations–Programme und Systembibliotheken unterschiedlich strukturiert.

Einige frühe Arbeiten über Quellcode–Bibliotheken wurden 1985 von J. James [5] durchgeführt.

Zukünftige Entwicklungen

Ich habe begonnen, Bibliotheks–relevanten Quellcode, der in vielen Forth–Dialekten formuliert war, Schritt um Schritt in ANS–Forth umzuwandeln, um eine Basis für künftige Applikationen zu schaffen. Mit mittlerer Priorität entstehen auf diese Weise ANS–Forth–Standard–Bibliotheken für Stringverarbeitung, Speicherverwaltung, Zustandsautomaten, Datenstrukturen, Kommunikation und anderes.

Ausblick

Das vorgeschlagene einfache Quellcode–Bibliothekssystem ist bei mir seit vielen Jahren in Benutzung und hat sich als angenehm zu benutzen herausgestellt. Um eine Standard–Anwendung zu konstruieren, kann ich sie ganz einfach auf vorhandene Komponenten aufbauen, indem ich lediglich ihre Abhängigkeiten benenne. Die Programmierung von Standard–Anwendungen wird so viel leichter.

Um allerdings Standard–Forth–Code zu *veröffentlichen*, muss ein zusätzlicher Schritt gemacht werden, denn die verwendeten Bibliotheks–Routinen sind anderen eventuell gar nicht bekannt. Es ist dann nötig, die Verweise auf Bibliotheks–Routinen durch die geladenen Definitionen zu ersetzen. Diese zusätzliche Arbeit bei der Veröffentlichung von Programmen wird durch die Einfachheit der Entwicklung mit Bibliotheken mehr als kompensiert.

[2] Siehe [1] 11.6.1.2165 Definition von `S"` im *Optional File Access word set*.

[3] Die Verwendung des Namens `REQUIRE` haben wir auf der Forth–Tagung 2008 heftig diskutiert, da sie in Konflikt mit dem Forth–2000x–Wort gleichen Namens steht. Über Vorschläge für eine konfliktfreie, eingängige Benennung der vorgestellten Worte wäre ich sehr dankbar.

Literatur

[1] dpANS: *Draft proposed American National Standard for Information Systems – Programming Languages – Forth* Secretariat Computer and Business Equipment Manufacturers Association, American National Standards Institute, Inc., 1993

[2] *Scientific Forth Library* der aktuelle Status kann unter `http://www.taygeta.com/fsl/sciforth.html` eingesehen werden.

[3] *Modular Forth: Import, Export and Linking*, Stephen Pelc and Neil Smith, 1986 FORML Conference Proceedings, Forth Interest Group, San Jose, 1987

[4] *Module Forth*, Ulrich Hoffmann, 1989 FORML Conference Proceedings, Forth Interest Group, San Jose, 1990

[5] *A Forth Component Library for Off–The–Shelf Distribution of Modules*, John S. James, 1985 FORML Conference Proceedings, Forth Interest Group, San Jose, 1986

[6] *Adding MODULEs to Forth*, D. Val Schorre, 1980 FORML Conference Proceedings, Forth Interest Group, San Jose, 1981

[7] *Forth Foundation Library*, `http://ffl.dvoudheusden.net/`

Glossar

In Applikations–Programmen, um auf die Bibliothek zuzugreifen:

`from` `( <spaces>ccc<spaces> -- )`
from wird in der Weise `from` <Bibliotheks-Name> verwendet, um die aktuelle Bibliothek auf den Wert <Bibliotheks-Name> zu setzen. <Bibliotheks-Name> ist der Name des Files, das den Bibliotheks–Quellcode enthält.

`required` `( c-addr len -- )`
required wird verwendet, um die durch den Identifikations–String c-addr len bezeichnete Definition (und ihre abhängigen Definitionen) aus der aktuellen Bibliothek zu laden. Der Identifikations–String ist typischerweise der Name einer Definition.

`require` `( <spaces>ccc<spaces> -- )`
require ist eine Präfix–Version von **required**. Sie wird in der folgenden Form verwendet: `require` <identification-string>.

In Bibliotheken:

`\\` `( -- )` immediate
\\ überspringt allen verbleibenden Quellcode im aktuell geladenen File.

`desired` `( c-addr1 len1 c-addr2 len2 -- c-addr1 len1 false | true )`
desired vergleicht die beiden durch c-addr1 len1 und c-addr2 len2 gegebenen Strings und liefert true, wenn beide Strings die gleichen Zeichen in der gleichen Reihenfolge enthalten. Wenn die Strings nicht gleich sind, liefert es c-addr1 len1 und false.[4]

`desires` `( <space>ccc<spaces> c-addr len -- c-addr len false true )|`
desires ist eine Präfix–Version von **desired**. Es wird in folgender Form verwendet: `desires` <Identifications-String>

[4] Es ist also ähnlich zum Wort **case?**, das folgendermaßen definiert werden kann:
`: case? ( x1 x2 - x1 false | true ) OVER = DUP IF NIP THEN ;`
Man beachte aber den Unterschied, dass **desired** Strings und keine Zell–Werte vergleicht.

Listing 1: Eine einfache Bibliothek mit sequentieller Suche

```
1   \ SAMPLE.LIB: Sample Library with sequential search  uho Mar95
2
3
4   ( c-addr u -- )
5
6
7   \ do not look up, if identification string is already
8   \ in dictionary
9
10  2DUP PAD CHAR+ SWAP MOVE DUP PAD C!
11  PAD FIND NIP [IF]  2DROP \\ [THEN]
12
13
14
15  CR .( looking for ) 2DUP TYPE SPACE
16
17
18  desires case? [IF] \\ ----------------------------------------
19
20  : case? ( flag -- ) OVER = DUP IF  NIP  THEN ;
21
22  \\ [THEN]
23
24
25  desires on [IF] \ ----------------------------------------
26
27  : on ( a-addr -- )  TRUE SWAP ! ;
28
29  \\ [THEN]
30
31
32  desires off [IF] \ ----------------------------------------
33
34  : off ( a-addr -- )  FALSE SWAP ! ;
35
36  \\ [THEN]
37
38
39  \ ------------------------------------------------------------
40
41  CR TYPE .(  NOT provided! )  QUIT
```

Listing 2: Eine Beispiel–Applikation

```
1   \ Sample Application Program
2
3   from sample.lib require on
4                    require off
5                    require case?
6
7   : cursormove ( k -- f )
8           [CHAR] j case? IF down  EXIT THEN
9           [CHAR] k case? IF up    EXIT THEN
10          [CHAR] h case? IF left  EXIT THEN
11          [CHAR] l case? IF right EXIT THEN
12          #insert  case? IF insert-mode on EXIT THEN
13
14          ... ;
```

Listing 3: Quellcode des einfachen Quellcode–Bibliothekssystems

```
\ LIBRARY.ANS: Simple Source Code Library system           uho Mar 95

\ $Id: library.ans,v 1.3 2004/03/24 19:41:03 uho Exp $

: \\ ( -- )
   SOURCE-ID  1+ 1 U>  IF  BEGIN  REFILL 0=  UNTIL THEN
   POSTPONE \ ; IMMEDIATE

: desired ( c-addr1 len1 c-addr2 len2 -- c-addr1 len1 false | true )
    2OVER COMPARE 0= DUP IF NIP NIP THEN ;

: desires ( <space>ccc<spaces> c-addr len -- c-addr len false | true )
    BL WORD COUNT desired ;

CREATE $library 256 CHARS ALLOT

: from ( <spaces>ccc<spaces> -- )
   BL WORD COUNT $library CHAR+ SWAP DUP >R CMOVE R> $library C! ;

: required ( c-addr len -- )
   DUP ALLOCATE THROW  DUP >R SWAP 2DUP 2>R CMOVE 2R>
   $library COUNT
   DUP ALLOCATE THROW  SWAP 2DUP 2>R 2DUP 2>R CMOVE 2R>
   ['] INCLUDED CATCH
   2R> 2DUP $library CHAR+  SWAP CMOVE  $library C!  FREE THROW
   R> FREE THROW  THROW ;

: require ( <space>ccc<spaces> -- )
   BL WORD COUNT required ;

from sample.lib

CR .( Simple Source Code Library System initialized! )


```

Cartoon von Leo Brodie aus Starting Forth
(Starting-Forth-Online unter `http://www.forth.com/starting-forth/`)

Forth von der Pike auf — Teil 10

Ron Minke

Die mit freundlicher Genehmigung der HCC–Forth–gebruikersgroep in der Vierten Dimension in einer Übersetzung von Fred Behringer wiedergegebene achtteilige Artikelserie erschien ursprünglich in den Jahren 2004 und 2005 in der Zeitschrift *Vijgeblaadje* unserer niederländischen Forth–Freunde. Die Firma Atmel hat inzwischen größere Bausteine ihrer Serie entwickelt. Das reizte den Autor dazu, sein Projekt auf einen größeren Mikro–Controller zu übertragen. Im Vijgeblaadje war Teil 9 der Serie (deutsche Übersetzung im VD–Heft 3–4/2007) erschienen. Und hier geht es nun weiter: Auch die vorliegende Übersetzung aus dem Niederländischen stammt von Fred Behringer.

Hier kommt nun Teil 10 der Wiedergabe des Versuchs, ein AVR–Forth–System mit der Voraussetzung *from scratch* zu erstellen.

Reihenfolge der Daten

Im vorliegenden Teil überlegen wir uns, ob die Entscheidungen, die wir in Teil 2 dieser Serie getroffen haben, gute Entscheidungen waren.

Wir trafen in Teil 2 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).

Die Entscheidung 1 bleibt so. Die Entscheidungen 2 und 3 haben mit der Art und Weise zu tun, wie der AVR–Prozessor selbst mit seinem eigenen (Return–)Stack umgeht.

Was bewirkt das?

Was geschieht da nun genau? Angenommen, ein willkürlich herausgegriffenes Stück AVR–Maschinencode leitet einen Unterprogramm–Aufruf (CALL) ein. Beim Abarbeiten des CALLs sieht die Reihenfolge (in Pseudocode, in Bytes) wie folgt aus:

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

Wir sehen hier, dass der Stack nach UNTEN wächst und dass ZUERST das Datenwort abgespeichert und dann erst weiterer Platz geschaffen wird. Der AVR–SP zeigt also offensichtlich auf den ersten freien Platz auf dem Stack. So hatten die Entwickler bei ATMEL sich das zumindest überlegt.

Beim Einbringen von Struktur in die Verwendung der Register (siehe Teil 5) haben wir uns an diese Reihenfolge (Entscheidung 3) gehalten. Benötigt ein Wort beispielsweise zwei Stack–Einträge, dann bekommen wir das Bild 1 (wir nehmen das Wort AND als Beispiel).

In diesem Beispiel wurde der Top–of–Datastack auf die Adresse 0x100 gesetzt. Das obere Byte eines Datenstack–Eintrages liegt an einer bestimmten Adresse, das untere Byte liegt genau eine Adresse höher. Auf diese Weise kam ein experimentelles Forth zustande, das ausgezeichnet arbeitet! Trotzdem wird man das Gefühl nicht los, dass es auch einfacher geht.

Ein Stück Forthcode

Irgendwo im Quelltext unseres Forth–Programms steht die Wortfolge:

```
... DUP OVER ...
```

Beim Übersetzen in Maschinencode durch den Atmel–Assembler kriegen wir im vorgefertigten Teil des Flash–Speichers:

AND (n2 n1 — n3)

	Input			Output		tmp. AVR-Reg.	Daten-Stack-Adresse
	0			0		—	0x100
	1	n2 unteres Byte		1	n3 unteres Byte	R20	0x0FF
	2	n2 oberes Byte	SP →	2	n3 oberes Byte	R21	0x0FE
	3	n1 unteres Byte		3		R22	0x0FD
SP →	4	n1 oberes Byte		4		R23	0x0FC
	5			5			

Bild 1: Darstellung der Stack–Einträge im Speicher am Beispiel von `AND`

		AND (n2 n1 — n3)				tmp. AVR-Reg.	Daten-Stack-Adresse
		Input			Output		
	0			0		—	0x100
	1	n2 oberes Byte		1	n3 oberes Byte	R20	0x0FF
	2	n2 unteres Byte	SP →	2	n3 unteres Byte	R21	0x0FE
	3	n1 oberes Byte		3		R22	0x0FD
SP →	4	n1 unteres Byte		4		R23	0x0FC
	5			5			

Bild 2: Neue Darstellung der Stack–Einträge im Speicher am Beispiel von `AND`

Flash–Adresse	Code	Quelle
⋮	⋮	⋮
0230	0526	.dw DUP
0231	078A	.dw OVER
⋮	⋮	⋮

Bei der *Außenbord*–Version von Forth, mit externem RAM–Speicher, wird dieser Code–Teil beim Hochfahren ins RAM kopiert. Um jedoch der Entscheidung 3 zu genügen, müssen wir die Reihenfolge der Bytes innerhalb eines Wortes im Flash umdrehen. Warum das? Wir holen ein Wort aus dem Flash über das Z–Register und den Befehl LPM ab. Aus dem unteren Byte des ersten Wortes holen wir uns 26, und aus dem oberen Byte 05. Wir halten uns an die Absprache: Das obere Byte kommt an eine bestimmte Adresse, das untere Byte genau eine Adresse darüber. Wenn die Flash–Daten in den RAM–Speicher kopiert werden, müssen wir also erst das obere Byte 05 abspeichern, und danach dann das untere Byte 26. Die Reihenfolge ist hier andersherum! Die Kopier–Routine erledigt das prima für uns, aber aufgepasst!

Stück Text

Schwieriger wird es, wenn ein Stück Text kopiert werden soll. Text steht im Quelltext (Übers.: Ich belasse es beim holländischen Original) so wie:

```
.db "Stukje tekst"
```

Dieser Text wird Buchstabe für Buchstabe paarweise in den Flash–Speicher gesetzt. (Der Flash enthält Wörter, keine Bytes.)

Flash–Adresse	Code	Quelle
⋮	⋮	⋮
0310	7453	`.db "Stukje tekst"`
0311	6B75	
0312	656A	
0313	7420	
0314	6B65	
0315	7473	
⋮	⋮	⋮

Wenn wir dieses Stück Text mit unserer Kopier–Routine ins RAM platzieren, merken wir, dass die Zeichen paarweise vertauscht erscheinen. Dadurch, dass wir erst das obere Byte abspeichern, und danach dann das untere Byte, werden die Buchstaben verkehrt herum abgelegt. Wie lösen wir das Problem? Ganz einfach: Im Quelltext setzen wir sie von in natürlicher Reihenfolge paarweise nach in umgekehrter Reihenfolge. Mit einem kleinen Software–Zusatz, einem Pre–Compiler, passen wir den Quelltext so an, dass Textteile paarweise umgedreht werden.

Flash–Adresse	Code	Quelle
⋮	⋮	⋮
0310	5374	`.db "tSkuejt kets"`
0311	756B	
0312	6A65	
0313	2074	
0314	656B	
0315	7374	
⋮	⋮	⋮

Was bringt uns nun diese ganze Mühe ein? Auf jeden Fall ein funktionierendes Forth–System! Aber geht das nicht auch einfacher . . .??

Es wird Zeit, darüber nachzudenken, ob unsere Entscheidungen 2 und 3 die richtigen waren.

Zurückdrehen

Die in Teil 2 gewählte Reihenfolge der Bytes auf dem Stack war an die Art und Weise gekoppelt, wie Atmel den Returnstack–Pointer des Systems implementiert hat. Für den Bau eines einfachen und begreifbaren Forth–Systems ist diese Reihenfolge unhandlich. Es ist aber nie zu spät einzusehen, dass eine Entscheidung nicht die richtige war!

Wir drehen die Reihenfolge der Bytes auf dem Stack einfach um. Das Beispiel mit dem Wort `AND` ändert sich nun, wie im Bild 2 zu sehen ist.

Wenn wir uns den Inhalt der Prozessor–Register anschauen, sehen wir, dass das untere Byte im *oberen* Register gelandet ist, und umgekehrt. Wie lösen wir nun wieder dieses Problem? Dazu schauen wir uns zunächst die Abbildung der AVR–Register auf den internen Speicher an.

	AND (n2 n1 — n3)					tmp. AVR-Reg.	Daten-Stack-Adresse
	Input			Output			
	0			0		—	0x100
	1	n2 oberes Byte		1	n3 oberes Byte	R19	0x0FF
	2	n2 unteres Byte	SP →	2	n3 unteres Byte	R18	0x0FE
	3	n1 oberes Byte		3		R17	0x0FD
SP →	4	n1 unteres Byte		4		R16	0x0FC
	5			5			

Bild 3: Endgültige Darstellung der Stack–Einträge im Speicher am Beispiel von `AND`

AVR–Register

Der AVR–Prozessor hat 32 frei verfügbare 8–Bit–Register an Bord: R00 bis einschließlich R31. Hiervon können (und sollen) 8 Register zu 4 Registerpaaren zusammengekoppelt werden. Sie lauten:

```
R30 - R31 Z R28 - R29 Y
R26 - R27 X R24 - R25 W
```

Die Register liegen der Reihe nach wie folgt im internen AVR–Speicher:

```
R31 - 0x1F ZH
R30 - 0x1E ZL
R29 - 0x1D YH
R28 - 0x1C YL
. .
. .
R01 - 0x01
R00 - 0x00
```

Wenn wir uns diese Reihenfolge genauer betrachten, fällt uns auf, dass bei den **Registerpaaren** das untere Byte an einem niedrigeren Speicherplatz liegt als das obere Byte. Das ist genau die Reihenfolge, die wir benötigen! Wir müssen die Reihenfolge beim Verarbeiten der Register im Beispiel beim Wort `AND` also **umkehren**. Wir wählen dabei `R16` als neuen Startpunkt. Die endgültige Version lautet dann, wie im Bild 3 zu sehen ist.

Mit dieser Maßnahme sorgen wir dafür, dass bei einem Dump des betreffenden Stücks Speicher (der Register, bzw. des Datenstacks) dasselbe Bild entsteht.

Stück Text II

Nun, da wir die Reihenfolge der Bytes innerhalb eines Wortes zurückgedreht haben, kommt das Textbeispiel *Stukje tekst* auch wieder in der richtigen Reihenfolge im Flash–Speicher zu liegen. Den Pre–Compiler für das Paarweise–Umdrehen haben wir nicht mehr nötig. Auch die Kopier–Routine braucht nicht mehr verkehrt herum zu kopieren; alles kann nun Byte für Byte 1 zu 1 kopiert werden. Es ist gelungen: Eine einfachere Version unseres Forth–Codes!

Die ursprünglichen Entscheidungen 2 und 3 erweisen sich im Nachhinein als keine glückliche Wahl. Die neuen Entscheidungen lauten:

1. Bleibt.
2. Das obere Byte eines 16–Bit–Wortes wird zuerst auf den Stack gelegt, darunter dann das untere Byte.
3. Der Forth–Pointer zeigt auf das **untere** Byte eines 16–Bit–Wortes (und also — in Abweichung von der Arbeitsweise des AVR–SPs — nicht auf den leeren Platz unter dem Wort).

Berichte generieren mit Hilfe von Gforth — ein Kinderspiel

Michael Kalus

Wer wie ich immer wieder kürzere Berichte abgeben muss — als Arzt im Krankenhaus sind das Befunde von oft angefertigten Untersuchungen, von denen viele standardisiert befundet werden können, so wie EKG, Lungenfunktionsprüfungen, Röntgenbilder des Thorax u.a. – wünscht sich einen Baukasten dafür. Hier eine sehr simple Lösung in Gforth.

Alle Sätze dieses Beispiels sind natürlich Blindtexte, da ich hier keine Patientendaten veröffentlichen kann. Aber es braucht nicht viel Fantasie, um den Nutzen auch so zu erkennen, hoffe ich zumindest. Nun, bevor ich in langatmige Erklärungen verfalle, seht selbst. Es ist so simpel, dass man es wohl kaum ausführlicher kommentieren muss.

Benutzt wird das Feature von Gforth, ein Input-File verarbeiten zu können und den Output sogleich in ein File zu schreiben:

```
PowerBook$ gforth in.fs > out.txt
```

Ferner benutzte ich einige Steuerelemente des Compilers:

`(`	Beginnt einen Kommentar; wird nicht ausgeführt.
`)`	Beendet den Kommentar.
`.(`	Beginnt einen Kommentar; wird nicht compiliert, sondern ausgegeben.
`INCLUDE` <*name*>	Fügt weitere Dateien ein.

Weitere Steuerelemente wie `[IF]` und `[THEN]` werden ebenfalls benutzt. Natürlich können auch Kalkulationen eingefügt werden.

Viel Vergnügen, Michael

Links

`http://www.jwdt.com/~paysan/gforth.html` — Current release 0.6.2
`http://www.newmediadesigner.de/` — Blindtext-Archiv

Listings

File: `in.fs`

```
1   \ Steuerdatei
2
3   \ Header
4   cr .( Bericht: )
5   .( G - 11.11.08 - 11:11Uhr ) \ Buchstabe, Tag, Monat, Jahr und Uhrzeit angeben
6   cr cr
7
8
9   \ 4 Sorten Berichte können ausgewählt werden:
10
11  include m.dep
12  \ include w.dep
13  \ include m.ang
14  \ include w.ang
15
16  cr cr .( Ennepetal, den <Datum> <Unterschrift> <Stempel> )
17
18  bye ( finis)
```

File: `m.dep`

```
1   \ Blocktexte für Bericht; Fall "dep", männliche fassung (m).
2   \ Es werden Blindtexte als Beispiel verwendet.
3
```

```
4   cr cr
5   \ Vorwort: (ohne Absatznummer)
6
7   .( Freilebende Gummibärchen gibt es nicht. )
8
9   .( Man kauft sie in Packungen an der Kinokasse. )
10   ( Man kauft sie in Packungen am Kiosk. )
11
12  .( Dieser Kauf ist der Beginn einer fast )
13          .( erotischen )
14          .( und sehr ambivalenten )
15           ( wenig erfreulichen )
16  .( Beziehung Gummibärchen-Mensch. )
17
18  .( Zuerst genießt man. )
19  .( Dieser Genuss umfasst alle Sinne. )
20  .( Man wühlt in den Gummibärchen, )
21          .( man fühlt sie. )
22           ( sie kleben etwas. )
23           ( kalt sind sie hart. )
24
25  .( Gummibärchen haben eine Konsistenz wie weichgekochter Radiergummi. )
26
27  .( Die Tastempfindung geht auch ins Sexuelle. )
28  .( Das bedeutet nicht unbedingt, dass das Verhältnis zum Gummibärchen ein
29   geschlechtliches wäre, denn prinzipiell sind diese geschlechtsneutral. )
30
31  cr cr
32  \ 1. Konsistenz
33
34  .( Nun sind Gummibärchen weder wabbelig noch zäh; )
35          .( sie stehen genau an der Grenze. )
36           ( erst kalt sind sie hart wie Stein. )
37
38  .( Auch das macht sie spannend. )
39
40  .( Gummibärchen sind auf eine aufreizende Art weich. )
41  .( Und da sie weich sind, kann man sie auch ziehen. )
42
43   ( Gummibärchen sind kalt völlig unatraktiv. )
44   ( Wenn sie so hart sind, kann man sie nur noch zerschlagen. )
45
46  cr cr
47  \ 2 Alternativen
48  .( Ich mache das sehr gerne. Ich sitze im dunklen Kino und ziehe meine
49  Gummibärchen in die Länge, ganz ganz langsam. Man will sie nicht kaputt machen,
50  und dann siegt doch die Neugier, wieviel Zug so ein Bärchen aushält. )
51   ( Manche Leute machen das sehr gerne. Sie sitzen im dunklen Kino und ziehen
52  ihre Gummibärchen in die Länge, ganz ganz langsam. Eigentlich wollen sie sie
53  nicht kaputt machen. Doch dann siegt die Neugier, wieviel Zug so ein Bärchen
54  aushält. )
55
56
57  \ 2. Zwischenspiel:
58  cr cr
59
60  false [IF]       \ use true or false to switch
61  include bio.txt
62  [THEN]
63
```

```
64  cr cr
65  \ 3. Climax und Kontemplation:
66
67  .( Forscherdrang und gleichzeitig das Böse im Menschen erreichen den Climax,
68   wenn sich die Mitte des gezerrten Bärchens von Millionen Mikrorissen weiß färbt
69   und gleich darauf das zweigeteilte Stück auf die Finger zurückschnappt. )
70  .( Man hat ein Gefühl der Macht über das hilflose, nette Gummibärchen. Und wie
71   man damit umgeht: Mensch erkenne dich selbst! )
72
73  .( Jetzt ist es so, dass Gummibärchen ja nicht gleich Gummibärchen ist. )
74  .( Ich bevorzuge das klassische Gummibärchen, )
75          .( künstlich gefärbt und aromatisiert. )
76          .( Und in der europäischen mittleren Größe. )
77           ( Nicht die USA Sorte, viel zu dick. )
78
79  .( Mag sein, dass es eine Sentimentalität ist. )
80  .( Jedenfalls halte ich nichts von neuartigen Alternativ-Gummibärchen ohne Farbstoff. )
81  .( Und auch unter den konventionellen tummeln sich schwarze Schafe: )
82      .( die schwarzen Lakritz-Bärchen. )
83
84  cr cr
85  .( Den ganzen Blindtext findest du hier: ) cr
86  .( http://www.newmediadesigner.de/11.htm )
87
88  ( finis)
```

File: `bio.txt`

```
1   \ Variables einfügen - Blindtext hier im Beispiel
2
3   .( Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed diam nonummy
4    nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat. Ut wisi
5    enim ad minim veniam, quis nostrud exerci tation ullamcorper suscipit lobortis
6    nisl ut aliquip ex ea commodo consequat. Duis autem vel eum iriure dolor in
7    hendrerit in vulputate velit esse molestie consequat, vel illum dolore eu
8    feugiat nulla facilisis at vero et accumsan et iusto odio dignissim qui
9    blandit praesent luptatum zzril delenit augue duis dolore te feugait nulla
10   facilisi. Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed diam
11   nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat
12   volutpat. Ut wisi enim ad minim veniam, quis nostrud exerci tation ullamcorper
13   suscipit lobortis nisl ut aliquip ex ea commodo consequat. Duis
14   autem vel eum iriure dolor in hendrerit in vulputate velit esse molestie
15   consequat, vel illum dolore eu feugiat nulla facilisis at vero et
16   accumsan et iusto odio dignissim qui blandit praesent luptatum zzril
17   delenit augue duis dolore te feugait nulla facilisi. Nam liber tempor
18   cum soluta nobis eleifend option congue nihil imperdiet doming id quod
19   mazim placerat facer possim assum. )
20
21  ( finis)
```

Forth–Gruppen regional

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

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

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

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

Gruppengründungen, Kontakte

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

µP–Controller Verleih

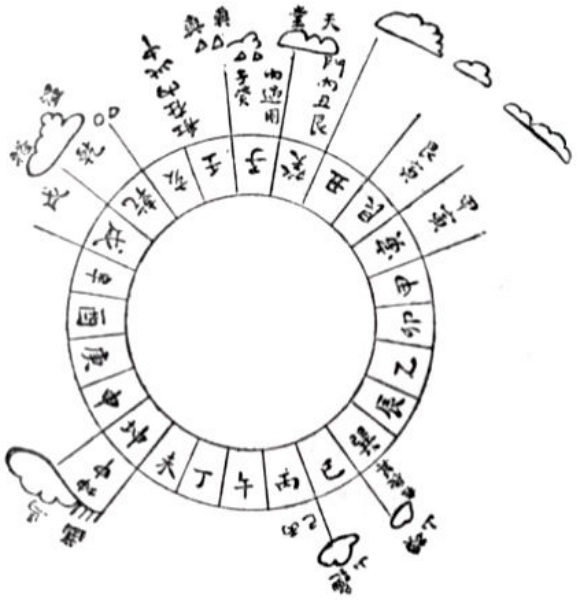
Carsten Strotmann
`microcontrollerverleih@forth-ev.de`
`mcv@forth-ev.de`

Spezielle Fachgebiete

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

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

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



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

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

EuroForth 2008 — 24th EuroForth Conference
Technische Universität Wien, Vienna, Austria
26th to 28th September, 2008

http://www.tuwien.ac.at/

EuroForth is an annual conference on the Forth programming language, stack machines, and related topics, and has been held since 1985. The 24th EuroForth will be held at TU Wien in Vienna, Austria. The conference will be preceeded by a Forth 200x standards meeting.

June 26:	Early registration deadline
June 27:	Deadline for draft papers (academic stream)
August 12:	Notification of acceptance of academic stream papers
August 19:	Registration deadline (later registration uncertain)
September 15:	Deadline for camera-ready paper submission (academic and industrial stream)
September 25-26:	Forth200x meeting
September 26-28:	EuroForth 2008 conference

For your electronic calendar: dates in iCal (.ics) and vCal (.vcs) format are avalaible at `http://www.complang.tuwien.ac.at/anton/euroforth/ef08.html` thanks to Bernd Paysan.

Information on earlier conferences can be found at the EuroForth home page at `http://www.euroforth.org/`

Contact Information

You can contact the conference organizers by phone (Anton Ertl: (+43-1) 58801 18515; Ewa Vesely (+43-1) 58801 18504), by fax ((+43-1)58801 18598), or by Email (see homepage).

Links

- Call for Papers (`http://www.complang.tuwien.ac.at/anton/euroforth/ef08/cfp.html`)
- Registration form and Invitation (`http://www.complang.tuwien.ac.at/anton/euroforth/ef08/registration.pdf`)
- Travel information (`http://www.complang.tuwien.ac.at/anton/euroforth/ef08/travel.html`)

EuroForth Home (`http://www.complang.tuwien.ac.at/anton/euroforth/index.html`)

Anton Ertl (`http://www.complang.tuwien.ac.at/anton/`)

Schloss Schönbrunn in Wien (Photo: Wikipedia)

This information can also be found at `http://www.complang.tuwien.ac.at/anton/euroforth/ef08.html`.