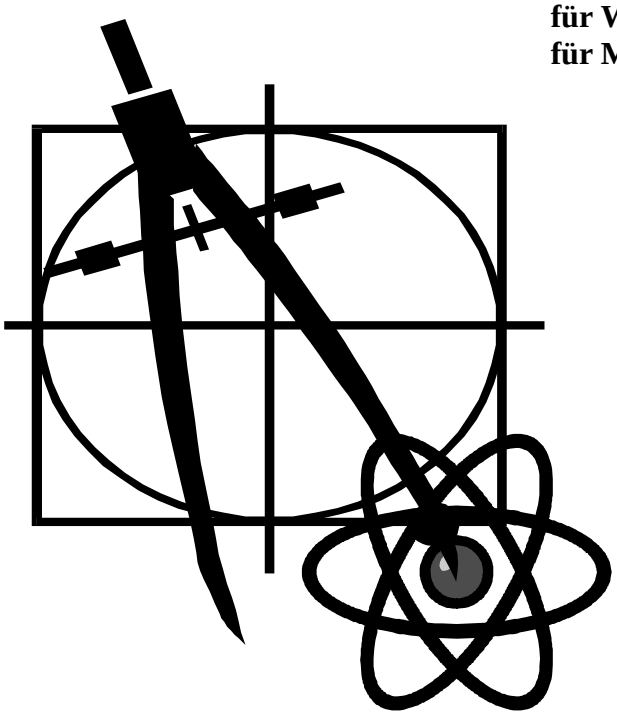
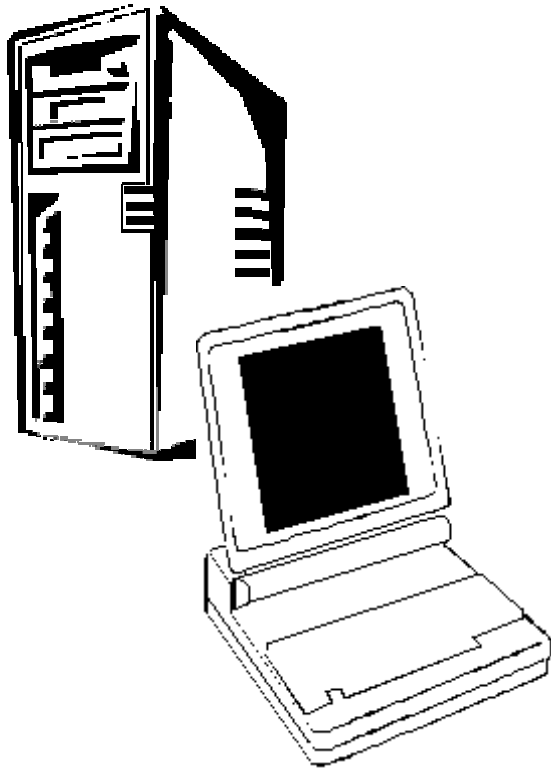
VIERTE DIMENSION

4 / 2002

Das FORTH-Magazin



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



In dieser Ausgabe:

18. Jahrgang - **Ausg. 4/2002** - 5,11 €

Organ der FORTH-Gesellschaft e.V.

Dienstleistungen und Produkte fördernder Mitglieder des Vereins

tematik GmbH
Technische Informatik

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

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

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

LEGO RCX-Verleih

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

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

Inhalt:

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

Anfragen bitte an

Martin.Bitter@t-online.de

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

Dipl.-Ing. Arndt Klingelnberg

Tel.: ++32 +87 -63 09 89 (Fax: -63 09 88)
Waldring 23, B-4730 Hauset, Belgien
akg@aachen.kbbs.org

Computergestützte Meßtechnik und Qualitätskontrolle, Fuzzy, Datalogger, Elektroakustik (HiFi), MusiCassette HighSpeedDuplicating, Tonband, (engl.) Dokumentationen und Bedienungsanleitungen.

Forth Engineering
Dr. Wolf Wejgaard

Tel.: +41 41 377 3774 - Fax: +41 41 377 4774
Neuhöflirain 10
CH-6045 Meggen http://holonforth.com

Wir konzentrieren uns auf Forschung und Weiterentwicklung des Forth-Prinzips und offerieren HolonForth, ein interaktives Forth Cross-Entwicklungssystem mit ungewöhnlichen Eigenschaften. HolonForth ist erhältlich für 80x86, 68HC11 und 68300 Zielprozessoren.

KIMA Echtzeitsysteme GmbH

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

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

FORTecH Software
Entwicklungsbüro Dr.-Ing. Egmont Woitzel

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

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

Ingenieurbüro
Dipl.-Ing. Wolfgang Allinger

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

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

Ingenieurbüro
Klaus Kohl

Tel.: 08233-30 524 Fax: - 9971
Postfach 1173
D-86404 Mering

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

Inhalt

In der nächsten Ausgabe finden Sie voraussichtlich:

- Nachträge zu den Besprechungen der ForthWrite und des Feigenblattes; Fred Behringer
- Threaded Code - Varianten und Optimierungen; Anton Ertl
- b16 - Ein Forth Processor im FPGA; Bernd Paysan

Editorial

IMPRESSUM

Name der Zeitschrift

Vierte Dimension

Herausgeberin

Forth-Gesellschaft e.V.
Postfach 16 12 04
D-18025 Rostock
Tel.: 0381-400 78 28
E-Mail:
SECRETARY@FORTH-EV.DE
DIREKTORIUM@FORTH-EV.DE

Bankverbindung: Postbank Hamburg
BLZ: 200 100 20
Kto: 563 211 208

Redaktion & Layout

Friederich Prinz
Hombergerstraße 335
47443 Moers
Tel.: 02841-58 3 98
E-Mail:
VD@FORTH-EV.DE
FRIEDERICH.PRINZ@T-ONLINE.DE

Anzeigenverwaltung

Büro der Herausgeberin

Redaktionsschluß

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

Erscheinungsweise

1 Ausgabe / Quartal

Einzelpreis

5,11 € + Porto u. Verpackung

Liebe Leser,

daß zu dieser Ausgabe das Weihnachtsfest schon beinahe vor der Türe steht, scheint mir eine besonders gute Gelegenheit zu sein, einmal denjenigen in unserer Gesellschaft Danke zu sagen, die sich immer wieder darum bemühen, die Forthgesellschaft im Fluß zu halten.
Da sind sicher als Erste die Autoren zu nennen, die uns allen immer wieder interessante Artikel – manches Mal ganze Artikelfolgen – Leserbriefe und andere Beiträge schicken. Ohne diese Arbeit gäbe es unsere Zeitschrift nicht. Ohne diese Arbeit gäbe es sicher unsere Gesellschaft schon lange nicht mehr.
Dem Forth-Büro, das für uns alle stille, aber unentbehrliche Dienste leistet, gebührt mindestens ebenso deutlicher Dank wie den Autoren. Ohne die Arbeit des Forth-Büros gäbe es ebenfalls keine „Vierte Dimension“. Druck, bzw. Vervielfältigung, Versand nach Rostock, Verpackung und Versand zu den Mitgliedern erledigen sich nicht von selbst. Und auch das Führen der Mitgliederliste, das Einziehen der Mitgliederbeiträge, das Führen unserer Konten verlangt nach einer ordnenden Hand und einem kühlen Kopf.
Die Menschen, die dabei helfen, die „VD zu machen“, müssen selbstverständlich in den Dank einbezogen werden. Ob es um die Übersetzungsarbeit zu den Briefen oder Beiträgen aus der „FIG Silicon Valley“ geht, oder um die Zusammenfassung jeder Ausgabe der VD für unsere englischsprache Schwesternzeitschrift, um die Korrekturen der Tipfehler des Editors, oder um die Organisation der Vervielfältigung der Zeitschrift – immer sind weitere fleißige Hände, kluge Köpfe und großzügige Herzen notwendig, die auf jeweils ganz unterschiedliche Weise mitwirken und ihren Teil zum Gelingen des Ganzen beitragen.
Und auch die Pflege der WEB-Seiten der Forthgesellschaft fordert genau so Zeit und Einsatz, wie es die vielfältigen Geschäfte tun, die im Laufe eines Jahres von den Mitgliedern des Direktoriums zu erledigen sind.

Es tut sich eben nichts von selbst. Einige aus unserer Mitte sind sehr aktiv; für uns. Und denen sage ich an dieser Stelle, sicher im Namen der ganzen Leserschaft, ein herzliches Dankeschön für ihren Einsatz.
Ein fröhliches Weihnachtsfest und einen „guten Rutsch“ in ein gesundes und erfolgreiches Jahr 2003 wünsche ich uns Allen. Und natürlich wünsche ich uns auch, daß wir nicht nur in 2003 immer genügend hilfreiche Hände, Köpfe und Herzen für alle anstehenden Arbeiten und Aufgaben haben werden.

Friederich Prinz

P.S.: Ein ganz besonderes Dankeschön sage ich den Direktoren der Forthgesellschaft, die der Redaktion der „Vierte Dimension“ einen funkelnagelneuen Drucker spendiert haben. Der Kyocera FS 1900 DN (Duplexeinheit und netzwerktauglich) wird mir die Arbeit, die ich bisher mit einem „steinalten“ Kyocera T 800 erledigt habe (ein DIN A 4 Blatt [graphisch] auszudrucken hat rund 3 Minuten benötigt) ganz erheblich erleichtern.

fep

Quelltext-Service

Die Quelltexte in der VD müssen Sie nicht abtippen. Sie können diese Texte auch direkt bei uns anfordern und sich zum Beispiel per E-Mail schicken lassen. Schreiben Sie dazu einfach eine E-Mail an die Redaktionsadresse. *fep*

Berichtigung

Manchmal dauert es lange, ehe eine Reaktion auf einen Artikel eintrifft – aber eine Freude ist es allemal! Vor einigen Jahren stand in der Vierten Dimension ein Artikel über den Lee-Effekt, besser ausgedrückt DAF (delayed acoustic feedback). Dort wurde die Geschichte des Lee-Effekts als die der Suche nach einer Methode Simulanten zu überführen, erklärt. Diese 'Geschichte' ist wohl nicht wahr – ob sie wenigstens schön ist? Hier also die fällige Richtigstellung, für die ich mich bei Herrn **Andreas Starke** herzlich bedanke.

Martin Bitter

Lieber Martin Bitter,
ich bin zufällig auf Deinen Artikel über den Lee-Effekt gestoßen. Daß Lee (Ohio State University) den "Lee-Effekt" (der übrigens nur in Deutschland so heißt, der Himmel weiß warum) entwickelt habe, um Taubheitssimulanten während des Vietnam-Krieges zu überführen, ist ein Ammenmärchen, das ich noch nie gehört habe. Da käme eher der Korea-Krieg in Frage! Die erste Arbeit von B.S. Lee zu dem Thema ist von 1950. "Lee BS. Effects of delayed speech feedback. J Acoust Soc Am 1950; 22; 824-26"

Klarstellung!

Bernd Paysan hat dankenswerter Weise einen Bericht über die Jahrestagung geschrieben. Darin hat er auch meinen kurzen Auftritt erwähnt. Dazu möchte ich zwei Dinge anmerken: Es ging mir darum, zu zeigen, dass mit **genau** den Möglichkeiten, die die damaligen Teilnehmer hatten, das Ziel erreicht werden konnte. Das Modell, das ich verwendete, war eines der vorjährigen Wettbewerbsmodelle. Insbesondere wurde kein zusätzlicher Sensor verwendet, sondern ein rein mechanischer Anschlag, den jeder hätte anbauen können. Zum zweiten: Leider war der Untergrund, der bei der Vorführung zur Verfügung stand, ein anderer als der, den ich zum Testen verwendete, deshalb ging die Vorführung recht daneben. In der Mittagspause gelang es, das Programm neu zu kalibrieren – und es ergab sich ein lesbarer Schriftzug. Zeugen: Ulrich Hofmann und Klaus Schleisik. Letzter fand die Schrift lesbar, aber mit zu großen Leerräumen.
Wäre ich ein größerer Geist, hätte ich diese Klarstellung nicht nötig.

Martin Bitter

Chuck Moore live erlebt!

Am 4. Mai 2002 war ich mal wieder beim Forth-Chat der FIG UK und beklagte mich gerade über das seltsame Verhalten von CATCH unter Win32for (ich glaubte es bei einer sehr-vielfach-Verzweigung benutzen zu müssen), als gegen Ende der Veranstaltung jemand mit der lapidaren Meldung “Chuck will be in a chat room in 25 minutes or so” herausrückte.
Bis alles Weitere mitgeteilt war (Wo? Wann?) und der Chat-Server nebst Chat-Kanal geändert war, verging einige Zeit, aber ich habe es geschafft – und konnte versuchen, den Diskussionen zu folgen. Ehrfurchtsvoll schwieg ich und lauschte lieber, wenn ich auch nicht alles verstand.
Aber eines fiel mir auf: Chuck Moore sagte (unter Anderem): “On the other hand, I don't like CATCH THROW Errors should be impossible. Or resolved immediately. over”
Mhm! Mhm! Für mich also sehr interessant, sollte ich umdenken?
Wer jetzt selbst nachlesen möchte, um Gedanken und Einsichten des Forth-Erfinders zu erfahren, kann einen Mittschnitt des Chats unter http://www.ultratechnology.com/chatlog.htm einsehen. Der Teil, bei dem ich zugegen war, kann per E-Mail mbitter@forth-ev.de bezogen werden (Log-Datei)

Martin Bitter

BACKLASH

Jim Lawless experimentiert mit einer forthähnlichen Sprache, die er BACKLASH nennt. Backlash kann mit Rückstoß, Rückschlag übersetzt werden. Mit diesem Namen will Jim Lawless eine Korrespondenz zur forthtypischen UPN und der Forth-Syntax ausdrücken.
BACKLASH ist in Java-Script geschrieben und soll Jim die Arbeit mit Server-Client-Anwendungen erleichtern. Er beabsichtigt nicht, aus BACKLASH ein vollständiges (Java-Script)-Forth zu entwickeln, hat aber nichts dagegen, wenn andere dies versuchen möchten. Da BACKLASH auf Java-Script aufbaut, kann es in allen Umgebungen, die Java-Script erlauben, wie z.B. Windows Scripting Host, verwendet werden. Das ist sein großer Vor- und Nachteil, da die Qualität der Programmausführung sehr von den Clientbedingungen abhängt. So läuft das Beispiel http://www.mailsend-online.com/bl.htm bei mir unter Opera schleichend langsam, während unter dem MISE das Ganze in Sekundenbruchteilen erledigt ist.

Martin Bitter

Embedded selbst downladen

Bereits zum zweiten Male stellt Rafael Deliano seine Zeitschrift 'embedded' on-line als PDF-Datei zum Download zur Verfügung. Das ist die unmittelbare Folge seines Erfolges, war es ihm doch nicht mehr möglich, die gewünschte Anzahl der Abonnenten weiterhin mit gedruckten (kopierten) Exemplaren zu versorgen.
Unter http://www.embeddedFORTH.de kann sich nun Jederman und jede Frau die bisher erschienenen Exemplare der 'embedded' besorgen. Und das ist überaus empfehlenswert.

Wie alle 'embedded's zuvor überzeugt auch die Ausgabe Nummer 7 durch eine gelungene Mischung aus Hard- und Softwaretips, so wie kurzen historischen Rückblicken.
Der zweite Teil zu zyklisch-fehlerkorrigierenden Codes in der Datenübertragung und -speicherung zeigt verschiedene Verfahren auf und erläutert die Möglichkeiten, falsche (gekippte) Bits zu erkennen und durch den Vergleich mit abgespeicherten Fehlermustern teilweise zu rekonstruieren.
Besonders interessiert hat mich ein 'neues' (für mich neu) Multiplikationsverfahren, das auf Addition, Subtraktion und Quadrierung beruht (Viertelquadratmethode). Ist erst einmal kompliziert, aber in der programmtechnischen Umsetzung durchaus handhabbar. Die Quadrate in eine Tabelle und Be-

achtung der Grenzfälle (FF 7F bzw. 128 -128 usw.) und 'schon' hat man einen 8-bit Multiplizierer, der knapp 51 (Programm) plus 128 (Quadrattabelle) Byte belegt und mit ca. 25- 33 µsec noch schneller ist als der 'native' Befehl der betrachteten CPU (Motorola 6502, 55 µsec).
Das absolute Schmankerl ist allerdings Rafael Delianos Tip, (gebrauchte) Motorola-FPUs MC68882 zur Rechenunterstüzung herkömmlicher Micros zu nutzen. Beschaltung, Opcodes, Softwarepaket zur Ansprache - alles vorhanden. Einfach toll! Noch einmal: einfach toll!
FIR (Finite Impulse Response) Filter sind gut geeignet, um in 'kleinen' Micros implementiert zu werden. Sie arbeiten in 'endlicher' Zeit und erzeugen ihre Antwort praktisch „on the fly“. Das heißt, im Gegensatz zu anderen Filterfunktionen, bei denen ein ganzer Wellenzug eingelesen und im Nachhinein in seine Frequenzen zerlegt wird, werden bei FIR die Wellenzüge durch Schieberegister gesandt, und zu festgelegter Zeit an bestimmten Positionen die jeweiligen Werte ausgelesen und aufaddiert. Dabei entsprechen die Positionen im Schieberegister den Koeffizienten der jeweils gewünschten Filterfunktion. Rafael Deliano liefert wieder komplett: Betrachtung, wie solche Koeffizienten zu ermitteln sind (Filtersample rein - Koeffizienten raus), ein Programm, das diese Arbeit übernimmt und Beispiele zu FIR in Hard- und Software (schnell und effizient).
Den Abschluss bildet der erste Teil über die TTY-Stromschleife. Der Uralt(nicht)standard wird in vielen Bereichen noch genutzt. Hier werden diverse Beschaltungen mit galvanischer Kopplung oder Optokopplung gezeigt, die große Entfernungen überbrücken können.
Zu allen Beiträgen gibt es vollständige Listings. Diese stehen in seperaten Dateien zum Download bereit.
Rafael Deliano leistet mit der 'embedded' erstaunliches. Das sollte mit anerkennenden Rückmeldungen unterstützt werden. Also - auf zum Download und zur E-Mail.

Martin Bitter

Zur Person:
Rafael Deliano hat E-Technik an FH-Regensburg studiert, und war danach als Angestellter in verschiedenen Unternehmen für Sprachverarbeitung und Fernmeldetechnik tätig. Seit ca. 10 Jahren betreibt er ein eigenes Ingenieur-Büro.
Erste Erfahrungen mit FORTH sammelte er 1983 auf AIM65. Er benutzt es immer noch beruflich auf Controllern.
Rafael Deliano ist eifriger Sammler technischer Bücher.

Lieber Fritz,
die außerordentlich gut besuchte Garmisch-Tagung hat ihre positive Auswirkung auf "many issues to come". Heft 3/2002 besticht durch seine Vielfalt an Artikeln, Leserbriefen und sonstigen Beiträgen der verschiedensten Autoren aus den verschiedensten Gebieten. Trotzdem: Auch wenn dank der wiedererwachten Schreibfreudigkeit unserer Mitglieder und Freunde aus aller Welt viel "Material" vorliegt, die Sichtung, die geeignete Auswahl und die geschickte Plazierung liegt in Deinen Händen. Und das machst Du gut. Wir danken Dir. Mach weiter so! Herzlichen Gruß

Fred Behringer

Lieber Fred, liebe Leser
Meinen Dank für das Lob! Einen Grund zum Ausruhen bringt es leider nicht mit sich. Mit der wiedererwachten Schreiblust unserer Mitglieder ist es nicht so furchtbar weit her. Tatsächlich „lebt“ die VD aktuell von den Tagungsbeiträgen unserer Versammlung in Garmisch. Die Halde ist nicht besonders groß. Das gilt auch, wenn ich NICHT die bei meinem Arbeitgeber entstandenen Definitionen von GROßEN HALDEN zugrunde lege ;-)

Glückauf *fep*

Lieber Herr Prinz,
hiermit sende ich Ihnen, die gefragte Geschichte von Atari/ Milan. Bezüglich der Weiterentwicklung von Forth und vom FG, bin ich sehr optimistisch. Die Hauptsache ist, dass mit Forth zu arbeiten, viel Spass macht. Forth ist eine unendliche Geschichte.
Herzliche Grüsse

Filippo Sala

Eine kurze Geschichte von Atari

Als 1994/95 Atari sich aus dem Computer-Bereich zurückzog, waren die vielen Atarianer (über 1 Mio in Deutschland) schwer enttäuscht. Die Zukunft des TOS-Betriebssystems war ungewiss. Aber es lebt heute Ende 2002 noch. Die heutige Situation könnte man mit dem Satz "TOS sucht eine neue CPU" zusammenfassen. Aber der Reihe nach.

Die Hardware

Der letzte Computer von Atari war der Falcon. Er hatte einen grossen Erfolg, besonders bei den Musikern und wird heute noch mit Preisen ab 800 DM gahandelt. Der Falcon hatte auch ein Multitasking-Betriebsystem (Multitos), aber er war zu langsam, insbesondere bei höheren Auflösungen in Farbe.
Danach machten sich die Clone-Hersteller an die Arbeit und schufen mehrere Geräte: die MEDUSA, den EAGLE und den HADES. Wegen der kleinen Stückzahlen waren alle sehr teuer: Medusa ab 5000 DM, Eagle ab 3000 und Hades ab 3500. Es gab auch eine Karte für den PC - die JANUS - mit der CPU MC68020 und war "wahnsinnig schnell und bunt" so die Werbung. Sie kostete 499 DM.

Ende 1998 schuf die Milan GmbH (eine ad-hoc Gesellschaft von Atari-Fans) den letzten Clone: der MILAN I mit MC68040-CPU, 16 MB Speicher, PCI-Bus und 2 MB-S3-Grafikkarte. Der Preis von 1500 DM war sensationell und für vielen Atari-Fans annehmbar. Die vielen teueren Original-Chips von Atari wurden mit Hilfe einem Logic-Chips einfach emuliert.
Durch die Verwendung der MC68040-CPU mit Daten- und Instruktion-Cache und der 32-Bit-Grafikarte S3 war die Arbeitsgeschwindigkeit für normale Anwendungen mehr als ausreichend.

... weiter auf Seite 22

Forth - die frühen Jahre

Chuck Moore chipchuck@mindspring.com <mailto: chipchuck@mindspring.com> 1991

Den Originaltext der nachfolgenden Übersetzung findet man im Internet unter:
http://www.mindspring.com/~chipchuck/HOPL.html

Für die Vierte Dimension mit freundlicher Genehmigung des Autors, ***Charles Moore***, übersetzt von *Friederich Prinz* und *Fred Behringer*, mit wertvollen Hinweisen von *Henry Vinerts* (SVFIG).

Zusammenfassung

Forth ist eine einfache, natürliche Computersprache. Sie hat da, wo es auf Leistungsfähigkeit ankommt, bemerkenswerte Akzeptanz gefunden. Sie entwickelte sich in den 60er Jahren auf der Wanderschaft von der Universität über die Wirtschaftswelt zum Labor. Dies ist die Geschichte eines einfachen Interpreters, der zu einer vollständigen Programmiersprache und zu einem Betriebssystem heranwuchs.

Nachtrag für 1999

Dieser Text wurde für die HOPL-II-Konferenz [History of programming languages] (Geschichte der Programmiersprachen) abgefaßt. Er wurde dort, wohl aus stilistischen Gründen, in Bausch und Bogen abgelehnt. Große Teile daraus wurden in die nicht abgelehnte Arbeit [Rather 1993] übernommen.

Die vorliegende HTML-Version (Übers.: das Original) wurde aus der ursprünglichen Schreibmaschinenvorlage heraus übertragen. Im Text wurden nur ein paar Kleinigkeiten verändert. Phil Koopman hat das Ganze durchgesehen und einige Beispiele zum Quelltext vorgeschlagen. Diese wurden noch nicht eingefügt.

Inhalt

Forth

Forth entwickelte sich in den 60er Jahren auf der Wanderschaft quer durch Amerika, von Universität über Wirtschaft zum Labor, inmitten etablierter Sprachen. In jener Zeit war ich der einzige Forth-Programmierer und die Sprache hatte bis zum Ende keinen Namen. Dieser Bericht wurde aus dem Gedächtnis zusammengetragen und stützt sich lediglich auf spärliche Aufzeichnungen und die wenigen Listings, die überlebt haben.

Forth liefert kaum etwas Neues, aber die Zusammensetzung seiner Bestandteile ist einzigartig. Ich bin den Menschen und Einrichtungen, die es mir - zumeist ohne ihr Wissen - erlaubt haben, Forth zu entwickeln, dankbar. Und dankbar bin ich Ihnen dafür, daß Sie genug Interesse aufbringen, etwas darüber zu lesen.

Forth ist eine einfache und natürliche Computersprache. In unseren Tagen findet sie als Sprache erster Wahl Anerkennung. Daß sie das ohne Unterstützung durch Industrie, Universität oder Regierung geschafft hat, ist ihrer Leistungsfähigkeit, ihrer Zuverlässigkeit und ihrer Vielseitigkeit zuzuschreiben. Zu Forth greift man, wenn die Leistungsfähigkeit stärker ins Gewicht fällt als die Beliebtheit anderer Sprachen. Besonders stark ist das in praktischen Anwendungen, wie in Steuerungs- und Nachrichtenübertragungssystemen, der Fall.

Viele Forth-Organisationen und ganze Heerscharen kleinerer Firmen bieten Systeme, Anwendungen und Dokumentationen an. In Nordamerika, Europa und Asien werden jährlich Konferenzen abgehalten. Demnächst werden wir den Entwurf eines ANSI-Standards erleben [ANS 1991].

Keines der Bücher über Forth fängt dessen Würze ganz ein. Ich halte das erste unter ihnen, "Starting Forth" von Leo Brodie [Brodie 1981], immer noch für das beste. Einen weiteren Einblick liefert das unschätzbare Titel- und Autorenverzeichnis [Martin 1987] von JFAR (Übers.: Journal of Forth Applications and Research).

Das klassische Forth, über das wir hier sprechen, stellt dem Programmierer ein Minimalgerüst zur Entwicklung einer auf seine Anwendungen hin zugeschnittenen Sprache zur Verfügung. Es ist für eine Arbeitsplatzumgebung gedacht: Tastatur, Bildschirm, Computer und Diskette.

Forth ist eine im wesentlichen kontextfreie textbasierte Sprache. Sie konstruiert neue "Worte" aus Worten, die durch Leerzeichen voneinander getrennt sind. Etwa 150 solcher Worte reichen für ein System aus, das folgendes leistet (Daten geben das Entwicklungsjahr wieder):

```
 SAO  1958  Interpreter
SLAC  1961  Daten-Stack
 RSI  1966  Tastatureingabe
            Bildschirmausgabe, OK
```

```
                Editor
  Mohasco  1968 Compiler
                Return-Stack
                Dictionary
                Virtueller Speicher
(Diskette)
                Multitasking
     NRAO  1971 Gefädelter Code
                Festkomma-Arithmetik
```

Ein solches System besteht aus 3-8 Kilobyte an Maschinencode, der aus 10-20 Seiten Quelltext heraus compiliert wird. Es kann leicht von einem einzigen Programmierer auf einen kleinen Computer implementiert werden.

Der vorliegende Bericht folgt notwendigerweise der Entwicklung meiner Karriere. Eigentlich beabsichtigt ist aber eine Autobiographie von Forth. Ich werde die oben angeführten Eigenschaften und die Namen der damit verknüpften Worte besprechen. Für viele Worte ergibt sich die Bedeutung von selbst. Ein paar Worte verlangen nach einer Beschreibung. Ein paar andere Worte würden den Rahmen dieser Arbeit sprengen.

Es folgt ein Überblick über den hier zu erwähnenden Teil des Forth-Dictionarys:

```
Interpreter
     WORD  NUMBER  INTERPRET  ABORT
     HASH  FIND  '  FORGET
     BASE  OCTAL  DECIMAL  HEX
     LOAD  EXIT  EXECUTE  (
Ein-/Ausgabegerät
     KEY  EXPECT
     EMIT  CR  SPACE  SPACES  DIGIT  TYPE
DUMP
Datenstack
     DUP  DROP  SWAP  OVER
     +  -  *  /  MOD  NEGATE
     ABS  MAX  MIN
     AND  OR  XOR  NOT
     0<  0=  =
     @  !  +!  C@  C!
     SQRT  SIN.COS  ATAN  EXP  LOG
Returnstack
     :  ;  PUSH  POP  I
Diskette
     BLOCK  UPDATE  FLUSH  BUFFER  PREV
OLDEST
Compiler
     CREATE  ALLOT  ,  SMUDGE
     VARIABLE  CONSTANT
     [  ]  LITERAL  ."  COMPILE
     BEGIN  UNTIL  AGAIN  WHILE  REPEAT
     DO  LOOP  +LOOP  IF  ELSE  THEN
```

MIT, SAO, 1958

Im Oktober 1957 kam Sputnik - eine sehr aufregende Zeit. Ich war im zweiten Studienjahr am MIT (Übers.: Massachusetts Institute of Technology) und leistete Werkstudentenarbeit bei SAO (Smithsonian Astrophysical Observatory, 14 Silben) an der Harvard Universität.

SAOs Aufgabe war die optische Verfolgung von Satelliten - visuelle Mondbeobachtung und Baker-Nunn-Zielverfolgungskameras. Von Sputnik aufgeschreckt, heuerte man bei SAO Studenten aus den ersten Semestern zur Berechnung von Vorhersagen mit Hilfe von Friden-Tischrechnern an. Von John Gaustad erfuhr ich einiges über die EDPM-704 von IBM am MIT und er lieh mir sein Fortran-II-Handbuch. Mein erstes Programm, Ephemeris 4, eliminierte meinen Job [Moore 1958].

Jetzt war ich Programmierer und arbeitete mit George Veis daran, dessen Verfahren der Kurvenanpassung nach der Methode der kleinsten Fehlerquadrate auf die Ermittlung von Objekten in der Umlaufbahn, auf Standortbestimmungen und schließlich auf die Erkundung der Form der Erde [Veis 1960] anzuwenden. Natürlich erstreckte sich diese Teilzeitbeschäftigung auf mindestens 40 Stunden und meine Prüfungen, was soll man sagen, waren beim Teufel.

Am MIT hielt John McCarthy eine unglaubliche Vorlesung über LISP. Hier wurde ich mit Rekursionen und mit der wundersamen Vielfalt der Computersprachen bekannt. Nach einer Bemerkung von Will Baden verhält sich LISP zum Lambda-Kalkül wie Forth zur Postfixnotation von Lukasiewicz.

APL, mit seinem höchst eigenartigen Parsen von rechts nach links, war ebenfalls aktuell. Ich bewundere an APL die Funktionen und Befehle, ich ahme sie nach, aber ich bin nicht davon überzeugt, daß sie ein optimales System darstellen.

Die Programmierumgebung der 50er Jahre war rauher als heute. Mein Quelltext füllte zwei Kästen mit Lochkarten. Die mußten herumgeschleppt und durch Maschinen gejagt werden, was ich meist selbst besorgte. Das Compilieren dauerte (wie bei C) 30 Minuten. Rechnerzeit war aber knapp und deshalb gab es nur einen Durchgang pro Tag, es sei denn, es wurde eine Extraschicht eingelegt.

So schrieb ich also diesen einfachen Interpreter, um Lochkarten einzulesen und den Programmablauf zu steuern. Er konnte auch Berechnungen dirigieren. Zu den fünf Umlaufbahn-Objekten gehörte je eine empirische Gleichung, die den Widerstand der Atmosphäre und die Tatsache, daß die Erde nicht ganz rund ist, verarbeitete. Ich konnte also für die verschiedenen Satelliten verschiedene Gleichungen zusammensetzen, ohne sie neu compilieren zu müssen.

In diesen Gleichungen wurden Ausdrücke wie P2 (Polynom zweiten Grades) und S (Sinus) addiert. Die Rechenzeit wurde hauptsächlich von 36-Bit-Gleitkomma-Operationen beherrscht und der Verwaltungsaufwand war dementsprechend gering. Ein Daten-Stack wurde nicht benötigt und ich kannte einen solchen wohl auch gar nicht.

Der Forth-Interpreter fing dabei mit den Worten

```
WORD  NUMBER  INTERPRET  ABORT
```

an. Sie waren Anweisungsnummern und wurden anders bezeichnet.

INTERPRET verwendet WORD zum Einlesen von Worten, die durch Leerzeichen getrennt sind, und NUMBER zur Umwandlung eines Wortes in Binärform (in diesem Fall als Gleitkommawert). Solche formatfreien Eingaben war ungewöhnlich, aber wirkungsvoller (kürzer und schneller) und zuverlässig. Fortran-Eingaben mußten sich an fest vorgegebene Spalten halten und Schreibfehler hatten zu unzähligen Verzögerungen geführt.

In diesem Interpreter wurde ein in Fortran programmierter "IF...ELSE IF"-Konstrukt verwendet, um nach einzelnen Zeichen zu suchen. Trat ein Fehler auf, wurde der Lauf abgebrochen. ABORT fragte dann, wie heute auch, den Benutzer, was weiter geschehen solle. Da die Lochkarten in derselben Reihenfolge abgelegt wurden, wie eingegeben und gelesen, wußte man, wo der Fehler steckt.

Stanford, SLAC, 1961

1961 ging ich nach Stanford, um Mathematik zu studieren. Stanford war gerade dabei, seine Abteilung für Computer-Wissenschaften aufzubauen, ich aber war eher an echtem Rechnen interessiert. Eindruck machte auf mich, daß die Stanforder in der Lage waren (sich trauten?), ihren eigenen Algol-Compiler zu schreiben. Und ich machte die schicksalhafte Bekanntschaft mit dem Burroughs-Computer B5500.

Ich übernahm wieder eine "Teilzeit"-Beschäftigung, diesmal bei SLAC (Stanford Linear Accelerator Center - 12 Silben!), wo ich Programme zur Optimierung der Strahlsteuerung für den beabsichtigten 2-Meilen-Elektronenbeschleuniger schrieb. Das war eine ganz natürliche Anwendung meiner Fehlerquadrat-Erfahrung auf den Phasenraum. Hal Butler leitete unsere Gruppe und das Programm, TRANSPORT, war recht erfolgreich.

Eine weitere Fehlerquadrat-Anwendung war das Programm CURVE, in Algol geschrieben (1964). Das ist ein allgemein verwendbares, nichtlineares, differentiell-ausgleichendes Daten-Anpassungs-Programm. Mit statistischer Strenge liefert es einen Einblick in die Übereinstimmung zwischen Modell und Daten.

Das Datenformat und die Modellgleichungen wurden interpretiert und ein Ablagestapel wurde eingerichtet, um die Berechnungen zu erleichtern. CURVE war ein beeindruckender Vorläufer von Forth. Um viel komplizierteren Modellen als nur Gleichungen entsprechen zu können, führte es die folgenden Worte ein:

```
+  -  *  NEGATE
IF  ELSE  THEN  <
DUP  DROP  SWAP
:  ;  VARIABLE  !  (
SIN  ATAN  EXP  LOG
```

Bezeichnet wurden sie ganz anders:

```
NEGATE  war  MINUS
  DROP       ;
  SWAP       .
     !       <
VARIABLE      DECLARE
     ;        END
( ...)        COMMENT ...;
```

Der Interpreter verwendete IF...ELSE IF, um ein Eingabewort von 6 Zeichen Länge, das ATOM (von LISP) genannt wurde, zu identifizieren. DUP DROP und SWAP sind Befehle von der 5500er. Die veränderten Bezeichnungen überraschen mich. Das Wort **:** wurde von der Label-Kennzeichung aus Algol übernommen und zum Parsen von links nach rechts umgestellt (damit der Interpreter nicht auf ein noch nicht definiertes Wort stößt).

```
Algol - LABEL:
CURVE - : LABEL
```

An sich wurde mit **:** eine Stelle im Eingabestrom gekennzeichnet, die später interpretiert werden sollte. Mit **;** hörte die Interpretation auf. Es gab eine Version von **:** , die DEFINE genannt wurde.

Der Abspeicheroperator (!) erschien in Verbindung mit VARIABLE. Das Werteholen (@) kam aber ganz automatisch. Zu bemerken ist, daß die Eingaben komplex genug geworden waren, um nach Kommentaren zu verlangen. Das gelegentlich kritisierte Postfix-Bedingungsstrukturwort hat hier seinen Ursprung:

```
Algol - IF ausdruck THEN true ELSE false
CURVE - stack IF true ELSE false THEN
```

Wenn "stack" ungleich Null ist, erscheint "true". THEN liefert einen eindeutigen Abschluß. In Algol hat mich das Fehlen eines solchen Abschlusses immer etwas verwirrt. Solche Ausdrücke wurden interpretiert: Mit IF wurde in einer Vorwärtssuche nach ELSE oder THEN gefahndet.

Mit dem Wort < kam die Übereinkunft, daß Relationen einen Wahrheitswert auf dem Stack hinterlassen, 1 für true und 0 für false. Die transzendenten Funktionen sind natürlich Bibliotheksaufrufe.

Freischaffender

1965 verließ ich Stanford und machte mich als freier Programmierer im Raum New York selbständig. Das war nichts Ungewöhnliches und ich bekam Programmieraufträge in Fortran, Algol, Jovial, PL/I und verschiedenen Assemblern. Ich schleppte buchstäblich mein Kartenpack mit mir herum und paßte es vom Programm her den jeweiligen Gegebenheiten an.

Minicomputer kamen auf, und mit ihnen die Terminals. Für Teletype-Eingaben war der Interpreter ideal und er wurde schnell um Befehle erweitert, die Ausgaben handhaben konnten. So kamen wir zu den Worten:

```
KEY  EXPECT
EMIT  CR  SPACE  SPACES  DIGIT  TYPE
```

EXPECT ist eine Schleife, die KEY aufruft, um eine gedrückte Taste einzulesen. TYPE ist eine Schleife, die EMIT aufruft, um ein Zeichen anzuzeigen.

Mit dem Fernschreiber kam der Lochstreifen auf, und die unangenehmste Software, die man sich vorstellen konnte - stundenlanges Editieren, und Stanzen, und Laden, und Assemblieren, und Ausdrucken, und wieder Laden, und Testen, und Wiederholen. Ich erinnere mich an einen schrecklichen Sonntag in einem Wolkenkratzer in Manhattan, als ich kein Lochstreifenklebeband finden konnte (was anderes ging nicht) und mir schwor: "Da muß es doch einen besseren Weg geben!".

Ich habe eine beträchtliche Zeit für Bob Davis bei Realtime Systems, Inc (RSI) gearbeitet. Es ging um dessen Zeitteilungseinrichtung (Ferneingabe in einen zentralen Rechner). Aus mir wurde ein 5500-MCP-Guru und ich schrieb einen Fortran-Algol-Übersetzer und Dienstprogramme zur Dateienverwaltung. Der Übersetzer lehrte mich Zwischenräume zwischen den Worten schätzen, die ja in Fortran nicht nötig sind.

Der Interpreter unterschied die Worte immer noch nur nach den ersten 6 Zeichen (die 5500 hatte 48-Bit-Worte). Es tauchen die Worte

```
LIST  EDIT  BEGIN  AGAIN  EXIT
```

auf, wobei BEGIN...AGAIN mit START...REPEAT bezeichnet und dazu benutzt wurde, die Editorkommandos

```
T  TYPE  I  INSERT  D  DELETE  F
FIND
```

die später im Editor von NRAO Verwendung fanden, einzuschließen. Das Wort FIELD wurde wie bei Mohasco und der Datenbankverwaltung von Forth, Inc eingesetzt.

Von hier stammt eine kennzeichnende Eigenschaft von Forth. Forth hängt jeder Eingabezeile, sobald die Interpretation abgeschlossen ist, ein OK an. Unter Umständen nicht ganz einfach. Sobald nämlich die Eingabe mit dem CR beendet wird, muß als Echo erst ein Leerzeichen und dann das CR zusammen mit dem OK ausgegeben werden. Bei RSI stand das OK auf der nächsten Zeile, aber es gab einem immer noch eine freundliche Versicherung der Richtigkeit bei einer einschüchternden Kommunikationszeile wie der folgenden:

```
56 INSERT ALGOL IS VERY ADAPTABLE
OK
```

Die Postfixnotation legt einen Daten-Stack nahe, der aber nur einen Eintrag tief zu sein brauchte.

Mohasco, 1968

1968 verwandelte ich mich in einen kaufmännischen Programmierer bei Mohasco Industries, Inc in Amsterdam NY. Das ist ein größeres Einrichtungshaus - Teppiche und Möbel. Bei RSI hatte ich mit Geoff Leach zusammengearbeitet und der überredete mich, ihm ins nördliche Hinterland zu folgen. Ich hatte gerade geheiratet und Amsterdam NY wartet im Gegensatz zur Stadt New York mit einer reizenden Kleinstadt-Atmosphäre auf.

Ich schrieb meine Programme nach COBOL um und lernte die ganze Wahrheit über Wirtschafts-Software. Bob Rayco hatte die Verarbeitung der Geschäftsdaten unter sich und wies mir zwei bedeutende Projekte zu:

Er leaste einen Minicomputer des Typs IBM 1130 mit dem graphischen Display 2250. Man wollte herausfinden, ob man mit Hilfe von Computer-Graphik Teppichmuster entwerfen kann. Die Antwort war "nicht ohne Farbe", und weg war die 1130.

Inzwischen hatte ich die allermodernste Minicomputer-Umgebung: 16-Bit-CPU, 8k RAM, Diskettenlaufwerk (mein erstes), Tastatur, Drucker, Lochkartenleser und -stanzer, Fortran-Compiler. Der Kartenleser und -stanzer hatte eine Möglichkeit zur Sicherung auf Diskette. Ich übertrug meinen Interpreter aufs neue (nach Fortran zurück) und fügte einen Cross-Assembler hinzu, um Programme für die 2250 zu erzeugen.

Das System war ein großer Erfolg. Es konnte bewegte 3D-Bilder zeichnen, wo doch IBM kaum mit statischem 2D zurechtkam. Da mir hier zum ersten Mal Echtzeit-Graphik zur Verfügung stand, schrieb ich Spacewar, jenes erste Videospiel. Ich übertrug auch mein Algol-Schachprogramm nach Forth und war gehörig davon beeindruckt, um wieviel einfacher das wurde.

Die Datei, die den Interpreter enthielt, war mit FORTH bezeichnet, für Software der 4ten (nächsten) Generation: "fourth generation" - aber das Betriebssystem ließ für Dateinamen nur 5 Zeichen zu.

Diese Umgebung zur Programmierung der 2250 war der

Fortran-Umgebung weit überlegen. Ich dehnte also den 2250-Cross-Assembler zu einem 1130-Compiler aus. Damit wurde eine Schar neuer Worte eingeführt:

```
DO  LOOP  UNTIL
BLOCK  LOAD  UPDATE  FLUSH
BASE  CONTEXT  STATE  INTERPRET  DUMP
CREATE  CODE  ;CODE  CONSTANT  SMUDGE
@  OVER  AND  OR  NOT  0=  0<
```

Auch die liefen unter anderer Bezeichnung:

```
 LOOP   war   CONTINUE
UNTIL         END
BLOCK         GET
 LOAD         READ
 TYPE         SEND
INTERPRET     QUERY
CREATE        ENTER
 CODE         das Cent-Zeichen
```

Die einzige Verwendung, die ich je für das Cent-Zeichen gefunden hatte. Index und Grenzen von Schleifen lagen auf dem Daten-Stack. DO und CONTINUE waren ein Tribut an Fortran.

BLOCK verwaltet eine Anzahl von Puffern, um den Diskettenzugriff zu minimieren. LOAD liest Quelltext von einem 1024 Byte langen Block aus ein. 1024 wurde als schönes Stück Diskettenaufteilungseinheit gewählt und hat sich als gute Wahl herausgestellt. Mit UPDATE kann man einen Block markieren und ihn später, wenn sein Pufferplatz benötigt wird (oder per FLUSH), auf Diskette zurückschreiben. Es implementiert virtuellen Speicher und verbirgt sich in den Abspeicherworten (!).

BASE erlaubt neben Dezimalzahlen auch Oktal- und Hexadezimalzahlen. CONTEXT war der erste Schritt zu Vokabularen und diente dazu, Editor-Worte abzugrenzen. STATE unterschied zwischen Compilation und Interpretation. In der Compilationsphase wurden das Längenbyte und die ersten drei Zeichen eines jeden Wortes zur späteren Interpretation compiliert. Eigenartigerweise konnten die Worte durch beliebige Begrenzungszeichen beendet werden, eine Verirrung, die schnell wieder aufgegeben wurde. Das Zeichen zum Hereinholen (@) erschien in vielerlei Gewand, da man zwischen dem Hereinholen von Variablenwerten, dem Hereinholen aus Feldern oder dem von Diskette unterscheiden mußte. DUMP wurde für die Untersuchung des Arbeitsspeichers wichtig.

Was aber am allerwichtigsten war: Jetzt gab es ein Dictionary. Die zu interpretierenden Worte hatten jetzt einen Namen und wurden in einer verketteten Liste auf Übereinstimmung mit der Eingabe verglichen. CREATE konstruiert den klassischen Dictionary-Eintrag:

```
Link zum vorigen Eintrag
Längenbyte und 3 Zeichen
auszuführendes Programmstück
Parameter
```

Eine wichtige Neuerung war das Codefeld: Ein indirekter Sprung war der einzige zusätzlich benötigte Aufwand, wenn ein Wort erst einmal aufgefunden war. Den Wert des Längenbytes zur Unterscheidung von Worten lehrten mich die Compiler-Schreiber bei Stanford schätzen.

Eine wichtige Klasse von Worten erschien mit CODE. Im Parameterfeld folgten Maschinenbefehle. Jetzt konnte also jedes Wort, das der Computer hergab, definiert werden. ;CODE legt das für eine neue Wortklasse auszuführende Programmstück fest, und damit wurde jener Begriff eingeführt, den man heute Objekt nennt.

Mit SMUDGE wurden Rekursionen während der Interpretation eines Wortes unterbunden. Weil das Dictionary beim Aufsuchen eines Wortes von der jüngsten zur ältesten Definition hin durchlaufen wird, treten dabei im Normalfall Rekursionen auf.

Schließlich erschien der Return-Stack. Bis dahin waren die Wortdefinitionen nicht verschachtelt gewesen oder sie verwendeten den Datenstack zur Aufbewahrung ihrer Rücksprungadressen. Alles in allem war das eine Zeit großer Innovationsschübe in der sprunghaften Entwicklung von Forth.

Die erste Arbeit über Forth, ein interner Mohasco-Bericht, wurde von Geoff und mir geschrieben [Moore 1970]. Sie hätte heute immer noch ihre Berechtigung.

1970 bestellte Bob eine Univac 1108. Ein ehrgeiziges Projekt zur Unterstützung eines Netzwerkes aus gemieteten Leitungen für ein Auftragseingangssystem. Ich hatte einen Berichtsgenerator in Forth geschrieben und war davon überzeugt, daß ich auch Auftragseingänge programmieren könne. Um glaubhafter zu wirken, übertrug ich Forth (als eigenständiges System!) auf die 5500. Im Unternehmen schrieb man aber in COBOL. Der fabelhafte Kompromiß bestand darin, ein Forth-System auf der 1108 zu installieren, das mit COBOL-Modulen zur Verarbeitung von Tätigkeitsberichten zusammenwirkte.

Ich erinnere mich noch lebhaft daran, wie ich in diesem Winter nach Schenectady hin- und herfuhr, um Extraschichtrechenzeit für die 1107 zu borgen. Bei meinem TR4-A (Übers.: Triumph Cabrio, Baujahr 1965) fehlten Bodenbeläge und Fenster, so daß das Ganze zu einem nächtlichen Überlebenstraining wurde. Aber dem System war ein unglaublicher Erfolg beschieden. Selbst Univac war von dessen Leistungsfähigkeit beeindruckt (Projektkoordinator war Les Sharp). Eigentlicher Maßstab war die Reaktionszeit. Ich war aber entschlossen, das System wartbar (klein und einfach) zu halten. Leider Gottes veranlaßte ein wirtschaftlicher Rückschlag die Geschäftsleitung, die 1108 aufzugeben. Das war kein guter Rückzieher. Davon bin ich noch heute überzeugt. Ich war der erste, der resignierte.

Das Forth für die 1108 mußte in Assembler programmiert vorliegen. Es pufferte Ein- und Ausgabepakete und teilte die

CPU-Zeit zwischen den Programmstücken, die die einzelnen Leitungen bedienten, auf. Hier haben Sie Ihr klassisches Betriebssystem! Es nahm aber auch die Interpretation der Eingaben vor und PERFORMte die entsprechenden COBOL-Module. Es verwaltete Trommelspeicherpuffer und gepackte und ungepackte Datensätze. Die Worte

```
BUFFER  PREV  OLDEST
TASK  ACTIVATE  GET  RELEASE
```

stammen aus dieser Zeit. BUFFER verhinderte Lesezugriffe auf die Diskette, wenn der angeforderte Block als leer bekannt war. PREV (previous = voriger) und OLDEST sind Systemvariablen, die eine Pufferverwaltung nach dem Prinzip des am längsten nicht mehr benutzten Puffers aufbauen. TASK definiert eine Task (Teilaufgabe) zur Boot-Zeit, und ACTIVATE startet sie, sobald sie benötigt wird. GET und RELEASE verwalten die gemeinsam benutzten Hilfsmittel (Trommelspeicher, Drucker). Das Wort PAUSE bestimmt, wie eine Task die Steuerung durch die CPU wieder abgibt. Es ist Bestandteil aller Ein- und Ausgabeoperationen und tritt im Geschäftsberichtsprogramm nicht unmittelbar in Erscheinung. Mit seiner Hilfe ist ein einfacher Zeitscheiben-Algorithmus möglich, bei dem keine der Tasks das System lahmlegen kann.

Nachdem ich gekündigt hatte, schrieb ich ein zorniges Gedicht und ein Buch, das niemals veröffentlicht wurde. Es beschrieb, wie man Forth-Software entwickelt, und rief zur Einfachheit und Innovationsfreudigkeit auf. Es beschrieb auch indirekt gefädelte Systeme. Diese wurden aber erst bei NRAO zum ersten Mal implementiert.

Ich schlug mich mit dem Begriff der Metasprache herum, Sprache, die über Sprache spricht. Forth konnte jetzt einen Assembler interpretieren, der einen Compiler assemblierte, welcher wiederum den Interpreter compilierte. Ich gelangte schließlich zur Überzeugung, daß mir die Terminologie nicht weiterhilft. Der Ausdruck "Meta-Compilieren" für das Compilieren von Forth ist aber immer noch in Gebrauch.

NRAO, 1971

George Conant bot mir eine Stellung bei NRAO (National Radio Astronomy Observatory, 15 Silben) an. Ich hatte ihn bei SAO kennengelernt und er mochte Ephemeris 4. Wir gingen also nach Charlottesville VA und verbrachten Sommer über Sommer in Tucson AZ, wenn das Radio-Teleskop auf dem Kitt Peak für Wartungsarbeiten zur Verfügung stand.

Es sollte ein Minicomputer des Typs 316 von Honeywell zur Steuerung einer neuen Filter-Bank für das 36-Fuß-Millimeterwellenteleskop programmiert werden. Der 316er hatte ein neunspuriges Band und ein Terminal mit einer Tektronix-Speicherröhre. George ließ mir bei der Systementwicklung freie Hand, war jedoch über das, was dabei herauskam, nicht erfreut. Bei NRAO war man auf Fortran fixiert und ich nannte Forth inzwischen eine Sprache. Er hatte recht: Die Firmen sollten sich auf eine und nur eine Sprache einigen. Jetzt wollten auch andere Programmierer ihre eigene Sprache haben.

Nun, ich hatte Forth auf dem IBM-360/50-Großrechner in Assembler geschrieben. Dann cross-compilierte ich es auf die 316. Dann compilierte ich es auf der 316 von neuem (ich hatte zwar auf dem 360 ein Terminal, jedoch mit furchtbar hohen Reaktionszeiten). Als das System erst einmal lief, war das Anwendungsprogramm schnell geschrieben. Es ging um zweierlei Beobachtungsarten, Kontinuum und Spektrallinien. Spektrallinien machten mehr Spaß, weil ich die Spektren so ausgeben lassen konnte, wie sie empfangen wurden, wobei ich die Linienverläufe nach der Methode der kleinsten Quadrate anpaßte [Moore 1973].

In Tucson hatte Ned Conklin die Verantwortung und das System kam dort gut an. Es brachte den Stand der Technik in der Online-Datenreduzierung ein gutes Stück voran. Die Astronomen entdeckten und kartographierten damit interstellare Moleküle, gerade als dieser Zweig der Forschung in den Brennpunkt des Interesses rückte.

Zur Unterstützung am Ort des Geschehens wurde Bess Rather verpflichtet. Sie mußte sich zunächst einmal mit dem Forth-System bekannt machen und es dann weitestgehend ohne meine Hilfe erklären und dokumentieren. Das Jahr darauf programmierte ich die DDP-116 zur Optimierung der Teleskopausrichtung um. Darauf hin ersetzten Bess und ich die 116 und die 316 durch eine DEC-PDP-11.

Die Neuerung, durch die das alles ermöglicht wurde, hieß indirekte Fädelung. Das war eine ganz natürliche Weiterentwicklung meiner Arbeit bei Mohasco. (Später hörte ich allerdings, daß man bei DEC in einem der Compiler direkt gefädelten Code benutzte). Statt den Text der Wort-Definitionen ständig neu zu interpretieren, werden die Adressen der einzelnen Dictionary-Einträge compiliert. Dadurch stieg die Leistungsfähigkeit enorm, da ja jeder Wortbezug nur 2 Bytes benötigte und ein Adressen-Interpreter die Wort-Definitionen sehr viel schneller durchlaufen konnte. Der Interpreter selbst bestand auf der 11 aus einem 2-Wort-Makro:

```
: NEXT   IP )+ W MOV  W )+ ) JMP ;
```

Jetzt war Forth fertig. Und das wußte ich. Ich konnte schneller Programme schreiben, die leistungsfähiger und zuverlässiger waren. Außerdem konnte man sie leicht auf andere Systeme übertragen. Ich fuhr damit fort, die 116 umzuprogrammieren, mit der das 300-Fuß-Green-Bank-Teleskop (Übers.: Reflektordurchmesser 91,5 m) eingestellt wurde, und den HP-Mini, mit welchem die VLBI-Astronomie (Übers.: Radiointerferometrie) begann. George gab mir ein ModComp und ich beschäftigte mich mit Fourier-Transformierten für die Interferometrie und für die Suche nach Pulsaren (64k Daten). Ich

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

HCC-Forth-gebruikersgroep.

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

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

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

konnte sogar zeigen, daß auf dem 360 die komplexe Multiplikation mit Forth um 20% schneller ging als mit Assembler.

Bei NRAO wurde sehr geschätzt, was ich da zusammengeschmiedet hatte. Man stand mit einer Beratungsfirma in Verbindung, die hinter neuen Nebenprodukt-Techniken her war. Die Frage der Patentierung von Forth wurde ausgiebig diskutiert. Aber weil Software-Patente umstritten waren und den Obersten Gerichtshof hätten auf den Plan rufen können, ließ NRAO von der weiteren Verfolgung dieser Angelegenheit wieder ab. Die Rechte gingen daraufhin wieder an mich zurück. Meiner Meinung nach sollten Ideen nicht patentierbar sein. Rückblickend bestätigt sich, daß die einzige Chance für Forth im Public Domain lag. Dort blüht und gedeiht es seither.

Die Fädeltechnik veränderte die Strukturworte (wie DO LOOP IF THEN). Deren Implementierung wurde eleganter, mit Adressen, die während der Compilation auf dem Daten-Stack lagen.

Jetzt hatte jedes Forth für den speziellen Computer, auf dem es lief, einen Assembler. Der arbeitet mit Postfix-Opcodes, setzt Adressen auf dem Daten-Stack zusammen und verwendet für Verzweigungen Forth-ähnliche Strukturworte. Die vom Hersteller stammenden Abkürzungen für die Maschinenbefehle werden über ;CODE als Wortklassen definiert. Kann an einem Nachmittag programmiert werden. Das obenstehende Makro für NEXT ist ein Beispiel.

Etwas ungewöhnlich aussehende Arithmetik-Befehle erwiesen sich als nützlich:

```
     M*   */   /MOD   SQRT   SIN.COS   ATAN
EXP   LOG
```

M* ist die übliche Hardware-Multiplikation zweier 16-Bit-Zahlen zu einem 32-Bit-Produkt (Argumente natürlich auf dem Daten-Stack). Dazu dann das */ mit der Division zur Implementierung einer Rationalzahl-Arithmetik. /MOD liefert sowohl den Quotienten wie auch den Rest und eignet sich in idealer Weise zur Lokalisierung von Datensätzen in einer Datei. SQRT liefert aus einem 32-Bit-Argument ein 16-Bit-Ergebnis. SIN.COS liefert als Ergebnis neben dem Sinus auch den Cosinus, was für die Vektorrechnung und den Umgang mit komplexen Zahlen (FFT) von Nutzen ist (Übers.: FFT = Fast Fourier Transformation = Schnelle Fourier-Transformation). ATAN, die hierzu inverse Operation, liefert quadranten-eindeutige Ergebnisse. EXP und LOG liefen zur Basis 2.

FIGUK

(Englische Forth-Gesellschaft)

Treten Sie unserer Forth-Gruppe bei.
Verschaffen Sie sich Zugang zu unserer umfangreichen Bibliothek.
Sichern Sie sich alle zwei Monate
ein Heft unserer Vereinszeitschrift.
(Auch ältere Hefte erhältlich)
Suchen Sie unsere Webseite auf:
www.users.zetnet.co.uk/aborigine/Forth.htm
Lassen Sie sich unser Neuzugangs-Gratis-Paket geben.
Der Mitgliedsbeitrag beträgt 12 engl. Pfund.
Hierfür bekommen Sie 6 Hefte unserer
Vereinszeitschrift Forthwrite.
Beschleunigte Zustellung (Air Mail)
ins Ausland kostet 20 Pfund.
Körperschaften zahlen 36 Pfund,
erhalten dafür aber viel Werbung.

Wenden Sie sich an:

Dr. Douglas Neale
58 Woodland Way
Morden Surrey
SM4 4DS
Tel.: (44) 181-542-2747
E-Mail: dneale@w58wmorden.demon.co.uk

Alle diese Funktionen arbeiteten mit Festkomma-Arithmetik - 14 oder 30 Bits rechts vom binären Komma (Punkt) für trigonometrische Funktionen, 10 für Logarithmen. Das wurde zu einem charakteristischen Merkmal von Forth. Festkomma ist einfacher, schneller und genauer als Gleitkomma. Hardware- und Software-Gleitkomma läßt sich aber leicht implementieren.

Mein Beifall gebührt der unschätzbaren Arbeit von Hart [Hart 1978], der Näherungstabellen verschiedener Genauigkeit für Funktionswerte aufgestellt und die an vorderster Front Kämpfenden damit von den Unzulänglichkeiten der existierenden Programmbibliotheken befreit hat.

Das Wort DOES> (in der Schreibweise **;:**) kam hinzu. Es definiert (genau wie ;CODE) eine Klasse von Worten, bei denen festgelegt wird, welche Definition ausgeführt werden soll, wenn sie aufgerufen werden. Es war gar nicht so einfach, DOES> zu erfinden, es erwies sich aber bei der Definition von Opcodes als ganz besonders nützlich.

Trotz allem gelang es mir nicht, die bei Charlottesville von den Vorteilen von Forth zu überzeugen. Sie erlaubten mir nicht, das VLA (Übers.: Very Large Array, Verbund aus Radio-Teleskopen) zu programmieren. In jeder Gruppe gab es 25%, die für Forth, und 25%, die streng dagegen waren. Es wurde heftig argumentiert und Kompromiß war keiner in Sicht. Diejenigen, die dafür waren, schlossen sich zusammen und gründeten die Firma Forth, Inc. Doch das ist eine andere Geschichte.

Moral

Die Forth-Geschichte erscheint einem wie ein Moralstück: Hartnäckiger junger Programmierer kämpft gegen die Gleichgültigkeit bei seiner Suche nach der Wahrheit und der Errettung seiner leidenden Kameraden. Aber Besserung ist in Sicht: Man beobachte nur Forth, Inc, wie die sich bei einem französischen Bankbearbeitungssystem IBM entgegenstemmen.

Ich weiß, daß Forth die beste Sprache ist, die es je gegeben hat. Ich freue mich über ihren Erfolg, besonders auf dem ultrakonservativen Schauplatz der künstlichen Intelligenz. Es macht mir zu schaffen, daß Leute, von denen ich es eigentlich erwartet hätte, nicht wahrhaben wollen, wie sehr Forth ihre eigene Beschreibung von einer idealen Sprache verkörpert.

Aber ich forsche immer noch ohne Lizenz. Forth hat zu einer Architektur geführt, die ein wunderbares Zusammenwachsen von Software und Silizium verspricht. Und eine weitere neue Programmierumgebung.

Literatur

[ANS 1991] Draft Proposed ANS Forth (vorläufige Fassung des vorgeschlagenen ANS-Forth), Dokument X3.215-199x, erhältlich von Global Engineering Documents, 2805 McGaw Ave., Irvine CA 92714.

[Brodie, 1981] Brodie, Leo, Starting FORTH, Englewood Cliffs NJ: Prentice-Hall, 1981, ISBN 0-13-842930-8.

[Hart, 1968] Hart, John F. et al, Computer Approximations. Malabar FL: Krieger, 1968; (zweite Ausgabe), 1978, ISBN 0-88275-642-7.

[Martin, 1987] Martin, Thea, A Bibliography of Forth References, 3. Ausgabe, Rochester NY: Institute for Applied Forth Research, 1987, ISBN 0-914593-07-2.

[Moore, 1958] Moore, Charles H. and Lautman, Don A., Predictions for photographic tracking stations - APO Ephemeris 4, in SAO Special Report Nr.11, Schilling G. F., Ed., Cambridge MA: Smithsonian Astrophysical Observatory, März 1958.

[Moore, 1970] --- and Leach, Geoffrey C., FORTH - A Language for Interactive Computing, Amsterdam NY: Mohasco Industries, Inc. (interne Veröffentlichung) 1970.

[Moore, 1972] --- and Rather, Elizabeth D., The FORTH program for spectral line observing on NRAO's 36 ft telescope, Astronomy & Astrophysics Supplement Series, Band 15, Nr. 3, Juni 1974, Proceedings of the Symposium on the Collection and Analysis of Astrophysical Data, Charlottesville VA, Nov. 1972, 13-15.

[Moore, 1980] ---, The evolution of FORTH, an unusual language, Byte, 5:8, August 1980.

[Rather, 1993] Rather, Elizabeth D., Colburn, Donald R. and Moore, Charles H., The Evolution of Forth, in History of Programming Languages-II, Bergin, T. J. and Gibson, R. G., Ed., New York NY: Addison-Wesley, 1996, ISBN 0-201-89502-1.

[Veis, 1960] Veis, George and Moore, C. H., SAO differential orbit improvement program, in Tracking Programs and Orbit Determination Seminar Proceedings, Pasadena CA: JPL, Februar 1960, 23-26.

Zyklische Codes

Raffael Deliano

Mit freundlicher Genehmigung des Autors entnommen aus:
„Embedded 7“ April 2002

FEC ("Forward Error Correction") wird in der Datenübertragung, z.B. 433 MHz Funkmodulen vermehrt angewendet. Dabei eignen sich besonders die hier dargestellten sequentiellen Verfahren, die auf Schieberegistern beruhen. Sie können bei niederigen Bitraten in Software implementiert werden.

$$b(x) = x^4 + x^2 + x + 1$$

$$1x^4+0x^3+1x^2+1x^1+1x^0$$

1 0 1 1 1

MSB LSB

Bild 1: Polynom auf Bits

Arithmetik

Etwas Theorie kann nicht völlig vermieden werden. Da Reed-Solomon-Codes vernachlässigt werden, wird nur mit binären Datenworten, hier Polynome genannt, gearbeitet. Diese haben meist krumme Länge, also nicht 8 Bit oder ein Vielfaches davon.

In Bild 1 ist die Umwandlung zwischen Polynomschreibweise und binärem Datenwort dargestellt. Sie erscheint umständlich, aber da in der Literatur statt Hex auch Oktalzahlen verwendet werden und das LSB nicht immer rechts ist, ist es die einzig sichere Notation.

Es gibt auch für Schieberegister Operationen mit dem Namen Addition, Subtraktion, Multiplikation und Division. Doch funktionieren sie anders als gewohnt, nämlich streng bitweise ohne Carry. Wie man in Bild 2 sieht, entsprechen sich damit Addition und Subtraktion. Der XOR-Befehl der CPU kann diese Funktionen für Datenworte direkt ausführen.

Etwas mühsamer ist Multiplikation, am Beispiel in Bild 3 dargestellt. Erinnert aber nicht ungefähr an Shift & Add-Multiplikation in der normalen Arithmetik. Das Wort „a“ wird schrittweise nach links geschoben, was die CPU mit LSL leicht ausführen kann.

1	1	0	0
+1	+0	+1	+0
0	1	1	0

(Carry)

1	1	0	0
-1	-0	-1	-0
0	1	1	0

(Borrow)

Bild 2: Addition und Subtraktion

Gleichzeitig erfolgt Addition durch XOR-Befehl, falls in Wort „b“ dieses Bit gesetzt ist.

Bild 4 zeigt die Division. Den Divisor aus der Ruhelage nach links schieben, bis sein oberstes Bit das oberste Bit des Dividenden abdeckt. Dieses oberste Bit im Ergebnis setzen, wenn es im Dividenden auch Eins ist. Dann Subtraktion, d.h. XOR durchführen und Divisor nach rechts schieben. Zum Schluß kann ein Rest R im Dividenden-Register stehen bleiben.

$$a(x) * b(x) = c(x)$$

$$a(x)=x^3+x+1$$

$$b(x)=x^4+x^2+x+1$$

b(x)	a(x)	x^3	x^2	x^1	x^0
x^0 1				1	0 1 1
x^1 1			1	0	1 1
x^2 1		1	0	1	1
0					
x^4 1	+1	0	1	1	
c(x) =	1	0	0	0	0 0 0 1

Bild 3: Multiplikation

```
      a          /b     =c
C=  1100110000 / 10011 = 110110
 1  10011 <-
    10101   5
 1   10011
     01100
 0   00000
     11000
 1   10011
     10110
 1   10011
     01010
 0   00000
     1010   R=
```

Bild 4: Division

Es empfiehlt sich, die gezeigten Funktionen zu programmieren, so daß wenigstens 16 Bit Hexzahlen verarbeitet werden können. Für Echtzeitanwendung zu ineffizient, aber zur Erzeugung von Testvektoren nützlich. Die Division funktioniert so meist nur, wenn der Divisor kürzer als der Dividend ist. Deshalb ist Vorsicht geboten. Mit der Multiplikation kann man die Funktion der Division prüfen, indem man Quotient und Divisor multipliziert und den Rest addiert. Was wieder den Dividenden ergeben sollte.

Konstanten

Schieberegisterschaltungen werden in der Literatur zeichnerisch vereinfacht dargestellt (Bild 5), Bild 7 zeigt dagegen die in der Schaltungstechnik übliche Darstellung.

Bei den FlipFlops wird angenommen, daß sie vor dem Beginn der Rechnung durch einen Clear-Eingang gelöscht wurden. Man beachte das auch für Software.

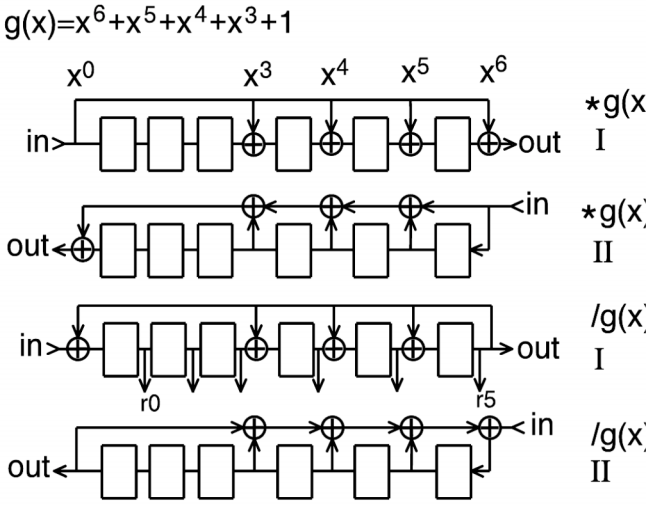


Bild 5: Multiplikation und Division von Konstanten

In Hardware sind große XORs aus Gates mit 2 Eingängen realisierbar (Bild 6). Verwendbar sind aber auch Paritygeneratoren wie 74HC280 oder CD4531.

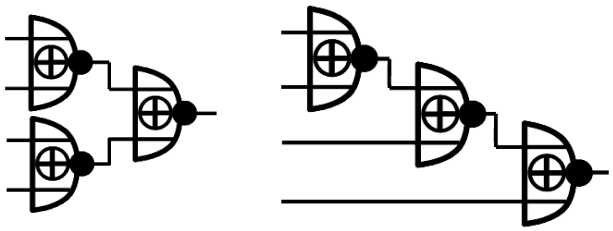


Bild 6: XOR mit mehreren Eingängen

Für die hier relevanten Funktionen gibt es Realisierungsvarianten I und II. In der neueren Literatur wird I als „Galois-Konfiguration“ und II als „Fibonacci-Konfiguration“ bezeichnet. In MSI-Hardware wird Anordnung II bevorzugt. Die Literatur folgt dieser Vorgabe. Anordnung I hingegen ist in Software günstiger, weil man XOR-Befehle besser nutzen kann. Die Umwandlung zwischen beiden Varianten ist nicht immer möglich.

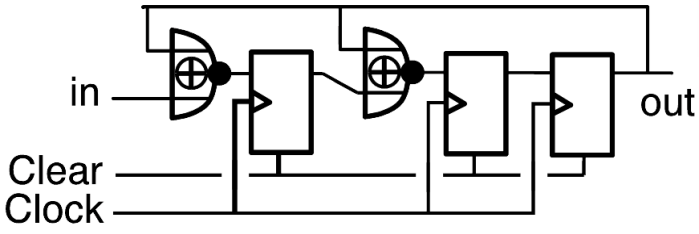


Bild 7: Hardwareschaltung

Multiplikation und Division mit Konstanten lassen sich mit den Schaltungen in Bild 5 durchführen. Wieviele FlipFlops man braucht, hängt von der Länge des fixen Divisors g(x) ab. Er bestimmt auch, wo die XORs eingefügt werden. Das Schema ist anhand des Beispiels wohl leicht zu erkennen. Im Betrieb wird das Datenwort MSB first eingeschoben und kommt MSB first am anderen Ende wieder heraus. Die Multiplikationsschaltung wird seltener benötigt. Bei den Divisionsschaltungen ist die Variante I nützlicher, weil bei ihr nach Ende der Rechnung der Rest R im Speicher stehen bleibt. (Es finden sich in der Literatur [2] vereinzelt Darstellungen wie man aus II den Rest erzeugen kann, aber experimentell hat das bei mir nicht funktioniert.)

Es empfiehlt sich die Division Typ I zu programmieren, weil sie wesentliche Grundlage der Fehler-Korrekturschaltungen ist. Übergibt man den Divisor g(x) als 16 Bitzahl auf dem Stack kann das Programm automatisch seine Länge bestimmen. Es muß dazu ja nur das oberste gesetzte Bit im Polynom suchen. Damit ist das Programm für unterschiedliche Polynome und Datenwortlängen direkt verwendbar. Für praktischen Einsatz zu kompliziert und langsam, dort ist Assembler nötig. Aber zur Simulation und Erstellung von Testvektoren nützlich.

Generator

„Linear Feedback Shift Register“ (LFSR) werden meist im Zusammenhang mit der Erzeugung von binären Pseudozufallszahlen dargestellt. Auch diese Funktion ist in Variante I und II darstellbar. Es handelt sich um Divisionsschaltungen, bei denen die Eingangsdaten konstant Null sind und damit das XOR am Eingang entfällt (Bild 8). Unter der Voraussetzung, daß das Register mit einem Wert ungleich Null geladen wurde, erzeugen bestimmte Polynome zyklisch alle für die Wortlänge möglichen Datenworte, ausser eben Null. Denn das ist die magische Zahl (demon state, hang-up), mit der die Maschine zu funktionieren aufhört und die deshalb auch im Betrieb als Generator nie auftauchen wird.

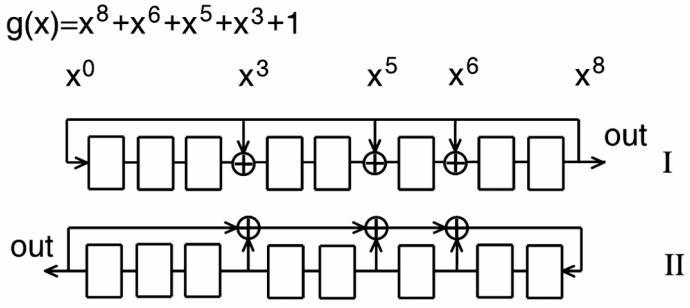


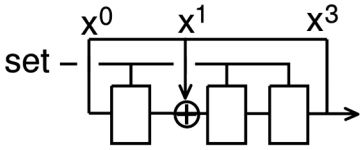
Bild 8: LFSR-Generatoren

In Tabelle 1 ist das erzeugte Muster für ein kurzes Register dargestellt, Startwert war 111b. Die Zykluslänge ist 2^3 - 1 = 7 (vgl. den Pfeil).

Nur „primitive Polynome“ zeigen dieses Verhalten. Tabelle 2 listet diese binär anhand der Wortlänge. Nur ein Teil (ca. 20%) der Rückkopplungskombinationen, die anhand der Wortbreite möglich sind, funktionieren als primitive Polynome. Tabelle 3 gibt eine kurze Aufstellung, die zeigt, daß trotz-

dem mit steigender Wortlänge viele Generatorpolynome verfügbar werden. Aber von diesen eignen sich wiederum nur wenige für fehlerkorrigierende Codes oder gute Rauschgeneratoren.

Aus jedem Polynom in Tabelle 2 läßt sich ein zweites, "reziprokes" Polynom direkt ableiten. Die nötige Umordnung der Bits erfolgt durch Spiegelung an der Mittellinie (Bild 9). (Angeblich erzeugen diese das Zyklusmuster rückwärts, aber experimentell hat das bei mir noch nicht funktioniert).



kurzes Register
zu Tabelle 1

Tabelle 1: Generatorlauf

out	register
0	110 <-
1	101
1	011
1	111
0	110
0	100
1	001
0	010
1	101
1	011
1	111
0	110 <-
0	100

Tabelle 2:
primitive Polynome

n	xn	x0
2	111	
3	1011	
	1101	
4	10011	
	11001	
5	100101	
	101001	
	101111	
	110111	
	111011	
	111101	
6	1000011	
	1011011	
	1100001	
	1100111	
	1101101	
	1110011	

Tabelle 3:
Zahl der primitiven Polynome für Wortlängen n

n=	prim. Polynome
2	1
3	2
4	2
5	6
6	6
7	18
8	16
16	2048
24	276480
32	67108864

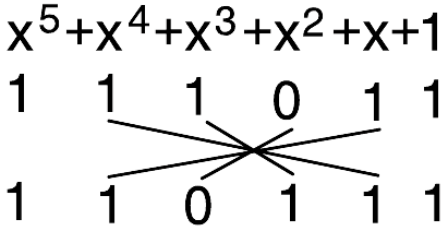


Bild 9: Reziprokes Polynom

Fehlerkorrektur

Für Gruppenbezeichnung und Datenformat der Codes wird in der Literatur die Notation „Gruppe (n,k)" verwendet. Der Gruppenname ist typisch der des Entdeckers, z.B. Hamming (7,4). Dabei ist 7 Bit die Länge der Gesamtnachricht, aber nur 4 Bit davon sind Nutzdaten. Der Code hat wenig praktische Bedeutung, ist aber das typische Einführungsbeispiel, weil man die Nachrichtenworte noch bequem ausdrucken kann.

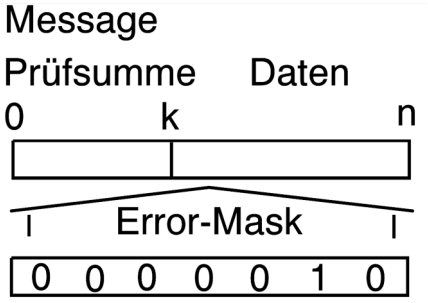


Bild 10: Datenformat und Einkopplung von Fehlern

Bei Schieberegisterschaltungen erfolgt die Datenübertragung seriell; erst kommen die Datenbits, dann die Prüfsummenbits (Bild 10). Das Auftreten von gekippten Bits im Kanal simmuliert man durch XOR mit einem Fehlerwort, das die Länge der Nachricht hat und bei dem die Fehlerbits gesetzt sind. Eine Schieberegisterschaltung ordnet einem Datenwort 00h auch eine Prüfsumme 00h zu. Verwendet man als Testdaten also 00h, kann man das Fehlerwort mit seinen einzelnen gesetzten Bits direkt als empfangene Nachricht ansehen. Aus Bequemlichkeit ist das eine übliche Vorgehensweise.

Coder

Zyklische Codes werden durch ihr Generatorpolynom definiert. Berechnung der Prüfsumme erfolgt durch Division des Datenworts d(x) durch das Generatorwort g(x), wobei der Rest r(x), als Prüfsumme an das Datenwort d(x) angehängt, übertragen wird. Als Beispiel wird hier der zyklische Hamming (7,4) verwendet. Sein Generatorwort g(x) ist 1011b, also 000Bh. Er korrigiert ein Fehlerbit. Die Generatorwörter entnimmt man Tabellen in der Literatur. Speziell [1] hat eine reichhaltige Sammlung.

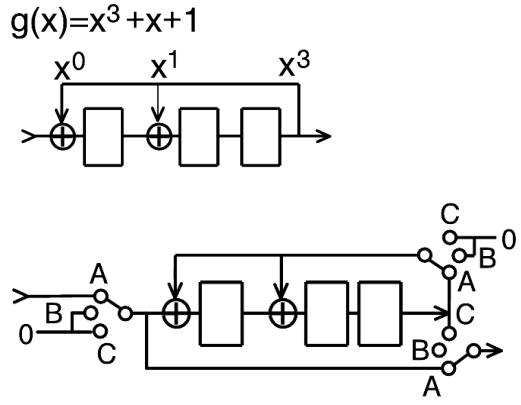


Bild 11: Simpler Coder

Die Divisionsschaltung I erfüllt die gewünschte Funktion, den Rest zu berechnen. Der Quotient, der dabei herausgeschoben wird, wird ignoriert. Für Implementierungen in Software ist sie (Bild 11) meist der geeignete Ausgangspunkt.

In Hardware sind etwa drei Codervarianten üblich, die aber nur verschiedene Möglichkeiten der Ablaufsteuerung sind. Sie sollen hier kurz dargestellt werden. Dabei wird das Schieberegister mit Clear ersteinmal gelöscht. Soweit mechanische Schalter abgebildet sind, kann man annehmen, daß die folgenden Eingänge pulldown-Widerstände haben.

Grundschaltung

In Bild 11 ist oben die Divisionsschaltung mit dem g(x) Polynom dargestellt. Darunter daraus der erste Coder. In Schalterstellung A wird in 4 Taktzyklen die Nachricht in Schieberegister und Kanal eingelesen. Dann sind in Schalterstellung B 4 Taktzyklen nötig, in denen 4 Nullen in das Schieberegister übertragen werden, während dieses die Division berechnet. In den letzten 3 Takten in Schalterstellung C wird dann der Rest, die Prüfsumme, in den Kanal gesendet. In dieser Schalterstellung sind die XORs gesperrt und es handelt sich nur noch um ein Schieberegister, kein Rechenwerk mehr. Die Schaltung funktioniert zwar, ist aber nicht sehr elegant.

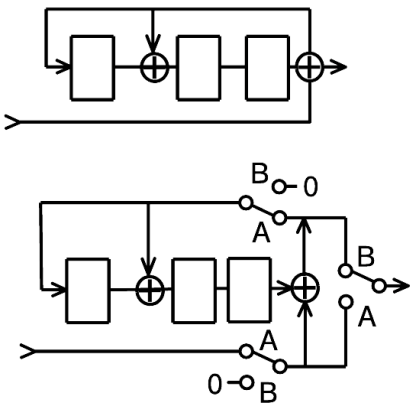
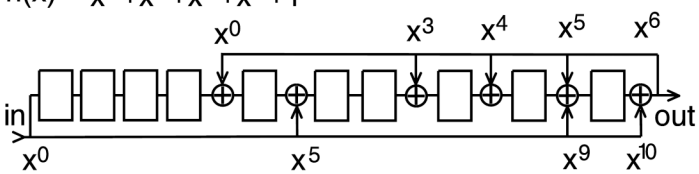


Bild 12: Coder mit Vormultiplikation

$$\frac{g(x)}{h(x)} = \frac{x^{10}+x^9+x^5+1}{x^6+x^5+x^4+x^3+1}$$



$$\frac{g(x)}{h(x)} = \frac{x^5+x+1}{x^6+x^5+x^4+x^3+1}$$

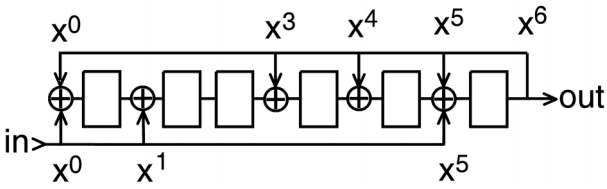


Bild 13: Beispiele für gleichzeitige Multiplikation und Division

$$h(x) = \frac{x^n+1}{g(x)} = \frac{x^7+1}{g(x)}$$

$$\frac{81}{B} = 17 = x^4 \quad +x^2 \quad +x^1 \quad +x^0$$

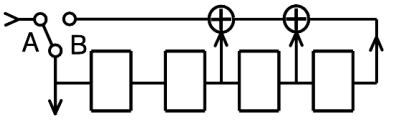


Bild 14: Coder 3. Variante

Mit Vormultiplikation

Mit der Vormultiplikation x^(n-k) wird die Schaltung etwas einfacher. Die dazu nötige Umordnung eines XORs zeigt Bild 12 oben. Darunter die funktionsfähige Beschaltung. Hier wird durch 4 Taktzyklen in Stellung A die Nachricht in den Kanal geschickt und gleichzeitig die Prüfsumme berechnet. In Schalterstellung B wird dann in 3 Takten die Prüfsumme angehängt.

Schaltungen zur gleichzeitigen Multiplikation und Division sind in Hardware sicher nützlich und finden sich häufig für gekürzte zyklische Codes („shortened cyclic codes“). In Software ist sowas weniger vorteilhaft. Bild 13 zeigt nochmal Beispiele zur Klärung des Aufbaus, sonst hier aber nicht weiter behandelt. Gekürzte Codes kann man in Software durch Wegwerfen und Einfügen von Datenbits einfacher verarbeiten.

3. Codervariante

Für diesen Coder (Bild 14) muß man aus dem Generatorpolynom g(x) das Prüfpolynom h(x) berechnen. Dazu die Division aus dem Abschnitt Arithmetik verwenden.

In Schalterstellung A überträgt man die Daten ins Schieberegister und in den Kanal. Dann legt man den Schalter auf Stellung B um, taktet den Inhalt des Schieberegister durch das XOR-Gate in den Kanal.

Decoder

Für 1 Bit korrigierende Hammingcodes eignet sich der Meggitt-Decoder. Er besteht im Wesentlichen aus 4 Baugruppen (Bild 15) und einem Schieberegister als Verzögerungsleitung, das so lang wie die Nachricht ist, gefolgt von einem XOR-Gate, das zum Zurückkippen des als fehlerhaft erkannten Datenbits dient. Einem Syndrom-Generator, der der ursprünglichen Divisionsschaltung des Coders entspricht. Und einem Logiknetz, das ein bestimmtes Syndrommuster dekodiert. Angenommen für Tests zur Bestimmung des Syndrommusters wird aus Gründen der Einfachheit die Übertragung des Datenwortes 0, Prüfsumme ist dann auch 0.

Wenn kein Fehler vorliegt, ist das Syndrom, der Rest, offensichtlich weiterhin 0 (Tabelle 4). Da man nur einen Fehler korrigieren kann, gibt es nur 7 „erlaubte“ Fehler, die alle unterschiedliche Syndrome erzeugen. Um abgestimmtes Verhal-

ten mit dem Verzögerungsregister zu erhalten, wäre es sinnvoll, den Fehler im obersten Bit des Datenworts zu erkennen. Beim Hamming (7,4) also 5 und dort funktioniert das auch. (Nicht so beim Hamming (15,11), dort funktionierte ein anderes Muster. Durch Probieren gefunden, Ursache unbekannt und noch nicht näher untersucht).

Keine Probleme mit dem Logiknetz gibt es beim Meggitt-Dekoder mit Vormultiplikation (Bild 16). Dort ist immer das oberste Bit 1 und alle anderen Bits 0. In den Zeichnungen ist mit den gestrichelten Linien eine Verfeinerung, die hier nicht verwendet wird, angedeutet, die „syndrome modification". Dabei korrigiert man nicht nur die Daten, sondern auch das Syndrom. Nützlich für Varianten zur Dekodierung von Mehrbitfehlern.

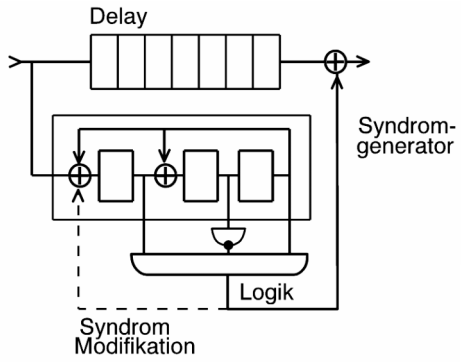


Bild 15:
Meggit-Decoder
Direktform

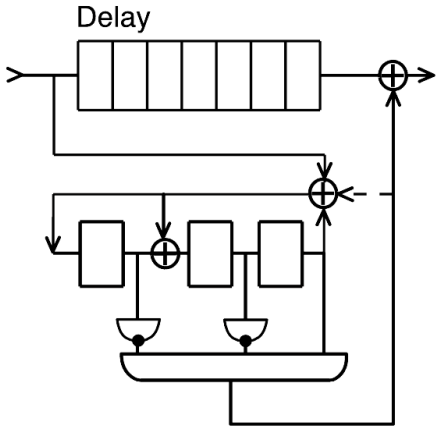


Bild 16:
Meggit-Decoder
mit
Vormultiplikation

Tabelle 4: Hamming (7,4)

Nachricht & Syndrom	
M=	R=
00	00 kein Fehler
01	01
02	02
04	04
08	03
10	06
20	07
40	05 <\<>- trigger

Division ?

Daß sowohl der Coder als auch der Decoder die Division verwenden, ist wenig intuitiv. Denn man sollte erwarten, daß der Coder eine Multiplikation mit dem Generatorpolynom durchführt. In diesem Zusammenhang ist aber zu berücksichtigen, daß das Datenwort hochgeschoben werden mußte, damit man es mit der Prüfsumme zu Nachricht vercoden konnte. Dieses Hochschieben entspricht einer Multiplikation mit der Konstante g^(n-k), die auszugleichen ist.

Wenn man den Coder in Software rein durch die Division (Bild 11) durchführt, muß man diese Multiplikation durch Shifts explizit ausführen, damit man ein paralleles Datenwort erhält.

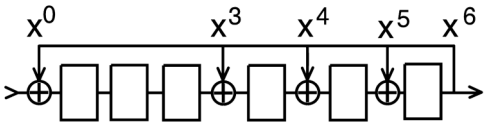


Bild 17:
Polynome (15,9)

Für weitere Experimente und eventuell praktische Anwendung empfielt sich z.B. der Hamming (15,11), der auch einen Fehler korrigiert, g(x) = 13h. Er entspricht bezüglich Coder und Decoder den gezeigten Verfahren.

CRC

Wem das alles irgendwie bekannt vorkommt: der Cyclic Redundancy Check funktioniert auch so, nur daß man sich dort damit begnügt nachzusehen ob der Divisionsrest Null ist und keine Korrektur vornimmt. Die Generatorpolynome würden sich dafür auch nicht eignen.

Burst Error Correcting Codes

Die bekanntesten Bündelfehlercodes dürften Fire-Codes sein, wie sie in den 70er Jahren in Harddisks und Magnetbändern verwendet wurden. Daneben findet man in [1] jedoch eine Liste von zum Teil recht kurzen Codes, die es erlauben, mehrere aufeinanderfolgende gekippte Bits zu korrigieren.

Der Hamming (15 ,9) (Bild 17) war als gekürzter Code (14,8) für das ursprüngliche Powerline Modem Protokoll von EHS vorgesehen. Er kann drei aufeinanderfolgende, zerstörte Bits korrigieren. Es ist also wohlgemerkt kein Code, der echte 3 Bits korrigiert, denn bei einem solchen könnten die Fehlerbits beliebig im Datenwort angeordnet sein.

Die Konstantendivision (Bild 17) bildet die Grundlage sowohl für Coder als auch für Decoder. Provisorisch implementierbar über den universellen Konstantendividierer kann man
einen Coder erstellen. Mit dem sind nun alle „legalen" Fehlervektoren durchzuprobieren (Tabelle 5). Man beachte erstens, daß die Fehlerbits am Ende der Tabellen in den Anfang zurückgefüttert werden. Zyklische Codes verdauen zyklische Fehler. Zweitens sind es nicht 7 Tabellen, sondern nur 4, weil manche Fehlermuster nur verschobene Varianten anderer Fehlermuster sind. Drittens wird das Muster vom MSB her eingespeist. Das liegt daran, daß das Datenwort (9 Bit) oben, die Prüfbits (6 Bit) unten liegen. Da man in der Praxis nur Fehlerbits im Datenwort korrigieren will, werden solche Fehlertabellen oft nur gekürzt auf die Datenbits erstellt, also nur 9 Zeilen.

Tabelle 5:
Fehlervektoren für
3 Bit Bündelfehler
und Syndrome dazu

E=	R=		
e = 100b		e = 100b	
0100000000000000	0000000000111100	0101000000000000	0000000000110011
0010000000000000	0000000000011110	0010100000000000	0000000000100101
0001000000000000	0000000000001111	0001010000000000	0000000000101110
0000100000000000	0000000000111011	0000101000000000	0000000000010111
0000010000000000	0000000000100001	0000010100000000	0000000000110111
0000001000000000	0000000000101100	0000001010000000	0000000000100111
0000000100000000	0000000000010110	0000000101000000	0000000000101111
0000000010000000	0000000000001011	0000000010100000	0000000000101011
0000000001000000	0000000000111001	0000000001010000	0000000000101001
0000000000100000	0000000000100000	0000000000101000	0000000000101000
0000000000010000	0000000000010000	0000000000010100	0000000000010100
0000000000001000	0000000000001000	0000000000001010	0000000000001010
0000000000000100	0000000000000100	0000000000000101	0000000000000101
0000000000000010	0000000000000010	0100000000000010	0000000000111110
0000000000000001	0000000000000001	0010000000000001	0000000000011111
e = 110b		e = 100b	
0110000000000000	0000000000100010	0111000000000000	0000000000101101
0011000000000000	0000000000010001	0011100000000000	0000000000101010
0001100000000000	0000000000110100	0001110000000000	0000000000010101
0000110000000000	0000000000011010	0000111000000000	0000000000110110
0000011000000000	0000000000001101	0000011100000000	0000000000011011
0000001100000000	0000000000111010	0000001110000000	0000000000110001
0000000110000000	0000000000011101	0000000111000000	0000000000100100
0000000011000000	0000000000110010	0000000011100000	0000000000010010
0000000001100000	0000000000011001	0000000001110000	0000000000001001
0000000000110000	0000000000110000	0000000000111000	0000000000111000
0000000000011000	0000000000011000	0000000000011100	0000000000011100
0000000000001100	0000000000001100	0000000000001110	0000000000001110
0000000000000110	0000000000000110	0000000000000111	0000000000000111
0000000000000011	0000000000000011	0100000000000011	0000000000111111
0100000000000001	0000000000111101	0110000000000001	0000000000100011

ROM

Von den 64 Syndromwerten, die man mit 6 Prüfbits darstellen kann, sind also 60 mit diesen Fehlern belegt. Es bietet sich an, mit dem Syndromgenerator eine Tabelle zu adressieren, die die entsprechenden 16 Bit Fehlervektoren enthält. Führt man mit dem Tabellenwert eine XOR-Verknüpfung mit dem empfangenen Datenwort durch, werden dessen Fehler korrigiert. Die 4 unbenutzen Syndrome, die ja keinen definierten Fehler darstellen, werden in der Tabelle mit dem Wert 0000h belegt, sodaß die XOR-Verknüpfung hier folgenlos bleibt. Mit ROM-Tabellen ist Dekodierung also sehr simpel, aber der Code darf nur wenige Prüfbits haben, weil sonst der Speicherverbrauch unakzeptabel wird.

Assembler

Nachdem der Ablauf nun prinzipiell in Hochsprache läuft, geht man an die Programmierung auf z.B. einem 8 Bit Controller.

Zum Verständnis der effizienten Programmierung eines Typ I LFSR, ist in Bild 19 ein einfacher Zufallszahlengenerator dargestellt. Bild 20 enthält das Flußdiagramm. In das 16 Bit Register SEED wird abhängig vom Zustand des obersten Bits durch einen ROL Befehl von unten ein Carrybit eingeschoben. Vorher erfolgt aber eventuell noch ein paralleles XOR mit einer Konstante MASK, hier D008h.

In der Fehlerkorrekturschaltung ordnet man die Taps in 3 Bytes nach oben verschoben an (Bild 20). Für den Coder kann man nun unten das Datenwort zu 15 Bits mit den Prüfbits gelöscht laden und in 15 Schritten hochschieben. Bei jedem Schritt erfolgt, wenn nötig, die Verknüpfung mit der XOR-Maske. Am Ende kann man das Syndrom als 6 Bits auslesen und in die leeren Prüfbits der Nachricht kopieren. Das gleiche Unterprogramm benutzt auch der Decoder. Unten wird hier das 15 Bit Datenwort eingelesen und in 15 Schritten hochgeschoben. Oben wird dann das Syndrom entnommen und damit die Tabelle adressiert.

Auf einem 2,45 MHz Mitsubishi-6502 kommen zum Code von etwa 140 Byte noch die Tabelle von 128 Byte. Der Encoder und Decoder benötigen jeweils 250usec.

Eine gekürzte Variante für (15,8) ist kaum schneller. Jedoch schrumpft die Tabelle auf 64 Byte.

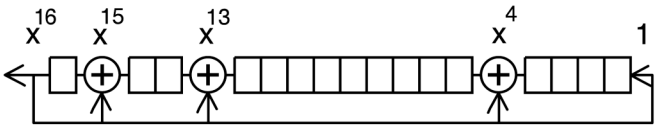


Bild 18: Schaltung Zufallszahlengenerator

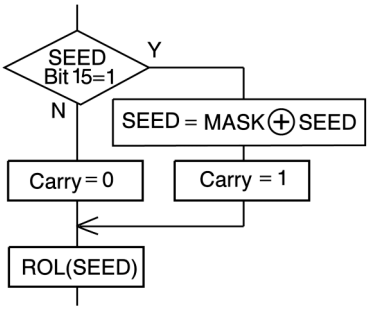


Bild 19: Flußdiagramm Zufallszahlengenerator

Error Trapping

Die Auswertung des Syndroms ist bis 8 Bit durch ROMs bequem möglich. In Software speziell in Form von Sprungzieltabelle. Bei längeren Syndromen ist der Speicherverbrauch dafür aber problematisch, andere Verfahren sind nötig.

Error Trapping ist eine Variante des Meggit Decoders, die 1961/62 von Kasami, Mitchell, Rudolph unabhängig voneinander gefunden wurde. Dieser Dekoder arbeitet sequentiell und kommt ohne Tabellen aus. Anwendbar auf Codes für 1 und 2 Bitfehler, praktische Obergrenze dürfte der Dekoder für den 3 Bit Golay-Code sein.

Tabelle 6: Ablauf ohne Syndrommodifikation

```
Nachricht 00 mit 2 Fehlerbits:     0010010000000000

Syndrom daraus nach 15 Schritten:  0000000001001000

weitere Syndromschritte:       -> 0 0000000010010000
                                  1 0000000011110001
                                  2 0000000000110011
                                  3 0000000001100110
                                  4 0000000011001100
                                  5 0000000001001001
                                  6 0000000010010010
                                  7 0000000011110101
                                  8 0000000000111011
                                  9 0000000001110110
                                 10 0000000011101100
                                 11 0000000000001001
                                 12 0000000000010010
                                 13 0000000000100100
                                 14 0000000001001000
                               -> 15 0000000010010000
                                 16 0000000011110001
```

Bild 20:
Passender Aufbau
für Syndromerzeugung
(15, 9)

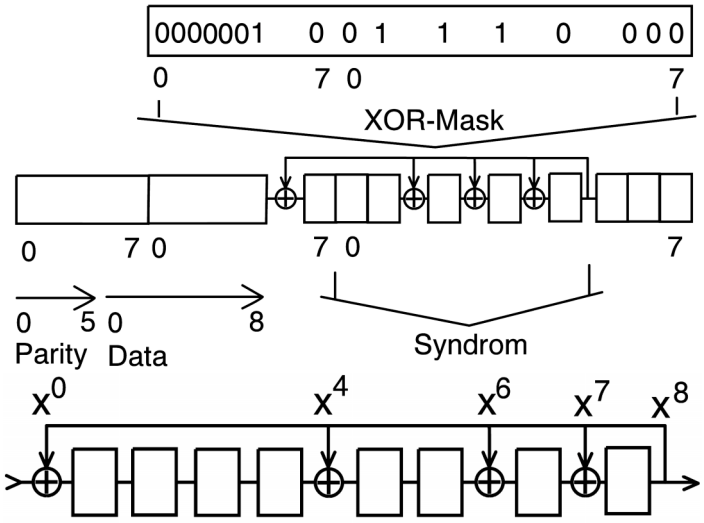


Tabelle 7: wie Tabelle 6 aber mit Syndrommodifikation

```
          0 0000000010010000 1
          1 0000000000100000 0
          2 0000000001000000 0
          3 0000000010000000 1
       -> 4 0000000000000000 0
```


Bild 21: Division mit Polynom (15, 7)

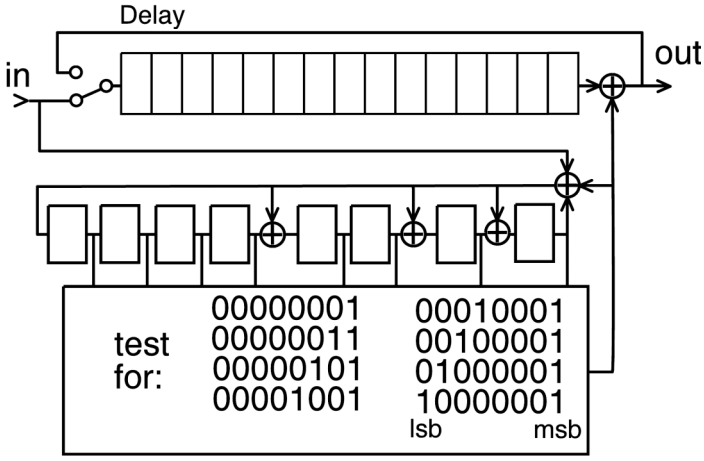


Bild 22: Error Trapping Decoder für (15, 7)

Besonders effizient, wenn die Fehler nahe beieinander liegen. Damit auch für Bündelfehler interessant.

Das übliche Beispiel ist der BCH (15,7). Zusammen mit dem Golay in den USA von Motorola in Pagern eingesetzt. Als Beispiel günstig, weil er der kürzeste Code ist, der 1 und 2 Bit Fehler korrigiert. Er würde sogar noch die Korrektur von 30% der 3 Bit Fehler ermöglichen, aber hier wird darauf verzichtet, weil die Komplexität dadurch ansteigt.

Der Coder beruht wieder konventionell auf Konstantendivision. Man kann damit wieder alle 1 und 2 Bit Fehlerkombinationen durchprobieren und ein Syndrom-ROM erzeugen. Man stellt aber fest, daß von dessen 256 Einträgen nur 120 tatsächlich belegt sind. Dekodierung über ROM ohne Berücksichtigung der 3 Bitfehler wäre also ineffizient. Prinzipiell hat man die Dekodierung aber wieder gelöst und hat Testvektoren für das effizientere Verfahren.

Für Error Trapping gibt es, genau wie bei den ursprünglichen Meggitt, Varianten mit und ohne Vormultiplikation für den Syndrom-Dekoder. Letztere Bauform ist schneller, wenn man viele Prüfbits und wenige Datenbits hat. D.h., abhängig vom Code haben beide Varianten praktische Bedeutung.

Hier sei die Version mit Vormultiplikation verwendet (Bild 22). Es erfolgt erstens eine Syndrommodifikation, zweitens können die Nachrichten in ihrem Schieberegister zirkulieren. Zuerst wird jedoch konventionell aus der empfangenen Nachricht in 15 Schritten das Syndrom berechnet. In Tabelle 6 sieht man die darauf folgende schrittweise Veränderung des Syndroms, ohne daß die Syndrommodifikation stattfindet. Da auch von der Datenleitung nur Nullen kommen, arbeitet die Schaltung wie ein LFSR-Generator. Die Zykluslänge ist 16 und würde sich endlos wiederholen (vgl. Pfeil).

Zyklische Codes

In Tabelle 7 ist die Auswirkung der Syndrommodifikation dargestellt, wobei das jeweils eingespeiste Korrekturbit rechts zu sehen ist. Das Ergebnis der beiden korrigierten Fehler ist somit, daß das Syndrom zu Null reduziert wird. In diesem Zustand sind alle Fehler korrigiert und der LFSR-Generator kann mit diesem Datenwort bekanntlich nicht mehr weiter laufen. Das Korrekturbit ist gesetzt, wenn das Syndrom einem der 8 Referenzmuster entspricht. Das festzustellen braucht man kein 256 Byte ROM, 8 Vergleichsbefehle würden genügen. Oder man testet, ob das oberste Bit gesetzt ist und läßt dann per Shifts ein Muster mit einem gesetzten Bit als Vergleichswert durchlaufen. Um zwei Fehler zu erledigen, sind im ungünstigsten Fall, wenn die Fehlerbits weit auseinanderliegen (Fehlermuster 0101h), mehr als 15 Korrekturschritte, hier 21, nötig. Das erschwert die Indexberechnung für die Korrektur des Datenworts, weshalb dafür eine Tabelle verwendet wird. In Software nimmt man natürlich ein explizites Schieberegister wie in Bild 19.

[1] Lin, Costello
„Error Control Coding“.
Prentice Hall 1983 (lieferbar)

[2] Blahut „Theory and Practice of Error Control Codes“
Addison Wesley 1983 (vergriffen)

Der Autor dieses Artikels hat der „Vierte Dimension“ freundlicherweise den Artikel sowohl als ASCII-Dateien zur Verfügung gestellt, als auch in einem PDF-File. Die Skizzen stehen in EPS zur Verfügung und als TIF Dateien. Alle Quellen können nach Rücksprache mit der Redaktion per EDV angefordert werden. *fep*

Fortsetzung von Seite 6

Der Erfolg war sehr gut und die Milan GmBH kündigte bald einen Nachfolger an, den Milan II mit der MC68060 und dreifache Performance. Ein Prototyp wurde gebaut und sogar im WDR-Computerclub vorgestellt. Die Produktion in grossen Stückzahlen und der Vertrieb in mehreren deutschen Kaufhäusern waren für Ende 1999 geplant.
Dann kam die grosse Entäuschung. Die Produktion des Logic-Chips, der die Atari-Bausteine emulierte, wurde eingestellt. Eine Produktion nur für den Milan wäre sehr teuer gewesen und somit der angepeilte Verkaufpreis von 1500 DM nicht mehr erreichbar.

Die Software

Das Betriebssystem TOS wurde ab 1995 mächtig weiter entwickelt und optisch perfektioniert. Die zwei TOS-Systeme (beide Multitasking), die heute mit Erfolg bei den Atari-Computer eingesetzt werden, heißen MAGIC und MINT/N_AES mit den Benutzeroberflächen JINNEE und THINGH. Beide sind sehr gut und schnell. Die Frage, welches besser ist, ist nicht zu entscheiden. MAGIC ist kompakt und einfach, während MINT/N_AES mehr in Richtung Linux-Komplexität tendiert.

Es gibt auch 4 Emulationen, die auf dem PC uneingeschränkt laufen: MACIC-PC, N_AES-PC, STEMULATOR und GEMULATOR. Die Emulation für den Apple gibt es auch: MAGIC-MAC.
Die Benutzeroberfläche - insbesondere JINNEE - ist wunderbar! Obwohl die 68000-CPU emuliert wird, ist die Arbeitsgeschwindigkeit sehr hoch.

Die Zukunft

Eine Gruppe von von Atari-Enthusiasten entwickelt zur Zeit ein Board mit einem modernen Prozessor, dem Coldfire MCF5407 von Motorola. Als Betriebsystem sind MAGIC und MINT/N_AES vorgesehen.

Internetauswahl: Software und Demo-Programme

```
http://www.user.uni-bremen.de/~peter
       Software
http://www.atari.org
       Allgemein
http://atos-magazin.de
       Internet-Zeitschrift
http://www.xtos.de
       Software
http://highwire.atari-users.net
       Browser Entwicklung
http://draconis.atari.org/german/index.htm
       Browser
http://www.student.informatik.
       tu-darmstadt.de
       Links
http://www.st-computer.net
       Allgemein
http://topp.atari-users.net
       Sotfware Entwicklung
http://www.calamus.net
       DTP Klasse!!
http://www.falke-verlag.de
       Zeitschriften
http://www.cs.uni-magdeburg.de/~fnaumann
       MINT Entwicklung
http://www.freemint.de
       Gnu/C/C++
http://www.woller.com
       MINT/N_AES (Demo)
http://www.application-systems.de
       MAGIC (Demo)
http://shop.falkemedia.de
       STEMULATOR (Demo)
http://www.atariuptodate.de
       Aktuelle Software
```

...und sie dreht sich doch, die Welt der Atarianer, immer noch, recht munter und voller Enthusiasmus. Die Welt der ‘‘Rechenboliden“ (erinnert sich noch jemanden an den Spaß, den wir mit dem Z 80 hatten ?), die Welt der bunten Fenster, der Betriebssystemkriege ... verstellt allzu oft den Blick auf fröhliche und forthige Alternativen. Vielen Dank für den Bericht, Herr Sala.

fep

Protokoll der Jahresversammlung 2002 der Forth Gesellschaft e.V.

Ort: Garmisch-Partenkirchen, Forsthaus-Graseck
Tag: 21. April 2002
Beginn: 09:07 Uhr
Ende: 12:30 Uhr

1. Begrüßung

Fred Behringer begrüßte die anwesenden Mitglieder zur Jahresversammlung der Forth Gesellschaft e.V. Martin Bitter, Drachenbesitzer im letzten Jahr, überreichte im Namen des Drachenrats den Drachen an Hans Eckes und lobte ihn in seiner poetischen Laudatio für seine Leistungen in unserer Gesellschaft.

2. Wahl des Schriftführers

Das Plenum schlug die Herren Ewald Rieger und Jens Wilke zum Schriftführer vor. Nach kurzer Diskussion benannten die anwesenden Mitglieder Ewald Rieger zum Schriftführer.

3. Wahl des Versammlungsleiters

Bernd Paysan wird per Akklamation mit der Leitung der Versammlung beauftragt.

4. Ergänzungen zur Tagesordnung

Zur Tagungsordnung lagen keine Ergänzungen vor.

5. Feststellung der Beschlußfähigkeit

Ute Woitzel berichtete, daß die FG zur Zeit 122 Mitglieder zählt. Die 19 anwesenden Mitglieder ergaben somit eine beschlußfähige Versammlung.

6. Bericht des Direktoriums

(Thomas Beierlein, Fred Behringer, Ulrich Hoffmann)

Thomas Beierlein lobte im Namen der Tagungsteilnehmer die hervorragende Vorbereitung und Ausrichtung der Tagung durch Fred Behringer, Ulrike und Heinz Schnitter.

Die Redaktion und das Layout der letzten fünf VDs übernahm vorübergehend Martin Bitter, der nun diese Tätigkeit wieder an Friederich Prinz zurück gibt. Thomas Beierlein würdigte Martin Bitter für die unermüdliche ehrenamtliche Arbeit, die VD in ihrer gewohnten Qualität erscheinen zulassen. Zum Dank überreichte ihm das Direktorium einen Blumenstrauß. In einer kurzen Stellungnahme berichtete Martin Bitter über seine Arbeit als Redakteur der VD. Dabei verwies er auf den akuten Mangel an neuen Artikeln und auf die Suche nach Korrekturlesern. Neben der großen Belastung freute er sich über neue Kontakte und die Tatsache, neue Informationen als erster zu erfahren.

Der Internetzugang der FG ist nun komplett von der Fa. Schlund und Partner zu Ulrich Hoffmann als Webmaster umgezogen. Das Verhältnis zur KBBS bei Holger Petersen ist inzwischen geklärt. Alle Aktivitäten sind gekündigt. Nutzer der KBBS müssen nun ihren Zugang dort selbst regeln. Die Struktur der Forth-eV-Webseite ist zufriedenstellend und müsse lediglich aktualisiert werden; z.B. ergänzt um ein Antragsformular zur Mitgliedschaft in der FG; um eine Domain für das Projekt über den FPGA-Forth-Prozessor b16 von Bernd Paysan.

Das Kapital der FG ist ohne Risiko angelegt und kurzfristig verfügbar.

Fred Behringer berichtete folgendes über die Kommunikation mit ausländischen Forthgruppen: Die Zusammenarbeit mit den Engländern und Holländern ist gut. In Amerika existiert nur noch eine kleine Gruppe im Silicon Valley, jedoch ohne Zeitschrift. Der Kontakt zu ihnen über Henry Vinerts und C. H. Ting ist gut. In Frankreich gibt es seit 1992 keine Forthgesellschaft mehr. Es besteht gelegentlicher E-Mail-Kontakt zu Marc Petremann. In Taiwan und in Rußland gibt es Forth-Freunde mit eigenen Homepages. Organisierte Vereine gibt es dort nicht. Wir stehen mit den Autoren in E-Mail-Kontakt. Aus Rußland haben wir einen Leserbrief in der VD veröffentlicht, aus Taiwan einen Bericht. In Australien gibt es eine kleine unorganisierte Gruppe, zu der wir keinen Kontakt haben. Österreich und die Schweiz (deutsch-sprachiger Raum) gehören zur FG e.V.

Mitgliederstand:.
Die FG verzeichnet einen weiter rückläufigen Mitgliederstand. Im Jahre 2000 zählte die FG 146 Mitglieder, 2001 waren es 134 Mitglieder und zur Zeit noch 122 Mitglieder.

7. Kassenbericht

Egmont Woitzel berichtete über den Kassenstand zum 31.12.2001

Einnahmen:

VD	DM	5569,95
Mitgliedsbeiträge	DM	5832,36
Jahrestagung 2001	DM	7037,00
Porto	DM	32,70
Spenden	DM	486,30
MWSt	DM	455,55
Summe Einnahmen	DM	19413,86

Ausgaben:

Jahrestagung 2001	DM	5601,32
Porto	DM	1979,50
Fracht	DM	216,00
Druckkosten	DM	1762,96
Versand VD	DM	906,00
Verwaltungskosten	DM	1050,96
Bankgebühren	DM	159,61
Internet	DM	480,24
Sonstiges	DM	68,72
Bürobedarf	DM	0,80
MWST	DM	464,18
Summe Ausgaben	DM	12690,09
Überschuß	DM	6723,77

Kontostände in	DM	Vorjahr
	(31.12.2000)	31.12.2001
Postgiro	34160,73	5884,50
Postsparbuch	373,85	373,85
Kapital Plus 90		25000,00
Kapital Plus 30		10000,00

Die Kasse prüfte Frau Elisabeth Rohrmayer. Sie bestätigte eine ordentliche und korrekte Führung der Kasse.

8. Entlastung des Direktoriums

Die Versammlung entlastete das Direktorium mit 18 Ja-Stimmen und einer Enthaltung.

9. Wahl des Direktoriums

Thomas Beierlein teilte der Versammlung mit, daß er nicht mehr für das Amt eines Direktors kandidieren möchte.
Die Direktoren Fred Behringer und Ulrich Hoffmann erklärten sich bereit, erneut zu kandidieren. Beide Direktoren wählte die Versammlung durch Akklamation mit 18 Ja-Stimmen und einer Stimmenthaltung.
Zur Wahl des dritten Direktors stellten sich Martin Bitter und Bernd Paysan zur Wahl. Klaus Schleisiek beantragte eine geheime Wahl. Im ersten Wahlgang stellten die Wahlleiter Heinz Schnitter und Klaus Schleisiek ein Patt mit 9 Stimmen für Martin Bitter und 9 Stimmen für Bernd Paysan, sowie einer Enthaltung und einer ungültigen Stimme fest. Im zweiten Wahlgang erreichte Bernd Paysan 11 Stimmen und Martin Bitter 9 Stimmen. Der neu gewählte Direktor Bernd Paysan nahm die Wahl an.

Mitgliedsbeiträge und Preisgestaltung der VD

Auf Grund des hohen Vermögens und der schwindenden Mitgliederzahlen erarbeitete die Versammlung den Vorschlag, die Preise wie folgt anzupassen:

Mitgliedsbeiträge: jeweils in Euro	bisher	neue
Fördernde Mitglieder	88	88
Mitglieder	40	32
Schüler/Studenten/Rentner	16	16
Preis der VD je Ausgabe	5,11	4

Die neue Preisgestaltung soll zum 1.1.2003 wirksam werden. Die derzeitige Preisauszeichnung der VD mit DM 10,00 soll ab sofort in Euro 5,11 geändert werden.
Diesem Antrag stimmten alle Mitglieder zu.

10. Verschiedenes

Im Laufe der Versammlung wurden nachfolgende Themen zu *Verschiedenes* gesammelt und priorisiert.

Themen	Priorität
VD als PDF	5
Antragsformular + Gimmics	4
Nächste Jahrestagung der FG	2
Forthbüro	1
Microcontrollerverleih	6
Wiki-Server	5
CD-Projekt	5
Webauftritt	5
Geld	3

Thema: Forthbüro
Ute Woitzel möchte das Forth-Büro zum Ende 2002 abgeben. Sie umriß kurz die anfallenden Arbeiten im Forth-Büro und deren Honorierung. Potentielle Interessenten aus dem Plenum konnten so einen Eindruck über den Umfang der Arbeit gewinnen. Leider erklärte sich niemand bereit, das Forth-Büro zu übernehmen. Die Versammlung beauftrage deshalb das Direktorium, rechtzeitig eine geeignete Lösung zu finden.

Thema: Nächste Jahrestagung der FG
Zur Ausrichtung der nächsten Jahrestagung meldeten sich gleich zwei Kandidaten. Ewald Rieger möchte die Tagung in Bad Dürkheim an der Weinstraße und Ulrich Hoffmann auf der Insel Fehmarn ausrichten. Beide Kandidaten und die versammelten Mitglieder einigten sich darauf, die Tagung 2003 durch Ewald Rieger in Bad Dürkheim auszurichten. Ulrich Hofmann wurde für die übernächste Tagung in 2004 vorgemerkt.

Thema: Geld
Das Kapital pro Mitglied der FG stieg in den letzten Jahren stetig an. Erreicht durch sparsames Wirtschaften und sinkende Mitgliederzahlen. Um neue Mitglieder zu werben, wurde durch Klaus Schleisiek angeregt, dem Direktorium ein Budget für Forth-Projekte zur Verfügung zu stellen. Über die Höhe wurde kontrovers diskutiert. Folgende Beträge kamen zur Abstimmung:

Kein Budget	keine Zustimmung
2048 Euro für µC-Projekte	keine Zustimmung
3072 Euro	Zustimmung mit einer Enthaltung

Thomas Beierlein beantragte Gelder für Forth-Projekte mit dem Ziel, Forth-Veröffentlichungen zu erreichen. Ein Limit des Budgets wurde nicht genannt.
Dieser Antrag wurde einstimmig angenommen.

Thema: CD-Projekt:
Neue Mitglieder sollen als Geschenk eine CD mit dem Inhalt früherer Jahrgänge der VD (VD-CD) oder/und eine Forth-CD erhalten. Das Stimmungsbild der versammelten Mitglieder ergab ein Votum zur Konzentration auf die Erstellung der VD-CD. Um die Arbeit auf mehrere Schultern zu verteilen, werden zum Scannen von alten VDs noch Freiwillige gesucht (Koordination: U.Hoffmann)

Thema: Webauftritt
Ulrich Hoffmann sucht zur Pflege des Wiki-Servers einen Moderator. Ein Antragsformular zur Mitgliedschaft soll online als PDF-Datei verfügbar sein. Die Satzung der FG soll online abrufbar sein.

gez. ***Ewald Rieger***
Protokollführer

Titelliste

Diese Liste wurde von Fred Behringer zusammengestellt. Sie enthält alle in unserer Zeitschrift "Vierte Dimension" jemals veröffentlichten Artikel - oder sollte sie jedenfalls enthalten. Fehlmeldungen bitte unter der E-Mail-Adresse behringe@mathematik.tu-munechen.de. Die Übertragung nach PDF für die elektronische Version besorgte Rolf Schöne. Die Sachgruppen sind alphabetisch geordnet, die Titel innerhalb einer Sachgruppe nach dem Erscheinungsheft, innerhalb eines Heftes nach der Plazierung im Heft. Manche Artikel sind mehrfach, unter verschiedenen Sachgruppen aufgeführt. Editorials, Direktorials, Meldungen und Leserbriefe sind nur in Auswahl vertreten. Wir haben vor, diese Liste alle Jahre wieder zu aktualisieren.

Parallel	Montvelishsky	95-4 Parallel-Forth: Der neue Weg
Patchen	Klingelnberg, A.	93-1 KEYB8B
Patchen	Woitzel, Egmont Schütz, Udo	94-3 Der totale Patch
Patchen	Behringer, Fred	98-4 Patchen ohne den Beigeschmack des Halbfertigen
Patchen	Behringer, Fred	99-4 BEGIN-UNTIL über 32K ab 80386 im ZF-Assembler
Philosophie	Moore, Chuck	87-3 FORTH - eine persönliche Sprache
Philosophie	Haydon, G. B.	89-1 WISC und das FORTH-Dilemma
Philosophie	Wejgaard, Wolf	89-3 Warum Postfix?
Philosophie	Holzapfel, Martin	89-4 Das "Greenhorn-Modul" (LOGO)
Philosophie	Goppold, A.	92-3 Forth und der Rest der Welt
Philosophie	Goppold, A.	93-1 Forth und der Rest der Welt
Philosophie	Goppold, A.	93-3 Object Code contra Metacode
Philosophie	Diaczyszyn, Z. Forster, H.-G.	93-3 BASIC oder Forth?
Philosophie	Canton, D. K. Rottenkolber, W.J. Rather, E.D.	93-4 Forth ohne Arbeit? (ein Schlagabtausch)
Philosophie	Symonds, M.	94-3 Forth als Metasprache
Philosophie	Knaggs, Peter	96-1 Forth auf eine professionelle Basis heben
Philosophie	Behringer, Fred	96-1 Forth braucht keinen qualitätsüberwachenden Fachverband
Philosophie	Beierlein, T.	96-2 Forth und der Anfänger
Philosophie	Kalus, Michael	96-3 Ein heiliger Krieg - das THEN im Forth
Philosophie	Freitag, Robert	97-2 Warum nicht Forth, sondern C?
Philosophie	Deliano, Rafael	97-3 PostScript
Philosophie	Vogt, Claus	98-3 Typüberprüfung in Forth
Philosophie	Ting, C.H.	00-1 Das Dao Forth
Philosophie	Aguilar, Hugh Jakeman, Chris Ouwerkerk, Willem Prinz, Friederich Bitter, Martin Behringer, Fred	00-1 Spaß mit Forth
Philosophie	Staben, Jörg	01-1 Bibliotheks- und Sicherheitskonzepte in Forth und anderswo
Philosophie	Staben, Jörg	01-1 Ein Lehr-Stück?
Philosophie	Staben, Jörg	01-1 Fünf Jahre später - eine Standortbestimmung
Philosophie	Staben, Jörg	01-2 Bibliothekskonzepte, JavaBeans

VD Titelliste (Teil III)

Philosophie	Behringer, Fred	01-2 Stimmen der anderen: Ist Forth keine Sprache?
Philosophie	Klimas, Andreas	01-4 Zur Softwareentwicklung
Philosophie	Tiedemann, S.	02-1 Stack-FORTH
Philosophie	Klimas, Andreas	02-1 Warum wird die Bedeutung von OO überschätzt?
Portierung	Deliano, Rafael	97-1 Nucleus für Controller
Portierung	Deliano, Rafael	97-2 Nucleus für Controller: Aithmetik
Postfix	Wejgaard, Wolf	89-3 Warum Postfix?
Postfix	Staben, Jörg	89-4 Leserbrief zum Thema Postfix
Postfix	Paysan, Bernd	91-4 Infix nach Postfix
Postfix	Klingelnberg, A.	94-3 Vorher <--> Nachher (Präfix - Postfix)
Praxis	Neuthe, Ralf	92-3 Automation eines Sägewerks mit Forth
Praxis	Allinger, W.	93-1 Forth spinnt (Spinnmaschine)
Probleme	Staben, Jörg	91-4 Keine zweite Shell unter 4DOS/volks4TH
Projekte	Brühl, Dirk	89-1 FORTH-Workstation
Projekte	Staben, Jörg	92-2 Projektförderung durch die FG
Projekte	Prinz, Friederich	98-2 Projekt: Ein "gutes" Forth
Projekte	Körber, Hellmut	98-4 Leserbrief zu "Ein gutes Forth(-Buch)"
Projekte	Tiedemann, S.	98-4 Leserbrief "Ein Forth von und für die FG"
Projekte	Ouwerkerk, W.	99-2 Das Igel-Arbeitsbuch (AT89Cx51)
Projekte	Jakeman, Chris	01-3 WebForth
Protokoll	Rieger, Ewald	00-3 Mitgliederversammlung 2000
Prozessor	Betancourt, Jose	88-1 Name That Architecture ...
Prozessor	Danile, Peter S. Malinowski, Ch.W.	88-1 FORTH Processor Core for Integrated 16-Bit Systems
Prozessor	Schleisiek, Klaus	88-1 Die Reparatur des Interrupts im NC 4000
Prozessor	anonym	88-2 Der RTX 2000
Prozessor	Vack, Gert-Ulrich	88-2 Hardware-Realisierung von FORTH
Prozessor	Stelzner, Peter	88-2 Modulares Mikrokontroller-Konzept mit LCD-Treiber von EUROSIL
Prozessor	Göttle, G. Kobenter, R.W.	88-4 MARC4 - ein applikationsorientierter Mikrocontroller für Forth
Prozessor	Paul, Ulrich	89-1 Das Mini-BEE-2000-System
Prozessor	Paul, Ulrich	89-2 Die Pals der RTX-2000-Mini-BEE
Prozessor	Willers, H.-G.	90-3 RISC-FORTH
Prozessor	Schnitter, Heinz Willers, Hans-Günther	91-3 Platine für den Zilog-Super8-FORTH-CHIP
Prozessor	Hoffmann, Ulrich	92-1 Der HP48SX - Ein Taschenrechner, Forth-ähnlich programmierbar
Prozessor	Kühnel, Claus	92-1 Fehlererkennung mit dem Cyclic Redundancy Code (CRC)
Prozessor	Klingelnberg, A.	92-3 FORTH-RISC-Prozessor FRP 1600
Prozessor	Michaels, Jan Dyja, Holger	93-2 68HC11
Prozessor	Dyja, Holger	93-4 68HC11 - noch mehr Forth - 2. Teil
Prozessor	Schemmert, W.	94-2 Der Dallas DS80C320
Prozessor	Deliano, Rafael	95-1 Forth im Orbit
Prozessor	Deliano, Rafael	95-2 Hochsprachen in Silizium
Prozessor	Deliano, Rafael	95-2 P21 (Chuck Moore)
Prozessor	Deliano, Rafael	95-3 FORTH-Computer ...
Prozessor	Deliano, Rafael	95-3 MARC4 (Arbeitsgruppe)
Prozessor	Deliano, Rafael	95-4 IX1 (Feldbusprozessor von DELTAt)
Prozessor	Deliano, Rafael	96-2 C auf Stackprozessoren
Prozessor	Deliano, Rafael	96-4 Stackprozessor als FPGA
Prozessor	Deliano, Rafael	97-1 MISC
Prozessor	Beierlein, T.	97-1 DN-1620 - ein Forth-Prozessor aus Weißrußland
Prozessor	Wilke, Jens	99-4 Patriot Scientific PSC1000 (Java, Forth und C)
Prozessor	Eckes, Hans	00-3 MiniModul 2M, eine Prozessorkarte incl. Forth
Prozessor	Schemmert, W.	00-4 "Avise" - AVR Virtual Stack Engine
Prozessor	Pöppe, Christoph	01-2 Rechnen mit garantierter Genauigkeit (XPA3233)
Prozessor	Pöppe, Christoph	01-3 Rechnen mit garantierter Genauigkeit (korrigierte Version)
Prozessor	Eckes, Hans	01-4 Die Forthbriefmarke - der Prototyp

Prozessor	Tiedemann, S.	02-1 MuP21/F21 - Bootprozeß
Rätsel	Klingelnberg, A.	94-1 Variablentausch ohne Hilfsvariable
Rätsel	Bitter, Martin	96-1 Rätsel (Tröpfelalgorithmus)
Rätsel	Bitter, Martin	96-1 Steter Tropfen hö(h)lt den Stein (Lösung)
Rätsel	Beierlein, Georg	99-3 Schlampiger Elektriker
Rätsel	Pohl, Joerg	99-4 Lösung zu "Schlampiger Elektriker"
Rätsel	Quick, Guido	99-4 Lösung zu "Schlampiger Elektriker"
Rätsel	Vinerts, Henry	99-4 Lösung zu "Schlampiger Elektriker"
Rätsel	Behringer, Fred	99-4 Rätsel - Versuch einer Lösung (Elektriker)
Rätsel	Behringer, Fred	00-2 Der verschlüsselte Schlüssel oder X S 2XOR
Rätsel	Behringer, Fred	00-2 Die Zahlen von 1 bis 50
Rätsel	Behringer, Fred	00-4 Des Rätsels Lösung (verschlüsselter Schlüssel)
Rätsel	Behringer, Fred	00-4 Des Rätsels Lösung (Viererproblem)
Rätsel	Behringer, Fred	01-1 Rätsel: Zahlendarstellung
Rätsel	Behringer, Fred	01-2 Metarätsel - Spielereien mit dem Allbegriff
Rätsel	Behringer, Fred	01-3 Des Rätsels Lösung: Zahlendarstellung
Rätsel	Behringer, Fred	01-4 Des Rätsels Lösung: Metarätsel aus Heft 2/2000
Rätsel	Bitter, Martin	02-1 Preisausschreibung (Codeverschiebung)
Regler	Deliano, Rafael	94-1 PID-Regler im Überblick
Regler	Führer, W.	96-4 'N bißchen wat Praxis
Roboter	Prinz, Friederich	93-1 Fis(c)hing Forth (Fischer-Technik)
Roboter	Deliano, Rafael	95-3 ICR Intermediate Code for Robots
Roboter	Behringer, Fred	01-1 RCX: Wie kommt Forth in den Roboter? - Erste Schritte
Roboter	Behringer, Fred	01-1 ASM2COM über Turbo-Forth: Warum meldet sich der RCX nicht?
Roboter	Bitter, Martin	01-1 Auf den vierten Platz geschlängelt
Roboter	Behringer, Fred	01-3 Lego-Roboter und arithmetisierte Logik in Forth
Roboter	Bitter, Martin Behringer, Fred	01-3 Den Lego-Roboter überlisten - mit und ohne Forth
Roboter	Kalus, Michael Krüger, Adolf	01-4 Tower forever - die zweite
Roboter	Kalus, Michael Krüger, Adolf	01-4 Zur Lego-Mindstorm-Infrarot-Datenübertragung
Roboter	Bitter, Martin	01-4 Assembler "brute force" für den Lego-RCX
Roboter	Kalus, Michael	02-1 RCX mit höherer Frequenz
Roboter	Schöne, Rolf	02-1 Tower forever - unter DOS
Sicherheit	Staben, Jörg Plewe, Jörg Hoffmann, Ulrich	93-4 Let's Gimpel ("Range Check")
Sortieren	Hohl, Heinrich	91-1 Fünf Beispiele für Sortieren in FORTH
Sortieren	Prinz, Friederich	98-3 Programmierwettbewerb: Äpfel, Birnen, Sortieren
Sound	Bitter, Martin	98-2 His Masters Voice: Lean is beautiful ?
Spaß	Symonds, M.	94-3 Die Neuentwicklung des Bits
Spaß	Vogt, Claus	96-1 Das müssen wir analysieren, ...
Spaß	Vogt, Claus	96-4 Freitexttabellenkalkulation
Spaß	Prinz, Friederich	97-3 Ein Super-DAU (dümmster anzunehmender User)
Spaß	Beetz, Jürgen	97-3 Eine kleine Geschichte
Spaß	Kohl, Klaus	98-1 Bill Gates in Irland
Spaß	Prinz, Birgit	98-1 Der sprechende Computer HAL
Spaß	Vogt, Claus	98-3 GM und Bill Gates: Autos und Computer
Spaß	Aguilar, Hugh Jakeman, Chris Ouwerkerk, Willem Prinz, Friederich Bitter, Martin Behringer, Fred	00-1 Spaß mit Forth
Spaß	Tiedemann, S.	00-1 Wie fange ich einen Elefanten?
Spaß	Behringer, Fred	00-1 Forth verändert die Welt
Spaß	Die Redaktion	00-4 Das Netzteil
Spaß	Bitter, Martin	01-1 Brückenbauers Traum

Spaß	Stejskal, B.-M.	01-4 Die Forth-Schatzinsel
Spaß	Bitter, Martin	01-4 Katzenjammer
Spiele	Döring, Andreas	90-1 Mensch-ärgere-dich-nicht in volksFORTH
Spiele	Plewe, Jörg	93-4 Krieg der Kerne (F-PC, F68K)
Spiele	Richter, Ulrich	98-1 Win32Forth: Programm mit mehrsprachiger Oberfläche (Schiebespiel)
Spiele	Richter, Ulrich	98-2 Win32Forth: Mein erstes Programm (Schiebespiel)
Spiele	Pilot, M.	99-4 Computerspiel zur Wirtschaftspolitik ???
Spiele	Tiedemann, S.	00-3 The Way of Stones (Runa Furtak oder Ishido)
Spiele	Prinz, Friederich	02-1 Pontifex (Brückenbau)
Stack	Hoekman, Doneil	86-1 Ein universelles Stackwort
Stack	Kalus, Michael	87-2 Stackakrobatik - Multiplikation mit 2
Stack	Kretzschmar, R.	88-2 Swopperatoren
Stack	Findewirth, A.	89-1 SWOPILER, ein Generator für Stackoperator-Worte
Stack	Stüss, Frank	89-2 Noch einmal: STACK
Stack	Findewirth, A.	89-2 Schneller Swopiler
Stack	Klingelnberg, A.	91-4 Die Stacks, TIB, PAD und HERE - Ein Einstieg nicht nur in F-PC
Stack	Behringer, Fred	99-2 Stack-Auslagerung in eine Datei
Stack	Tiedemann, S.	02-1 Stack-FORTH
Standard	Haydon, G. B.	87-1 A Forth Standard?
Standard	Ouverson, Marlin	87-2 State of the Standard
Standard	Kalus, Michael	91-2 BASIS15 - ANS-Forth-Entwurf 4/91
Standard	Kalus, Michael	91-2 Neues im ANS-Forth Core Word Set
Standard	Kalus, Michael	91-3 Vergleich ANS-Forth Core Word Set und Forth-83 Required Word Set
Standard	Bradley, Mitch	91-4 dpANS
Standard	Paysan, Bernd	93-1 ANS-Forth - der letzte Stand
Strings	Schleisiek, Klaus	86-1 Stringstack
Strings	Konrad, Karsten	89-3 Read-only-Stringfelder in FORTH
Strings	Scheller, Konrad	89-4 Neuer Stringstack mit Heap
Strings	Hohl, Heinrich	91-2 Der dynamische Stringbuffer
Strings	Vogt, Claus	93-3 Vom Suchen, Finden und Wegwerfen (F-PC)
Strings	Eilers, Nils	99-2 Strings in Forth (F83 und ANS)
Strings	Baden, Wil	99-2 Zeilen und Strings
Systeme	Bruziks, George	86-3 Public Domain Forths für den IBM-PC
Systeme	Blum, Holger	86-3 Mach1, ein Forthsystem für den Macintosh
Systeme	Mertins, Rainer	87-1 Public Domain Forth F83
Systeme	Petremann, Marc	88-1 Turbo-Forth made in France
Systeme	Kempf, Stefan	89-3 MULTI-FORTH, der Einstieg
Systeme	Plewe, Jörg	90-1 Schnelles FORTH für den MC68000
Systeme	Epprecht, Robert	90-2 Patch fürs volksFORTH (auf ATARI ST)
Systeme	Plewe, Jörg	90-1 Schnelles FORTH für den MC68000 - Teil II
Systeme	Strobel, Max	90-1 FORTH-Implementation auf Großcomputer in zwei Wochen
Systeme	Klingelnberg, A.	90-3 Zimmer-FORTH v. 3.50a.k.
Systeme	Plewe, Jörg	90-4 Ein schnelles Chamäleon (F68K)
Systeme	Hoffmann, Ulrich	91-1 Ein Weg, wie man FORTH klein macht
Systeme	Prinz, Friederich	91-2 FORTH ganz praktisch
Systeme	Bradley, Mitch	92-1 Open-Boot-Firmware
Systeme	Grehan, Rick	92-1 Yerk kommt zum PC
Systeme	Plewe, Jörg	92-3 F68K - das erste Jahr
Systeme	Plewe, Jörg	92-3 F68K-Neuigkeiten in letzter Minute
Systeme	Schemmert, W.	92-4 eForth => 68HC11
Systeme	Golf, Burkhard Schönlau, Rolf Pleßmann, K.W.	93-1 Forth++ (1)
Systeme	Major, Michael Prinz, Friederich	93-2 FORTH/2 - ein 32-Bit-System für OS/2 2.x
Systeme	Diaczyszyn, Z.	93-3 F68K konnte es, F68KANS besser
Systeme	Prinz, Friederich	93-3 Das legendäre ZF
Systeme	Klingelnberg, A.	94-1 Moderne Betriebssysteme
Systeme	Kohl, Klaus	94-2 Der neue volksFORTH-Vertrieb
Systeme	FS F-SYSTEME	94-3 LMI WINFORTH

Systeme	Schütz, Udo	94-3 Programmierung mit comFORTH für Windows
Systeme	Goddard, Roy	94-3 MPE ProForth für Windows
Systeme	Saric, Rainer	95-1 PowerGEM für Forthmacs
Systeme	Schütz, Udo	95-1 Windows unter Windows (comFORTH)
Systeme	Zembrod, Philip	95-2 ultraFORTH83 rev.3.83
Systeme	Staben, Jörg	95-4 Das Werkzeug Win32Forth
Systeme	Führer, W.	96-1 Portables Forth: Atari-Portfolio
Systeme	Maier-Schuler, P.	96-1 cpForth für OS/2
Systeme	Deliano, Rafael	96-1 Jeff Raskins "Canon Cat"
Systeme	Raisig, Gerhard	96-2 Eine SPS-Simulation in PC-volksFORTH
Systeme	Prinz, Friederich	96-2 HOLON - Das ganz andere FORTH
Systeme	Köller, Malte	96-3 Ein Rahmen für modular aufgebaute Forth-Systeme
Systeme	Paysan, Bernd	97-1 Gforth auf MISC portiert
Systeme	Prinz, Friederich	97-2 HOLON (Forth) lebt
Systeme	Wejgaard, Wolf	97-3 Die Entwicklung von HolonForth - Teil 1
Systeme	Bitter, Martin	98-2 Platz da ?! - "Nachtrag" zum ZF
Systeme	Freitag, Robert	98-2 Ein neues DOS ?? (Open DOS)
Systeme	Wejgaard, Wolf	98-2 Die Entw. von HolonForth, Teil 2: Wie man Forth echt interaktiv macht
Systeme	Behringer, Fred	98-2 Real-Mode-32-Bit-Erweiterung für Turbo-Forth
Systeme	Bitter, Martin	98-2 Fred Behringers Real-Mode-32-Bit-Forth für ZF
Systeme	Vogt, Claus	98-3 Real-Mode-32-Bit-Erweiterung
Systeme	Behringer, Fred	99-1 So kriegt man Forth auch klein (Turbo-Forth)
Systeme	Tiedemann, S.	99-1 MISC - Minimal Instruction Set Computer
Systeme	Jakeman, Chris	01-3 WebForth
Systeme	Allinger, W.	01-4 QUARTUS Forth - erste Erfahrungen
Tagung	Shifrin, Jerry	87-3 August 1987 ANS Forth Meeting Notes
Tagung	Kalus, Michael	89-2 Bericht von der FORTH-Tagung 1989 in Aachen
Tagung	Krinninger, Ch.	89-4 euroFORML 1989
Tagung	Staben, Jörg Plewe, Jörg	90-2 Bericht von der FORTH-Tagung 1990
Tagung	Klingelnberg, A.	90-3 Echtzeit '90
Tagung	Kalus, Michael	91-2 FORTH '91 - Jahrestagung in München
Tagung	Hoffmann, Ulrich	91-3 Echtzeit '91 - FORTH-Teams an der Spitze
Tagung	Hoffmann, Ulrich	91-4 euroFORML '91
Tagung	Kalus, Michael	92-2 Tagung '92 in Rostock
Tagung	Schleisiek, K.-P.	92-2 Programmierwettbewerb '92
Tagung	Klingelnberg, A.	92-2 Echtzeit '92
Tagung	Wejgaard, Wolf	92-4 euroFORTH '92
Tagung	Plewe, Jörg	94-3 Das war der Programmierwettbewerb auf der Echtzeit '94
Tagung	Plewe, Jörg	95-2 Das war die Forth-Tagung '95
Tagung	Vogt, Claus	95-3 Abstract der euroForth '94
Tagung	Beierlein, T.	95-4 euroFORTH '95
Tagung	Plewe, Jörg	96-2 Jahrestagung der Forthgesellschaft 1996
Tagung	Steffenhagen, B. Hoffmann, Ulrich	96-3 Echtzeit 96
Tagung	Deliano, Rafael	97-1 Embedded Systems '97
Tagung	Altvater, Walter	97-2 Jahrestagung der Forthgesellschaft 1997
Tagung	Dahm, Markus	97-3 12th euroForth conference
Tagung	Paysan, Bernd	98-1 euroForth '97
Tagung	Paysan, Bernd	98-4 Forth-Tagung '98
Tagung	Paysan, Bernd	99-3 Forth-Tagung '99 in Oberammergau
Tagung	Prinz, Friederich	00-3 Forth-Tagung 2000 in Hamburg
Tagung	Paysan, Bernd	01-3 Forth-Tagung 2001 in Hamminkeln-Dingden
Tagung	Prinz, Friederich	01-3 Forth-Tagung 2001 in Hamminkeln-Dingden
Tagung	Hoffmann, Ulrich	02-1 euroFORTH 2001
Tastatur	Hoffmann, Ulrich	91-1 F-PC spricht Deutsch
Tastatur	Vogt, Claus	92-3 Internationaler Zeichensatz im F-PC
Tastatur	Klingelnberg, A.	93-1 KEYB8B
Tastatur	Groß, Heiko	97-2 Digitales Potentiometer (Tastaturersatz)
Terminal	Pennemann, B.	86-2 Ein Terminalprogramm in Forth
Terminal	Vogt, Claus	90-2 Datenübertragung von Commodore-Plus4 auf den PC
Terminal	Deliano, Rafael	96-3 XMODEM (nanoFORTH)

Test	Schultheis, M.	89-4 Testbericht: Ein Weg zum professionellen M680X0-FORTH
Test	Staben, Jörg	89-4 Testbericht: DRUMA-FORTH - ein interessantes FORTH-System?
Test	Vogt, Claus	89-4 ASYST oder ein FORTH-Programm muß nicht immer klein sein
Test	Schultheis, M.	90-1 Testbericht: FFORTH aus heimischen Landen
Test	Staben, Jörg	90-1 Der Mensch als Jäger und Sammler (Test POLYMATH und FIFTH)
Test	Schultheis, M.	90-3 Testbericht: bigFORTH
Test	Deliano, Raphael	93-3 Benchmark für 8-Bit Microcontroller
Test	Deliano, Raphael	93-3 Erbsenzähler (Anwendungshäufigkeit von Forth)
Test	Tiedemann, S.	99-2 Benchmark-Tests mit dem F21-Quicksort
Text	Saupe, Peter	89-2 Ein einfacher Textinterpreter, der Rechnen und auch Formeln kann
Text	Plewe, Jörg Staben, Jörg	91-1 Universelles Filterwerkzeug
Text	Haase, H.-J.	94-2 WordWriter 94 (F-PC)
Text	Vogt, Claus	96-4 Freitexttabellenkalkulation
Timer	Beierlein, T.	92-4 Was tickt denn da ...
Timer	Klingelnberg, A.	94-1 Pudding mit Sahne (FUDGE, MS)
Timer	Paysan, Bernd	94-2 Zeitmessung auf dem PC
Timer	Beuster, Bernd	95-2 Multitaskfähige Timerfunktionen (PowerForth)
Timer	Kohl, Klaus	95-3 Programmierung des Timer0-Interrupts
Tips	Allinger, W.	93-4 Tips und Tricks
Tips	Prinz, Friederich	98-1 "Kerniges" zum Win32For
Tools	Pennemann, B.	86-4 Alert Boxen
Tools	Bradley, Mitch	86-4 Self-Understanding Programs
Tools	Hoffmann, Ulrich	87-1 Menüs in FORTH
Tools	Pennemann, B.	87-1 Die "Haeh?"-Falle
Tools	Pennemann, B.	87-1 Die Fileselector-Box unter volksFORTH83 auf dem Atari ST
Tools	Wenrich, Carl A.	87-1 Screenless Forth
Tools	Pennemann, B.	88-1 Accessories und das volksFORTH83
Tools	Redeker, Markus	88-1 Screens verschieben mit AROUND
Tools	Jung, Thomas	88-2 Der VBL-Interrupt und das volks- FORTH83
Tools	Hoffmann, Ulrich	91-1 Ein Weg, wie man FORTH klein macht
Tools	Staben, Jörg	91-2 Problembeschreibung: CURSOR
Tools	König, H.	91-3 Wo ist der Flaschenhals? (Laufzeitverhalten)
Tools	Staben, Jörg	92-3 Ein klein, fein Helferlein (Print Screen)
Tools	Staben, Jörg	92-3 Swopp mal wieder
Tools	Beierlein, T.	92-4 Was tickt denn da ...
Tools	Klingelnberg, A.	92-4 NUN endlich (Tastatur-LEDs)
Tools	Schleisiek, K.-P.	93-1 Hard-DisCo (8x8-LED-Feld)
Tools	Neumann, Helge	93-1 Ein FILE DUMPen
Tools	Klingelnberg, A.	94-1 FSSS - Forth simple, short and safe
Tools	Prinz, Friederich	94-1 DOS unter Kontrolle
Tools	Schemmert, W.	94-1 Aus Alt mach Neu (Redefinition von Worten)
Tools	Vogt, Claus	94-3 Stilles Forth - Standardausgabe weggeleitet
Tools	Prinz, Friederich	95-1 Rename und Copy (ZF)
Tools	Haase, H.-J.	95-1 Zugehörig oder nicht? (CASE)
Tools	Klingelnberg, A.	95-2 An oder Aus, High oder Low (Bitmanipulation)
Tools	Scholz, Michael	96-1 F-PC, (kooperatives Multitasking)
Tools	Staben, Jörg	96-1 Timestamping a file
Tools	Klingelnberg, A.	96-1 Noch Platz ?!? (Get Free Disk Space)
Tools	Klingelnberg, A.	96-3 Who is Who? What is TOS?
Tools	Bitter, Martin	96-3 WinView-Einstellungen unter Windows 3.1 dauerhaft speichern
Tools	Klingelnberg, A.	96-4 An oder Aus, High oder Low - da capo
Tools	Bitter, Martin	96-4 Die 64 Constant-Falle im ZF
Tools	Beuster, Bernd	96-4 Intelhex-Format
Tools	Krieger, Klaus	97-1 Low-Level-Debug-Werkzeuge für fieldFORTH
Tools	Schröder, M.	97-1 WIN32FOR: Von DOS zu Windows - Nehmen wir unsere Umlaute mit?
Tools	Bitter, Martin	97-2 Jetzt aber mal Punkt (Ordnernamen, ZF)
Tools	Behringer, Fred	97-3 Das Wort Doppelcolon aus Turbo-Forth
Tools	Prinz, Friederich	97-3 Dynamische Macros (Holon)
Tools	Bitter, Martin	98-1 Im Gleichschritt ... (Cursor in Win32Forth)
Tools	Behringer, Fred	99-1 TF, Conways "Life" und die DNF als IF-Bedingung

Tools	Prinz, Friederich	99-2 FreeMem - Speichermanagement (Win32Forth)
Tools	Bitter, Martin	99-3 Wordinfo (Win32Forth)
Tools	Behringer, Fred	99-4 BEGIN-UNTIL über 32K ab 80386 im ZF-Assembler
Tools	Behringer, Fred	99-4 Eakers CASE für Assembler in ZF und Turbo-Forth
Tools	Allinger, W.	00-1 CFA2NAME (ZF)
Tools	Behringer, Fred	01-3 Lego-Roboter und arithmetisierte Logik in Forth
Transputer	Behringer, Fred	93-4 Transputer gefällig?
Transputer	Behringer, Fred	95-4 Umlaute in den Namen von Forth-Worten
Transputer	Behringer, Fred	96-3 Forth-Server bei Forth-Host-Target-Interfaces
TSR	Prinz, Friederich	92-4 Die Uhr (1)
TSR	Prinz, Friederich	93-1 Die Uhr (2)
TSR	Prinz, Friederich	93-1 Fis(c)hing Forth (Fischer-Technik)
TSR	Prinz, Friederich	93-2 Die Uhr (3)
Umsteiger	Behringer, Fred	01-3 Forth öffnet die Türen
Umsteiger	Behringer, Fred	01-4 CODE-Definitionen ohne CODE und END-CODE
Umsteiger	Behringer, Fred	02-1 FINDRAMD.COM - Assemblerprogrammierung in Forth
Variablen	Neumann, W.-H.	93-2 Mengenrabatt - Variable und Konstante elegant deklariert
Verbund	Krüger, Adolf	86-1 Rechner sprechen Forth miteinander
Viren	Plewe, Jörg	93-4 Krieg der Kerne (F-PC, F68K)
Wanted	Staben, Jörg	92-2 SYSINFO - In der Klemme - Interrupts - Keyboard- und Geräte-Treiber
Wanted	Staben, Jörg	92-3 dBASE-Dateien - Phonetische Suche - Bibliotheken in Forth
Wanted	Klingelnberg, A.	93-1 Inhaltsangaben der Forth Dimensions
Wanted	Kretzschmar, R.	93-1 Guter Aufnehmer (Fehlerkorrektur)
Wanted	Behringer, Fred	00-2 Analogrechner
Wettbewerb	Zembrod, Philip	95-3 The Freiburg-Zurich-Nashville-Story
Wettbewerb	Vogt, Claus	95-4 Dateikonvertierung
Zeitmessung	Pennemann, B:	87-2 Laufzeitmessung von Forthworten
Zufall	Kalus, Michael	88-1 Zufallszahlen
Zufall	Plewe, Jörg Staben, Jörg	93-1 Zufallsdaten? - F-PC in den Händen der Magier

Dies ist das – vorläufige – Ende der ungeheuren Fleißarbeit, die Fred Behringer für uns geleistet hat. Alle bis zum Beginn dieser Arbeit jemals in der „Vierte Dimension“ erschienenen Artikel sind hier erfaßt und in nachvollziehbare Kategorien eingeordnet. Fred Behringer will diese Arbeit jährlich aktualisieren – wenn Sie, liebe Leser, Interesse und Bedarf an dieser Arbeit haben. Bitte schreiben Sie uns hierzu. *fep*

Leserbrief " **Keine Angst vor LOCALS**"

Lieber Herr Behringer,

vor kurzem las ich erneut in VD 2/2002 den vorzüglichen Artikel von Bernd Beuster über die verschiedenen Sortiermethoden. Weil die Methode Quicksort nicht aufgeführt war, habe ich sie gesucht und bin fündig geworden.
Es ist mit Forth manchmal zum Verzweifen. Wenn man nach langer Zeit ein Wort wieder liest, versteht man es kaum. Die vielen forthigen Worten DUP ROT 2DUP 2OVER ... , erschweren das Verstehen sehr. Man muss mühsam den ganzen Datenstack per Hand untersuchen, um zu begreifen, wie das Ganze funktioniert. Besondere die letzten 3 Zeilen von (qsort) waren eine harte Nuss.
Ich bin seit einiger Zeit ein Anhänger von LOCALS geworden, und habe deswegen (qsort) unter Verwendung von LOCALS umgeschrieben. Ich bin sicher, dass man, wenn man nach langer Zeit die Locals-Version von (qsort) wieder liest, sie auf Anhieb versteht.
Hinzu kommt gratis, dass (qsort) mit LOCALS 5% schneller ist.
Herzliche Grüsse

Filippo Sala

...weiter auf der nächsten Seite

Listing

```
\ =================
\ QUICKSORT
\ Von C. A. R Roar
\ =================

BASE @
DECIMAL

\ ---------------------------------------
\ Kurze Beschreibung

\ : qsort ( adr1 adr2 -- ) recurse

\ Vergleichswert v festlegen
\ adr1                          adr2   BEGIN
\ p -->                         <-- q
p+ q- wiederholen bis
\        p                 q
p@ >= v  &  v >= q@
\        p                 q
Inhalte vertauschen
\         p              q
p+ q-
\
bis q<p
\ adr2           q p            adr2   UN-
TIL
\
\ wenn adr1 < q  Rekursion mit den Parame-
tern  adr1 q
\ wenn p < adr2  Rekursion mit den Parame-
tern  p adr2
```

\ Forth(ige) Version

```
: (qsort) ( adr1 adr2 -- ) recurse
  dup @ >r ( R: adr2 @ )
  2dup swap
  BEGIN
    BEGIN  dup @ r@  u< WHILE  1 cells +  REPEAT
swap
    BEGIN  r@ over @ u< WHILE  1 cells -  REPEAT  swap
    2dup u<
    IF
      true
    ELSE
      2dup 2dup @ >r
      @ swap !
      r> swap !
      swap 1 cells - swap 1 cells +
      false
    THEN
  UNTIL
  r> drop
  rot 2over 2over - + > IF  2swap  THEN
  2dup u< IF  (qsort)  ELSE  2drop  THEN
  2dup u< IF  (qsort)  ELSE  2drop  THEN
```

==========
Die Seite für den Umsteiger
==========

Ein Assembl(i)eraufruf (3/3)

Julian Noble <jvn@virginia.edu>

Institut für Kern- und Teilchenphysik der Universität von Virginia, Charlottesville, VA 22901

Dritter Teil eines von Fred Behringer übersetzten Artikels, der ursprünglich für die nun leider eingestellte amerikanische Zeitschrift Forth Dimensions vorbereitet wurde. Der Artikel erschien kürzlich in der Forthwrite (113-115) der englischen Forth-Freunde. Wir danken der Forth Interest Group UK für die freundliche Befürwortung der deutschen Übersetzung.

Stichworte: Assembler, Tutorial, Forth-Philosophie, FPU, Gleitkommazahlen, Besselfunktionen, F-PC, Win32Forth

Sphärische Besselfunktionen

Wir bringen jetzt ein hinreichend kompliziertes Programmstück aus einer zahlenverschlingenden Anwendung zur Berechnung der Auswirkung einer 3D-Schwingung an beliebigen Punkten. Diese Funktion wurde laufend aufgerufen und mußte daher in Assembler programmiert werden.

Wenn man lediglich eine einzige sphärische Besselfunktion, $j_n(x)$, benötigt, ist es normalerweise das beste, sie einfach über $\sin(x)$, $\cos(x)$ und Polynome in $1/x$ zu berechnen. Braucht man aber mehrere solcher Funktionen, besonders wenn es sich um Funktionen höherer Ordnung handelt, setzt man sinnvollerweise Rekursionen ein. Die sich sofort anbietende Aufwärtsrekursion, die von der Beziehung

$$j_{n-1}(x) = (2n+1)x^{-1} j_n(x) - j_{n+1}(x)$$

mit expliziten Formeln für $j_0(x)$ und $j_1(x)$ ausgeht, ist jedoch unstabil und verliert schnell an numerischer Genauigkeit. Wir setzen daher die von Abramowitz and Stegun [1] empfohlene Abwärtsrekursion ein, mit den Anfangswerten

$$j_N = 1,\ j_{N+1} = 0$$

(für genügend großes N), und wenden dann die Beziehung

$$\sum_{k=0}^{N}(2k+1)[j_k(x)]^2 = 1$$

an, um zur Normalisierung zu gelangen. In Forth könnte das wie folgt aussehen:

```
\ Datenstrukturen
10 REAL*8 #CELLS 1ARRAY JBES{      \ j0-j9 aufbewahren
FVARIABLE SUM                      \ Zum Herunterladen vom FPU-Stack
FVARIABLE X

: SETUP   ( F: x --- 0 1 )    ( --- 79)
   X DF!   79 S>F   SUM DF!
   F0.0 F1.0   79 ;

: NORMALIZE
   SUM DF@ FSQRT 1/F
   10 0 DO  FDUP  JBES{ I } ( [2] ) DUP DF@ F* DF!
   LOOP
   FDROP ;

: DO_X=0
   FDROP  F1.0  JBES{ 0 }  DF!
   10 1 DO  F0.0  JBES{ I }  DF!  LOOP ;
```

```
: ITERATE ( F: jn+1 jn --- jn jn-1 ) ( 2n+1 --- 2n-1 )
   DUP SF FOVER F*                   ( F: jn+1 jn jn*[2n+1] )
   X DF@ F/ FROT F-                  ( F: --- jn jn-1 )
   FDUP F^2                          ( F: --- jn jn-1 jn-1^2 )
   2- DUP                            ( --- 2n-1 2n-1 )
   S>F F*
   SUM DF@ F+
   SUM DF! ;

: SPHBES ( F: x --- )
   FDUP F0=
   IF DO_X=0 EXIT THEN
   SETUP  11 39 DO ITERATE  -1 +LOOP
   0 9 DO ITERATE
      FDUP  JBES{ I }  DF!
   -1 +LOOP
   DROP FDROP FDROP                  \ Stacks aufräumen
   NORMALIZE ;
```

Die Übersetzung dieser Routine in die Assemblersprache macht den Hauptarbeitsanteil des vorliegenden Artikels aus. Sie ist ziemlich lang und stellt die oberste Grenze dessen dar, was man in der nie enden wollenden Suche nach Schnelligkeit gerade noch in einer einzigen Routine abzuarbeiten vertreten kann. Übergangswerte und Zwischenergebnisse werden wir auf dem eingebauten Stack des Gleitkommaprozessors halten, um den Transport zum (langsameren) Hauptspeicher und zurück auf ein Minimum zu beschränken. Das Public-Domain-Forth-System F-PC enthält keine Assembler-Erweiterung für den 80x87. Zum Programmieren und Austesten dieses Programmteils müssen wir daher einen der folgenden Wege beschreiten:

- Man erweitere den F-PC-Assembler um die benötigten Teile. (Robert L. Smith hat das mit seiner Gleitkomma-Erweiterung ffloat.seq getan, die auf einer Reihe von Forth-Archiven liegt.)
- Man verwende den Mikromini-Assembler aus dem zweiten Teil dieses Artikels.
- Man verwende ein Forth-System mit einem vollständigeren Assembler, wie beispielsweise Win32Forth von Tom Zimmer.

Die Gleitkomma-Einheit (FPU) der Intel-Mikroprozessoren enthält einen eingebauten Stack von 8 Einheiten Tiefe [3]. Bevor wir das eigentliche Programmstück starten, müssen wir den FPU-Stack initialisieren, so daß er nichts mehr enthält. Wir deuten das wie folgt an:

Die ersten Schritte bestehen aus der Initialisierung. Der FPU-Stack enthält dann **x**, das Argument der Besselfunktion(en), sowie die Anfangswerte von **jn, jn+1,** und was sonst noch nötig ist. Das sieht dann wie folgt aus:

```
st(7) ...        st(7) ...
st(6) ...        st(6) ...
st(5) ...        st(5) ...
st(4) ...        st(4) x
st(3) ...        st(3) sum
st(2) ...        st(2) 2n+1
st(1) ...        st(1) jn+1
st(0) ...        st(0) jn
```

In jeder weiteren Iteration geht der Stack über in:

```
st(7) ...    ->    st(7) ...
st(6) ...          st(6) ...
st(5) ...          st(5) ...
st(4) x            st(4)   x
st(3) sum          st(3)   sum + (2n+1)*jn*jn
st(2) 2n+1         st(2)   2n-1
st(1) jn+1         st(1)   jn
st(0) jn           st(0)   jn-1
```

Fangen wir bei den Initialisierungsschritten an:

```
finit                          \ FPU-Stack löschen

mov ecx, FSP [edi]             \ fstack ptr holen
sub ecx, # B/FLOAT             \ Um Datenbreite dekrementieren
js L$2                         \ -> Fehlerbehandlung
fld FSIZE FSTACK [ecx] [edi] ( 87: x)
mov FSP [edi], ecx             \ fstack ptr anpassen
push ebx                       \ TOS -> mem
push # 4F                      \ 79d=4Fh auf den Datenstack legen
fldz                           \ fload 0 ( 87: 0 x )
fild dword 0 [esp]             \ 79d - st(0)
fldz                           \ fload 0
fld1                           \ fld 1 ( 87: 1 0 79 0 x )
pop ebx                        \ ebx = 79 ( 87: jn jn+1 2n+1 sum x )
```

Die Initialisierungsroutine löscht den Stack der Gleitkomma-Einheit (FPU) und holt **x** aus dem im Speicher befindlichen **fstack** zur FPU. (Dieser Teil wurde direkt aus dem Wort **fpop** in **float.f** übernommen.) Schließlich werden die numerischen Konstanten geladen.

Als nächstes schauen wir uns an, was bei den einzelnen Iterationen passiert: Wir müssen sorgfältig auf den FPU-Stack aufpassen, da sich dort nach der Initialisierung 5 Einträge befinden. Wir wissen, daß wir den Faktor $(2_{n+1})\times j_n$ an zwei Stellen benötigen: Zum einen, um j_{n-1} zu berechnen, und zum anderen zur Berechnung des nächsten Ausdrucks in der Summe. Zur Ausarbeitung der benötigten Schritte geben wir nach jedem Maschinenbefehl den FPU-Stack an:

```
       FXCH ST(1)   FLD ST(1)          FMUL ST(0), ST(3)
st(7) ...           ...                ...
st(6) ...           ...                ...
st(5) ...           x                  x
st(4) x             sum                sum
st(3) sum           2n+1               2n+1
st(2) 2n+1          jn                 jn
st(1) jn            jn+1               jn+1
st(0) jn+1          jn                 jn*(2n+1)

       FLD ST(0)    FMUL ST(0),        ST(3) FADDP ST(5), ST(0)
st(7) ...           ...                ...
st(6) x             x                  ...
st(5) sum           sum                x
st(4) 2n+1          2n+1               sum'
st(3) jn            jn                 2n+1
st(2) jn+1          jn+1               jn
st(1) jn*(2n+1)     jn*(2n+1)          jn+1
st(0) jn*(2n+1)     jn*(2n+1)*jn       jn*(2n+1)

       FDIV ST(0), ST(5) FSUBRP ST(1), ST(0) FLD1
st(7) ...           ...                 ...
st(6) ...           ...                 ...
st(5) x             ...                 x
st(4) sum'          x                   sum'
st(3) 2n+1          sum'                2n+1
st(2) jn            2n+1                jn
st(1) jn+1          jn                  jn-1
st(0) jn*(2n+1)/x   jn-1                1

FSUB ST(3),         ST(0) FSUBP ST(3), ST(0)
st(7) ...           ...
st(6) ...           ...
st(5) x             ...
st(4) sum'          x
st(3) 2n            sum'
st(2) jn            2n-1
st(1) jn-1          jn
st(0) 1             jn-1
```

Das heißt, die vollständige Befehlssequenz zur Ausführung eines Iterationsschrittes sieht wie folgt aus:

```
fxch   st(1)          ( 87: jn+1 jn k=2n+1 sum x )
fld    st(1)          ( 87: jn jn+1 jn k sum x )
fmul   st(0), st(3)   ( 87: k*jn jn+1 jn k sum x )
fld    st(0)          ( 87: k*jn k*jn jn+1 jn k sum x )
fmul   st(0), st(3)   ( 87: k*jn^2 k*jn jn+1 jn k sum x )
faddp  st(5), st(0)   ( 87: k*jn jn+1 jn k sum' x )
fsubpr st(1), st(0)   \ Das ist ein Druckfehler in 486asm.f
  \ ^^^^^^
fld1
fsub   st(3), st(0)
fsubp  st(3), st(0)   ( 87: jn-1 jn 2n-1 sum' x)
```

Wie können wir das jetzt auf Fehlerfreiheit überprüfen? Das Schöne am Austesten einer Assembler-Routine in einer Forth-Umgebung ist, daß wir dazu keinerlei Linker benötigen. Wir können größere und immer größere Teile des **CODE**-Wortes assemblieren und den jeweils ausgetesteten Teil sofort **FORGET**ten, um zum Austesten des nächsten Teils überzugehen. (Vorausgesetzt natürlich, daß wir bei unseren Hantierungen das System nicht zum Absturz gebracht haben.)

Das Wort **ITERATE** wurde stufenweise aufgebaut und auf jeder Stufe interaktiv ausgetestet. Auf der letzten Stufe wurde eine **BEGIN .. UNTIL** - Schleife eingefügt. Viele Forth-Assembler stellen für diesen Zweck Makros zur Verfügung. Da es jedoch mein Ziel war, eine Routine zu erstellen, die (nach Einbau eines geeigneten Anpassungsteils an Anfang und Ende) leicht auf andere High-Level-Sprachen übertragen werden kann, habe ich von den Forth-spezifischen Makromöglichkeiten keinen Gebrauch gemacht.

Wie man sieht, wird am Anfang einer jeden Iteration der momentane Wert der Besselfunktion (natürlich noch nicht richtig normalisiert) in der dafür vorgesehenen Komponente des Arrays **jbes{** gespeichert. Hierzu wird zuerst mit Hilfe des Ausdruckes **jbes { 0 }** die Basisadresse berechnet, welche dann zu den durch die Indexregister **ebx** und **edi** gegebenen Offsets hinzuaddiert wird. Der Array-Index scheint, wie man meinen könnte, mit 4 (Bytes) multipliziert zu sein, was einer 32-Bit-Genauigkeit entspräche. Zu diesem Abspeicherzeitpunkt hat **ebx** jedoch den Wert **2n**, da **ebx** schon einmal dekrementiert wurde. Das heißt also, daß die Routine in Wirklichkeit zur Aufnahme von 64-Bit-Gleitkommazahlen geschrieben wurde - sehr wichtig, da die Größe der nichtnormalisierten Funktionen (von der der Normalisierungssumme erst gar nicht zu sprechen) ganz leicht den vom IEEE vorgesehenen Platz zur Aufnahme von 32-Bit-genauen Zahlen sprengen kann.

Die erste "**dec ebx**"-Anweisung (die in **ebx** den Wert **2n** hinterläßt) kennzeichnet den Anfang der Schleife. Die zweite "**dec ebx**"-Anweisung kennzeichnet den letzten Berechnungsschritt der Schleife. Wir markieren den Anfang der Schleife mit Hilfe der Assembler-Möglichkeit für lokale Labels (mit dem Ausdruck **L$1:**) und verwenden den Intel-Befehl **jns** (”jump not sign”), um wieder dorthin zurückzuspringen, wenn der Dekrementierungsvorgang das Indexvorzeichen im Register **ebx** nicht verändert hat (solange also $2n\text{-}1 \geq 0$ bleibt).

Schließlich müssen wir noch die Stacks aufräumen. Der Endwert des Indexes (--1) im Register **ebx** (das in Win32Forth als Aufbewahrungsort für den obersten Datenstackeintrag verwendet wird) muß wieder durch denjenigen Wert, welcher auch immer, ersetzt werden, der sich vor Einsprung in die Routine im obersten Stackeintrag befunden hat. Das wird von der Anweisung "**pop ebx**" geleistet. Da es an sich nicht darauf ankommt, wann das erfolgt, erledigen wir das als letztes. Die einzige Zahl, die wir vom FPU-Stack zurückbehalten wollen, ist die Summe. Wir "**pop**"pen also die obersten drei Einträge durch dreimalige Anwendung der Anweisung **fstp st(0)**. Dann übertragen wir die Summe nach **fstack** im Speicher (indem wir uns einfach die Befehlsfolge von **fpush** für diesen Zweck kopieren) und schließlich entfernen wir unter abermaliger Anwendung von **fstp st(0)** den Wert **x** vom FPU-Stack.

Ob Sie es glauben wollen oder nicht, als ich dieses Programmstück einfügte und das im Listing weiter unten angegebene High-Level-Wort **sphbes** austestete, funktionierte das auf Anhieb, und zwar gut. Die gesamte Testabfolge, einschließlich der benötigten Korrektur eines Fehlers, dauerte 15-20 Minuten. Ich glaube nicht, daß MASM oder TASM das in auch nur annähernd so kurzer Zeit geschafft hätten.

Mit der nun erfolgenden Angabe des vollständigen Programms zur Berechnung sphärischer Besselfunktionen will ich meinen Aufruf zu mehr Assembler-Programmierung schließen. Ich danke Ihnen für Ihre Aufmerksamkeit.

Anhang

Es folgt das vollständige Programm zur Berechnung sphärischer Besselfunktionen, bei welchem die Iterationsschleife in Assembler gefaßt wurde.

```
FALSE [IF]
Reguläre sphärische Besselfunktionen j_n(x), n=0-39
(Assembler-Version, geeignet für Win32Forth)
© J.V. Noble 1999. Darf für jeden Zweck verwendet werden, vorausgesetzt, diese Co-
pyright-Bemerkung wird beibehalten.

Verwendet die Methode von Miller zur Abwärtsrekursion, welche in Abramowitz & Ste-
gun, "Handbook of Mathematical Functions", 10.5 ff, beschrieben wird.

Die Rekursion lautet
j(n-1) = (2n+1) j(n) / x - j(n+1)

Die Abwärtsrekursion fängt mit j40 = 0, j39 = 1 an.
Die sich ergebenden Funktionen werden normalisiert über
Sum (n=0 bis inf) { (2n+1) * jn(x)^2 } = 1 .

Verwendung: Zur Berechnung von j0-j39, sagen wir

   3.0e0 sphbes
Zum Zugriff auf einen Wert / zur Anzeige eines Wertes, sagen wir
   jbes{ 3 } F@ F. .1520516620 ok
[THEN]

marker -jbes
include arrays.f
40 long 1 dfloats 1array jbes{
FVARIABLE x
HEX

code ITERATE ( f: x --- ) \ Initialisierung
   finit                \ FPU-Stack löschen
   mov ecx, FSP [edi]   \ x vom FSTACK nach st(0) bringen
   sub ecx, # B/FLOAT
   js L$2               \ -> Fehlerbehandlung
   fld FSIZE FSTACK [ecx] [edi] ( 87: x )
   mov FSP [edi], ecx   \ Übertragung beendet
   push ebx
   push # 4F            \ 79d auf den Datenstack
   fldz                 ( 87: 0 x )
   fild dword 0 [esp]   ( 87: 79 0 x )
   fldz
   fld1                 ( 87: 1 0 79 0 x )
   pop ebx              \ ebx = 79 = 2N+1
   ( 87: jn jn+1 2n+1 sum x ) \ Ende der Initialisierung
L$1:
   dec ebx              \ Hier beginnt die Schleife
   fst double jbes{ 0 } [ebx*4] [edi]
\  fwait                \ Wird eventuell benötigt
   fxch st(1)           ( 87: jn+1 jn k=2n+1 sum x )
   fld st(1)            ( 87: jn jn+1 jn k sum x )
   fmul st(0), st(3)    ( 87: k*jn jn+1 jn k sum x )
   fld st(0)            ( 87: k*jn k*jn jn+1 jn k sum x )
   fmul st(0), st(3)    ( 87: k*jn^2 k*jn jn+1 jn k sum x )
   faddp st(5), st(0)   ( 87: k*jn jn+1 jn k sum' x )
   fdiv st(0), st(5)    ( 87: k*jn/x jn+1 jn k sum' x )
   fsubpr st(1), st(0)  \ Das ist ein Schreibfehler in 486asm.f
\  ^^^^^^               \ --- sollte fsubrp heißen
   fld1
```

```
    fsub st(3), st(0)
    fsubp st(3), st(0)    ( 87: jn-1 jn 2n-1 sum' x )
    dec ebx
    jns L$1               \ Hier endet die Schleife
                          ( 87: j0 j1 -1 sum x )
    fstp st(0)            ( 87: j1 1 sum x )
    fstp st(0)            ( 87: 1 sum x )
    fstp st(0)            ( 87: sum x )
    mov ecx, FSP [edi]    \ sum-fstack
    fstp FSIZE FSTACK [ecx] [edi]
    fwait
    add ecx, # B/FLOAT
    mov FSP [edi], ecx
    fstp st(0)            ( 87: x --- )
    pop ebx               ( -1 --- )
    jmp L$3
L$2:
mov esi, # ' FSTKUFLO body \ Fehlerbehandlung
add esi, edi
L$3:
next,
end-code

DECIMAL

: DO_X=0 \ Behandlung des Sonderfalls x=0
    FDROP F1.0 JBES{ 0 } DF!
    10 1 DO F0.0 JBES{ I } DF! LOOP ;

: NORMALIZE ( f: sum --- )
    FSQRT F1.0 FSWAP F/
    39 0 DO FDUP JBES{ I } DUP F@ F* F! LOOP
    FDROP ;

: SPHBES ( f: x --- )
    FDUP F0= \ x=0 ?
    IF DO_X=0 ELSE ITERATE NORMALIZE THEN ;
```

[1] M. Abramowitz and I.A. Stegun, Handbook of Mathematical Functions (Dover Publications, Inc., New York, 1965), S.452.

[2] Diese Bezeichnungsweise wurde in meinem Buch "Scientific Forth" eingeführt und wurde von dem von Skip Carter organisierten Projekt "Forth Scientific Subroutine Library" als Standardbezeichnung übernommen.

[3] Die Stacknotierung (87: --) bezieht sich auf den 8 Eingänge tiefen FPU-eigenen Stack. (Die Reihe der Intel-FPUs begann als separate Chips mit den Bezeichnungen 8087/80287/80387, bevor die FPU-Chips dann in den 80486 und den Pentium integriert wurden.)

Der Assembl(i)eraufruf endet hier, liebe Leser; zumindest vorerst. War das „harte Kost“ - oder ein kurzweiliger Exkurs? Daß Assembler nach wie vor unverzichtbar sind, geht manchmal in der Hetze des (Programmier-)Alltags verloren. Immer leistungsfähigere Prozessoren erlauben es uns, immer mehr Rechenleistung sozusagen zum Fenster hinaus zu pusten. Erst Lernen, dann über ein gegebenes Problem nachdenken, dem Prozessor abverlangen, was er zu leisten vermag und am Ende etwas abliefern, von dem „man wirklich etwas hat“, das scheint heute nicht sehr gefragt zu sein. Ich habe in diesen Tagen in Stuttgart in der S-Bahn-Linie 3 ab der Haltestelle Universität auf dem Weg in die Innenstadt einer Unterhaltung zwischen Informatikstudenten zuhören dürfen oder müssen. Da fragt eine junge Frau, was sie denn nun schon wieder mit den Booleans solle, und wozu sie diese überhaupt brauche. Ein Konsemester antwortet ihr, daß es da mal vor mehr als zwanzig Jahren einen Programmierer gegeben hätte. Der hätte Boolean oder so ähnlich geheißen und ein paar coole Statements abgeliefert, die aber heute eigentlich Niemand mehr braucht.
Ich frage mich, was dieser junge Geck wohl mit „seinem“ Java anstellt, wenn er Problem ähnlich diesem hier vorgelegt bekommt. Ich denke, wir brauchen viel mehr solcher Beiträge. *fep*

Forth-Gruppen regional

Moers

Friederich Prinz
Tel.: 02841-58398 (p) (Q)
(Bitte den Anrufbeantworter nutzen !)
(Besucher: Bitte anmelden !)
Treffen: 2. und 4. Samstag im Monat
14:00 Uhr, **MALZ, Donaustraße 1**
47441 Moers

Mannheim

Thomas Prinz
Tel.: 06271-2830 (p)
Ewald Rieger
Tel.: 06239-920 185 (p)
Treffen: jeden 1. Mittwoch im Monat
Vereinslokal Segelverein Mannheim e.V.
Flugplatz Mannheim-Neuostheim

München

Jens Wilke
Tel.: 089-89 76 890
Treffen: jeden 4. Mittwoch im Monat
Ristorante Pizzeria Gran Sasso
Ebenauer Str. 1
80637 München

µP-Controller Verleih

Thomas Prinz
Tel.: 06271-2830 (p)
micro@forth-ev.de

Gruppengründungen, Kontakte

Hier könnten SIE
sich zur Gründung einer
lokalen Gruppe zur Verfügung stellen !

Forth-Hilfe für Ratsuchende

Forth allgemein

Jörg Plewe
Tel.: 0208-49 70 68 (p)

Jörg Staben
Tel.: 02103-24 06 09 (p)

Karl Schroer
Tel.: 02845-2 89 51 (p)

Spezielle Fachgebiete

Arbeitsgruppe MARC4 — **Rafael Deliano**
Tel./Fax: 089-841 83 17 (p)

FORTHchips
(FRP 1600, RTX, Novix) — **Klaus Schleisiek-Kern**
Tel.: 040-375 008 03 (g)

F-PC & TCOM, Asyst (Meßtechnik), embedded Controller (H8/5xx// TDS2020, TDS9092), Fuzzy — **Arndt Klingelnberg**, Consultants
akg@aachen.kbbs.org
Tel.: ++32 +87 -63 09 89 (pgQ)
(Fax -63 09 88)

KI, Object Oriented Forth, Sicherheitskritische Systeme — **Ulrich Hoffmann**
Tel.: 04351 -712 217 (p)
Fax: -712 216

Forth-Vertrieb
vlksFORTH
ultraFORTH
RTX / FG / Super8
KK-FORTH
Ingenieurbüro **Klaus Kohl**
Tel.: 08233-3 05 24 (p)
Fax : 08233-99 71
mailorder@forth-ev.de

Möchten Sie gerne in Ihrer Umgebung eine lokale Forthgruppe gründen, oder einfach nur regelmäßige Treffen initiieren ? Oder können Sie sich vorstellen, ratsuchenden Forthern zu Forth (oder anderen Themen) Hilfestellung zu leisten ? Möchten Sie gerne Kontakte knüpfen, die über die VD und das jährliche Mitgliedertreffen hinausgehen ?

Schreiben Sie einfach der VD - oder rufen Sie an - oder schicken Sie uns eine E-Mail !

Hinweise zu den Angaben nach den Telefonnummern:

Q = Anrufbeantworter
p = privat, außerhalb typischer Arbeitszeiten
g = geschäftlich

Die Adressen des Büros der Forthgesellschaft und der VD finden Sie im Impressum des Heftes.

4 / 2002

Eine ganz besonders schöne E-Mail

Hallo ,
die nächste Jahrestagung findet nicht wie angekündigt in Bad Dürkheim statt. Die Kosten waren dort einfach zu hoch; oder die Räumlichkeiten waren nicht passend. Die Tagung findet nun in der Pfalzakademie in Lambrecht bei Neustadt/Weinstrasse vom 11.04.2003 bis 13.04.2003 statt. Dort habe ich 15 Einzelzimmer und 10 Doppelzimmer und einen Tagungsraum für 35 Personen reservieren lassen. Lambrecht, eine kleine Stadt mit rund 5000 Einwohnern, liegt bereits im Pfälzerwald 7 km westlich von Neustadt an der Weinstrasse. Seine umliegenden Wälder, Burgen und die Hütten des Pfälzer Waldvereins laden neben der Tagung zu einer erholsamen Wanderung ein. Die naheliegende Weinstrasse kann in wenigen Autominuten oder mit der Bahn im Halbstundentakt erreicht werden.

Als Anhang habe ich zunächst das Anmeldeformular zur Jahrestagung 03 beigefügt *[Das Formular liegt dieser Ausgabe der VD bereits bei!]*. Zur Gestaltung der Einladung in der VD werde ich noch ein paar Bilder von der Tagungsstätte und Umgebung anfertigen.

Viele Grüße *Ewald Rieger*

Hambacher Schloß und Lambrecht

Weinstube

Tagungsraum