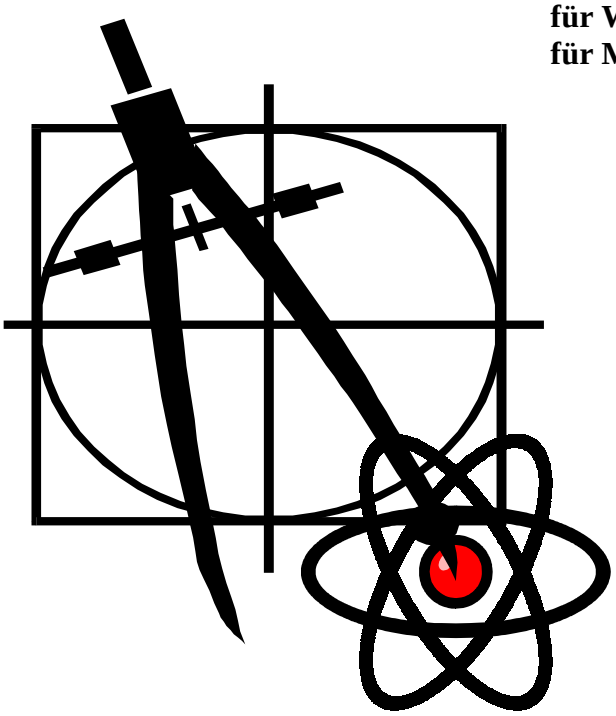
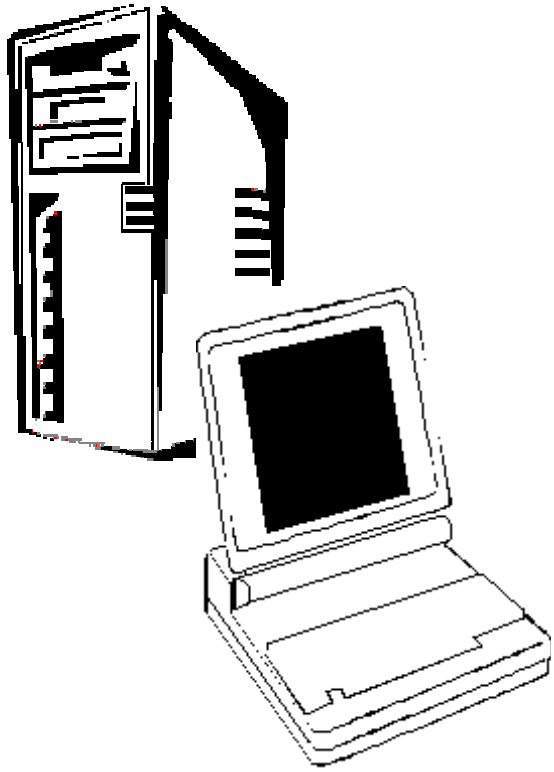
# VIERTE DIMENSION

3 / 2002

## Das FORTH-Magazin

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



### In dieser Ausgabe:




18. Jahrgang - **Ausg. 3/2002** - 5,11 Euro

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, !

Martin Bitter

## 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 HighSpeed-Duplicating, 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

Joachim-Jungius-Straße 9 D-18059 Rostock
Tel.: (0381) 405 94 72 Fax: (0381) 405 94 71

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:

- die Fortsetzung der VD Titelliste,
- Zyklische Codes (R.Deliano),
- Threaded Codes, Varianten und Optimierungen (A.Ertl)
- Debugging mit einem Mixed Signal Oszilloskop (K.Zobawa)
- b16 Forthprozessor im FPGA (B. 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 Euro + Porto u. Verpackung


**Manuskripte und Rechte**



Liebe Leser,

Wenn Sie diese Zeitschrift auf Ihren Schreibtisch bekommen, dann sind sehr viele von Ihnen entweder bereits aus dem Urlaub zurück oder stehen unmittelbar vor einer Reise. Ich wünsche Ihnen darum, sich gut zu erholen und entspannt und ausgeruht in die zweite Hälfte des Jahres "durchstarten" zu können. Wenn Sie die Vierte Dimension an Ihren Urlaubsort mitgenommen haben, dann würde es mich schon interessieren zu lesen, an welchen Orten überall in der Welt unsere Zeitschrift in diesem Jahr zu lesen war.

*Friederich Prinz*

*Es ist nicht immer einfach, den für das „Editorial" vorgesehenen Platz mit intelligent klingenden Sätzen zu füllen. Ich lasse das dieses Mal lieber und gebe statt dessen einige ‚Hinweise' wieder, die von verschiedenen Lesern hier eingereicht wurden...*

*fep*

## Wenn Betriebssysteme eine Airline betreiben würden ...

**DOS Airlines:**
Alle schieben das Flugzeug an, bis es abhebt. Dann springen alle auf und lassen das Flugzeug trudeln, bis es wieder auf dem Boden aufschlägt. Dann schieben wieder alle an, springen auf ...

**Mac Airlines:**
Alle Stewards, Stewardessen, Piloten, Gepäckträger und Ticketverkäufer sehen gleich aus, bewegen sich gleich und sagen das Gleiche. Wenn man nach Details fragt, bekommt man immer die gleiche Antwort: das müsse man nicht wissen, wolle es auch nicht wissen, und alles laufe schon richtig. Man solle also lieber gleich still sein.

**Windows Airlines:**
Das Flughafenterminal ist schön bunt, die Stewards und Stewardessen freundlich. Man gelangt ohne Probleme an Bord, ein reibungsloser Start... Plötzlich stürzt das Flugzeug ohne jegliche Vorwarnung ab.

**OS/2 Airlines:**
Um an Bord des Flugzeugs zu kommen, muß man sein Ticket zehnmal stempeln lassen und in zehn verschiedenen Schlangen anstehen. Dann füllt man ein Formular aus, in dem man angeben muß, wo man sitzen möchte und ob der Sitzplatz wie in einem Schiff, einem Bus oder einem Zug aussehen soll. Wenn es einem gelingt, an Bord zu kommen, und wenn das Flugzeug tatsächlich vom Boden abhebt, hat man einen wunderbaren Flug... - außer wenn die Höhen- und Seitenruder einfrieren. In diesem Fall hat man jedoch immer noch genügend Zeit, sich auf den Absturz vorzubereiten.

**Unix Airlines:**
Jedermann bringt ein Stück des Flugzeugs zum Flughafen mit. Alle gehen auf die Startbahn und setzen das Flugzeug Stück für Stück zusammen. Dabei diskutieren sie fortwährend, welche Art von Flugzeug sie gerade zusammenbauen.


***...weiter Seite 10***


**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*

Hallo Fritz,

auf der Website der Physical Review Letters habe ich folgenden Abstract aus der Ausgabe vom 10 Juni 2002 gefunden.
(http://prl.aps.org/ Suchmaske Vol: = 88 Article: = 237901 oder gleich http://link.aps.org/abstract/PRL/v88/e237901)

Die Rechenfähigkeit des Universums, von Seth Lloyd.
d'Arbeloff Laboratory for Information Systems and Technology, MIT Department of Mechanical Engineering, MIT 3-160, Cambridge, Massachusetts 02139
(Eingang: 23. Januar 2002; veröffentlicht: 24. mai 2002)

Jedes physikalische System enthält und verarbeitet Informationen. Durch die Naturgesetze ist festgelegt, welche Menge an Information (in Bits) ein physikalisches System enthalten kann und welche logischen Elementaroperationen (OPS) ein solches System ausführen kann. Das gesamte Universum ist ebenfalls ein physikalisches System. Die Gesamtmenge der Informationen und Rechenoperationen, die es in seiner Entwicklung bisher speichern und ausführen konnte, wurde berechnet.
Demnach konnte das Universum bisher 10^120 logische Elementaroperationen mit 10^90 Bits (unter Einbezug der gravitativen Freiheitsgrade 10^120 Bits) ausführen.

(Ende der Übersetzung)
Auf der Website kann - wer will - den gesamten Artikel für gutes Geld 'runtersaugen'.

Gruß Martin *(Martin Bitter)*

Moin moin Fritz,

bei uns KüstenFORTHlern hat sich einiges geändert. Zum einen die Adresse wo wir uns treffen und zum Anderen die Zeiten. Und zwar wie folgt:

KüstenFORTH
Klaus Schleisiek, kschleisiek@send.de
Treffen 1 x im Quartal, Termin auf Anfrage
Fa. SEND,
Rostocker Str. 20
D-20099 Hamburg
Tel: +49 40 375008-13

Klaus *(Klaus Zobawa)*

Lieber Friederich,

neulich stöberte ich in alten Heften der 'Vierten Dimension' und blieb für einen Moment hängen bei der allerersten Ausgabe. Darin formulierte ein Programierer seine Befürchtungen in Bezug auf Forth. Zitat:
"Die unvorstellbare Komplexität der EDV-Systeme ist so allverbreitet, daß es undenkbar ist, die Notwendigkeit dieser Situation in Frage zu stellen. Ebenso unvorstellbar ist es für einen Praktiker auf diesen Systemen, eine Möglichkeit, daß es anders sein könnte, auch nur im Bereich seines geistigen Horizonts zu dulden. Aber eben genau das repräsentiert FORTH. Und deshalb ist es unmöglich, in der EDV-Szene ein Wort über FORTH zu sagen, und ausgehört zu werden."
(Andreas Goppold, in: Vierte Dimension Vol-I/No.1 Oktober 1984, Seite 20.)

Und nun frage ich mich, wie es denn heute damit steht. EDV-Systeme sind, so vermute ich, inzwischen viel komplexer als selbst G. damals ermessen konnte. Und wo steht Forth heute? Was ist seien Bedeutung jetzt und rückblickend. Und was ich damit nicht meine ist folgendes: Allgemeine Statements irgendwo zwischen jauchzen oder jammern. Sondern lieber konkret und persönlich geantwortet: Was war für mich an Forth wichtig und wann war das? Und wie ist das heute?
Das, lieber Friederich, wüßte ich gern mal von den Vereinsmitgliedern.

*Michael Kalus*

*Also schreiben Sie bitte etwas dazu, liebe Vereinsmitglieder und/oder weitere Leser.*

*fep*

Hallo Fritz,

die neue 4D ist gerade angekommen - vielfältig, und wie ich sehe hast Du jetzt auch noch einiges auf 'Halde'.
Die Meldungen über die Forth-Kategorie bei Lycos und Enth sind von mir (waren in meinem Materialordner).
Dein Rätsel soll im Hintergrundbild versteckt sein? Na -ja. Ich hab' es mal abgespeichert und schau es mir gleich genau an.
Erst einmal frohe Pfingsten Euch beiden.

Gruß Martin *(Martin Bitter)*

*Die „Halde“ der VD hält sich, wie immer, in engen Grenzen. Aber diese Ausgabe läßt sich noch leicht füllen, und mit ein wenig Mitarbeit der Leser unserer Zeitschrift bekommen wir sicher auch noch die vierte Ausgabe dieses Jahres gefüllt.*
*Da ich bisher in Bezug auf das „Rätsel“ noch gar keine Meldungen bekommen habe, lasse ich es einstweilen noch ohne Auflösung. Wenn diese Ausgabe der VD ihre Leser erreicht, sind viele entweder auf dem Weg in den Urlaub oder bereits wieder zurück. Vielleicht ist dann ausreichend Muße, sich um das Rätsel zu kümmern.*

*fep*

An: "Friederich Prinz"
The new 2.1.3 Version of pbForth has only minor fixes thanks to the investigative efforts of David Rojas and Rob Harvey. They were able to provide detailed, reproducible bug reports, which makes fixing bugs so much easier. Thanks guys!
Here's what has changed...

1. Fixed stack error in SOUND_TONE. It used to leave an extra parameter lying around on the stack with predictable results if you use this word in a loop.
2. Fixed a nasty communications bug which made the RCX hang under certain conditions - like shining a bright light at the RCX. The serial interface on the micro does not set RDRF if the character has an error.

Here's where to get the new files:

<http://www.hempeldesigngroup.com/lego/pbForth/download.htm>

Remember to send me an email if you are using pbForth in schools or if you have a website with some interesting bots programmed with pbForth!

Please give it a try and note any installation problems. Report any problems to the robotics.rcx.pbforth group at <http://www.lugnet.com> or send me an email.

And be prepared for a nice surprise in a few weeks. A well-known pbForth user is just about ready to release some VERY interesting software....
Cheers,

*Ralph Hempel*

*Ralph Hempel, seine Mannschaft und seine Mitstreiter weltweit sind sehr rührig. Wenn Sie entsprechende Mitteilungen über Änderungen und/oder neue Versionen des Interpreters gerne schneller verfügbar haben wollen als über unsere Zeitschrift, empfiehlt es sich, sich direkt in Ralph's Mailingliste aufnehmen zu lassen. Bitte schauen Sie dazu auf seiner Webseite vorbei.*

*fep*

Hi Friederich,

einen GPS-Empfänger wirst Du wohl nicht selber bauen. Das ist Hochfrequenztechnik „at it's best". Und hat mit Forth nichts zu tun. Es gibt etwa zwei Dutzend verschiedene Module auf dem Markt, mit mehr oder weniger eingebauter Intelligenz. Die reinen Empfänger liegen so bei $20-40. Für $80 kriegt man ein komplettes Handheld in der Größe einer Zigarettenschachtel.

Die erreichbare Genauigkeit hängt wesentlich von den Fähigkeiten des Empfängers ab. Die zwei wichtigsten Parameter sind die Anzahl der Satelliten, die er gleichzeitig empfangen kann und seine Eingangs-Empfindlichkeit. Die Signale kommen mit einer Empfangsleistung von ca. -200dBm an, also mit ca. 2pW (2*10 ** -12). Da muß schon ausgefeilte Technik herhalten.

Was aus dem Empfänger kommt, ist ein Bitstrom für jeden Satelliten. Der Satellit schickt die in Frames von 1500 Bits mit 50bps. Darin kodiert sind im wesentlichen zwei Informationen: Die genaue Zeit und die Bahndaten (Ephemeriden) des Satelliten. Durch "pseudo-ranging" und Zeitvergleich der Uhren der verschiedenen Satelliten ist damit eine Positionsbestimmung in 3D möglich.

Da das DoD sich verpflichtet hat Selective-Availability (SA) nicht mehr zu aktivieren, kommt man mit so einem Gerät auf Genauigkeiten von 3 -30m, je nach Umgebung.

Ich habe noch nichts selber programmiert, da die Berechnung ziemlich aufwendig ist. Allein das Einrasten auf einen Satelliten ist schon ein Prográmmchen für sich. Die Information ist nämlich verschlüsselt. Der Code ist zwar öffentlich bekannt, aber trotzdem muß man durch Korrelation feststellen, wo im Segment der Satellit gerade seine Bits herholt. Grob abschätzen kann man das über die Zeit (wenn der Empfänger eine genaue Uhrzeit hat), aber das läßt immer noch eine Ungenauigkeit von ein paar Bits übrig.

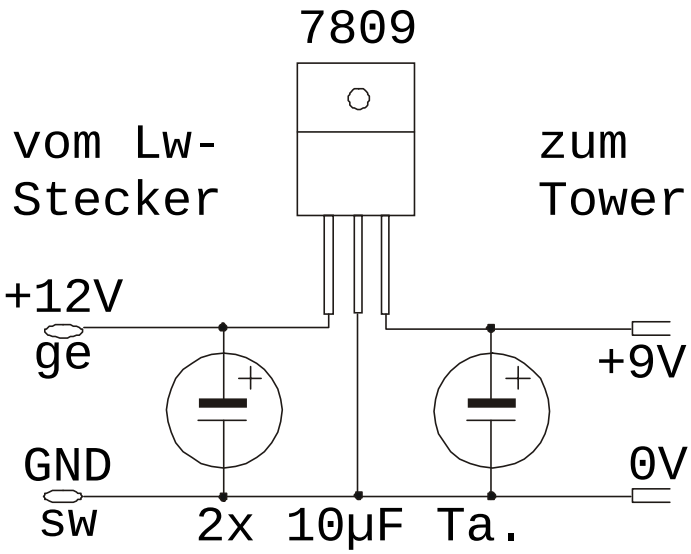
CU Uli *(Ulrich Paul)*

*Vielen Dank für diese erste, bereits sehr aufschlußreiche Information! Ich werde meine Aktivitäten in diese Richtung wohl doch auf den nächsten Winter verlegen ;-)*
*Tatsächlich möchte ich gerne zwei Dinge mit einem GPS-System erreichen. Ich möchte Wege (meist mit dem Motorrad zurückgelegt) dreidimensional „mitschreiben" und später auswerten können. Vor allem eine ausreichend genaue (+-5m) Höhenmessung möchte ich aber auch online vornehmen und während der Fahrt (oder Wanderung) ablesen können. Ihren Ausführungen entnehme ich, daß es allemal einfacher (und vermutlich auch preiswerter) ist, hier auf eines der Produkte der diversen Anbieter zurück zu greifen.*

fep

Tower-Power, Rolf Schöne

Der LEGO IR-Turm ist über ein serielles Nullmodem-Kabel dauernd mit dem PC verbunden. Es bietet sich daher an, die Batterie aus dem Turm zu entfernen und ihn aus dem PC zu versorgen. Dazu sind die im PC an einem Laufwerkstecker abgreifbaren +12V auf +9V zu reduzieren, was mit der folgenden kleinen Schaltung geschehen kann, deren +9V-Ausgang mit dem +9V-Anschluß auf der Turm-Platine zu verbinden ist:




Für die Art der Ausführung sind keinerlei Grenzen gesetzt. Man kann das fliegend verdrahten, auf eine Lochrasterplatine löten, die Schaltung unterwegs offen liegen lassen, in eine Schachtel einbauen, in den PC schrauben, auf einem Slotblech montieren, im geräumigen Batterieschacht des Towers oder im Tower selbst unterbringen, ... Sie haben sicher noch andere, womöglich bessere Ideen.

Sollten Sie, wie ich, das COM-Port ausschließlich für den Tower verwenden, so gibt es noch eine weitere saubere Lösung, welche den zusätzlichen +9V-Draht zum Tower einspart:

Das mitgelieferte serielle Kabel ist nicht voll belegt. Die Pins 1 (DCD), 6 (DSR) und 9 (RI) sind frei, allesamt Eingänge zum PC und können gefahrlos mit +9V beaufschlagt werden. Bringen Sie die kleine Schaltung im PC unter, löten Sie an einen der Pins 1, 6 oder 9 die +9V an (notfalls über einen Adapter, wenn Sie auf ATX-Boards nicht so gerne herumlöten), machen Sie ein neues Kabel (s. VD 04/2001 S. 10), bei dem auch der gewählte Pin 1:1 verdrahtet ist und verbinden den gewählten Pin auf der Tower-Platine mit dem +9V-Anschluß.

Als 7809 ist jeder beliebige Typ verwendbar. Ich habe den L7809CV von ST im Gehäuse TO-220 verwendet, der 1A liefern kann. Der Tower benötigt nur 10mA, eine Kühlung ist daher nicht erforderlich. Als Kondensatoren empfehlen sich Tantal-Elkos mit mindestens 4,7µF.

Fragen bitte an rolf@rolf-schoene.de

*Rolf Schöne*

Betreff: Letztes Treffen der Rhein-Neckar-Gruppe
Hallo Leute, der erste Mittwoch ist rum 8-)

Themen waren:
Schrittmotoransteuerung (Stepper Motor)
Bedienung aus Bigforth
Rampenprogrammierung usw.

Hardware zu demselben Zweck von Conrad
Ewald hat dort das Modell SMC-1500 erstanden,
eine Bipolare Schrittmotor-Steuerkarte.
Hersteller: EMIS Zur Drehscheibe 4
92637 Weiden i.d.Opf.
Modellansteuerung für Spielebedienung
Das Modell aus dem Amendt-Metallbaukasten konstruiert
Mitgliederwerbung in unserer Umgebung

PS : Hat noch jemand eine Schrittmotorkarte w.o.
SMC-800 oder SMC-1500 zu verkaufen ?

Tschüss aus Eberbach / Odenwald

*Thomas Prinz*

*Zwei weitere Hinweise von R.Hempel zum pbForth:*

With the release of pbForth 2.1.2 last week, I've finally got things stable enough to warrant adding some really neat tutorials.

This week, thanks to Darin Johnson, I am happy to announce that you can write new words in assembler right on the RCX! Darin has been kind enough to allow me to distribute the source code for his H8/300 assembler after doing some minor rework to allow it to work with pbForth.

See the new tutorial at:

<http://www.hempeldesigngroup.com/lego/pbForth/scripts/howtoRCXAssembler.html>

The new 2.1.4 Version of pbForth has a minor fix that affected programmers using the MARKER word. I recommend you get the new 2.1.2 GUI while you're at it because it is less agressive in stripping comments when uploading scripts. This was causing some problems for certain cases with the comment character embedded in words.

Here's where to get the new files:

<http://www.hempeldesigngroup.com/lego/pbForth/download.htm>

This week, I've added a new tutorial on playing music on the RCX. Near the end of the article, there's a description of how to hook into the 1 msec timer tick of the RCX - it's a perfect way to make a simple multi-tasker!

Thanks go out to Guido Truffelli of the Italian LEGO Users Group for providing the motivation and sample data for the song player.

See the new tutorial at:

<http://www.hempeldesigngroup.com/lego/pbForth/scripts/howtoRCXMusic.html>

*Ralph Hempel*

***Termine – Termine – Termine – Termine***

Die 18. EuroForth-Konferenz findet vom 6.9.2002-8.9.2002 in Wien statt.

Wichtige Daten:

| | |
|---|---|
| 27.6.: | Termin für draft papers (refereed stream) |
| 15.7.: | Termin für abstracts (non-refereed stream) |
| 1.8.: | Benachrichtigung über die Annahme des Papers |
| 6.8.: | Termin für Hotel-Reservierungen |
| 19.8.: | Termin für die endgültige Version des Papers |
| 6.9.: | Konferenz |

Näheres siehe:
<http://www.complang.tuwien.ac.at/anton/euroforth2002/>.

*M. Anton Ertl*

Hallo Friederich,

Microcontroller-Ecke
Das Mitsubishi Board ist erstmal bei Dirk Brühl gelandet, er wird sich um eine Portierung bemühen.
Da es noch kein Forth dafür gibt und sich bisher auch niemand gemeldet hat, ist das erstmal der beste Weg.

*Thomas Prinz*

# Neues aus der FIG Silicon Valley

Lieber Fred, Friederich und Chris,
der Anhang ist für Euch alle gedacht, wozu auch immer er wert ist. Chris, ich glaube, daß es ausreichend und ökonomischer wäre, wenn Du mir nur eine Kopie der Forthwrite schickst. Ich würde sie in der SVFIG Bibliothek ablegen. Bei meinem geringen technischen Computerverständnis, habe ich meine eigene Kopie nicht verdient. Daneben ist es besser für mich, meine "Berichte" in der feinen deutschen Übersetzung von Dr. Beierlein zu sehen (Thanks for compliments, Henry. T. Beierlein), anstatt im ursprünglichen kalifornischen Englisch.
Die Besten Wünsche für Euch alle,

*Henry*

Hallo meine Freunde jenseits des Großen Teiches!
Die Vierte Dimension erreichte mich am 22. Mai, vier Tage nach dem SVFIG-Treffen des letzten Monats. Ich habe mit Freude die fröhlichen Berichte über die deutsche Forth Konferenz in Garmisch gelesen, und es wärmt mir das Herz zu wissen, daß die nächsten beiden Konferenzen - in 2003 und 2004 - praktisch schon gesichert sind. Das Wissen, daß seine Erfindung viele Leute zu einer dauerhaften Freundschaft und Verbundenheit bewegt hat, sollte auch Chuck Moore's Herz erwärmen. Ja, Dr. Behringer, die Verbundenheit ist wichtiger als Forth selbst, aber, wie wir sehen, Forth wird noch einige Zeit diese Welt begleiten.
Nun zu meinem Bericht... Ich wünschte, ich wüßte und verstünde mehr von der Basic Briefmarke und Hans Eckes' Forth Briefmarke, damit ich einige intelligente Bemerkungen über Al Mitchell's Präsentation machen könnte, die das Interesse von knapp 20 Teilnehmern den ganzen Tag lang fesselte.
Ihr wißt, Al hat sein eigenes AMRBASIC in Gforth geschrieben (das geistige Produkt von Ertl, Paysan und Wilke, wenn ich mich richtig erinnere). Es läuft unter Linux, da Al seit einigen Jahren Windows ablehnt. Mittels Subroutine Threading und neuen 8051-kompatiblen Chips ist Al in der Lage, sein BASIC ca. 500 mal schneller laufen zu lassen als irgendeine der gegenwärtig verfügbaren Basic Briefmarken. Sein AMRBASIC ist weitgehend interpretativ und bietet Erweiterbarkeit wie das Forth, auf welchem es läuft. Er hofft Nutzer über das Medium BASIC mit Forth bekannt zu machen, sobald er das System nach Windows portiert hat. Al's Website (www.amresearch.com) mag zur Zeit nicht viel zu diesem Thema enthalten; aber einige Ankündigungen sind bald zu erwarten, da das System nahezu komplett ist.
Da ich Englisch besser verstehe als Computerwissenschaften, möchte ich einige von AL's Kommentaren zitieren, die mir besonders interessant erschienen. "Forth findet Anklang bei den Leuten, die ein Problem wirklich verstehen." "Forth ist das leistungsfähigste Werkzeug, welches zur Maschinensteuerung möglich ist." "Leute, die Forth benutzen, leben außerhalb der ANSI (American National Standards Institute) Schachtel."
"Jede Applikation ist ein Dialekt von Forth."
Der letzte Gedanke erinnert mich an meinen Glauben daran, daß die besten Produkte einzelnen Gehirnen zu entspringen scheinen, und nicht aus gemeinsamen Anstrengungen entstehen. Aber es bleibt mir die Frage, ob die Welt nicht besser geworden wäre, wenn Leibniz und Newton zusammengearbeitet hätten, falls Aiken Zuse gekannt hätte, oder gar falls Al Mitchell und Hans Eckes sich ab und zu bei einer Tasse Kaffe treffen könnten.
Das ist es für heute. Wie Ihr wohl bemerkt habt, war Dr. Ting abwesend, vielleicht entgegen seinen besseren Wünschen. Aber da Cogswell College das Datum unseres Treffens auf den gleichen Samstag verschoben hatte, zu dem Dr. Ting's Tochter heiratete (wie ich hörte), gewann die Familie die Oberhand über den 'Verein'.

Cheers, *Henry*

Gentlemen,
der Anhang geht wieder an Euch alle drei. Wie üblich, um so kürzer ich es machen will, um so länger wird es.
Ich weiß, daß der Einsendeschluß für die VD jeweils in der dritten Woche des Juni, September, Dezember und März liegt. Ich habe den Juni-Termin nicht geschafft, Friederich. Ich möchte Chris' Termine für die Forthwrite wissen, und natürlich sind jegliche Kommentare von Euch allen willkommen, die das Schreiben und die Veröffentlichung erleichtern. Soll ich weiterhin versuchen 12 eigenständige Berichte im Jahr verfassen, vorausgesetzt, daß ich kein Treffen verpasse? Oder soll ich lieber versuchen, vier mal im Jahr kürzere Berichte zu schreiben? Sollte ich bei den Tatsachen bleiben oder lieber bei den Meinungnen (insbesondere meinen eigenen)?
Ehrlich gesagt, ist es schwierig für mich, meine Arbeitsweise zu ändern. Ich möchte weniger Zeit mit Schreiben verbringen und manchmal (falls es Interessenkonflikte gibt) auch weniger Zeit bei den Treffen. Aber ich habe nicht vor jetzt aufzuhören, jedenfalls nicht, solange ich der Forth-Gemeinschaft nützen kann. Bitte kritisiert mich, falls nötig, und kürzt heraus, was nicht zum Charakter Eurer Zeitschrift paßt. Wie immer, für die ganze Arbeit die Thomas Beierlein tut, sagt ihm mein Bewunderung und leitet seine Kommentare an mich weiter.

Cheers, *Henry*

Lieber Henry,
Deine Berichte sind für uns alle interessant und wertvoll. Bitte schreibe uns so, wie es Dir am leichtesten fällt, Deine Arbeit für die Forthwrite und die VD zu organisieren!
Uns interessieren hier die „Tatsachen“ ebenso wie die Meinungen und Stimmungen, insbesondere Deine Meinung und Sicht auf die Dinge, die Du bei Euren Treffen beobachtest.
Idealerweise bekämen wir die vielen interessanten Vorträge, die Du bei Euren Treffen hören und sehen kannst, hier zumindest in Papierform zum Nachdruck zur Verfügung gestellt Das wird aber auch in absehbarer Zukunft nicht zu leisten sein. Da reicht es uns schon zu lesen, was da alles in Eurem College vorgestellt wurde. Wenn Du uns zusätzlich schreibst, wie diese Dinge auf Dich gewirkt haben, welche Eindrücke Du gewonnen hast, was Dir die einzelnen Vorträge gegeben haben, dann meine ich, daß wir zumindest einen guten Überblick darüber vermittelt bekommen, was sich auf der anderen Seite des großen Teiches tut.
In diesem Sinne bitte ich Dich im Namen unserer Leser: mache weiter, so wie es Dir am angenehmsten ist.
Glückauf

*Fritz*

Grüße aus Nord Kalifornien!
Am 22. Juni traf sich die SVFIG und feierte die Ankunft eines neuen sonnigen Sommers, zumindest meteorologisch gesprochen. Die Schlagzeile am nächsten Tag im San Jose Mercury News, der führenden Lokalzeitung, las sich wie folgt: "High-tech slowdown may be worsening," ("Die Verlangsamung der High-Tech kann sich noch verschlimmern") und weiter, "Silicon Valley's high-tech hangover isn't over. In fact, it may be getting worse." ("Silicon Valley's High-Tech Überhang ist nicht vorbei. In Wahrheit kann es noch schlimmer werden.") Nach Angaben der Zeitung wuchs die Arbeitslosigkeit im Santa Clara County von 1,7 % im Jahr 2000 auf 7 % in 2001.
Ich vermute, daß die typischen SVFIG Mitglieder aus dieser Statistik herausfallen. In den vergangenen Jahren hat die Teilnahme an unseren Treffen keine wesentlichen Schwankungen aufgewiesen. Ebenso gab es keine unübliche Zahl von Nachfragen nach Jobs. Es wäre interessant, einige statistische Angaben über die "Forther" zu sammeln, um zu sehen wie und warum sie sich vom lokalen Durchschnitt unterscheiden und ob es Ähnlichkeiten unter ihnen im globalen Maßstab gibt.
Dr. Ting war ganz sicher in letzter Zeit beschäftigt, hauptsächlich mit Forth in Taiwan. Sein Bericht füllte die beiden Stunden am Morgen ohne irgendeine Pause. Zuerst ging es um seine P8 und P16 Prototypen Chips, die in Taiwan gefertigt wurden, und wie sie unter eForth arbeiten. Danach berichtete er, wie der Nintendo Game Boy, mit seinem ARM7 Mikroprozessor, (siehe www.gameshark.com) als eine sehr gute Platform für Forth dient und eine portable Technologie für die Erzeugung chinesischer Zeichen bietet.
Obwohl Win32Forth auf diesem System arbeiten würde, sagt Ting, daß sich Win32Forth als zu kompliziert für ihn erwiesen hat und daß er niemals ein gutes Handbuch dafür schreiben konnte. Daher entwickelte er F#, eine 32-Bit, Protected Mode, Subroutine-gefädelte eForth-Implementation mit einem Windows Interface. Mittels F# hilft er bei der Feinabstimmung oder ästhetischen Korrektur der 70.000 chinesischen Zeichen, die von einem taiwanesischem Programm, welches unter Windows XP läuft, erzeugt werden.
Unglücklicherweise mußte Ting seine Lektion ohne eine Demonstration beenden, da kein Windows-System zum SVFIG-Treffen verfügbar war. Nach der Mittagspause (und das war die einzige Unterbrechung bis zum Schluß um 4 Uhr Nachmittags), berichtete ein junger (nicht ganz Vierzig) Veranstaltungsneuling, Scott ... (ich habe seinen Nachnamen nicht verstanden), über seine 20-jährige Arbeitserfahrung, beginnend mit Forth (MVP, F83) für Maschinensteuerungen, und endend - aber noch nicht fertiggestellt - mit Forth (in TCL geschrieben!) in seinen Versuchen Web-präsentierbare automatisierte Fehler-Code-Sucher zur Störungssuche in Ladder-Logic-Systemen (Kontaktplandarstellung), wie sie z.B. in SPS (Speicher Programmierbare Steuerungen) verwendet werden, zu erstellen.
Er möchte sein Programm FRED nennen (für Forth-based Relay Engine for Diagnostics). (In Anbetracht dessen, werde ich meinen Satz am Anfang dieses Paragraphen nicht kürzen!)
Es tat gut, den früheren (und letzten?) Präsidenten der FIG, Skip Carter, zu sehen. Kann es sein, daß die abstürzende Ökonomie ihm wieder etwas Zeit für Forth gegeben hat? Skip schloß unser Treffen mit einer Demonstration eines brandneuen PDA von Sharp in Japan, dem "Sharp PersonalMobile Tool" SL-5000, der weniger als 600 Dollar in den USA kostet. Da es sofort mit Linux arbeiten kann und der GCC Cross-Compiler von Sharp frei verfügbar ist, hat Skip keine Zeit verschwendet und Anton Ertel's Gforth auf darauf portiert. Unter Nutzung eines drahtlosen Browsers (ich glaube von www.caphnet.com), welcher auf dem PDA läuft, und einem Mobiltelefon kann Skip den SL-500 mit einer drahtlosen Netzwerkkarte für den Internetzugriffe von überall her nutzen. Mehr Details gibt es auf Skip's Website, www.taygeta.com.
Zufällig sagte Skip, daß Mobiltelefone in Japan als PDA benutzt werden können. Ich frage mich, wie lange es dauert, bis die Mobiltelefone den Schulranzen ersetzen, den die Kinder mit zur Schule nehmen? Oder machen Sie es unnötig überhaupt zur Schule zu gehen? Bisher jedoch haben die Mobiltelefone keine Hilfe bei der Bekämpfung der gewaltigen Waldbrände in den USA geleistet, aber es würde mich nicht überraschen, wenn für sie mehr Geld ausgegeben wird, als für die Bekämpfung von Naturkatastrophen.
Carpe diem! *Henry*

# MicroCore

**an Open-Source, Scalable,**

**Dual-Stack, Harvard Processor**

**Synthesisable VHDL for FPGAs**

**Klaus.Schleisiek@hamburg.de**
**www.microcore.org**

**Zusammenfassung**

Forth hat den "Compiler-aided-Programmer" hervorgebracht. Ich denke, die Zeit ist reif, dieses Konzept auf die Hardware weiterzuführen: Zum "Core-aided-Programmer".

Wenn man einen einfachen, erweiterbaren, in einem FPGA realisierten Prozessorkern als Basis eines Systems benutzt, so wird der Programmierer endlich von den Zwängen einer statischen Prozessorarchitektur befreit – sei diese nun CISC, RISC, WISC, FRISC oder sonstwie. Kein "Drumherumprogrammieren" um bekannte Hardwarefehler mehr! Man hat die Wahl, ob eine bestimmte Funktion besser in Software oder in Hardware realisiert werden sollte: Es kann einfach die am wenigsten komplexe, effizienteste Lösung realisiert werden, in einem konkreten Projekt.

Natürlich ist die Nutzung von FPGAs mit einem Einbruch der "MIPS Zähler" verbunden. Aber die Nutzung eines FPGAs erlaubt es, zeitkritische und unter Umständen äußerst komplexe Funktionen in Hardware in genau der Weise zu realisieren, wie es die Anwendung erfordert, so dass der Prozessor sich um eine entsprechende zeitraubende "innere Schleife" gar nicht zu

kümmern braucht.

Die FPGA-Herangehensweise macht den Anwender auch unabhängig von Bauteilabkündigungen, die ein Dauerproblem der Elektronikindustrie bei sicherheitskritischen Anwendungen sind. Und schließlich: Wenn der Prozessorkern im FPGA steckt, wird eines der Hochsprachenparadigmen hinfällig: Nämlich das der - in Wirklichkeit nur erhofften - Code-Portabilität. Wenn ich meinen eigenen Instruktionssatz realisieren kann, dann gibt es keine Notwendigkeit mehr, den Code auf eine andere Architektur portieren zu müssen, so dass der einzige Grund, sich eines konventionellen Programmierstils zu befleißigen darin besteht, dass Programme von anderen Programmierern wartbar sein müssen.

Übrig bleibt nur noch, im Hinblick auf eine spezifische FPGA-Familie die Hardware in einer herstellerunabhängigen Beschreibungssprache zu beschreiben. MicroCore ist in VHDL realisiert und läuft auf dem MTI Simulator. Der identische Quellcode erzeugt identische Ergebnisse für Xilinx und Altera FPGAs unter Nutzung des Synplify oder Leonardo Synthesizers.

MicroCore ist nicht darauf beschränkt, in Forth programmiert zu werden – es basiert aber auf der virtuellen Forthmaschine und spricht Forth als "Assembler". Die Unterstützung lokaler Variablen durch relative Adressierung in den Return-Stack ist einfach und scheint alles zu sein, was nötig ist, aus MicroCore auch eine gute "C-Maschine" zu machen. Ihre Brauchbarkeit für Java wird noch untersucht.

*Klaus Schleisiek*

*Der Beitrag ist eine Zusammenfassung des Autors zu seinem in Garmisch-Partenkirchen gehaltenen Vortrag anläßlich der Jahrestagung der Forthgesellschaft e.V.:*

*fep*

**...weitere, merkwürdige "Vögel"**

**VMS Airlines:**
Megacarrier mit weltweiter Ju52-Flotte. Passagiere streiken zur Zeit gegen Umstellung auf moderneres Fluggerät. Spezialisiert auf Formationsflug in kleinen Gruppen mit spektakulären Crashmöglichkeiten.

**NT Airlines:**
Alle gehen auf die Startbahn, sagen im Chor das Passwort und bilden die Umrisse eines Flugzeugs. Dann setzen sich alle auf den Boden und geben Geräusche von sich, als würden sie wirklich fliegen.

***...weiter auf Seite 16***

# RCX am Draht

**Adolf Krüger**
**Michael Kalus**

Lieber Martin, neulich hast du gefragt, ob es ginge, die serielle Schnittstelle des RCX 1.0 als Kabel herauszuführen und mit dem PC zu verbinden. Hier unser Vorschlag.

**Klar** ist uns wieder geworden, daß eine gute und einfache Verbindung mit rein passiven Bauteilen nicht möglich ist: Die RS232 verwendet inverse Logik, aktive Bits sind LOW, auch am H8. Auch muß eine Pegelumsetzung von 0/+5V auf bis zu -12/+12V gemacht werden – so weit also das klassische Problem. In Frage kommt die Verbindung mittels Optokoppler (TLP621-2) oder mit Hilfe eines seriellen Schnittstellen Bausteins (MAX232).

Bei der Suche nach geeigneten Anschlußstellen fanden wir heraus, daß der IR-Empfänger TSOP1138 an seinem Datenausgang höchstens 3,3V verträgt. Dieser Datenausgang muß also weggeschaltet werden, will man über einen Draht an Pin2 des Prozessors ein Signal einspeisen.

Sodann sollten die Stellen an der Platine des RCX robust genug sein, um daran herum zu löten, ohne den RCX zu zerstören. Das Ganze soll möglichst in das RCX-Gehäuse hineinpassen - und darin ist wirklich wenig Platz. Auch sollte der Baustein wie gewohnt benutzt werden können, wenn das serielle Kabel wieder abgezogen ist. Als Verbindung zum PC sollten die LEGO-Kabel genommen werden können.

So kamen wir darauf, eine Stereo-Klinkenbuchse mit 2 Schaltern zu verwenden und den Optokoppler zu nehmen.

**Auf der Suche** nach geeigneten Stellen auf der Platine des RCX um sich einzuklinken, fahndeten wir nach gut zugänglichen Durchkontaktierungen, an denen angelötet werden kann, damit die Multilayerplatine nicht beschädigt wird - einfache Pads halten Zug vom Draht bei Vibrationen oder Sturz des Klotzes nicht aus! Und es dürfen keine Eingriffe in irgendwelche Leiterbahnen erfolgen.

Die folgenden Stellen sind geeignet: Direkt unter der Lötstelle für den Out-Pin des TSOP11 ist die Platine durchkontaktiert, da geht's zum Prozessor H8 und seinen RxD-Eingang (Pin2), die weitere Leiterbahn liegt ganz unter dem H8 - nicht zu erreichen. Der Daten-Pin des TSOP11 wurde also abgelötet und etwas hochgebogen, die Durchkontaktierung **A** bot der roten Litze Halt. Der Daten-Pin des IR-Empfängers war Lötstelle **B**. Dieses twisted pair bildete die RxD-Schleife. Praktischerweise kann gleich daneben an **E = Gnd** die blaue Litze angeschlossen werden. Der TxD-Ausgang des H8 (Pin1) zieht ebenfalls unter dem Chip durch, wechselt dann die Platinenseite für ein kurzes Stück und zieht weiter auf der Unterseite gleich neben dem H8 zu einem Widerstand R=10K (SMD 103) [Basiswiderstand zum Transistors T1, in welchem TxD mit der Blitzfrequenz 38KHz gemischt wird; diese kommt vom Timer output Pin30 des H8 dazu]. Die Stelle ist gut zu sehen, liegt frei, ist durchkontaktiert **C**. (Ob der SMD103-Widerstand abgelötet und als

R=10K konventioneller Widerstand in die Leitung zur Klinkenbuchse hin integriert wird, ist optional. Damit kann bestimmt werden, ob der RCX neben der Kabelverbindung auch IR senden soll oder nicht. (Siehe Bild "Einbau").

Es zogen so nun 5 Leitungen von der Platine zur Klinkenbuchse: Ein Paar um den IR-Empfängers abzuschalten und statt dessen das PC-Signals einzuspeisen, das andere Paar, um den IR-Sender still zu legen und das H8-Signal heraus zu führen und dann noch eine Leitung für Masse. Die Fotos zeigen diese Stellen auf der Platine des RCX.

**Der Optokoppler** TLP621-2 war dann bald verdrahtet. Als +12V Quelle diente PC-seitig das Signal DTR, die -12V wurden aus dem TxD(PC)-Signal generiert mittels einer Gleichrichterschaltung. Das hat zwar den Nachteil, daß diese Spannungsquelle leer laufen kann, wenn lange Zeit nur Nullen geschickt werden – aber kann man damit als Forth-Bastler leben?

Das pbForth war geladen worden, der Turm stand im Dauerbetrieb bereit, Forth war mit Hypertext zu bedienen und gab brav sein OK - via IR. Doch das Terminal im Laptop, der per Kabel an den RCX angeschlossen war, zeigte nichts! Selbst der IR-Turm empfing nun nichts mehr, obwohl wir doch die IR-Dioden auf dem RCX nicht abgekoppelt hatten, um über diesen Weg zu prüfen, ob der Kabelanschluß das gleiche zeigt wie die IR-Übertragung. Hm - also nachmessen, was eigentlich geschieht.

Das Signal TxD(RCX) wurde durch den Optokoppler zu sehr abgesenkt, die invertierende Verstärkerkette auf dem RCX schaltete nicht mehr aus und sandte Dauer-IR, der Spannungsregler an Bord wurde nett heiß - gut das wir die Lehrerversion mit integriertem Netzteil hatten, sonst hätten leere Batterien unserem Forscherdrang schon hier ein Ende gesetzt. Der Begrenzungswiderstand (**R4**) vor der IR-Diode im Optokoppler brauchte also doch einen höheren Wert als zunächst geschätzt - 1kΩ reichten dann aber aus. Das Signal lief wieder durch zum IR-Sender und, nach dem noch eine schlechte Lötstelle im Klinkenstecker beseitigt war, auch bis zum Optokoppler. Doch es kamen noch immer keine Zeichen auf den Laptop. Nachmessen - aber die Leitungen funktionierten. (Erste Zweifel am eigenen Verstand!?)

Und das Signal RxD(RCX)? Da kann man ja - eigentlich - nichts falsch machen. Die Ausgabe vom PC über Diode ist simpel - und doch fehlte die Reaktion am RCX - es gab keinen Pegel an RxD(RCX). Ah ja, der Datenausgang des IR-Empfängers ist ja abgeschaltet, die Leitung hat gar keine Spannung mehr, der interne Pullup des H8 reichte nicht aus. Nun gut, spendieren wir also einen 10K Pullup. Günstige Stellen auf dem RCX sind rar - aber eine kleine Bauform - ging unter des IR Empfängers Beinchen, da liegen ja auch die 5V und gleich daneben dir RxD-Leitung - paßt. Und nun ging auch das Signal durch zum Eingang. Aber der RCX antwortete trotzdem nicht - abgestöpselt hingegen zeigte er eine einwandfreie Funktion, nichts abgestürzt, Test-LOOPs liefen.

Es blieb dabei: Mit Stecker ging nichts, ohne ging alles. Aber unsere Hardware bestand alle Tests, war richtig und funktionierte. Software? Inzwischen war es spät - eigentlich schon wieder früh - geworden. Wir verfolgten sequentielle Zeichen mit dem Speicher-Oscilloskop, (ja, es ist ein "X") - aber wieso brachte der RCX nun den Testoutput

```
: x  10 0 DO I . LOOP ;
```

nicht raus? Erst mit dem Testwort

```
: y  10 0 DO KEY EMIT LOOP ;
```

dämmerte es uns so langsam - draußen auch.

**Na klar!** Das Übersprechen der IR-Sendedioden in den IR-Empfänger nebenan muß natürlich auch simuliert werden. Also haben wir eine extra Diode **(D2)** spendiert. Sie läßt das Signal von TxD(RCX) nach RxD(RCX) durch, so daß nun bei jeder Ausgabe auch der eigene Eingang ein Echo erhält und das Verhalten des RCX wieder so ist, als ob die Übertragung per IR erfolgt. Und nun lief alles wie gewünscht, endlich online –RCX am Draht.

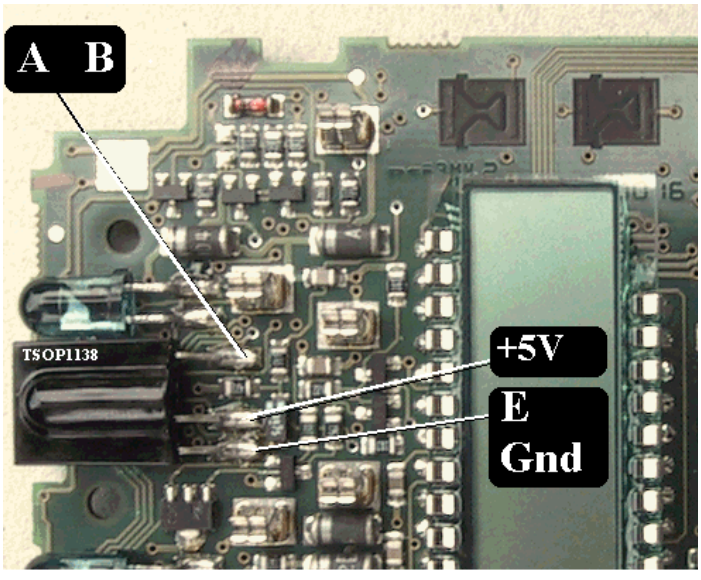
**Wer das nachbauen will** soll sich bitte gut informieren - *bevor* der RCX hin ist. Unsere Beiden leben noch. Aber wir hatten auch eine potentialfreie Lötstation. ALSO: JEDER BASTELT AUF EIGENE GEFAHR! KEINE GARANTIE FÜR NIX.

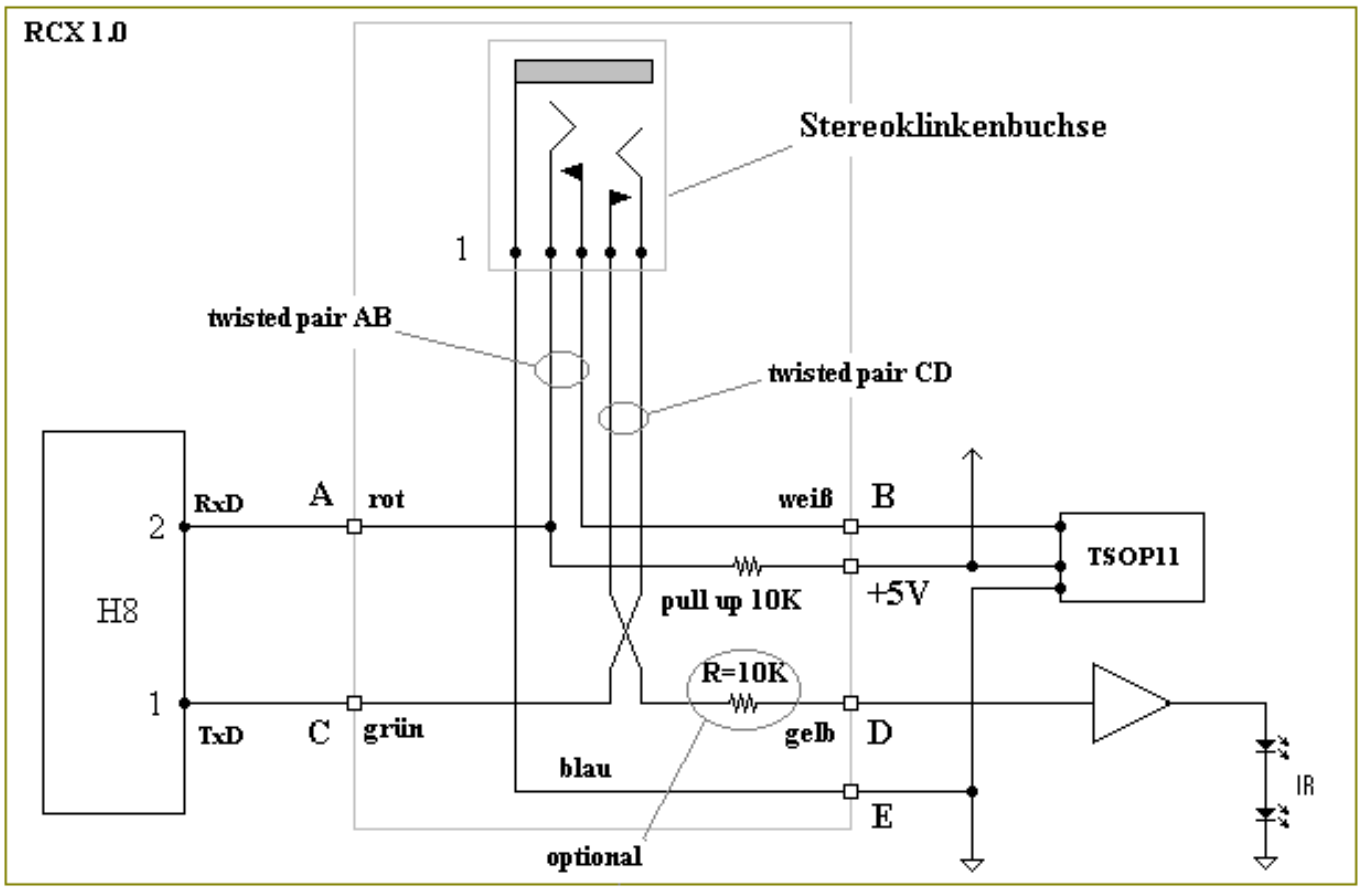
Wir sind nun gespannt zu erfahren, was Andere damit weiter machen werden, freuen uns auf Folgebeiträge und hoffen hilfreich gewesen zu sein. Mit besten Wünschen an alle Forthler, Euer Adolf und Michael.

PS: Eine Platine für diesen Adapter können wir auf Wunsch auch beisteuern.

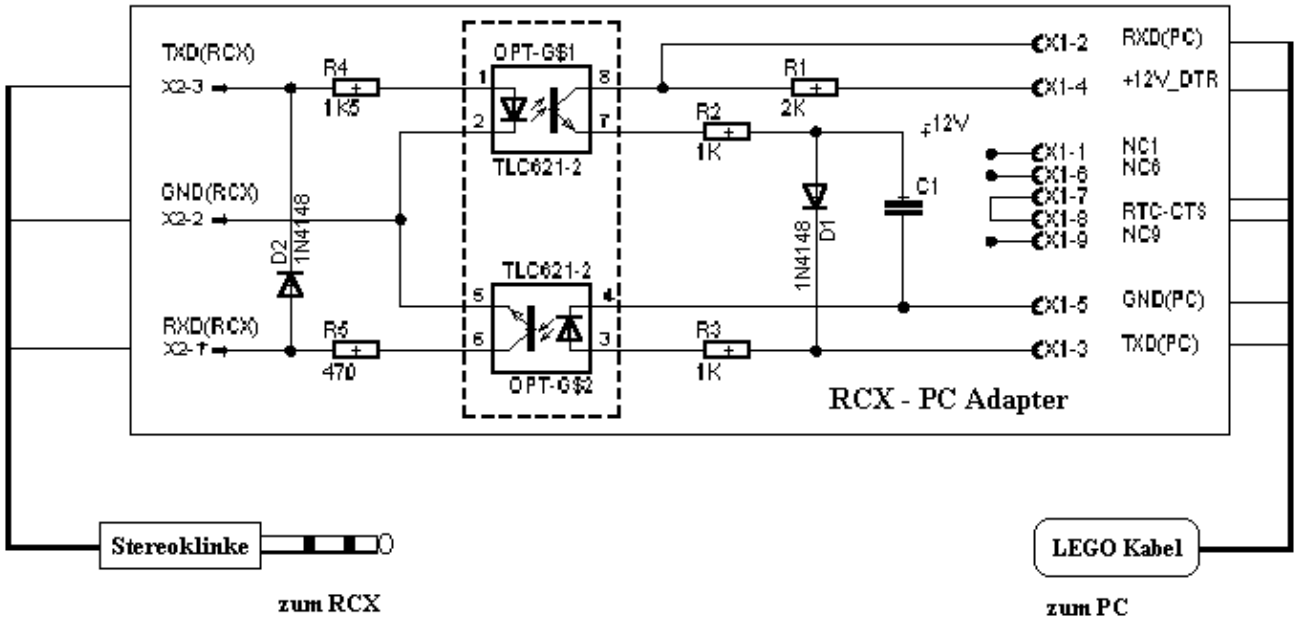
Anlagen:
Schaltung des RCX-PC Adapters,
Einbauplan für die Klinkenbuchse,
Bilder der RCX-Platine mit den Punkten A bis E.

*Adolf Krüger*
*Michael Kalus*






Anschluß der Stereoklinkenbuchse im RCX



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.

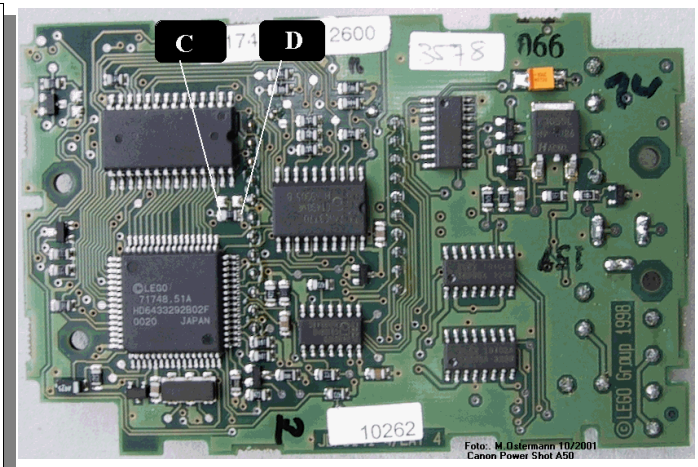



**Bild C-D** zu dem Beitrag RCX am Draht.

Die vier Darstellungen zu diesem Beitrag sind insgesamt rund 1,5 MByte groß. Bei Bedarf (bessere Bildqualität) können Sie die entsprechenden Dateien gerne per Mail anfordern. *fep*

***Auf den folgende Seiten finden Sie, liebe Leser, zwei sehr interessante Beiträge unserer niederländischen Forth-Freunde Albert Nijhoff und Willem Ouwerkerk, die uns zur Jahrestagung der Forthgesellschaft in Garmisch-Partenkirchen mit ihrer Anwesenheit eine große Freude bereitet haben. Die Beiträge waren, neben den vielen Diskussionen denen Beide sich am Rande der Tagung immer wieder gestellt haben, auch Beiträge der Tagung selbst.***
***Unsere Kollegen „machen praktisches Forth" wie wir es uns schöner eigentlich kaum vorstellen können. Vielen Dank für diese Beiträge!***

*Fep*

*Willem Ouwerkerk*

FIG UK

(Englische Forth-Gesellschaft)

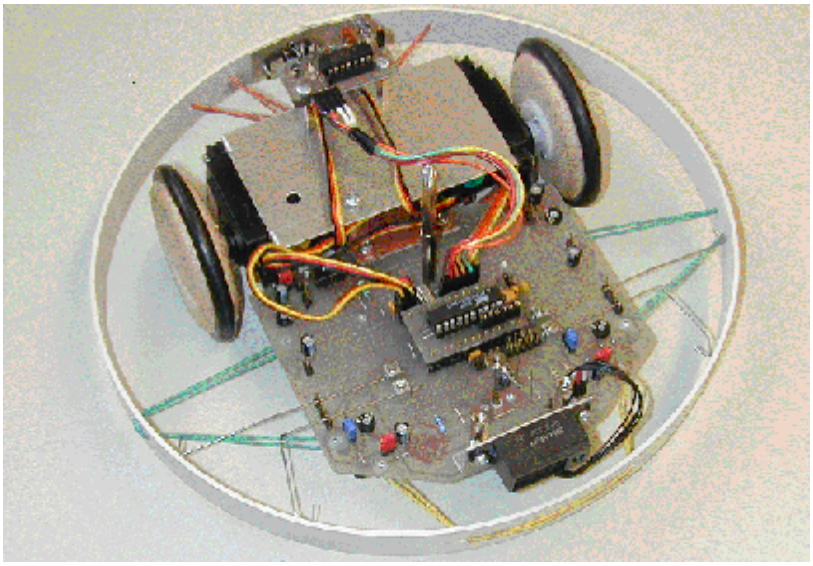
Treten Sie unserer Forth-Gruppe bei.
Verschaffen Sie sich Zugang zu unserer umfangreichen Bibliothek.
Sichern Sie sich alle zwei Monate
ein Heft unserer Vereinszeitschrift.
(Auch ältere Hefte erhältlich)
Suchen Sie unsere Webseite auf:
*www.users.zetnet.co.uk/aborigine/Forth.htm*
Lassen Sie sich unser 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**

# Die Arbeitsgruppe Ushi

**Willem Ouwerkerk**, HCC Forth-gebruikersgroep
Übersetzt von Fred Behringer



Ushi, Draufsicht

## Die Idee von Ushi

Die Idee kam mir auf den HCC-Tagen von 2000, wo ich viele Leute sah, die mit nicht (gut) funktionierenden Robotern beschäftigt waren. Viel Hardware und wenig Software. Da beschloß ich, einen einfachen autonomen Roboter zu beschreiben. Der Roboter sollte eine gehörige Anzahl verschiedener Aufgaben erfüllen können. Nachdruck sollte auf die Software gelegt werden. Der Roboter (namens Ushi) war vor allem dazu bestimmt, zum Programmieren anzuregen. Es wurden daher eine Anzahl von Spielideen entworfen, bei welchen die Software ausschlaggebend ist. Alle Teilnehmer sollten prinzipiell dieselbe Hardware zur Verfügung haben.

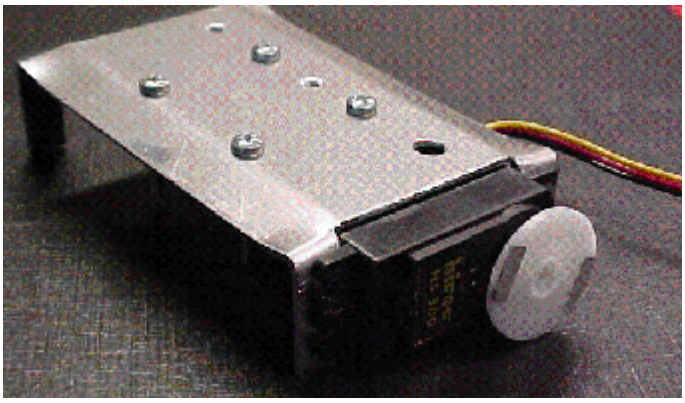
Es folgt eine vorläufige Liste von Spiel- und Wettstreit-Ideen.

a) Ein nicht vom Tisch fallender und Hindernissen ausweichender Roboter
b) Irrgarten
c) Fangemann
d) Leichtgewicht-Sumo-Ringer
e) Ein Roboter, der zeichnen oder/und schreiben kann
f) Ein Roboter, der einer Linie folgen kann

## Die Arbeitsgruppe Ushi

Nach dem Bau des ersten Mustermodells, nach einer Einleitung im Vijgeblaadje (unserer Vereinszeitschrift) und nach einer Vorführung auf einer unserer Treffen habe ich mit fünf anderen eine Arbeitgruppe ins Leben gerufen. Ziel der Arbeitsgruppe ist die Entwicklung von Ushi zu einem Selbstbauprojekt mit Platinen und Handbuch. Zur Wahl steht auch die Entwicklung eines vollständigen Selbstbaupakets. Inzwischen sind wir soweit, daß jedes Mitglied unserer Arbeitsgruppe eine oder mehrere funktionierende Ushis hat. Jetzt sind wir damit beschäftigt, genau zu formulieren, wie wir das Projekt abrunden können. Hierzu schauen wir uns den heutigen Stand der Hardware, der Software und der Durchführbarkeit der verschiedenen Spielideen an.

## Die Hardware von Ushi



Motorenbefestigung und ein Servo

Die Hardware von Ushi geht auf das ‚Igel-Arbeitsbuch' der HCC Forth-gebruikersgroep zurück. In diesem Arbeitsbuch werden allerlei kleinere Hardware-Projekte beschrieben, von denen einige in den Roboter eingeflossen sind. Ushis Aufbau ist rundum modular. An die 15 Ein- und Ausgänge eines Mikrocontrollers sind eine größere Anzahl von Modulen angeschlossen. So zum Beispiel ein RC5-Empfänger, um den Roboter über eine Fernseh-Fernbedienung steuern zu können, drei IR-Augen, die nachsehen, ob die Tischkante erreicht ist, fünf Barthaare, die fühlen, ob der Roboter sich gestoßen hat, ein den Abstand messender IR-Sensor und natürlich die zwei Servomotoren. Der Mikrocontroller wird auf eine lose Aufsteckplatine montiert. Dadurch kann er gegen einen anderen Mikrocontroller ausgetauscht werden. Dieser kann wiederum mit mehr I/O ausgerüstet sein, mit eingebauten A/D-Wandlern, mit mehr Programmspeicher usw. Auf der Hauptplatine ist auch zum späteren Einfügen neuer Hardware Platz. Zur Zeit besteht Ushi aus vier Platinen. Einige von uns sind dafür, Ushi auf einer Platine in ein Kästchen zu setzen, mit der Platine als Deckel (wir verfolgen das noch weiter). Neben dem IR-Abstandsmesser GP2D02 kann als Sensor-Alternative auch eine kleine Ultraschalleinheit verwendet werden.

## Software für Ushi

Beispielsroutinen für die Grundhardware wurden im Vijgeblaadje beschrieben. Einen einfachen autonomen und fernsteuerbaren Roboter findet man im Vijgeblaadje 28 veröffentlicht. Genau wie die Hardware ist auch die Software modular aufgebaut. Mit einfachen Befehlen ist der Roboter leicht zu steuern. Um ein paar zu nennen: VOORUIT (VORWÄRTS), ACHTERUIT (RÜCKWÄRTS), LINKS, RECHTS, STOP, SNELHEID (GESCHWINDIGKEIT), TAFELRAND? (TISCHKANTE?) usw. Zeitkritische Routinen zur Ansteuerung der Servomotoren und zum Decodieren der RC5-Befehle wurden in Maschinensprache geschrieben. Sie können aber über einfache Forth-Worte angesprochen werden.

Die Software wurde sowohl in 8051- als auch in AVR-ByteForth getestet. Es läßt sich auch eine Platine mit einem Forth-Interpreter aufsetzen, die sich besonders für einen vom Club organisierten Kurs eignet. Die Grundsoftware für dieses System wird in groben Zügen mit der 8051-ByteForth-Version übereinstimmen.

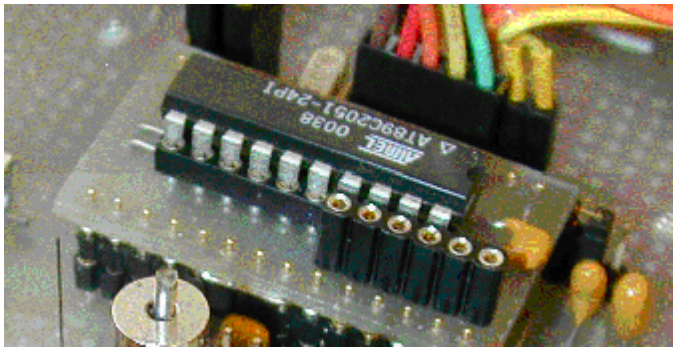
Die Grundsoftware für Ushi besteht aus:

a) Eingaberoutinen für alle Sensoren, die in der Grundausführung verwendet werden.
b) Eine Eingaberoutine für RC5-Infrarot-Code.
c) Ausgaberoutinen, die die Servomotoren als Antriebsmechanismus verwenden.
d) Ein Pseudozufallsgenerator.
e) Vier Timer.
f) Ein Roboter mit IR-Fernsteuerung, der Hindernissen ausweicht.
   Dazu der Code, der Ushi dabei hilft, nicht vom Tisch zu fallen.

```
\ Probier, von der Tischkante wegzukommen. Stop, wenn es nicht gelingt.

: ONTSNAP ( -- )                              \ ENTKOMMEN
   DRAAIWEG  5 100MS                          \ WEGDREHEN
   TAFELRAND? IF  DRAAIWEG  3 100MS  THEN     \ TISCHKANTE?
   BEGIN  TAFELRAND? WHILE  STOP  REPEAT ;
```

## (AVR) ByteForth



Aufsteckplatine für den AT89C2051 oder den AT90S2313

Ushi soll in erster Linie über ByteForth programmiert werden. ByteForth ist ein interaktiver Cross-Compiler mit eingebautem Assembler, Disassembler, Simulator und einem FLASH-Programmierer. Von ByteForth aus können all diese Funktionen beispielsweise mit Textbefehlen, wie sie in Forth gebräuchlich sind, verwendet werden.
Der Simulator ist (durch Verwendung von FLYER) völlig transparent implementiert, d.h., wenn wir DUP eintippen, wird DUP compiliert, ausgeführt und wieder entfernt. Man kann auch selbstgeschriebenen Code testen:

```
: 2*  DUP + ;

4 2* .
```

'Two-star' bringt Code für das von uns gewählte Ziel hervor. Sobald wir 2* eintippen, wird der Simulator aktiviert und der Code ausgeführt. Das Ergebnis wird wie gewöhnlich wieder auf den Stack zurückgeschrieben. Außer der speziellen CPU-Hardware können wir jeden Code interaktiv testen. Sobald die Software steht, kann der erzeugte Code beispielsweise über den ISP-Programmierer in das FLASH-ROM des Controllers gelegt und direkt auf dem Roboter getestet werden.
Für Forth-Programmierer ist das eigentlich nichts Neues. Sie sind es gewohnt, interaktiv zu entwickeln. Moderne Mikrocontroller können meistens über eine serielle Schnittstelle programmiert werden. Wir können daher Anwendungen fast wie im interaktiven Forth Schritt für Schritt entwickeln und austesten.

## Dokumentation von Ushi

Der Roboter wurde in einer ersten Serie von Artikeln beschrieben. Auf diesen Artikeln basiert die Baubeschreibung von Ushi. Die Dokumentation für Ushi wird aus einer Baubeschreibung der diversen Hardware-Bestandteile, einer Beschreibung der Beispielssoftware und einer genauen Beschreibung der einzelnen Spielaufgaben bestehen.

In der Arbeitsgruppe sind wir hart am Überlegen und Diskutieren, wie wir diese Ideen verbessern können. Wir sind bestrebt, die Baubeschreibung im November in gedruckter Form zur Verfügung stellen zu können.

Da die Forth-gebruikersgroep eine Untergruppierung der HCC ist, sind wir auch immer auf den HCC-Tagen vertreten. Unsere Arbeitsgruppe wird da das Bauanleitungsbuch präsentieren und mit einer Anzahl von Spieldemonstrationen aufwarten.

*Willem Ouwerkerk*

**...noch ein "seltener Vogel"**

**Linux Airlines:**
Wir sammeln die Teile des Flugzeugs, die wöchentlich in einer Illustrierten als Bastelbogen erscheinen, schneiden sie schön zurecht, kleben sie alle zusammen und malen sie dann noch bunt an. Dann starten wir das Flugzeug und hoffen, daß es bei einer Ozeanüberquerung nicht ins Wasser fällt und die Pappteile sich dabei auflösen. Benutzen wir NFS, kann man auf den Ozean verzichten, um das Auflösen hervorzurufen. In diesem Fall entstehen auch keine Zeugen des Unglücks. Sitze und Passagiere lösen sich vorher in Nebel auf.

*...weiter Seite 34*

# TO - Ein Mechanismus mit vielen Möglichkeiten

Albert Nijhof, HCC Forth-Gebruikersgroep
(Übersetzt von Fred Behringer)

## 1. Einleitung

Forth ist eine Low-Level- und gleichzeitig eine High-Level-Programmiersprache. Datenworte legen eine Adresse auf den Stack, wohin man dann anschließend etwas schreibt oder woraus man etwas ausliest. Bei Low-Level-Code gibt einem diese Arbeitweise alle Freiheiten, die man sich wünschen kann. High-Level-Code stellt andere Anforderungen. Die Adressen der Daten sind dann von geringerer Bedeutung. Es geht um die Daten selbst. Adressen auf dem Stack sind dann lediglich ein vom System aufgezwungener Zwischenschritt und vor allen Dingen auch eine mögliche Fehlerquelle. Zudem machen die vielen Lese- und Schreibworte den Code fehleranfälliger und gleichzeitig schlechter lesbar.

Mit der Verwendung von VALUEs in Verbindung mit TO wurde ein kleiner Schritt in Richtung auf eine Lösung hin eingeleitet. VALUEs sind in ANS-Forth jedoch nur eine Randerscheinung, nicht mehr (lediglich den Locals ist TO auch bekannt), und der Standard bietet keine wirksame Methode, mehr Datenworte auf TO vorzubereiten oder neue Vorsetzworte ("Präfixe") zu definieren. Vor allem Strings flehen geradezu nach einer Möglichkeit der High-Level-Behandlung, denn der interaktive Umgang mit MOVE in Verbindung mit Adressen auf dem Stack läßt das wünschenswert erscheinen.

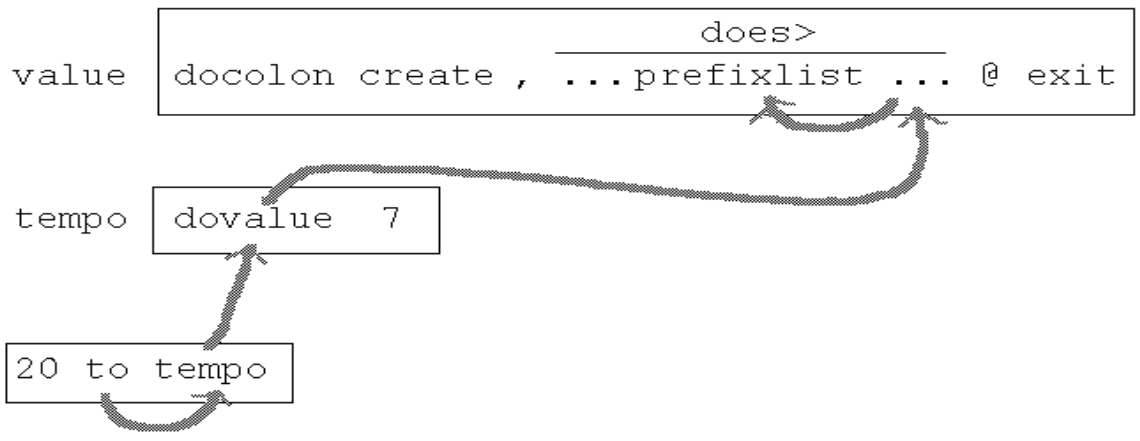
Die TO-Implementation, die ich Euch beschreiben möchte, verwende ich bereits seit 1990 (JinForth für den Atari ST). Die diesem Konzept zugrunde liegende Idee ist recht einfach und auf der Hand liegend. Sie liegt so sehr auf der Hand, daß sicher auch schon andere auf denselben Gedanken gekommen sind. Mir ist jedoch noch nichts darüber zu Ohren gekommen. Ich kann nur hoffen, daß auch für Euch noch etwas Neues dabei ist.

"Mein" TO ist fehlersicher und hat den "Nebeneffekt", daß der Programmierer auf ganz normale Forth-Art, ohne Patchen oder spezielle Syntax, für jede Anwendung unbeschränkt allerlei Präfixe für allerlei Worte (im Prinzip für alle Worte) definieren kann.

Die Aufgabe von störanfälligen STORE-Worten, wie ! 2! C! +! MOVE ON OFF ..., wird von den stärker fehlersicheren Vorsetzworten ("Präfixen") übernommen, die natürlich sowohl innerhalb als auch außerhalb der Definitionen funktionieren. Leseworte vom Typ @ werden überflüssig, da das Datenwort jetzt, da es sich um die Adresse nicht mehr zu kümmern braucht, die Leseaktion selbst ausführen kann.

Allerdings muß der TO-Mechanismus vom Forth-Erbauer vorbereitet werden, ein kleiner, aber fundamentaler Eingriff: Einleitend zu DOES> und ;CODE-Routine wird jedesmal eine eindeutig bestimmte Wortliste compiliert. An sich hat das auf den Arbeitsablauf von Forth keinerlei Einfluß. Das System wird halt nur um einige zig Zellen größer - im allgemeinen (es scheint ein Forth zu geben, das für eine neue Wortliste 64 Kilobytes benötigt).

## 2. Der Mechanismus



Wir benötigen die folgenden vier Zutaten:

(1) Ein Forth mit einem angepaßten DOES>
(2) Das Wort PREFIXLIST
(3) Das Wort PREFIX
(4) Einsicht in den Mechanismus

```
: VALUE create , does> @ ;
   7 value TEMPO
  20 to tempo
```

(1) Unmittelbar vor einer jeden Does-Routine compiliert DOES> eine eindeutig bestimmte Wortliste, die PREFIXLIST, in welcher die internen Aktionen definiert werden.

(2) Ein Wort zum Auffinden der Präfixliste (in einem indirekt gefädelten Forth):

```
: PREFIXLIST ( xt -- wid ) dup @ cell- @ swap ?doer ;
```

?DOER sieht nach, ob die CFA des Datenworts auf ein DOES> oder auf eine ;CODE-Routine zeigt.

```
' tempo prefixlist ( -- wid )
```

(3) Mit PREFIX werden globale Präfixe definiert:

```
: PREFIX ( «globaler name» «interner name» -- )
  prefix TO /to
```

(4) TO sieht bei TEMPO nach, findet die Präfixliste und sucht darin nach /TO.

## 3. Anwendungen

Bei allen Codebeispielen, hier und im folgenden, beachte man, daß ich von einem indirekt gefädelten Forth ausgehe!

Beispiel A ist trivial und kaum sinnvoll, gibt aber einen sehr guten Einblick in die Funktionsweise. Ich definiere Vorsetzworte ("Präfixe") für Variablen vom Typ BASE (Uservariablen). Man kann diese dann über GET und SET ansprechen, aber auch noch über @ und ! . Die beiden Methoden beißen sich nicht.
Ist die interne Aktion immediate? Dann wird sie vom globalen Vorsetzwort ausgeführt. Das Datenwort-Token liegt dabei auf dem Stack.

### Beispiel A

```
' base  prefixlist set-current

: /SET ( dataword-xt -- )  compile,  postpone ! ; immediate
: /GET ( dataword-xt -- )  compile,  postpone @ ; immediate

forth definitions
prefix SET /set
prefix GET /get

( get base wird base @  )

decimal
: HEX 16 set base ;
  hex get base decimal . [rtn] 16  ok
```

Man sieht, daß nur das Compilezeit-Verhalten von /GET und /SET codiert wird. GET und SET arbeiten jedoch auch interaktiv. Und wie das? Dafür sorgt FLYER (mehr darüber später).

**Beispiel B**

```
: VALUE ( x «name» -- ) create , does> @ ;
  prefix TO /to

0 value TEST
  ' test prefixlist set-current
  forget test

: /TO ( dataword-xt -- ) >body postpone literal postpone ! ; immediate
  forth definitions

( to test wird literal test-body !  )
```

In Beispiel C ist die interne Aktion nicht immediate: Das globale Vorsetzwort compiliert die interne Aktion und setzt die BODY-Adresse des Datenworts dahinter (Inline-Adresse).

**Beispiel C**

```
7 value #DAYS

  ' #days  prefixlist set-current

: /TO  ( x «inlinea» -- )  inline-address ! ;
: /+TO ( x «inlinea» -- )  inline-address +! ;

  forth definitions

prefix +TO /+to
prefix EXTRA /+to

( to #days wird /to inline #days-body  )

1 +to #days     #days . [rtn] 8  ok
12 extra #days  #days . [rtn] 20  ok

prefix NACH /to
0 nach #days    #days . [rtn] 0  ok
```

## 4. Does> anpassen

Jedes DOES> und ;CODE compiliert eine PREFIXLIST in die Zelle unmittelbar vor der Doer-Routine. Dafür müssen sie angepaßt werden. Die Abänderungen (Hinzufügungen) im Code sind unterstrichen:

```
: (;CODE) r> n + youngest ! ;
: DOES> postpone (;code) wordlist ,
        does-intro, ; immediate
: ;CODE postpone (;code) wordlist ,
        postpone [ assembler ; immediate
```

Das n in n + entspricht der Anzahl von Bytes, die durch wordlist , compiliert werden. Das Komma nach wordlist ist wahrscheinlich überflüssig, da WORDLIST selbst in den meisten Forth-Systemen die «wid» schon compiliert.

## 5. Doprefix

Mit PREFIX ( «global name» «internal name» -- ) werden Vorsetzworte compiliert.

| | Header | CFA | Body |
|---|---|---|---|
| Vorsetzworte: | global name | Doprefix | internal name (counted string) |

«global name» ist der Name des neuen Vorsetzwortes.
«internal name» wird als Counted String im Body des Vorsetzwortes abgelegt.

Die Adresse von DOPREFIX (DOES-Teil von PREFIX) kommt in die CFA des Vorsetzwortes. Doprefix geht zur Präfixliste des Datenwortes und sucht nach «internal name».

Die Funktionsweise von DOPREFIX:

(1) Geh zum Datenwort, finde dessen Präfixliste und suche darin nach der internen Aktion. Die Token des Datenwortes und der internen Aktion liegen jetzt auf dem Stack.

(2) Verwende FLYER (siehe folgenden Abschnitt), wodurch dann das interaktive Verhalten nicht mehr codiert zu werden braucht.

(3) Ist die interne Aktion immediate?
- NEIN: Leite die Standardbehandlung ein: COMPILE, >BODY , d.h., compiliere die interne Aktion und danach (inline) die Body-Adresse des Datenwortes.
- JA: Führe sie aus.

**Der Code** (mit Kommentar entsprechend dem Beispiel TO TEMPO ):

```
: PREFIX ( «name1» «name2» -- )
  create immediate
  bl parse
  string, align
  does> ( DOPREFIX )            ( TO-body )
  ' swap                        ( TEMPO-xt TO-body )
  over prefixlist               ( TEMPO-xt TO-body Wid )
  >r count r>                   ( TEMPO-xt "/TO" Wid )
  search-wordlist               ( TEMPO-xt /TO-xt? -1,1,0 )
  dup
  if flyer                      ( TEMPO-xt /TO-xt )
     0<                         \ Not immediate?
     if compile, >body , exit   \ Standardaktion
     then execute exit          \ Immediate
  then -32 throw ;              \ /TO not found.
```

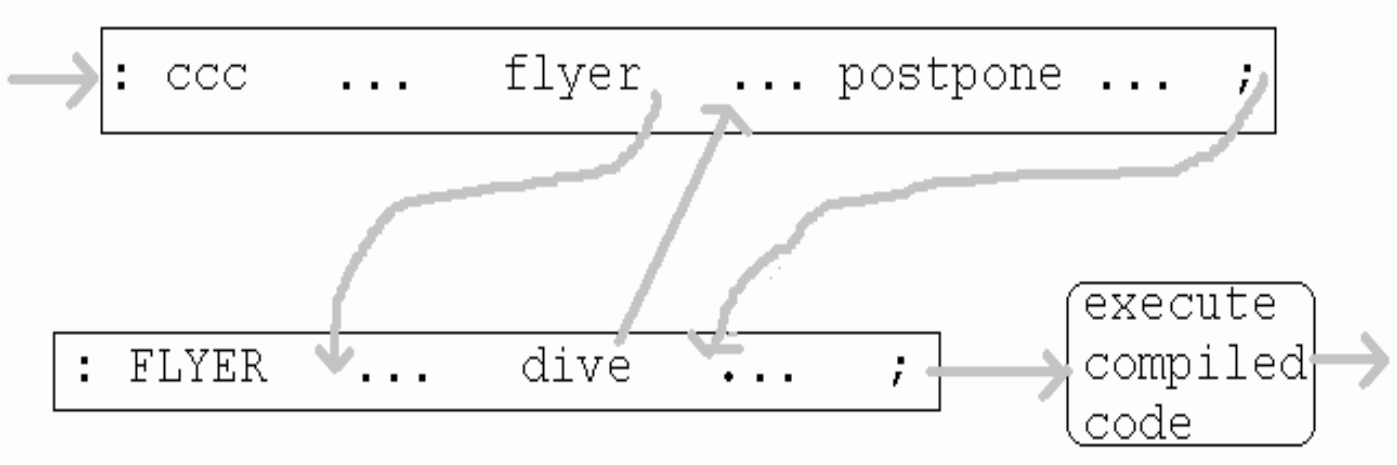
Eine immediate interne Aktion gibt uns große Flexibilität. Eine nicht immediate interne Aktion ist starr, ermöglicht aber oft eine effizientere Codierung in Assembler mit Inline-Adressen.

## 6. Flyer

In State-Smart-Worten sorgt FLYER dafür, daß man nur die Compilezeit-Aktion zu programmieren braucht, da das interaktive Verhalten damit auch schon bestimmt ist. Mit FLYER kann man also State-Smart-Worte auf einheitliche Weise behandeln.

```
: ." flyer postpone ." ; immediate
```

Angenommen, das urspüngliche ." funktioniert nur innerhalb von Definitionen. Dann funktioniert das neue ." mit dem darin enthaltenen FLYER interaktiv und in Definitionen.




```
: DIVE ( -- ) r> r> 2>r ; \ IP <--> Return address.

: FLYER ( -- )
  state @ if exit then
  here r> 2>r
  ] dive              \ Execute rest of caller and return.
  postpone exit
  postpone [
  r@ here - allot ;   \ Execute the newly compiled code.
```

Dieser vereinfachte "Mini-FLYER" genügt für DOPREFIX. Das FLYER, das ich selbst verwende, befördert HERE während des Compilierens in einen Ringpuffer, worin dem zu compilierenden Code und den Daten ein längeres Leben beschert bleibt. Das braucht man für S" und beispielsweise für Locals. Man vergleiche den gleich folgenden Code.

```
.... value FLYBUF
256  value FLYBUFLEN
0    value THERE
0    value FLYING? \ Flag

: HERE/THERE ( -- ) there
  flying? 0= dup to flying?
  if flybuflen over flybuf -
     u<         \ Overflow?
     if drop flybuf then
  then here dup to there
  - allot ;

: DIVE ( -- ) r> r> 2>r ;

: FLYER  ( -- ) state @
  if exit then flying? here/there
  if abort" Flyer nesting" then
  here r> 2>r ] !compilersecurity
  dive
  postpone exit postpone [
  here/there ?compilersecurity ;

( THERE is the alternative HERE in the circular flyer buffer. Keep 264 bytes free for
buffer overflow. )
```

### 7. Anwendung auf Strings

```
  : STRING ( maxlen «name» -- )
    create dup c,         \ maxlen
    0 c,                  \ actual length
    allot align           \ for the string itself
    does> char+ count ;
    10 string X
    ' x prefixlist set-current
    forget x

  : /TO ( a n «inlinea» -- )
    >r inline-address count r> umin swap 2dup c! char+ move ;
    code /+TO ( a n «inlinea» -- ) «ass» END-CODE
    code /INCR  (ch «inlinea» -- ) «ass» END-CODE

forth definitions
prefix INCR /incr
```

Diese internen Aktionen kann man zweckmäßiger und wahrscheinlich auch leichter in Assembler codieren. Je nach Wunsch können sie auch völlig fehlersicher gemacht werden.

```
16 string M
s" ANS"     to m   m type [rtn] ANS ok
char -    incr m   m type [rtn] ANS- ok
s" Forth"  +to m   m type [rtn] ANS-Forth ok
m        extra m   m type [rtn] ANS-ForthANS-For ok

  : SUBSTRING ( a n pos len -- a n ) >r over umin /string r> umin ;

m 4 5 substring type [rtn] Forth ok
here 0 to m   m type [rtn] ok
```

==========

# Die Seite für den Umsteiger

==========

# Ein Assembl(i)eraufruf (2/3)

**Julian Noble <jvn@virginia.edu>**

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

Zweiter 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, Bitmanipulation, Mikromini-Assembler, Zeichenumwandlung, Stringverarbeitung

**Umwandlung in Groß- oder Kleinbuchstaben**

Viele Sprachen enthalten eine Bibliotheksfunktion zur Umwandlung von Strings in lauter Groß- oder lauter Kleinbuchstaben, wobei Ziffern und Satzzeichen unberücksichtigt bleiben. In den neuen ANS-Forth-Standard-Empfehlungen [1] wird eigenartigerweise keine solche Routine verlangt, obwohl die meisten Forth-Systeme in ihrem Compile-Mechanismus ein Wort, das wie **UCASE** wirkt, enthalten.

Zunächst müssen wir uns über unsere Vorgehensweise einigen. In QuickBasic (QB) von Microsoft (R) werden Strings von N Zeichen Länge in N aufeinanderfolgenden Bytes im vorgegebenen Datensegment des Arbeitsspeichers abgelegt. Zum Aufruf dient ein 4-Byte-String-Descriptor, bei welchem die ersten zwei Bytes die Länge als vorzeichenbehaftete 16-Bit-Ganzzahl enthalten, und die zweiten zwei Bytes den Offset des Stringanfangs im Datensegment. Strings in QuickBasic können also bis zu 32 KB lang sein. C von Microsoft speichert Strings in zusammenhängenden Abschnitten von N+1 Bytes, wobei das (N+1)te Byte 0 enthält (Standard-Stringabschluß in C). Strings werden dort über die Adresse ihres ersten Bytes angesprochen.

Forth dagegen arbeitet normalerweise mit "counted Strings", die bis zu 255 Bytes lang sind und deren Länge (das "Count") im ersten Byte enthalten ist. Diese Unterschiede in den verschiedenen Sprachen stellen ein gewisses Problem dar, wenn man Unterprogramme zur Behandlung von Strings entwerfen möchte, da man das in Forth etwas anders machen muß als in QB oder C. Am einfachsten geht es, wenn man den Code in zwei Teile aufspaltet, in ein sprachabhängiges Kopfstück (Header) und einen allgemein verwendbaren Hauptteil. Wir wollen das mit Kopfstücken für die in Forth, QuickBasic und C geltenden Vereinbarungen zur Stringablage erläutern.

Wie sieht der Code für den Hauptteil aus? Der Entwurf wird klar, wenn wir das zunächst in High-Level-Forth schreiben [2].

```
: lcase?           ( char -- flag)     \ TRUE, wenn Kleinbuchstabe
   DUP [CHAR] a <  ( char f1)          \ TRUE, wenn char < "a"
   SWAP [CHAR] z > ( f1 f2)            \ TRUE, wenn char > "z"
   OR              ( not[flag] )       \ Flags vereinen
   INVERT ;                            \ Logische Negation

: UCASE            ( beg len)
   0 DO                          \ String von links nach rechts abarbeiten
      DUP C@       ( -- adr char)      \ Zeichen holen
      DUP lcase?   ( -- adr char flag)
      32 AND       ( -- adr char 32 bei lcase | 0 sonst)
      -                          \ 32 nur von Kleinbuchstaben abziehen
      OVER C!                    \ Durch abgeändertes Zeichen ersetzen
   CHAR+ LOOP                    \ Adresse um 1 erhöhen und an Schleifenanfang
   DROP ;                        \ Stack aufräumen
```

Wir gehen also den String Byte für Byte von Anfang bis Ende durch und prüfen, ob das Zeichen ein Kleinbuchstabe oder etwas anderes ist. Wenn Kleinbuchstabe, dann in Großbuchstabe umwandeln, sonst nichts unternehmen. Die Großbuchstaben haben einen ASCII-Code, der um **32d** kleiner ist als der des entsprechenden Kleinbuchstaben. Also erreichen wir das Umschalten von Klein- nach Großbuchstaben dadurch, daß wir **32d** vom Code des Ersteren abziehen. Man beachte, daß diese Programmiermethode bei den Worten **lcase?** und **UCASE** das Ergebnis nicht durch Entscheidung findet, sondern durch Berechnung. Ganz ohne Entscheidungen auszukommen, ist nicht immer praktizierbar [3], zu einem guten Stil gehört aber, daß man Verzweigungen aus dem Weg geht, wo es nur möglich ist.

Die Assemblerversion zu konstruieren, bereitet keine Schwierigkeiten. Man beginne mit **lcase?** und drücke alles direkt in Assemblerbefehlen aus:

```
CODE lcase?        ( char --- flag)
   POP BX          \ char - BL
   MOV AX, BX      \ Nach AL kopieren
   SUB AL, # 96    \ AL = char - 96
   CBW             \ Vorzeichen AL - AH = Flag1
   XCHG AX, BX     \ Vertausche Register, BH = Flag1
   SUB AL, # 123   \ AL = char - 123
   CBW             \ Vorzeichen AL - AH = Flag2
   OR AH, BH       \ AH = Flag1 oder Flag2
   XCHG AL, AH     \ AL = ~Flag
   NOT AL          \ AL = Flag
   CBW             \ Wandle 8-Bit- in 16-Bit-Flag um
   PUSH AX         \ Flag - TOS
NEXT END-CODE      \ CODE-Definition beenden
```

# Ein Assemblieraufruf

Wir probieren das aus:

```
CHAR A DUP . lcase? . 65 0 ok
CHAR a DUP . lcase? . 97 -1 ok
CHAR z DUP . lcase? . 122 -1 ok
CHAR & DUP . lcase? . 38 0 ok
```

(Anmerkung: **TRUE** - alle Bits auf **1** gesetzt - wird von "**.**" in den hier betrachteten Forth-Systemen als Ganzzahl **-1** interpretiert. ANS-Forth nimmt, verglichen mit anderen Sprachstandardisierungen, eine Sonderstellung ein: Es sieht Portierbarkeit zwischen der üblichen Zweierkomplement-Ganzzahlarithmetik und zwei anderen Darstellungsarten vor.)

Der vorstehende Test verlief zu unserer Zufriedenheit - wir können wirksam testen, ob ein gegebenes Zeichen ein Kleinbuchstabe ist. Als nächstes wollen wir **lcase?** und **UCASE** zu einer einzigen Routine zusammenführen. Dazu brauchen wir eine Schleifenkonstruktion. Die in **STIB** verwendete Schleife eignet sich hierfür gut, da die Schleife wieder eine vorbestimmte Anzahl von Malen durchlaufen wird. Wir brauchen wieder einen Kopfteil, der die Stringlänge in Bytes (den "count") in das Register **CX** legt, und die Adresse des ersten Stringbytes nach **BX**. Den Kopfteil wollen wir diesmal aber im Assembler-Unterprogramm als getrennten Programmabschnitt fassen, da wir diesen dann weiter unten in einer für eine andere Sprache als Forth geeigneten Form verwenden wollen.

In F-PC besteht der Kopfteil aus den folgenden Befehlen:

```
POP CX       \ Stringlänge nach CX
POP BX       \ Datenanfang nach BX
PUSH DI      \ Register DI (Index) aufbewahren
MOV DI, BX   \ Anfang-1 nach DI
```

Beim Aussprung stellen wir **DI** über **mov BX DI** wieder her. Das wird vom letzten Befehl vor

```
NEXT END-CODE
```

übernommen. Zum Vergleich: Ein für QuickBasic geeigneter Kopfteil würde wie folgt aussehen [4]:

```
PUSH BP          ; BP aufbewahren
MOV BP, SP       ; BP als Stack-Pointer verwenden
PUSH DI          ; Register DI aufbewahren
MOV BX, 6 [BP]   ; Adresse des String-Descriptors ins Register BX
                 ; Anmerkung: CX braucht nicht initialisiert zu werden
MOV CX, 0 [BX]   ; Stringlänge ins Register CX
ADD BX, 2        ; Offset zum Stringanfang nach BX
```

Und der entsprechende QB-Nachspann (für einen sauberen Aussprung) sieht wie folgt aus:

```
POP DI
POP BP           ; Register wiederherstellen
```

Das vollständige Programm in F-PC -Assembler liest sich dann wie folgt:

```
CODE UCASE           \ Anfang des Vorspanns
   POP CX            \ Stringlänge holen
   POP BX            \ Anfang holen
   PUSH DI           \ DI aufbewahren
   MOV DI, BX        \ Ende des Vorspanns, Anfang des Hauptteils
HERE:                \ Schleifenanfang
   INC DI            \ Auf nächstes Byte zeigen
   MOV BL, 0 [DI]    \ Byte holen
   MOV AX, # 96      \ Auf groß oder klein testen
   SUB AL, BL
   CBW
   XCHG AX, BX
   SUB AL, # 123
```

```
   CBW
   AND AH, BH          \ AH = FF|0
   AND AH, # 32        \ AH = 32, wenn Kleinbuchstabe, sonst 0
   SUB 0 [DI], AH      \ Buchstabe im String umwandeln
   LOOP HERE           \ Schleife wiederholen, wenn CX >= 0
                       \ Ende des Hauptteils, Anfang des Nachspanns
   POP DI              \ DI wiederherstellen
   NEXT                \ Ende des Nachspanns
END-CODE
```

Dieses Unterprogramm ist schwer zu lesen, selbst mit Einrückungen in den Kommentaren (gerade darum verwenden wir ja statt Assembler lieber Hochsprachen), es besteht aber aus denselben Teilen wie die entsprechende High-Level-Definition: Ein SET-UP-Abschnitt, der die Stringlänge und den Datenanfang hereinholt, ein Hauptteil, der den String in einer Schleife durch-**LOOP**t, ein Test, der feststellt, ob ein gegebenes Zeichen ein Kleinbuchstabe ist und ihn gegebenenfalls in einen Großbuchstaben umwandelt, und schließlich ein Nachspann, der alle irgendwie auf dem Stack aufbewahrten Register wiederherstellt und dann das Programm sauber wieder verläßt. Man beachte, daß wir hier drei überflüssige Befehle eliminieren konnten:

```
XCHG AL, AH
CBW
PUSH AX
```

Ihr einziger Zweck in **lcase?** als **CODE**-Definition war die Umwandlung eines 8-Bit-Flags in eine 16-Bit-Ganzzahl, die wir dann auf den Stack legen konnten. Der Code für **UCASE** ist in etwa so kompakt, wie es sich in einer solchen Routine nur machen läßt. Da wir aus Zeitüberlegungen heraus Assembler verwendet haben, ist es interessant, Zeitvergleiche anzustellen [5]. Schlagen wir bei der Zahl der Taktzyklen pro Befehl für den Intel-80286 nach, so finden wir:

```
CODE UCASE      \ 0 (Assembler-Anweisung)
POP CX          \ 5
POP BX          \ 5
PUSH DI         \ 3
MOV DI, BX      \ 2
          \ insgesamt = 15 für den Vorspann
HERE:           \ 0 (Assembler-Anweisung)
INC DI          \ 2
MOV BL, 0 [DI]  \ 5
MOV AX, # 96    \ 2
SUB AL, BL      \ 2
CBW             \ 2
XCHG AX, BX     \ 3
SUB AL, # 123   \ 3
CBW             \ 2
AND AH, BH      \ 2
AND AH, # 32    \ 3
SUB 0 [DI], AH  \ 7
LOOP HERE       \ 9
          \ insgesamt = 42 für den Hauptteil
POP DI          \ 5
NEXT            \ 5 (hängt vom Forth-System ab)
          \ insgesamt = 10 für den Nachspann
END-CODE        \ 0 (Assembler-Anweisung)
```

Die Befehle mit der Kommentarbezeichnung "Assembler-Anweisung" werden in der Compilations-Phase ausgeführt und verursachen keinen Laufzeitüberhang. Vorspann und Nachspann werden nur je einmal ausgeführt und ihre **25** Taktzyklen sind demzufolge bei nicht zu kurzen Eingabestrings vernachlässigbar. Die Umwandlung eines Kleinbuchstaben in einen Großbuchstaben erfordert, wie man sieht, **42** Taktzyklen, also etwa **1,3** µsec auf einer 33-MHz-Maschine. Mit Hilfe der Testschleife

```
: TEST0 0 DO PAD COUNT UCASE LOOP ;
: TEST1 0 DO 10000 TEST0 LOOP ;
```

können wir genügend viele Schleifendurchläufe erzeugen, um zu brauchbaren Überprüfungsergebnissen zu gelangen: Geben wir

**10 TEST1** ein, so bekommen wir **10^5** Durchläufe. Die zur Umwandlung von **4500000** Zeichen benötigte Zeit beträgt **7** Sekunden, was einer Umwandlungszeit von **1,6** µsec pro Zeichen entspricht. Das stimmt mit unserer von den Maschinenzyklen herrührenden Abschätzung recht gut überein. Es ist **24** mal schneller als mit der F-PC-Forth-Version [6]. Wenn wir viele Strings zu konvertieren haben, lohnt sich also die Optimierung auf jeden Fall.

Zur Abwechslung hier noch eine Version, die mit 0-terminierten Strings im Stile von C arbeitet. Es gibt zwei naheliegende Arten, wie man das Problem anpacken kann: Zum einen könnte man die Schleife in **UCASE** so abändern, daß sie ihr Ende erreicht hat, sobald das eingeholte Byte **0** ist (nicht zu verwechseln mit ASCII-"0"). Wenn wir eine schnelle Methode zur Verfügung hätten, die Länge eines Strings zu ermitteln, könnten wir andererseits den zuvor gebrachten Code auch unverändert übernehmen. Nun, das einzige, was wir von einem C-String kennen, ist dessen Anfangsadresse. Zur Ermittlung der Länge eines C-Strings müßten wir ihn also über einen schrittweise weiterschaltenden Zähler so lange durchlaufen, bis wir das Terminierungszeichen gefunden haben. In High-Level-Forth würde sich das Unterprogramm hierzu wie folgt lesen:

```
: GET_LEN   ( beg --- len)
   DUP            ( beg beg)
   BEGIN                        \ Beginn Schleife von unbestimmter Länge
      DUP C@                    \ Zeichen holen
      0 <>        ( beg adr flag)
   WHILE CHAR+    ( beg adr+1)
   REPEAT         ( beg end+1)   \ Schleife wiederholen bis Zeichen = 0
   SWAP -         ( -- len)      \ Länge berechnen
;
```

Das Programm ist sehr langsam. Wenn man keinen besonderen Grund für eine Funktion hat, die die Länge eines 0-terminierten Strings ermittelt, so ist nicht einzusehen, warum man gerade diese Funktion extra fassen sollte, nur um ein Programmstück wiederzuverwenden, das eigentlich für "counted Strings" gedacht war. Hier haben wir es mit einer Situation zu tun, wo es vorteilhafter ist, **UCASE** ganz neu zu fassen. Wir wollen die neue Version **UCASE.C** nennen.

Und wieder fangen wir zunächst einmal damit an, einen schnellen Entwurf in High-Level-Forth zu programmieren, um daraus dann eine **CODE**-Definition zu machen. Wir wollen das obenstehende **GET_LEN** mit **UCASE.C** kreuzen, d.h., wir ersetzen die Schleife von definierter Länge durch eine von undefinierter Länge.

```
: UCASE.C    ( beg -- )
   BEGIN                        \ Beginn Schleife von unbestimmter Länge
      DUP C@   ( -- adr char)
      DUP      ( -- adr char flag)
   0<> WHILE                    \ Ende noch nicht erreicht
      lcase?   ( -- adr flag)
      32 AND   ( -- adr char 32 falls lcase | 0 sonst)
      -                         \ 32 nur von Kleinbuchstaben abziehen
      OVER C!                   \ Durch abgeändertes Zeichen ersetzen
      CHAR+    ( -- adr+1)
   REPEAT                       \ Schleife wiederholen bis Zeichen = 0
   DROP                         \ Stack aufräumen
;
```

Die Assembler-Version ist schnell programmiert. Der Trick mit **CBW** ("convert byte to word" - Byte nach Doppelbyte wandeln) hilft uns, Entscheidungen dadurch zu vermeiden, daß wir (in der oberen Hälfte des Registers AX) ein Flag berechnen, das vom Vorzeichen des Subtraktionsvorgangs abhängt.

```
CODE UCASE.C
   MOV DX, DI       \ DI (in DX) aufbewahren
   POP DI           \ DI = beg
1 $:                \ Rücksprunglabel
   MOV BL, 0 [DI]   \ Byte holen
   CMP BL, # 0      \ Ist es 0 ?
   JZ 2 $           \ Ans Ende springen, wenn 0
   MOV AX, # 96     \ 97d ist ASCII-'a'
   SUB AL, BL       \ Ist das Byte ein 'a' ?
   CBW              \ Wenn BL = 97, dann AH = FFh; sonst AH = 0
```

```
   XCHG AX, BX
   SUB AL, # 123     \ Ist das Byte ein 'z' ?
   CBW               \ Wenn AL = 122, dann AH = FFh; sonst AH = 0
   AND AH, BH        \ AH = FFh, wenn 'a' <= Byte <= 'z'; sonst AH = 0
   AND AH, # 32      \ AH = 32 oder 0
   SUB 0 [DI], AH    \ Byte in String umwandeln
   JMP 1 $           \ Schleifenwiederholung
2 $:                 \ Ende
   MOV DI, DX        \ DI wiederherstellen.
   NEXT
END-CODE
```

**Mikromini-Assembler**

Ich habe zwar die Verwendung von Forth-Assembler im Zusammenhang mit der schnellen Entwicklung von Programmen in Maschinensprache oder/und als ein Propagandamittel zur Erweckung von Forth-Interesse bei Außenstehenden diskutiert, man sollte dabei aber natürlich nicht vergessen, daß es sich hier auch um ein wirkungsvolles Werkzeug in der Vorratskiste des Forth-Programmierers handelt. In meiner eigenen Arbeit habe ich mir nie allzu viele Gedanken darüber gemacht, daß die meisten Forth-Systeme [7] im allgemeinen etwas langsamer als optimierte C-Programme sind, da ich wußte, daß ich ohne viel Mehraufwand durch Handprogrammierung einer inneren Schleife jederzeit aufs Gas treten konnte, wenn ich es wirklich nötig hatte. (Es gab eine Zeit, noch gar nicht so lange her, da ich mich von dieser Vorgehensweise so überwältigen ließ, daß ich **CODE**-Definitionen für Forth-Worte nur so aus dem Ärmel schüttelte, nur weil es so leicht ging. Ich brauche nicht zu betonen, daß mir meine eigene Arbeit später ganz schön zu schaffen machte, als es darum ging, die Programme nach ANS-kompatiblen Forth-Systemen zu übertragen. Man sollte sich nicht um Kopf und Kragen **CODE**ieren.)

Wenn der Speicherplatz knapp ist und nur wenige **CODE**-Worte definiert werden sollen, zahlt es sich aus, die Befehle direkt in den Rumpf des **CODE**-Wortes zu schreiben, statt den gesamten Assembler zu laden. Normalerweise sind das Zahlen von der Größe eines Bytes im Hexadezimalformat, die per **C,** wie folgt (für F-PC geeignet) eingefügt werden können:

```
CODE MY@ HEX 5B C, FF C, 77 C, NEXT END-CODE
```

Wenn man mehr als nur ein paar solcher Worte hat, deshalb aber den Assembler noch nicht anwerfen möchte, könnte das folgende Wort von Nutzen sein:

```
\ Mikromini-Assembler, geeignet für F-PC
HEX
: <% BASE @ HEX                  \ Basis 16
   BEGIN BL WORD %NUMBER
   WHILE DROP C,
   REPEAT 2DROP BASE !           \ Basis wiederherstellen
   HERE 1+ @ 3E25 <> ABORT" Missing %> !" ; IMMEDIATE
DECIMAL
\ Verwendung wie folgt: CODE MY@ <% 5B FF 37 %> NEXT END-CODE
\ Achtung: Soll das Obenstehende in ANS-Forth funktionieren, müssen wir
\ %NUMBER unter Verwendung von NUMBER ausdrücken.
\ : %NUMBER 0.0 ROT COUNT NUMBER NIP ;
```

[1] Der endgültige Auszug des ANS-Forth-Standard-Dokuments, X3J14 dpANS-6, kann in einer Anzahl von verschiedenen Maschinenformaten von der Website http://www.taygeta.com heruntergeladen werden. Unter diesen Formaten sind auch F-PC-Hypertext, Microsoft-Word und HTML.

[2] Es gibt viele Möglichkeiten, **lcase?** zu definieren, darunter auch das Auslesen aus einer Tabelle. Der hier gezeigte Weg wurde gewählt, um die Verwendung von Assembler zu demonstrieren.

[3] J.V. Noble, Computers in Physics, Jul/Aug 1991, S. 386.

[4] Bemerkung: Wenn wir dieselbe Funktion zur Anbindung an C konstruieren wollten, müßten wir den Umstand beachten, daß in C Strings auf 0 enden. Wir würden dann wohl eine andere Schleifenstruktur benötigen, da die Stringlänge nicht unmittelbar zur Verfügung stünde.

[5] Abrash, im Vorhergehenden bereits erwähnt, diskutiert eingehend die Fallstricke, denen man begegnet, wenn man die von Intel gegebenen Befehlszeiten zu genau nimmt.

[6] F-PC ist ein direktgefädeltes Forth. Forth-Systeme, die optimieren und Maschinencode erzeugen, sind viel schneller (so schnell wie optimierende C-Compiler), jedoch nicht so schnell wie von Hand geschriebene Assemblerprogramme.

[7] Das VFX-Forth von MPE Ltd ist eine moderne Ausnahme.

## 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-muenchen.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.

| | | |
|---|---|---|
| Darstellung | Schleisiek, Klaus | 86-4 Die Kunst, Namen zu geben |
| Darstellung | Berlev, Finn | 87-2 A Forth dictionary |
| Darstellung | Jones, R. | 88-2 'Peerless' Namen in FORTH |
| Darstellung | Plewe, Jörg | 89-2 Über das Wesen der UPN |
| Dateien | Burger, A. | 90-3 Blocks World - Ein hierarchisches Filesystem in FORTH |
| Dateien | Neumann, Helge | 93-1 Ein FILE DUMPen |
| Dateien | Behringer, Fred | 99-2 Stack-Auslagerung in eine Datei |
| Datentypen | Hoffmann, Ulrich | 87-2 Strukturierte Datentypen |
| Datentypen | Staben, Jörg<br>Schleisiek, Klaus | 89-1 :DOES> -- Datenstrukturen in FORTH |
| Datentypen | Scheller, Konrad | 89-3 Neue Datentypen in FORTH |
| DFÜ | Schnitter, Heinz | 90-4 DFÜ-Aktivitäten "FORTH-eV.de" |
| DFÜ | Wilke, Jens | 91-2 Die neue Münchener FORTH-Box |
| DFÜ | Wilke, Jens | 93-1 Lust statt Frust - ein Einstieg in die DFÜ |
| DFÜ | Wilke, Jens | 93-2 Mailbox/DFÜ-News |
| DFÜ | Diaczyszyn, Z. | 94-2 DFÜ mit der Forth-Mailbox |
| DFÜ | Hoffmann, Ulrich<br>Petersen, Holger | 95-1 100 Tage neue Mailbox |
| DFÜ | Prinz, Friederich | 95-1 Z-Netz/Sprachen/Forth |
| Direktorial | Beierlein, T.<br>Hoffmann, Ulrich<br>Staben, Jörg | 93-3 Bericht des Direktoriums |
| Direktorial | Direktorium | 94-2 Brief des Direktoriums |
| Direktorial | Plewe, Jörg | 95-1 Wie schnell so ein Jahr vergeht ... |
| Direktorial | Direktorium | 97-3 Direktorial ... |
| Direktorial | Beierlein, T.<br>Woitzel, E.<br>Prinz, F. | 98-1 Bericht des Direktoriums |
| Direktorial | Beierlein, T.<br>Woitzel, Egmont<br>Prinz, Friederich | 98-3 Bericht des Direktoriums |
| Direktorial | Directors | 99-2 Happy Birthday FIG UK |
| Direktorial | Prinz, Friederich | 99-3 Direktorentreffen in Schierke |
| Direktorial | Behringer, Fred | 00-3 Ein neuer Direktor stellt sich vor |
| Direktorial | Beierlein, T. | 00-4 Zum Zustand der FIG US |
| Direktorial | Hoffmann, Ulrich | 01-3 Ein neuer Direktor stellt sich vor |
| Direktorial | Behringer, Fred | 02-1 Begrüßung eines neuen Mitglieds |
| Dokumentation | Pennemann, B. | 87-2 Quelltextdokumentation |
| Dokumentation | Vogt, Claus | 90-4 Autom. Dokum. von FORTH-Programmen (PC-FORTH-Plus 3.2) |

| | | | |
|---|---|---|---|
| DPMI | Schröder, M. | 95-1 | Extending Forth - ein Protected-Mode-Experiment |
| Echtzeit | Werner, Marcus | 90-2 | Realzeit mit KIROS |
| Echtzeit | Klingelnberg, A. | 90-3 | Echtzeit '90 (Bericht) |
| Echtzeit | Hoffmann, Ulrich | 91-3 | Echtzeit '91 - FORTH-Teams an der Spitze |
| Echtzeit | Schleisiek, K.-P. | 92-2 | Programmierwettbewerb '92 |
| Echtzeit | Klingelnberg, A. | 92-2 | Echtzeit '92 |
| Echtzeit | Woitzel, Egmont | 94-2 | Programmierwettbewerb Echtzeit '94 |
| Echtzeit | Kern, Ralf | 94-3 | Eingebettete Systeme und Echtzeit-Datenverarbeitung |
| Echtzeit | Klingelnberg, A. | 94-3 | Windows-Applikationen und sichere Echtzeitsysteme |
| Echtzeit | Plewe, Jörg | 94-3 | Das war der Programmierwettbewerb auf der Echtzeit '94 |
| Editor | Mensing, Rudolf | 87-3 | F83-Editor für C128 CPM+ und CPC464/6128 |
| Editor | Raschke, Frank | 89-4 | Ext. Editor auf Basis eines allg. Overlay-Treibers für IBM-volksFORTH |
| Editor | Hohl, Heinrich | 91-4 | Ein intelligenter Editor für Eingabefehler |
| Editorial | Luda, Denise<br>Aumiller, Rainer | 89-4 | Forth in der DDR |
| Editorial | Kretzschmar, R. | 93-3 | Der Ärger |
| Editorial | Vogt, Claus | 95-4 | "Ordentlich fett ..." |
| Editorial | Behringer, Fred | 96-2 | Forth-Tagung 96 - ein Erfolg |
| Editorial | Vogt, Claus | 96-3 | Von Sommer- und anderen -löchern |
| Editorial | Vogt, Claus | 96-4 | Multimedial |
| Editorial | Vogt, Claus | 97-2 | Ausflüge in die roten Zahlen |
| Editorial | Prinz, Friederich | 98-1 | Liebe Leser ... |
| Editorial | Prinz, Friederich | 98-2 | Liebe Leser, |
| Editorial | Prinz, Friederich | 98-3 | Liebe Leser, |
| Editorial | Ouverson, Marlin | 98-3 | Rund um die Welt |
| Editorial | Bitter, Martin | 98-4 | Liebe Leser, |
| Editorial | Prinz, Friederich | 99-1 | Liebe Leser, (VD im Englischen?) |
| Editorial | Prinz, Friederich | 99-2 | Liebe Leser, (CEBIT und die "digitale Zunft") |
| Editorial | Prinz, Friederich | 99-3 | Liebe Leser, (Untergang von C++ ?) |
| Editorial | Prinz, Friederich | 99-4 | Liebe Leser, (Rätsel gut, mehr VD-Mitarbeit) |
| Editorial | Prinz, Friederich | 00-1 | Liebe Leser, (VD im Web) |
| Editorial | Prinz, Friederich | 00-2 | Liebe Leser, (Was wird aus FIG US?) |
| Editorial | Prinz, Friederich | 00-3 | Liebe Leser, (Bernd Paysans Web-Server) |
| Editorial | Prinz, Friederich | 00-4 | Liebe Leser, (Aufruf zu mehr Mitarbeit) |
| Editorial | Bitter, Martin | 01-1 | Liebe Leserinnen und Leser, (eingesprungen) |
| Editorial | Bitter, Martin | 01-2 | Meeresrauschen und schalltote Räume |
| Editorial | Bitter, Martin | 01-3 | Vereinsmeierei und Glücksgefühle |
| Editorial | Bitter, Martin | 02-1 | Fünf Ausgaben der Vierten Dimension! |
| England | Behringer, Fred | 95-2 | Forth International |
| England | Behringer, Fred | 95-4 | Zeitschrift "Forthwrite" |
| England | Behringer, Fred | 96-2 | Forthwrite, August 1995 |
| England | Behringer, Fred | 97-1 | Forthwrite 90 (November 1996) |
| England | Behringer, Fred | 97-2 | Forthwrite 91 (Februar 1997) |
| England | Behringer, Fred | 98-2 | Forthwrite 92 (August 1997) |
| England | Behringer, Fred | 98-2 | Forthwrite 93 (November 1997) |
| England | Behringer, Fred | 98-2 | Forthwrite 94 (Januar 1998) |
| England | Behringer, Fred | 98-3 | Forthwrite 95 (März 1998) |
| England | Behringer, Fred | 98-4 | Forthwrite 96 (Mai 1998) |
| England | Behringer, Fred | 98-4 | Forthwrite 97 (Juli 1998) |
| England | Behringer, Fred | 99-1 | Forthwrite 98 (Oktober 1998) |
| England | Behringer, Fred | 99-1 | Forthwrite 99 (November 1998) |
| England | Behringer, Fred | 99-1 | Erste Schritte auf einer englischen IRC-Sitzung |
| England | Behringer, Fred | 99-2 | Forthwrite 100 (Januar 1999) |
| England | Behringer, Fred | 99-3 | Forthwrite 101 (April 1999) |
| England | Behringer, Fred | 99-3 | Forthwrite 102 (Juni 1999) |
| England | Behringer, Fred | 99-4 | Forthwrite 103 (August 1999) |
| England | Behringer, Fred | 00-2 | Forthwrite 104 (November 1999) |
| England | Behringer, Fred | 00-2 | Forthwrite 105 (Januar 2000) |
| England | Behringer, Fred | 00-3 | Forthwrite 106 (April 2000) |
| England | Behringer, Fred | 00-4 | Forthwrite 107 (Juni 2000) |
| England | Behringer, Fred | 00-4 | Forthwrite 108 (August 2000) |
| England | Behringer, Fred | 01-2 | Forthwrite 110 (Januar 2001) |
| England | Behringer, Fred | 01-3 | Forthwrite 111 (April 2001) |

| | | | |
|---|---|---|---|
| England | Behringer, Fred | 01-4 | Forthwrite 112 (Juli 2001) |
| England | Behringer, Fred | 02-1 | Forthwrite 114 (November 2001) |
| Ein-/Ausgabe | Kalus, Michael | 87-1 | Kleinkram laden |
| Ein-/Ausgabe | Molte, B. | 87-2 | Form. Zahlenausg. in F83 mit Dezimalkomma und Tausenderpunkten |
| Ein-/Ausgabe | Hoffmann, Ulrich | 87-3 | Frei programmierbare Funktionstasten |
| Ein-/Ausgabe | Berlev, Finn | 87-3 | Kleinkram laden, Antwort |
| Ein-/Ausgabe | Behringer, Fred | 99-2 | Stack-Auslagerung in eine Datei |
| Eingebettet | Kern, Ralf | 94-3 | Eingebettete Systeme und Echtzeit Datenverarbeitung |
| Eingebettet | Hoffmann, Ulrich | 00-4 | Digitale Signaturen und eingebettete Systeme |
| Entwicklung | Goppold, A. | 88-2 | Software-Engineering auf Personal Workstations |
| Expertensystem | Flögel, Ekkehard | 88-4 | ESY - Die Wissensbasis für ein Expertensystem |
| Fonts | Carl, Andreas | 86-3 | Zeichensatz-Generator für C64 |
| Fraktale | Krinninger, Ch. | 88-4 | Rössler-Attraktor |
| Fraktale | Krinninger, Ch. | 89-2 | Fraktale Berge |
| Fraktale | Krinninger, Ch. | 89-3 | Mandelbrot für das volksFORTH |
| Fraktale | Krinninger, Ch. | 89-4 | Graphtale Pflanzen |
| Frankreich | Behringer, Fred | 95-2 | Forth International |
| Frankreich | Behringer, Fred | 95-3 | FASTGRAF (für Turbo-Forth) |
| Geschichte | unbekannt | 87-2 | Zur Geschichte der Forth-Prozessoren |
| Geschichte | Deliano, Rafael | 96-4 | Schachcomputer |
| Geschichte | Deliano, Rafael | 97-3 | PostScript |
| Graphik | Krinninger, Ch. | 89-1 | DRAGON |
| Graphik | Kühnel, Claus | 89-2 | Nutzung der Firmware des CPC 6128 unter F83 und CP/M+ |
| Graphik | Krinninger, Ch. | 89-4 | Graphtale Pflanzen |
| Graphik | Stüss, Frank | 90-1 | Ein Treiber für die Hercules-Grafik-Karte im volksFORTH |
| Graphik | Beuster, Bernd | 93-2 | Vergleich: F-PC-Grafiktreiber |
| Graphik | Prinz, Friederich | 97-1 | Graphik "ohne Ende" (Turbo-Forth-Fastgraf) |
| Graphik | Bitter, Martin | 99-2 | Hilfefunktion für Fastgraf unter ZF |
| Graphik | Paysan, Bernd | 99-3 | Dragon Graphics (3D-Turtle-Graphics) |
| Graphik | Staben, Jörg | 01-1 | Win32Forth und Graphik |
| Grundlagen | Deliano, Rafael | 94-1 | PID-Regler im Überblick |
| Grundlagen | Allinger, W. | 98-4 | CRC für Dummies |
| Grundlagen | Behringer, Fred | 98-4 | Patchen ohne den Beigeschmack des Halbfertigen |
| Grundlagen | Prinz, Friederich | 99-4 | Hashing, Teil 1 |
| Grundlagen | Prinz, Friederich | 00-1 | Hashing, Teil 2 |
| Grundlagen | Pöppe, Christoph | 01-2 | Rechnen mit garantierter Genauigkeit (XPA3233) |
| Grundlagen | Pöppe, Christoph | 01-3 | Rechnen mit garantierter Genauigkeit (korrigierter Version) |
| Grundlagen | Tiedemann, S. | 02-1 | Stack-FORTH |
| Hardware | Schleisiek, K.-P. | 93-1 | Hard-DisCo (8x8-LED-Feld - ICM-7218) |
| Hardware | Klingelnberg, A. | 93-4 | Forth von TRIANGLE (TDS2020 mit H8/532) |
| Hardware | Klingelnberg, A. | 93-4 | Neuheiten zum Forth Controller TDS2020 |
| Hardware | Klingelnberg, A. | 94-3 | Neu: TDS9092 ControllerBoard |
| Holland | Behringer, Fred | 95-2 | Forth International |
| Holland | Behringer, Fred | 95-2 | Vijgeblad 39 |
| Holland | Behringer, Fred | 95-2 | Vijgeblad 44 |
| Holland | Behringer, Fred | 95-3 | Forth International |
| Holland | Behringer, Fred | 95-3 | Vijgeblad 45 |
| Holland | Behringer, Fred | 95-3 | Vijgeblad 46 |
| Holland | Behringer, Fred | 95-4 | Vijgeblad Notausgabe 1 |
| Holland | Behringer, Fred | 95-4 | Forth auf der Suche nach einem Redakteur |
| Holland | Behringer, Fred | 97-1 | Vijgeblaadje 2 (Oktober 1996) |
| Holland | Behringer, Fred | 97-1 | Vijgeblaadje 3 (Februar 1997) |
| Holland | Behringer, Fred | 97-2 | Vijgeblaadje 4 (April 1997) |
| Holland | Behringer, Fred | 97-3 | Vijgeblaadje 5 (Juni 1997) |
| Holland | Behringer, Fred | 98-1 | Vijgeblaadje 6 (Oktober 1997) |

| | | |
|---|---|---|
| Holland | Behringer, Fred | 98-1 Vijgeblaadje 7 (November 1997) |
| Holland | Behringer, Fred | 98-2 Vijgeblaadje 8 (Februar 1998) |
| Holland | Behringer, Fred | 98-3 Vijgeblaadje 9 (April 1998) |
| Holland | Behringer, Fred | 98-4 Vijgeblaadje 10 (Juni 1998) |
| Holland | Behringer, Fred | 99-1 Vijgeblaadje 11 (Oktober 1998) |
| Holland | Behringer, Fred | 99-2 Vijgeblaadje 12 (Dezember 1998) |
| Holland | Behringer, Fred | 99-3 Vijgeblaadje 14 (April 1999) |
| Holland | Behringer, Fred | 99-3 Vijgeblaadje 15 (Juni 1999) |
| Holland | Behringer, Fred | 00-2 Vijgeblaadje 16 (Oktober 1999) |
| Holland | Behringer, Fred | 00-2 Vijgeblaadje 17 (Dezember 1999) |
| Holland | Behringer, Fred | 00-2 Vijgeblaadje 18 (Februar 2000) |
| Holland | Behringer, Fred | 00-3 Vijgeblaadje 19 (April 2000) |
| Holland | Behringer, Fred | 00-3 Vijgeblaadje 20 (Juni 2000) |
| Holland | Behringer, Fred | 00-4 Vijgeblaadje 21 (August 2000) |
| Holland | Behringer, Fred | 01-1 Vijgeblaadje 22 (Oktober 2000) |
| Holland | Behringer, Fred | 01-1 Vijgeblaadje 23 (Dezember 2000) |
| Holland | Behringer, Fred | 01-2 Vijgeblaadje 24 (Februar 2001) |
| Holland | Behringer, Fred | 01-3 Vijgeblaadje 25 (April 2001) |
| Holland | Behringer, Fred | 01-3 Vijgeblaadje 26 (Juni 2001) |
| Holland | Behringer, Fred | 01-4 Vijgeblaadje 27 (August 2001) |
| Holland | Behringer, Fred | 02-1 Vijgeblaadje 28 (Oktober 2001) |
| Holland | Behringer, Fred | 02-1 Vijgeblaadje 29 (Dezember 2001) |
| Info | Steffenhagen, B. | 93-4 Fuzzy and Forth |
| Interna | Pennemann, B. | 86-1 Status anzeigen in Forth |
| Interna | Scheller, Konrad | 86-3 Defining Words, eine Einführung in die Anwendung |
| Interna | Perry, Michael | 86-3 Performance Analysis in Threaded Code Systems |
| Interna | Pennemann, B. | 86-3 Queues in Forth |
| Interna | Scheller, Konrad | 86-4 Defining Words II |
| Interna | Baden, Wil | 87-2 Ultimate CASE Statement |
| Interna | Klingelnberg, A. | 91-4 Direct Threaded Code am Beispiel F-PC - Create Does en Detail |
| Interna | Klingelnberg, A. | 91-4 Die Stacks, TIB, PAD und HERE - Ein Einstieg nicht nur in F-PC |
| Interna | Höhenleitner, T. | 95-3 DOER & MAKE kurz vorgestellt |
| Interna | Behringer, Fred | 95-4 Umlaute in den Namen von Forth-Worten |
| Interrupt | Stüss, Frank | 89-3 High-Level-Interrupts im IBM-volksFORTH |
| Interview | Ham, Mick | 87-2 FORTH-Profil(e): Martin Tracy |
| Interview | Ham, Mick | 88-4 FORTH-Profil(e): John D. Hall |
| Interview | VD | 93-2 Interview mit Tom Zimmer |
| Interview | Paysan, Bernd | 94-3 Der STREICH feiert seinen 25. Geburtstag |
| Interview | Lawlee, Jim | 02-1 Ein Interview mit Tom Zimmer |
| Lernen | Findewirth, A. | 90-3 FIT - Konzept einer computergestützten Lernhilfe für FORTH |
| Lernen | Prinz, Friederich | 92-1 Forth-Kurs in Moers |
| Lernen | Prinz, Friederich | 92-1 FORTH für Einsteiger - Kurs in Moers |
| Lernen | Kretzschmar, R. | 92-2 Eine Einführung in die Arbeit mit Forth |
| Lernen | Kretzschmar, R. | 92-4 DisCo - ein Lichtspiel |
| Lernen | Kretzschmar, R. | 92-4 Wie das Licht aufging |
| Lernen | Schleisiek, K.-P. | 92-4 DisCo mit System |
| Lernen | Schleisiek, K.-P. | 93-1 Hard-DisCo (8x8-LED-Feld) |
| Lernen | Prümm, Michael | 93-2 Die Schnecke (DisCO) |
| Lernen | Schleisiek, K.-P. | 93-3 Schneckenrennen im DisCo |
| Lernen | Prinz, Friederich | 95-3 EXCEL für Einsteiger - Werbung für die Forthler |
| Lernen | Schleisiek, Klaus | 96-2 DisCo-light |
| Lernen | Behringer, Fred | 01-3 Forth öffnet die Türen (für Umsteiger) |
| Lernen | Behringer, Fred | 01-4 CODE-Definitionen ohne CODE und END-CODE |
| Lernen | Behringer, Fred | 02-1 FINDRAMD.COM - Assemblerprogrammierung in Forth |
| Listen | Plewe, Jörg<br>Stüss, Frank<br>Staben, Jörg | 90-3 Linked Actions, Teil I |
| Listen | Plewe, Jörg<br>Stüss, Frank<br>Staben, Jörg | 90-3 Linked Actions, Teil II |
| Listen | Staben, Jörg | 92-3 Linked Actions und kein Ende |

| | | |
|---|---|---|
| Literatur | Staben, Jörg | 89-2 Bücherecke |
| Literatur | Hom, Werner | 90-3 Bücherecke |
| | Finsterbusch, Horst | |
| Literatur | Klingelnberg, A. | 90-4 Bücherecke |
| Literatur | Schnitter, Heinz | 92-3 Buchbesprechung (Parallele Prozesse) |
| Literatur | Klingelnberg. A. | 92-4 Buchbesprechung (F-PC 3.5 - Ting) |
| Literatur | Staben, Jörg | 93-1 Sprachenstreit in der Bücherecke (3 Bücher) |
| Literatur | Staben, Jörg | 93-1 Buch + Hardware (7 Bücher) |
| Literatur | Klingelnberg, A. | 93-1 Undocumented DOS |
| Literatur | Prinz, Friederich | 93-2 Bücherliste |
| Literatur | Wendler, W. | 93-2 Forth in der Elrad (Payne Forth) |
| Literatur | Klingelnberg, A. | 93-3 Tischer: PC intern |
| Literatur | Klingelnberg, A. | 93-3 CombiBus |
| Literatur | Klingelnberg, A. | 93-3 Buch: Dr.DOS und Ms.DOS |
| Literatur | Klingelnberg, A. | 93-3 Forth in der Elrad (PIC) |
| Literatur | Klingelnberg, A. | 93-4 2 Bücher über Fuzzy |
| Literatur | Freitag, Robert | 95-2 Hendtlass: Real Time Forth |
| Literatur | Prinz, Friederich | 95-2 Forth auf CD |
| Literatur | Behringer, Fred | 96-2 Bücher zu Forth in München wenig gefragt |
| Literatur | Behringer, Fred | 96-2 Sethi: Programming Languages |
| Literatur | Behringer, Fred | 97-1 Dalheimer: Java Virtual Machine |
| Literatur | Deliano, Rafael | 97-1 Raymond: The New Hacker's Dictionary |
| Literatur | Deliano, Rafael | 97-1 Gerdsen/Kröger: Digitale Signalverarbeitung |
| Literatur | Skirl, Ekkehard | 97-2 Archiv der Forthliteratur (Deutschland-Ost) |
| Literatur | Behringer, Fred | 97-3 Bücher (Java) |
| Literatur | Deliano, Rafael | 97-3 Bücher |
| Literatur | Behringer, Fred | 97-3 Bücher aus Frankreich |
| Literatur | Bretschneider, G. | 97-3 Archiv der Forthliteratur (Ost, Teil 2) |
| Literatur | Behringer, Fred | 98-2 Arndt und Haenel: Pi |
| Literatur | Richter, Ulrich | 99-2 Vogeler und Wieland: Borland C++ Builder 3.0 |
| Literatur | Bitter, Martin | 99-3 Staas: StarBasic-Programmierung |
| Literatur | Merkel, Joachim | 00-3 Hodges: Alan Turing, Enigma |
| Literatur | Behringer, Fred | 00-3 RCX-Ecke, Selbstbau (HTML) |
| Literatur | Behringer, Fred | 00-3 Auch Mathematiker können bissig sein |
| Literatur | Aguilar, Victor | 00-3 Axiomatische Wirtschaftstheorie |
| Literatur | Behringer, Fred | 00-4 RCX-Literatur |
| Literatur | Behringer, Fred | 00-4 RCX und Linux |
| Literatur | Prinz, Friederich | 01-3 Bünning und Krause: Windows 2000 |
| Literatur | Prinz, Friederich | 01-3 Badach und Hoffmann: Technik der IP-Netze |
| Literatur | Bitter, Martin | 01-4 Literaturdienst (mit Fred Behringer) |
| | | |
| Logik | Behringer, Fred | 99-1 TF, Conways "Life" und die DNF als IF-Bedingung |
| Logik | Behringer, Fred | 01-3 Lego-Roboter und arithmetisierte Logik in Forth |
| | | |
| Lokale Daten | Hoffmann, Ulrich | 88-1 Nocheinmal: Parameter und lokale Variablen in Forth |
| Lokale Daten | Hansen, Henning | 88-1 Local Variables |
| Lokale Daten | Yli-Nokari, Jyrki | 90-2 Lokale Variablen und Argumente (F83) |
| Lokale Daten | Hayes, John R. | 90-2 Lokale Variablen - ein anderes Verfahren (F83) |
| Lokale Daten | Lehnhardt, Rolf | 90-4 Lokale Variablen (volksFORTH-83) |
| Lokale Daten | Plewe, Jörg | 91-1 Die unendliche Geschichte oder Lokale Variablen Teil MCXVII |
| Lokale Daten | Stüss, Frank | 91-4 Swapping Data - Handhabung großer lokaler Datenmengen |
| | | |
| Macros | Prinz, Friederich | 97-3 Dynamische Macros (Holon) |
| | | |
| Mailbox | Pauck, Marco | 86-2 Der ForthTREE |
| Mailbox | Pauck, Marco | 87-2 Neues vom ForthTREE |
| Mailbox | Krinninger, Ch. | 89-2 Die FORTH-Box München stellt sich vor |
| Mailbox | Teich, Johannes | 92-3 Forth in Mailboxen |
| Mailbox | Deliano, Rafael | 92-4 Mailboxing |
| Mailbox | Wilke, Jens | 93-2 Mailbox/DFÜ-News |
| Mailbox | Diaczyszyn, Z. | 94-2 DFÜ mit der Forth-Mailbox |
| Mailbox | Hoffmann, Ulrich | 95-1 100 Tage neue Mailbox |
| | Petersen, Holger | |
| Mailbox | Kalus, Michael | 96-4 Schöne Sachen selber saugen |
| | | |
| Meldung | Prinz, Friederich | 94-1 FORTH/2 0.39 |
| Meldung | Deliano, Rafael | 95-2 Forth inside IPS (Flugrechner) |

| | | |
|---|---|---|
| Meldung | Prinz, Friederich | 98-1 HolonForth |
| Meldung | Vogt, Claus | 98-2 Java auf Silikon-Forth |
| Meldung | Vogt, Claus | 98-3 Real-Mode-32-Bit-Erweiterung |
| Meldung | Zimmer, Tom | 98-3 Update four (4) for Win32Forth |
| Meldung | Schnitter, Ulrike | 98-3 Forth-Büro nach Rostock |
| Meldung | Vogt, Claus | 98-4 JForth frei für AMIGA |
| Meldung | Prinz, Friederich | 98-4 JForth ist Freeware |
| Meldung | Prinz, Friederich | 98-4 Rafael Delianos Newsletter |
| Meldung | Whitt, John | 99-1 DBMS tools for Win32forth |
| Meldung | Sadler, John | 99-1 Ficl |
| Meldung | Prinz, Friederich | 99-1 Mountain View Press (Glen B. Haydon) |
| Meldung | Paul, Ulrich | 99-4 REORDER im Internet |
| Meldung | Schleisiek, Klaus | 99-4 MicroCore-Patent |
| Meldung | Prinz, Friederich | 00-2 SETI@home |
| Meldung | Behringer, Fred | 00-3 Verlautbarung (Drachen an Egmont Woitzel) |
| Meldung | Prinz, Friederich | 00-4 The FIG UK Awards of 1999 |
| Meldung | Behringer, Fred | 00-4 RCX-Selbstbau (HTML) |
| Meldung | Prinz, Friederich | 00-4 Atomzeit via Internet |
| Meldung | Zobawa, Klaus | 01-1 Küstenforth (Regionale Forth-Gruppe) |
| Meldung | Bitter, Martin | 01-1 strongForth 0.03 liegt vor |
| Meldung | Die Redaktion | 01-1 Microsoft gegen Bundesgerichtshof |
| Meldung | Bitter, Martin | 01-2 LEGO-RCX-Verleih |
| Meldung | Kalus, Michael | 02-1 RCX mit höherer Frequenz |
| Meldung | Bitter, Martin | 02-1 pbForth Version 2.0.0 |
| | | |
| Messen | Beuster, Bernd | 94-1 Messen via Centronics (A/D-Wandler) |
| Messen | Schemmert, W. | 94-1 Der I^2C-Bus - in Forth realisiert |
| Messen | Plewe, Jörg<br>Allinger, Wolfgang | 95-2 Forth denkt, Windows lenkt |
| Messen | Kohl, Klaus | 95-2 Meßtechnik mit dem PC |
| Messen | Kohl, Klaus | 95-3 Programmierung des Timer0-Interrupt |
| Messen | Kohl, Klaus | 95-4 Der A/D-Wandler am Druckerport |
| Messen | Kohl, Klaus | 96-2 Die serielle Schnittstelle |
| Messen | Kohl, Klaus | 96-3 Der Joystick |
| Messen | Kohl, Klaus | 96-4 PC-Meßkarten |
| Messen | Führer, W. | 96-4 'N bißchen wat Praxis |
| | | |
| Mini-Forth | Hoffmann, Ulrich | 91-1 Ein Weg, wie man FORTH klein macht |
| Mini-Forth | Behringer, Fred | 99-1 So kriegt man Forth auch klein (Turbo-Forth) |
| Mini-Forth | Tiedemann, S. | 99-1 MISC - Minimal Instruction Set Computer |
| Mini-Forth | Jakeman, Chris | 99-2 OO-Forth, klein und fein |
| | | |
| Mitteilung | Schnitter, Ulrike | 92-3 Mitteilung aus dem Forth-Büro |
| Mitteilung | Deliano, Rafael | 92-4 Microcontroller zu verleihen! |
| Mitteilung | Schnitter, Ulrike | 92-4 Wunschzettel |
| Mitteilung | Deliano, Rafael | 93-1 Neu: Fachgruppe "Roboter" |
| Mitteilung | Kretzschmar, R. | 93-1 Schamlos ... (Darius-Glasig-Verlag) |
| Mitteilung | Behringer, Fred | 93-4 Transputer gefällig? |
| Mitteilung | Deliano, Rafael | 94-2 Neues vom Microcontroller-Verleih |
| Mitteilung | Kohl, Klaus | 94-2 Der neue volksFORTH-Vertrieb |
| Mitteilung | Schnitter, Ulrike | 94-3 FORTH-Gesellschaft intern |
| Mitteilung | Klingelnberg, A. | 94-3 OOCOBOL auch für oder gerade für Forthler |
| Mitteilung | Schnitter, Ulrike | 95-4 Das Forth-Büro |
| Mitteilung | Prinz, Friederich | 99-1 Ein Logo für die FG wird gesucht |
| Mitteilung | Prinz, Friederich | 99-2 Logo für die FG wird immer noch gesucht |
| Mitteilung | Prinz, Friederich | 99-3 VD im Internet |
| Mitteilung | Die Redaktion | 99-3 Gratulation an Fred Behringer (SWAP-Drachen) |
| Mitteilung | Prinz, Friederich | 99-4 VD im Internet |
| Mitteilung | Prinz, Friederich | 99-4 Ein Logo für die FG |
| Mitteilung | Prinz, Friederich | 99-4 Englischsprachiger Sonderdruck der VD ??? |
| Mitteilung | Bitter, Martin | 01-3 3 neue Mitglieder |
| Mitteilung | Bitter, Martin | 01-4 Literaturdienst (mit Fred Behringer) |
| Mitteilung | Zobawa, Klaus | 02-1 Klaus Zobawa stellt ein neues Mitglied vor |
| Mitteilung | Behringer, Fred | 02-1 Direktorial und Begrüßung eines neuen Mitglieds |
| | | |
| Modem | Teza, Jeffrey R. | 88-2 Forth83-Multitasking-Modem-Paket |
| Modem | Deliano, Rafael | 90-4 Der Modemanschluß |

| | | |
|---|---|---|
| Modem | Wilke, Jens | 93-1 Lust statt Frust - ein Einstieg in die DFÜ |
| Multitasking | Scholz, Michael | 96-1 F-PC, (kooperatives Multitasking) |
| Musik | Rothkamm, F. | 95-3 Brief aus der Diaspora |
| Musik | Horch, Helge | 97-2 Mac, MIDI und "Musik" - das große Abenteuer |
| Neues Wort | Schleisiek, Klaus | 87-1 Das Wort des Monats #3 |
| Neues Wort | Redeker, Markus | 90-2 Verzögerte Ausführung von Worten |
| Neues Wort | Behringer, Fred | 97-3 Das Wort Doppelcolon aus Turbo-Forth |
| Neues Wort | Vogt, Claus | 99-1 Das neue IF{ |
| Neues Wort | Vogt, Claus | 99-1 Das neue Colon |
| Neues Wort | Hoffmann, Ulrich | 99-1 Fleischerhaken |
| Neues Wort | Allinger, W. | 00-1 CFA2NAME (ZF) |
| Neues Wort | Behringer, Fred | 00-3 GGT ohne Division für ZF und Turbo-Forth in 32-Bit Breite |
| Neues Wort | Nemtsev, N. | 00-4 Leserbrief zum "Größten gemeinsamen Teiler" |
| Neues Wort | Bitter, Martin | 01-3 Reorder und die Folgen |
| Online | Stoyke, Olaf | 96-1 Forth Online |
| Online | Stoyke, Olaf | 96-2 Forth Online |
| Online | Stoyke, Olaf | 96-3 Forth Online |
| Online | Stoyke, Olaf | 96-4 Forth Online |
| Online | Stoyke, Olaf | 97-1 Forth Online |
| Online | Stoyke, Olaf | 97-2 Forth Online |
| OOP | Rayburn, Terry | 87-3 METHODS> OBJECT-ORIENTED EXTENSIONS REDUX |
| OOP | Golf, Burkhard<br>Schönlau, Rolf<br>Pleßmann, Klaus W. | 93-1 Forth++ (1) |
| OOP | Golf, Burkhard<br>Schönlau, Rolf<br>Pleßmann, Klaus W. | 93-2 INFOSYS-106 |
| OOP | Paysan, Bernd | 94-2 Object Oriented bigFORTH |
| OOP | Steffenhagen, B.<br>Köller, Malte | 94-3 Paralleles Garbage Collecting in OO-Softwaresystemen |
| OOP | Köller, Malte | 94-3 OOP in der Automatisierungstechnik mit der Option ... |
| OOP | Maier-Schuler, P. | 96-1 cpForth für OS/2 |
| OOP | Schleisiek, Klaus | 98-3 OO-Konzept naxh Manfred Mahlow - Prelude |
| OOP | Wejgaard, Wolf | 98-3 Objekte in HolonForth |
| OOP | Prinz, Friederich | 98-3 OOP in Forth? Muß man das wirklich haben? |
| OOP | Woitzel, Egmont | 98-3 Objektorientierte Programmierung in comFORTH 4 |
| OOP | Prinz, Friederich | 98-3 Programmierwettbewerb: Äpfel, Birnen, Sortieren |
| OOP | Woitzel, Egmont | 98-4 OOP in comFORTH 4, Teil 2 |
| OOP | Woitzel, Egmont | 99-1 OOP in comFORTH 4, Teil 3 |
| OOP | Jakeman, Chris | 99-2 OO-Forth, klein und fein |
| OOP | Prinz, Friederich | 99-3 Wieviel Windows braucht der Mensch? |
| OOP | Klimas, Andreas | 02-1 Warum wird die Bedeutung von OO überschätzt? |
| OS/2 | Major, Michael<br>Prinz, Friederich | 93-2 FORTH/2 - ein 32-Bit-System für OS/2 2.x |
| OS/2 | Maier-Schuler, P. | 96-1 cpForth für OS/2 |

**...den “Flieger” nehmen wir auch noch mit.**

**Solaris 2.3 Airlines:**

Wir besteigen das wartende Flugzeug, werden von dem netten Flugpersonal begrüßt und nehmen bequem Platz. Das Flugzeug startet normal, aber in ca. 1000m Höhe stellen wir fest, daß Piloten und Stewardessen soeben mit einem Fallschirm abgespungen sind, ohne uns vorher zu warnen. Leider sind auch keine weiteren Fallschirme an Bord und der Autopilot ist direkt auf den Südpol eingestellt. Der Versuch "man autopilot" in den Bordcomputer einzugeben, wird von diesem freudig mit einem "Shutdown started" begrüßt. Danach setzt sich automatisch der Bordfilmprojektor in Gang und während wir uns in den Sinkflug versetzt fühlen, erfreut uns dieser mit Dr. Seltsam - oder wie ich lernte die Bombe zu lieben.

# Instant

## Cross-Compiling ohne Einschränkungen
## Forth-Tagung 2002, Garmisch-Partenkirchen

Jens Wilke (wilke@jwdt.com)
19. - 20. April 2002

**Motivation**

Der typische Forth Ansatz für cross-plattform Entwicklung ist es, einen minimalen Kernel mittels Cross-Compiler zu erstellen und den Rest der Applikation im Zielsystem zu compilieren. Neben den vielen Vorteilen (rapid prototyping), hat diese Vorgehensweise aber auch Probleme: Die Werkzeuge, die zum Bauen einer neuen Software benötigt werden, sind auf mehrere Umgebungen verteilt (cross-compiler und Target). D.h. man benötigt zwangsweise das Zielsystem zum erneuten Übersetzen bei einer kleinen Änderung.

Das Instant-Projekt ist eine Erweiterung des Cross-Compilers, die es ermöglicht, ANS-Forth kompatible Programme ohne Anpassung durch den Cross-Compiler laufen zu lassen.

Ziel dieses Papiers ist es, einen knappen Einblick in die Thematik zu geben und zu vermitteln, welcher Weg zur Lösung eingeschlagen bzw. welche Erfahrungen dabei gesammelt wurden.

**Aufbau**

Der Aufbau des Dokumentes ist wie folgt: Für Neulinge der Thematik gibt es eine kurze Einführung, was ein Cross-Compiler tut und wie die Neuerstellung eines F83 oder GForth-Systems aussieht. Es folgt eine Diskussion, welche Nachteile sich aus diesem Ansatz ergeben.

Als Nächstes werden unterschiedliche Lösungsansätze für das Instant Projekt diskutiert und dann der konkret eingeschlagene Weg vorgestellt. Zum Abschluss werden denkbare Anwendungsszenarien vorgestellt und nochmals über den gewählten Lösungsansatz reflektiert.

**Aufgaben eines Cross-Compilers**

Die grundsätzliche Aufgabe (auch in der Nicht-Forth-Welt) eines Cross-Compilers ist es, auf einem Wirtssystem eine Applikation zu compilieren, die für eine andere Systemplattform gedacht ist (andere CPU, andere Speicherkonfiguration, Betriebssystem). In der Forth-Welt ist das Programm, welches cross-compiliert werden soll, typischerweise ein Forth-System plus Anwendungsprogramm.

Der Cross-Compiler reserviert sich zunächst einen großen Speicherbereich, der das Abbild des Programms auf dem Zielsystem darstellt. Die typischen Speicherzugriffsbefehle (@, !, ...) werden entsprechend der Zielplattform (Zellenbreite, Big/Little Endian, Adressen) angepaßt bereitgestellt. Hier gilt es die Zellenbreite, big oder little endian, oder bestimmte Addressbereiche variabel zu unterstützen.

Arbeitend auf diesem Speicherabbild des Zielsystems, stellt uns der Cross-Compiler nun Spezialversionen der Wörter bereit, die zum Compilieren eines Forth-Programms benötigt werden: Create, :, ;, IF, BEGIN, etc. Dies geschieht in einem separaten Forth-Vokabular, so dass alle Wörter identisch zu ihrem "Original" benannt werden.

Ein Programm, das cross-compiliert werden kann, sieht damit (idealerweise) genauso aus wie ein normales Forth-Programm.

**Erzeugung eines Forth-Systems**

Im Gegensatz zu einem normalen Forth kann man aber beim Cross-Compilieren die definierten Wörter nicht sofort ausführen. Da dies für einen Forth-Programmierer ziemlich schmerzlich ist, findet man in der Praxis (F83, GForth) bei der Compilierung eines neuen Forth-Systems einen zweistufigen Ansatz:

Zuerst wird der sog. Kernel mittels Cross bzw. Meta-Compiler (F83 Begriff) übersetzt. In dieser Phase finden sich überwiegend Definitionen die die Basis eines Forth-Systems ausmachen: die Primitives, Arithmetik, (Zahlen)-I/O, Parser, Interpreter. Hier läßt sich das Manko, die definierten Wörter nicht ausführen zu können, verkraften.

In der zweiten Phase wird der Kernel auf dem Zielsystem gestartet und weitere Teile des Forth-Systems und eventuell auch eine Applikation nachgeladen: Vocabularies, See, Debugger, etc. Bei der Definition von Vocabularies wird nun von der Ausführung der definierten Wörter Gebrauch gemacht, z.B. um

die Basis-Vocabularies ROOT, ONLY und FORTH anzulegen. Da wir uns jetzt direkt auf dem Zielsystem bewegen, ist dies auch kein Problem.

Wollten wir die Vocabulary-Implementierung auch Cross-Compilieren, so hätten wir zwei Möglichkeiten: Die Aktionen der Wörter, die wir ausführen möchten, könnten wir "per Hand" auscodieren, mit Create und "," in das Target schreiben; oder aber wir würden den Cross-Compiler um die Unterstützung von Vocabularies erweitern. Das Erste ist unelegant, das Zweite scheidet durch den Programmiererehrencodex aus: Keinen Code doppeln!

**Einschränkungen eines Cross-Compilers**

Möchte man also ein Applikationsprogramm cross-compilieren, hat man eigentlich zwei Probleme: Definierter Code darf nicht ausgeführt werden, das Forth-System ist unvollständig, da Erweiterungen wie Vocabularies erst in der zweiten Phase geladen werden.

Das bedeutet konkret, dass man keine Chance hat, ein Programm das ANS-Forth konform ist, durch einen Cross-Compiler zu bekommen. Bleibt also nur die Variante, dass die Erweiterungen und das Applikationsprogramm auf das Zielsystem geladen werden.

Möchte man Code für eingebettete Systeme erzeugen, ergeben sich folgende Probleme:

–Zum Erstellen der Software bzw. eines Binär-Images wird immer das Zielsystem benötigt

–Es ist kein durchgängiger (automatisierbarer) Prozess (Makefile, Shell-Sript) zum Erstellen des Target-Images definierbar

Der eigentliche Auslöser für Instant ist aber folgender Sachverhalt: Für die Zielplattform eines Projektes stand nur ein Emulator zur Verfügung, der um einige Größenordnungen langsamer war als die tatsächliche (aber noch nicht vorhandene) Hardware. Verschärfend kam noch hinzu, dass der Software-Emulator nicht in der Lage war, ein Speicherabbild abzuspeichern, und dass das Applikationsprogramm relativ groß war, so daß in dieser Konstellation mehrere Stunden zum Compilieren benötigt wurden. Die Turn-Around Zeiten bei einer Code-Änderung waren also unakzeptabel.

Aus dieser Not heraus musste eine Möglichkeit geschaffen werden, das Zielsystem schneller zu emulieren.

**Lösung**

Dieser schnellere Emulator wurde dann letztendlich einfach in den Cross-Compiler integriert.

Für diesen Emulator gibt es prinzipiell drei Lösungsansätze: Interpretation des erzeugten Fädelcodes, Erzeugung von separatem Emulationscode für eine VM im Cross-Compiler, Compilierung von "normalen" Forth-Wörtern im Host-System.

Interpretiert man den Fädelcode, den man für das Target erzeugt hat (man erstellt also eine virtuelle Maschine für den Targetcode), ist es nicht nötig die Codeerzeugung im Cross-Compiler anzupassen. Es ist aber u.U. nötig den Emulator für jedes Target anzupassen.

Erstellt man eine "einheitliche" VM (sie muss zumindest bei den Zellbreiten wie des Targets variabel sein), so muss die Codeerzeugung im Cross-Compiler entsprechend erweitert werden. Dafür ist der erzeugte Code für die VM aber weitestgehend unabhängig vom erzeugten Targetcode. Für die Emulation ist es dabei ohne Belang, ob es sich um direkt oder indirekt gefädelten Code oder gar Bytecode handelt.

Die letzte Variante, also die direkte Erzeugung von Forth-Code im Wirtssystem, hat neben dem sehr schnell zu sein, auch noch einen weiteren Vorteil: Debugger sowie Decompiler können vom Forth-System des Wirtsrechners verwendet werden.

**Cross-Compiler Erweiterung**

Um unterschiedliche Plattformen zu unterstützen, gibt es im GForth Cross-Compiler ein sog. Plugin-Interface (ähnlich wie Defered Wörter). Dieses wurde erweitert, um nicht nur die Codeerzeugung für den Target zu verwalten, sondern auch um die Codeerzeugung für einen Emulator "einzuklinken".

In der Erzeugung des Emulationscodes wird zuerst die entsprechende Aktion der Targetcodeerzeugung aufgerufen und dann der Code im Wirtssystem erzeugt. In der Abbildung findet man ein Beispiel für das Plugin *colon*, das den Aufruf einer Colon-Definition in das Target compiliert. Die Codeerzeugung im Wirtssystem kann über ein Flag abgeschaltet werden. Dies ist nötig, wenn sich die Plugins gegenseitig aufrufen. Z.B. könnte *if* seinen Job dadurch erledigen, indem es *?branch* mit *colon* in das Target compiliert. Da wir aber mit der Erweiterung des Plugins *if* bereits alles im Wirtssystem für die Ausführung von IF im Target getan haben (nämlich einfach das IF des Wirtssystems verwenden), wollen wir nicht nach *colon* das *?branch* in das Wirtssystem compilieren. Die Codeerzeugung für das Wirtssystem wird also bei der Abarbeitung des *if*-Plugins zur Targetcodeerzeugung unterbunden.

Die nötigen Primitives für die Emulation des Speicherzugriffs kommen bereits aus dem Cross-Compiler selbst. Die Arithmetik wird aus dem Wirtssystem übernommen und die Ergebnisse auf die im Target signifikanten Stellen gekürzt.

```
\ compiles call to cfa at current position
Plugin colon, (tcfa – )

\ normal gefädelt
: (cc) T a, H ;                          ´ (cc) plugin-of colon,

\ call gefädelt mit peephole Erweiterungen
: (callc) compile call T >body a, H ;    ´ (callc) plugin-of colon,

pa: colon,
  dup TPA colon,
  skip-colon @ IF drop ELSE tcfa>icfa compile, THEN ;pa

pa: if, ( CS: – x )
  TPAI if, cs-push POSTPONE if ;pa

pa: else, ( CS: x – x )
  cs-pop TPAI else, cs-psuh POSTPONE else ;pa

pa: then, ( CS: x – )
  POSTPONE then cs-pop TPAI then ;pa
```

*Abbildung 1: Definition des Plugins colon, sowie die Codegeneratoren für if, else, und then.*

**Erfahrungen**

Wie so immer steckt der Teufel im Detail. Nun werden ein paar Fallen beschrieben, die sich bei dieser Unternehmung auftun:

Das Problem der separaten Speicher (von Target und Host) taucht immer wieder auf. Möchte die Applikation z.B mit *word* eine Eingabe parsen, so muss zunächst das Wort *word* im Hostsystem ausgeführt werden und dann das Ergebnis in den Speicherbereich des Targets kopiert werden. Genau anders herum verhält es sich bei *evaluate*. Hier muss vom Targetspeicher auf Hostspeicher umgesetzt werden.

Andere Fallen ergeben sich durch die zahlreichen Variablen im ANS-Standard. Werden Variablen z.B. beim Parsen (>in) im Hostsystem verändert, so kann dies beim Übergang in den Targetcode vorher in den Targetspeicher gespiegelt werden, sofern die entsprechenden Variablen im Target existieren. Andersherum ist es ebenfalls möglich: Wird >in vom Applikationscode im Target verändert (was übrigens standard-konform wäre), müßte dies ebenfalls vom Host übernommen werden, da das Parsen des Quelltextes das Forthsystem im Host bewerkstelligt. Die letzte Variante ist momentan noch nicht vollständig gelöst.

Das Wörter wie *>does* und *postpone* noch etwas heimtückischer sind als ohnehin, sei nur am Rande erwähnt.

Ein weiteres Problem hat sich bei den Vorwärts-Referenzen ergeben. Da während des Compilierens nicht bekannte Wörter als Vorwärts-Referenz angelegt werden, wird dafür ein Eintrag in der Symboltabelle angelegt. Die Symboltabelle ist aber nichts anderes als ein Dictionary. Da bei Instant der Emulationscode parallel mitcompiliert wird, gibt es nun einen Konflikt, da ja auch der Dictionary-Eintrag gemacht werden soll. Mittels Umsetzen des DP wurde dies aufgelöst. Diese Vorgehensweise ist allerdings nicht standard-konform.

**Anwendungsmöglichkeiten**

Die Anwendungen von Instant sind vielfältig:

–Das Debugging bei Änderungen am Forth-Kernel wird erleichtert, da bereits See und Debugger zur Verfügung stehen.

–Automatisiertes Testen von Teilen des Anwendungsprogramms in der Simulationsumgebung

–Definierter Erstellungsprozess des Target-Images

–ANSForth konforme Spracherweiterungen könnten lediglich zur Compile-Zeit verwendet werden, indem sie in einen Speicherbereich compiliert werden, der nicht in das Target aufgenommen wird;
Denkbar wären also objekt-orientierte Applikationen in einem System mit lediglich 8kb

**Abschlussbemerkungen**

Egmont Woitzel hat im fieldFORTH System ähnliche Ansätze verfolgt. Zur Emulation wurde ein Instruktionssatz ähnlich dem Prozessor RTX2000 verwendet.

Ob die gewählte Lösungsvariante, den cross-compilierten Code auch im Hostsystem parallel zu übersetzen, die sinnvollste ist, muss letztendlich offen gelassen oder weiter diskutiert werden, da es an sinnvollen Vergleichen fehlt.

Dieses Papier soll nun als Ausgangspunkt für die weitere Diskussion dienen.

*Jens Wilke*

*Der vorliegende Aufsatz war ein Beitrag des Autors zur Tagung der FG in Garmisch-Partenkirchen. fep*

## PbForth 2.15 erschienen

Ralph Hempel ist sehr fleißig. Vor kurzem stellte er erst die Version 2.13 vor, und nun gibt es schon eine (leicht) überarbeitete Version 2.15. Einige Fehler sind bereinigt und sein Tcl/Tk Programm, das zur Kommunikation zwischen dem PC und dem Lego-RCX Stein benutzt werden kann, ist noch einmal verbessert worden.
Mit der Version 2.00 wurde pbForth vor einiger Zeit völlig überarbeitet. Ralph hat sich konsequent an den MAF (Minimal Ansi Forth) Vorschlag von Chris Jakeman gehalten, ja er ist im Verkleinern noch weiter gegangen und hat selbst aus diesem Wordset noch Einiges weggelassen.
So ist das 'neue' pbForth wesentlich kleiner geworden. Natürlich bringt das den Verzicht auf Kompatibilität zu den Vorgängerversionen mit sich. Ob die Vorteile eines kleinen, fehlerfreieren, leicht zu wartenden Kernels diesen Verzicht ausgleichen, muss ein Jeder mit sich selbst ausmachen.
Ralph hat die Quellcodes (wieder) verfügbar gemacht und so umgestellt, dass sie von Tcl/Tk aus zu nutzen und einzusehen sind. Wer will - und kann - ist also in der Lage, pbForth weitgehend zu ändern.
Neu sind die Möglichkeiten, einen Assembler zu benutzen, den *Darin Johnson* zur Verfügung gestellt hat. Er beansprucht im RCX ca. 4 kB und macht Assemblerprogramierung recht bequem (Mein hardcodierter 'Assembler' funktioniert nicht mehr - die Headerstruktur hat sich ein wenig geändert. VD 4/2001 S. 31).
Auf seiner Homepage zeigt Ralph sein didaktisches Geschick: Es gibt ein ausführliches Tutorial zur Benutzung des Assemblers, das kaum Fragen offen lässt.
Innerhalb von Interpret hat Ralph das vektorisierte Wort 'UserIdle eingebaut. In seinem HowTo zum 'Monitoring' (Überwachung) des RCXs benutzt er es in sinnfälliger Weise um in die Vorteile der Programmierung mit solchen vektorisierten Worten einzuführen.
In den downloadbaren Beispielen findet sich eine weitere schöne Anwendung: Mittels SOUND_TONE und 'UserIdle spielt der RCX beständig eine Melodie im Hintergrund.
Alles in allem: pbForth wächst und gedeiht.
Ich vermisse seit den letzten Versionen Wordlist und >Words<. Da Ralph die Quellcodes öffentlich gemacht hat, ist es aber möglich, sein eigenes „Words“ zu schreiben.

*Martin Bitter*

```
\ ein WORDS für pbForth 2.31
\ Aus den Quellcodes interp.tcl und compile.tcl
\ konnte ich Folgendes extrahieren:

\ pbAsm::Secondary {FIND} Find 0 CORE
\ pbAsm::Cell {}          Dup          \ +2
\ pbAsm::Cell {}          Count        \ +4
\ pbAsm::Cell {}          ParseFind    \ +6 !!!
\ pbAsm::Cell {}          ZBranch
\ pbAsm::Cell {}          FIND1
\ pbAsm::Cell {}            Rot
\ pbAsm::Cell {}            Drop
\ pbAsm::Cell {}          Branch
\ pbAsm::Cell {}          FIND2
\ pbAsm::Cell {FIND1}       TwoDrop
\ pbAsm::Cell {}            False
\ pbAsm::Cell {FIND2}     Exit
```

```
\ pbAsm::Secondary {ParseFind} ParseFind 0 {}
\ pbAsm::Cell      {} Dup         \ +2
\ pbAsm::Cell      {} ZBranch     \ +4
\ pbAsm::Cell      {} PFIND1      \ +6
\ pbAsm::Cell      {} TwoDup      \ +8
\ pbAsm::Cell      {} Chars       \ +10
\ pbAsm::Cell      {} Plus        \ +12
\ pbAsm::Cell      {} Dup         \ +14
\ pbAsm::Cell      {} ToR         \ +16
\ pbAsm::Cell      {} Dup         \ +18
\ pbAsm::Cell      {} CharFetch   \ +20
\ pbAsm::Cell      {} ToR         \ +22
\ pbAsm::Cell      {} Blank       \ +24
\ pbAsm::Cell      {} Swap        \ +26
\ pbAsm::Cell      {} CharStore   \ +28
\ pbAsm::Cell      {} Over        \ +30
\ pbAsm::Cell      {} Head        \ +32 !!!
\ pbAsm::Cell      {} Fetch
\ pbAsm::Cell      {} Dup
\ pbAsm::Cell      {} ZBranch
\ pbAsm::Cell      {} PFIND3
\  ....
\ Mittels ' FIND bekam ich die xt-Adresse von
\ FIND und konnte dann einfach auszählen,
\ an welcher Adresse das 'verborgene'
\ ParseFind liegen musste, konnte danach
\ in ParseFind weitersuchen (bzw. zählen) wo
\ die 'verborgene' Variable > head < lag.
\ Ich zähle im Dezimalsystem

DECIMAL
  ' FIND 6 + @ 32 + @

\ Jetzt galt es nur noch, das 1 Cell breite
\ DOVAR zu überspringen

  2 +

\ und zu benennen:

  VALUE head

\ und schon ist head sichtbar geworden.
\ Jetzt noch die Informationen zum Aufbau
\ eines Wortes kennen (aus compile.tcl):
\ siehe Abb. nächste Seite.
\
\ In head ist die xt-Adresse des letzten
\ definierten Wortes gespeichert.
\ Von dort aus springen wir erst einmal zwei
\ Cells zurück.

  head @ 2 -

\ Von nun an geht‘s (maximal dreimal)
\ rückwärts um die 'padding' Nullen zu
\ überspringen.

  BEGIN DUP C@ 0<> WHILE 1 - REPEAT

\ jetzt habe ich ein Countbyte gefunden. Das
\ hole ich mir und springe entsprechend
\ zurück. Hier liegt der Anfang des
\ Namensstrings

  DUP C@ -

\ und zwei Bytes davor das Linkfeld!

  2 -
```

*weiter Seite 40*

## 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<br>Tel./Fax: 089-841 83 17 (p) |
| FORTHchips<br>(FRP 1600, RTX, Novix) | Klaus Schleisiek-Kern<br>Tel.: 040-375 008 03 (g) |
| F-PC & TCOM, Asyst<br>(Meßtechnik), embedded<br>Controller (H8/5xx//<br>TDS2020, TDS9092),<br>Fuzzy | Arndt Klingelnberg, Consultants<br>akg@aachen.kbbs.org<br>Tel.: ++32 +87 -63 09 89 pgq<br>( Fax -63 09 88 ) |
| KI, Object Oriented Forth,<br>Sicherheitskritische<br>Systeme | Ulrich Hoffmann<br>Tel.: 04351 -712 217 (p)<br>Fax: -712 216 |
| Forth-Vertrieb | volksFORTH / ultraFORTH<br>RTX / FG / Super8 / KK-FORTH<br>Ingenieurbüro Klaus Kohl<br>Tel.: 08233-3 05 24 (p)<br>Fax : 08233-99 71<br>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.

Was noch in dieser Ausgabe Platz finden sollte...

Hi,

unter http://www.frank-buss.de/forth/index.html habe ich meinen ForthInterpreter als Java-Applet abgelegt. Er kann noch nicht viel, ist aber auch mit dem Ziel entwickelt worden, möglichst klein zu sein, und ich habe dadurch was mehr darüber gelernt, Programmiersprachen zu implementieren.

Was noch nicht so gut ist, ist die Implementierung des Call-Stacks, der über den impliziten Java-Stack gelöst ist und die Unterscheidung zwischen Immidiate und Compiled Words, die bis jetzt nicht dynamisch für neue Words definierbar ist, was ja, glaube ich, in Forth möglich ist. Ich habe allerdings noch nicht so viel Erfahrung mit Forth, aber vielleicht kennt ja jemand im Internet eine gute Beschreibung, wie man einen Forth-Interpreter besser implementiert und ein gutes Tutorial, das möglichst alle mandatory und optionalen Standard-Wörter anhand von Beispielen erklärt (deutsch oder englisch)?

*Frank Buß*, fb@frank-buss.de

```
\ # -------------------------------------------------------------------------------------
\ # Design Note: Word List
\ # -------------------------------------------------------------------------------------
\ # Word names are recorded in headers forming a simple linked list. The headers have the
\ # following structure:
\ #
\ # -LF&-----NF&-----------------------------------------CF&---- -PF&--------------------
\ # Link   | Name       | Length | Padding | Immediate | Code   | Parameter
\ # Field  | Field      | Count  | Chars   | Flag      | Field  | Field
\ # 1 cell | 1-31 chars | 1 char | n Chars | 1 char    | 1 cell | n cells
\ #
\ # -------------------------------------------------------------------------------------
```

***compile.tcl*** *(R.Hempel)*

```
: head->link ( -- n )   \ Adr. des 'letzen' Linkfeldes
  head @ 2 -
  BEGIN DUP C@ 0 = WHILE 1 - REPEAT
  DUP C@ - 2 - ;

\ Einige Werte für die formatierte Zeichenausgabe

 0 VALUE COL#           \ Spaltenzähler; aktuelle Zeichenposition in der Zeile
18 VALUE FREESPACES     \ soviel soll mindestens in der Zeile frei sein
70 VALUE LINELEN        \ solang ist meine Zeile

: NEXT_COL ( -- )       \ erhöht den Zeichenzähler um Eins
  COL# 1 + TO COL# ;

: ?CR ( -- )            \ führt ein CR aus, wenn in der Zeile nicht mehr genug Platz ist
  LINELEN COL# - FREESPACES <
  IF CR 0 TO COL# THEN ;

\ Jetzt kann ich, vom Namensfeld aus, den Namen ausdrucken.
\ Der Namensstring wird von seinem Countbyte gefolgt!!!

: .Name ( adr -- )      \ druckt Wortnamen aus
  ?CR                   \ evtl. eine neue Zeile
  0 SWAP                \ Zähler (für ausgegebene Zeichen)
  BEGIN COUNT DUP       \ Zeichen holen und kopieren
        ROT ROT         \ (ROT ROT = -ROT )
        2SWAP OVER      \ Zeichen und Zähler als oberste Einträge
  <>                    \ vergleichen
  OVER 31 <             \ zur Sicherheit: maximale Länge erreicht
  AND                   \ solange weder Count oder max. Länge erreicht
  WHILE 1 + ROT ROT     \ Zähler erhöhen
        EMIT NEXT_COL   \ Zeichen ausgeben, Spalte anpassen
  REPEAT                \ evtl. wiederholen
  1 SPACES NEXT_COL     \ ein Leerzeichen anhängen
  2DROP DROP ;          \ Stack aufräumen

\ Trara! Alles ist bereit für WORDS !

: WORDS ( -- )          \ gibt die Worte im System formatiert aus
  CR 0 TO COL#          \ neue Zeile; Spaltenzählen angleichen
  head->link            \ erstes Linkfeld finden
  BEGIN DUP @ 0<>       \ Wert auslesen, solange er nicht Null ist (letzter Eintrag)
  WHILE DUP 2 + .Name   \ auf das Namensfeld positionieren und Name ausgeben
        @               \ nächstes Linkfeld holen und ...
  REPEAT                \ evtl. von vorne weitermachen
  DROP ;                \ Stack aufräumen
```