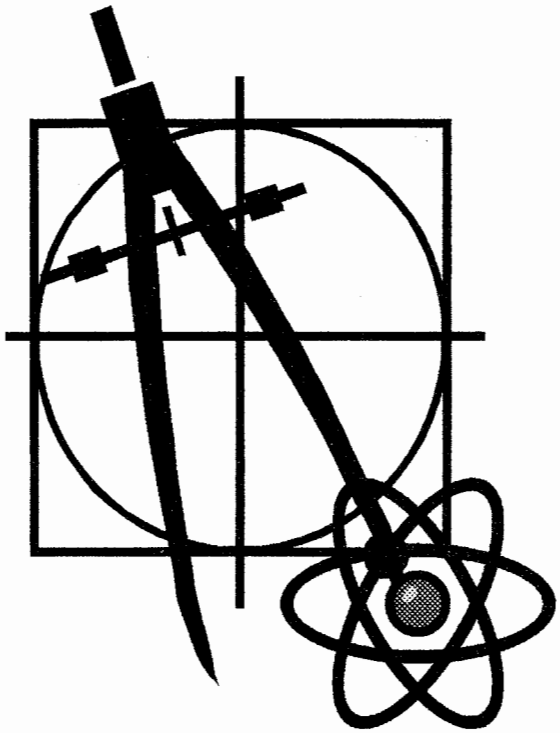
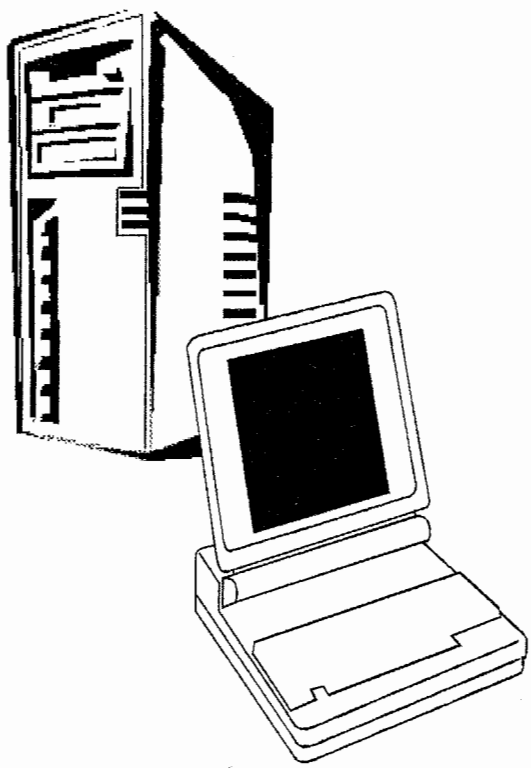
# VIERTE DIMENSION

4 / 2001

## Das FORTH-Magazin


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





### In dieser Ausgabe




17. Jahrgang - **Ausg. 4/2001** – 10,-DM

Organ der FORTH-Gesellschaft e.V.

# Dienstleistungen und Produkte fördernder Mitglieder des Vereins

## tematik GmbH
## Technische Informatik

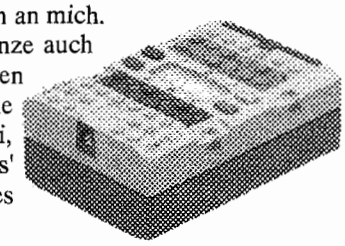
Feldstrasse 143
D-22880 Wedel
Fon 04103 - 808989 - 0
Fax 04103 - 808989 - 9
mail@tematik.de
www.tematik.de

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 SwiftX68k und seit kurzem mit Holon11 und MPE IRTC für Atmel AVR.

## LEGO RCX-Verleih

Seit unserm Gewinn (VD 1/2001 S.30) verfügt unsere Schule über so ausreichend viele RCX Komponenten, dass ich meine privat eingebrachten Dinge nun Anderen, vorzugsweise Mitgliedern der Forth Gesellschaft e.V. zur Verfügung stellen kann!
Angeboten wird: Ein komplettes LEGO-RCX-Set, so wie es für ca. 450 DM im Handel zu erwerben ist. Inhalt:

1 RCX, 1 Sendeturm, 2 Motoren, 4 Sensoren und ca. 1000 LEGO© Steine.
Anfragen über die Redaktion an mich.
Letztendlich enthält das Ganze auch nicht mehr als einen Mikrocontroller der Familie H8/300 Familie von Hitachi, ein 'paar' Treiber und 'etwas' Peripherie. Zudem: dieses Teil ist 'narrensicher'!



Martin Bitter

## 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.

## Dipl.-Ing. Arndt Klingelnberg

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

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

## KIMA Echtzeitsysteme GmbH

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

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

## FORTecH Software
## Entwicklungsbüro Dr.-Ing. Egmont Woitzel

Budapester Straße 80A 18057 Rostock
Tel.: +49 (0381) 46139910 Fax: +49 (0381) 4583488

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.

# Inhaltsverzeichnis

# Editorial


## IMPRESSUM

**Name der Zeitschrift**

**Vierte Dimension**

**Herausgeberin**

Forth-Gesellschaft e.V.
Postfach 16 12 04
D-18025 Rostock
Tel.(Anrufbeantw.): 0381-400 78 28
Fax: 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** *(vorübergehend)*

Martin Bitter
Möllenkampweg 1a
46499 Hamminkeln
Tel.: 02857-1419
E-Mail: VD@FORTH-EV.DE
martin.bitter@forth-ev.de

**Anzeigenverwaltung**

Büro der Herausgeberin

**Redaktionsschluß 2001**

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

**Erscheinungsweise**

1 Ausgabe / Quartal

**Einzelpreis**

DM 10,- zzgl. Porto u. Verp.




(Hier steht im Original wirklich nichts.)

11.9. - sprachlos

## 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.

*MB*

Mit Bedauern und Bestürzung haben wir zur Kenntnis genommen, dass im Juni 2001 Herr Ingwald Bernock. im Alter von 80 Jahren verstarb. Leider wissen wir wenig über die näheren Umstände, obwohl Herr Bernock 10 Jahre lang Mitglied der Forth Gesellschaft war.

das Direktorium
die Redaktion

## Neue pbForth Version

Ralph Hempel gibt bekannt, dass er eine neue gründlich überarbeitete Version von PbForth Ende der ersten Novemberwoche ins Netz stellen wird.
http://www.hempeldesigngroup.com/lego/pbFORTH/index.html

## Literaturdienst?

Es gibt einige Bücher über Forth. Eine handvoll davon sind – zurecht – berühmt, einige – zu unrecht – in Vergessenheit geraten. Viele sind alt! Fast alle sind vergriffen!
Manchmal, selten, besteht Bedarf in so ein Werk hineinzuschauen. Aber: Es fehlt eine zentrale Sammelstelle, die bei der Suche und Vermittlung hilft.
Als erster Schritt dazu nun der Aufruf an alle belesenen Mitglieder: mailt oder sendet mit der gelben Post Eure Titel, d.h. Die Titel der Forthbücher, auf die Ihr Zugriff habt und die ihr im Falle eines Falles bereit seid auszuleihen! Ja, wenn jemand entsprechende Literatur weggeben möchte; auch über solche Sendungen freuen wir uns.
Wir werden versuchen, diese Angaben und Werke zu sammeln und auf Anfragen hin weiter zu leiten.

Fred Behringer co. Rohrmayer
Johann-Strauß-Str. 16, 85591 Vaterstetten

Martin Bitter, Möllenkampweg 1a, 46499 Hamminkeln

## Server in Forth

http://www.eserv.ru ist die Internetadresse von Etype.Net einer Firma in Tula in der Nähe Moskaus (naja, wenn 200km nahe sind). Sie bietet Software rund um Server an (web, ftp, smtp etc). E-Mail-Server (SMTP, IMAP4 und POP3), Verbindung eines LANs mit dem Internet, notfalls mit nur einem Modem, FTP-Server. Unter Windows 3.1, 9x, NT/2000/XP, Unix, Mac OS, OS/2 und sogar unter DOS! Die Quellen sind frei erhältlich und das gesamte Bündel ist in Forth geschrieben! Wenn ich es richtig verstanden habe in SP-Forth, dass man unter anderem bei http://www.enet.ru/win/cherezov/sp-forth.html (Viel Spaß auch mit dem kyrillisch-kryptischen :-( oder bei http://spf.sf.net (englisch) downloaden kann. Dimitri Yakimov einer der Ko-Autoren ist überzeugt davon, dass SP-Forth das schnellste (Windows) Forth unter der Sonne ist!

MB

## Die Forth-Schatzinsel

Bernd-M. Stejskal hat einer Unterabteilung seiner Homepage diesen Namen gegeben. Seine sehr persönliche Sicht auf Betriebssysteme, Ein-Platinen-Computer und und 'große' und 'kleine' Forthes lohnt einen Besuch alle mal. Zitat: „ ... eine Programmiersprache ..., interaktiv wie Basic, schnell wie Assembler, strukturiert wie Pascal und beliebig erweiterbar. Eine solche Sprache ist Forth."

In der Rubrik Humor findet man unter vielem anderen:

**Wie tötet man einen Verein?**
10 Gebote für notorische Miesmacher:

1. Bleibe grundsätzlich jeder Versammlung fern. Läßt sich Dein Erscheinen aber wirklich nicht vermeiden, dann komme zu spät.
2. Wenn Du schon zu einer Versammlung gehst, dann finde Fehler in der Arbeit der Vorstandsmitglieder, vorwiegend in der des Vorsitzenden.
3. Lasse Dich nie für ein Amt oder einen Ausschuss benennen oder wählen. Es ist viel leichter zu kritisieren als irgend etwas selbst zu tun. Sei jedoch stets beleidigt, wenn Du für ein Amt nicht benannt wirst.
4. Wenn Dich der Vorsitzende bittet, Deine Meinung zu einer wichtigen Angelegenheit zu äußern, dann sage ihm, dass Du dazu nichts zu sagen hast. Später erzähle jedem, was eigentlich hätte getan werden müssen.
5. Mach nichts selbst. Wenn andere Mitglieder Gemeinschaftsarbeiten verrichten,dann grolle und erkläre öffentlich, dass der Verein von einer Clique beherrscht wird.
6. Höre grundsätzlich nicht zu und sage später, dass Dir niemand was gesagt hat.
7. Stimme für alles und tue das Gegenteil.
8. Stimme mit allem überein, was während der Versammlung gesagt wird, und erkläre Dich nach dem Schlusswort damit nicht einverstanden.
9. Beanspruche alle Annehmlichkeiten, die Du durch die Vereinszugehörigkeit erlangen kannst, doch trage selbst nichts dazu bei.
10. Wenn Du gebeten wirst, Deinen Beitrag zu entrichten, dann empöre Dich über eine solche Impertinenz.

(kein Kommentar M.B)

http://www.stejskal.de/web/computer/forth/index.html

## Neues Mitglied

*Herzlich begrüße ich (wenn auch leicht verspätet) Andreas Klimas in der Forth Gesellschaft. Er ist seit Anfang dieses Jahres Mitglied. Auf meine Bitte hin hat er folgende Selbstdarstellung verfasst:*

*MB*

Meine Gedanken zur aktuellen Situation in der Softwareentwicklung:

Ich bin seit zehn Jahren selbständiger Softwareentwickler, komme aus der typischen Business Ecke und habe den Fortschritt der letzten 15 Jahre kontinuierlich mitgemacht. Angefangen von DBase über C nach Smalltalk und heute Java. Auch die Methodiken haben sich von strukturiert nach objektorientiert geändert. Meine größte Erfahrung jedoch ist die, daß ich immer noch selber programmieren muß. Ich empfinde den Fortschritt der Softwareentwicklung, verglichen an dem der Hardware, als jämmerlich. Der Resourcenverbrauch ist gigantisch. Die Frameworks lösen Probleme, die nicht existieren und das eigentliche Problem bleibt unerkannt. Wir können leider nicht schneller Software bauen als denken. Womit wir beim Thema angekommen sind. Ein Problem muß zunächst verstanden werden, genauso wie das Werkzeug mit dem es gelöst werden soll. Erst dann können ein Design entworfen, und erste Gehversuche programmiert werden. Dieser Prozeß hat viele Namen, mir erscheint in diesem Zusammenhang Xtreme Programming als wichtigster Vertreter.
Nun, offensichtlich wird hierzu ein dynamisches System benötigt. Smalltalk ist in dieser Beziehung wunderbar flexibel und einfach, hat aber eine Menge Nachteile.
Auf meiner Suche nach einer schlanken Sprache bin ich auf Forth gestoßen. Eigentlich aus reinem Zufall, ich hatte für C einen Multitaskingmechanismus gesucht. Irgend jemand (ich weiß leider nicht mehr wer) erwähnte 'Multitasker' in diesem Zusammenhang. Nach kurzer Recherche im Internet hatte ich eine mögliche Antwort auf meine Fragen erhalten: Forth.
Durch Forth war ich mit einem Werkzeug konfrontiert, das sich bis ins letzte Detail verstehen und verändern ließ. Inspiriert durch Chuck Moore's ColorForth ging ich noch einen Schritt weiter, ich wollte meine Softwareentwicklung auf einem C128 betreiben. Als Beweis gegen den Overkill der heute betrieben wird. Und das ist auch mein aktuelles Projekt - ColorForth für den C64 bzw.C128, danach ein richtiges Projekt, laufend auf einem C64 oder C128.
Der größte Teil der Forthgemeinde beschäftigt sich mit Embedded Systemen und Microcontrollern. Davon verstehe ich rein gar nichts, und das ist auch nicht mein Ziel. Mein Geschäft ist die Anwendungsentwicklung. Mit cleveren HTML Frontends wird die Anwendung von GUI Logik befreit (meiner Meinung nach der einzige Ort wo wirklich Vererbung gebraucht wird). Die Daten werden einfach via 'Block' Mechanismus übers Netz geschaufelt. Eine einfachere Softwareentwicklung habe ich noch nirgends gesehen.
Ich bin mir natürlich bewußt daß ich meine Meinung nicht mit vielen Leuten teile. Das ist auch der Grund weswegen ich mit Java mitmache (mitmachen muß). Die EDV Gemeinde verdient sehr gut am Umstellen der Smalltalkprogramme nach Java. Und wenn dann C# die Oberhand gewinnt, werden wir gerne alle Programme nach C# umstellen ... und so weiter. Aber programmieren müssen wir bis dahin die Lösungen trotzdem immer noch selbst.

In diesem Sinne
Grüße vom Bodensee
Andreas Klimas

http//www.klimas-consulting.com

## Aus der Redaktion:

zuerst: eine Korrektur: In Fred Behringers Artikel über Lego-Roboter und arithmetisierte Logik fehlt in der Formel auf Seite 11 ein SIGMA.

Statt: $$f(x)=\sum\left[a_b*\left(x_1^{b_1}*\cdots*x_n^{b_n}\right)\right]$$

ist dort $$f(x)=\quad\left[a_b*\left(x_1^{b_1}*\cdots*x_n^{b_n}\right)\right]$$

zu lesen.
Wohin das SIGMA verschwunden ist – rätselhaft. Denn: Sein Platz ist frei! Es scheint da zu sein - nur eben unsichtbar.
Der geneigte Leser hat nun mehrere Möglichkeiten:

- Gar nichts tun.
- Die Vierte Dimension 3 2001 aus dem Regal nehmen und
- entweder per Hand das fehlende Sigma eintragen
- oder das korrekte SIGMA vom unteren Seitenrand ausschneiden und in die Lücke kleben
- oder per Fotokopierer ein SIGMA herstellen und
- ...

## Die Artikellage

Beim Erscheinen einer Ausgabe der Vierten Dimension liegen in der Regel Vorankündigungen oder teilfertige Artikel vor, die dem Redakteur des sichere Gefühl vermitteln: Die nächste Vierte Dimension wird gefüllt! Herzlichen Dank an dieser Stelle an alle regelmäßigen Schreiber und Einsender.
Nun diesmal ist es nicht so. Die Januarausgabe scheint dünn zu werden. Aus diesem Grund verschiebe ich den Karatsuba Artikel noch eine Ausgabe weiter (scheint eh. wenige zu interessieren). So ist garantiert, dass neben Fred Behringers Namen noch ein anderer auftaucht.
Wie gehaltvoll die Januarausgabe wird, liegt nun an Ihnen liebe Leser.

## Internet?!

M.Kalus@t-online.de (M.Kalus)
Englisch: "internet" zu Deutsch: "Begrabe dein Netto!"
denn:

to inter -> bestatten, beerdigen, begraben
net -> netzartig, das Netz, Netto
to net -> netto abwerfen, vernetzen

Grüße aus Bochum, Michael

## Katzenjammer

(zur Einstimmung auf die Rätselrubrik)
Der Syllogismus als logische Form ist bekannt.

Bsp. Alle Menschen müssen sterben. (praemissa major)
Sokrates ist ein Mensch. (praemissa minor)
Also muss Sokrates sterben! (conclusio)

Aber?!

Eine Katze hat einen Schwanz mehr als keine Katze. (praemissa major)
Keine Katze hat zwei Schwänze. (praemissa minor)
Also: Eine Katze hat drei Schwänze! (conclusio)

qed. (:-)

# Des Rätsels Lösung
# Metarätsel aus Heft 2/2000

**Fred Behringer**

Es ging um die uralte logische Antinomie des Kreters, der behauptete, alle Kreter lügen (hat er gelogen?) oder des Barbiers, der versprach, zum Neujahrstag alle diejenigen zu rasieren, die sich nicht selbst rasierten (läuft er immer noch mit dem Bart herum?) oder der Menge aller Mengen, die sich nicht selbst als Element enthalten (enthält sie sich selbst?). Insbesondere wollte ich darauf aufmerksam machen, daß eine solche Antinomie auch bei endlichen Begriffsmengen auftreten kann und man gut beraten ist, bei der Verwendung des Wortes "alle" doppelt vorsichtig zu sein.

Was mir als mögliche Abänderungen des Wortlautes vorschwebte, die die Antinomie beseitigen würden, war:

(1) Man erweitere V um eine Colon-Definition A, die alle solche Colon-Definitionen von V enthält, die sich nicht selbst enthalten.

Mögliche Lösung: V(vorher) = {EXIT,RECURSE} - V(nachher) = {A,EXIT,RECURSE} mit : A EXIT RECURSE ;

(2) Man erweitere V um eine Colon-Definition A, die nur solche Colon-Definitionen von V enthält, die sich nicht selbst enthalten.

Mögliche Lösung: V(vorher) = { } - V(nachher) = {A} mit : A ;

Viele Antinomien gehen auf linguistische Verwischungen zurück. Was ist zu der folgenden, in Forth dauernd anzutreffenden Situation zu sagen?

V(vorher) = {A} mit : A ; - V(nachher) = {A} mit : A A ;

Was heißt "erweitern", was heißt "enthalten"? Beim Programmieren spricht man von "Überladen" (ein in der Mathematik althergebrachter Vorgang) und von "Instanzen". Welche Regeln unserer Sprachen stellen sicher, daß wir nicht Begriffe mit ihren Bezeichnungen verwechseln?

Ulrich Paul hat sich mit der Antinomie beschäftigt, die in dem von mir etwas ungenau so bezeichneten "Rätsel" enthalten war. Danke Uli. Ich gebe hier seine Zuschriften wieder:

Hi Fred,

die Aufgabe ist unlösbar: Enthält A nicht sich selber, dann bezieht es sich selbst nicht ein und müßte daher in sich beinhaltet sein (gemäß der Aufgabe). Enthält sich A aber selbst, dann verletzt man die Aufgabe ebenfalls. Kurz gesagt: Enthält A sich selbst, dürfte es es nicht und enthält es sich nicht selbst, dann sollte es es.

Das erinnert mich an eine uralte Jesuitenfrage: Kann Gott einen Stein erschaffen, der so schwer ist, daß er ihn nicht heben kann? Die Theologie (wie auch der Rest der Wissenschaft) haben diese Frage nicht beantworten können. Es liegt an der universalen Definition von der "Allmacht Gottes". Sie erstreckt sich auf das gesamte Universum und darüber hinaus (z.B. Himmel und Hölle).

Dein Rätsel wäre lösbar, wenn man A in ein anderes Verzeichnis legen dürfte, sich also damit außerhalb des zu bearbeitenden Raumes begeben dürfte. Die Aufgabe wäre dann sogar trivial: A enthält dann mindestens eines, nämlich RECURSE, und höchstens so viele Elemente wie es Worte in A gibt, die sich nicht selbst referenzieren. (RECURSE kann sich nicht selbst

referenzieren, da es in diesem Fall niemals terminiert und somit unbrauchbar wäre. Würde man diese Forthigkeit vernachlässigen, dann wäre das Minimum gleich null.)

Aber als Praktiker gebe ich Dir Näherungslösungen: A enthält mindestens 0,5 und maximal soviele Elemente wie es nicht selbsreferenzierende Worte in V gibt (W) plus 0,5. Jeweils mit einer Fehlerspanne von +/-0,5. Oder mit der Intervallschreibweise: der minimale Wert liegt im Bereich 0;1 und der maximale im Bereich W;W+1, die Intervallgrenzen jeweils eingeschlossen. Da übliche Forth-Systeme eine gegen unendlich strebende Anzahl von Worten besitzen geht der o.g. Fehler -prozentual gesehen - gegen null und ist in der Praxis zu vernachlässigen ;-)

Verwandt mit Deinem Rätsel, aber ins Humoristische gezogen, sind die Schildbürgerstreiche (z.B. den Beschluß, das neue Rathaus an die gleiche Stelle wie das alte und mit den alten Ziegeln zu bauen. Das alte Rathaus darf aber erst abgerissen werden, wenn das neue fertig ist) und die angeblich existierende Clubsatzung, die besagt, daß in diesem Club nur Personen sein dürfen, die keinem Club angehören.

CU
Uli
--

Ulrich Paul (1)

Hi Fred,

> Meine Frage bei dem Metarätsel war: Wie kann man die Formulierung abändern, damit das Rätsel selbst eine Lösungsmöglichkeit bekommt.

Das habe ich doch in meiner Mail schon gemacht: (Ich zitiere daraus)

...
Dein Rätsel wäre lösbar, wenn man A in ein anderes Verzeichnis legen dürfte, sich also damit außerhalb des zu bearbeitenden Raumes begeben dürfte. Die Aufgabe wäre dann sogar trivial: ...

Aber gut, wenn Du darauf bestehst, dann eben nur durch Streichungen (keine Ersetzungen oder Einfügungen!) in dem Satz: "Man erweitere V ...":

1.) Streiche das "alle". Dann ist es freigestellt, ob A in sich aufgenommen werden soll/kann/muß.

2.) Streiche das "nicht". Das kehrt zwar die Aufgabe um, aber Du willst es ja so ;-) A enthält sich dann immer, widerspruchsfrei.

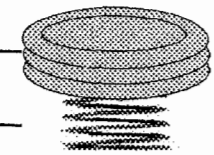
Zu 1.) Du kannst auch "alle" und "und nur solche" streichen. Erfüllt den gleichen Zweck, nämlich die Aufnahme von A in sich selbst dem Leser freizustellen.

Damit ist der linguistische Teil erledigt. Aber gleichzeitig ist das Rätsel keines mehr. Oder ich verstehe die ganze Sache von vorne herein nicht: Was ist das für ein Rätsel, bei dem ich die Fragestellung beliebig ändern darf?

CU
Uli
--

Ulrich Paul (2)

mailto:upaul@paul.de
http://www.paul.de



# MickerForth oder was tue ich mir eigentlich an? MACRO4th.asm !

**Dipl.-Ing. Wolfgang Allinger**

*Zusammenfassung:*

*MACRO4th stellt FORTH ähnliche MACROs mitsamt 16 und 32bit Grundrechnungsarten etc. zur Verfügung. Hier beispielhaft für einen 8051, geht aber für jeden beliebigen µController.*

Jeder richtige Forth Programmierer schreibt sein eigenes Forth. Ich habe mir das bisher verkniffen, denn ich muß meine Brötchen damit verdienen, daß ich Applikationen zum Laufen bringe. Und das möglichst schnell. An meinen Werkzeugen will ich eigentlich nur rumfummeln, um sie zu verbessern.

Ich stand vor einigen Wochen vor dem Problem, daß ich für einen Kunden ein weiteres Meßgerät als Prototyp auf einem AduC812 von Analog Devices zum Laufen bringen mußte. Der AduC812 ist im Prinzip ein 8052 mit zusätzlicher Hardware an Bord: 8kB Code-Flash, 640 byte User-Flash, 256 byte I-RAM, 8ch 12bit ADC, 2 12bit DAC, I²C, SSI etc.

Es ergab sich eine üble Mathematik. Ich brauchte bis zu 32bit Multiplikation und Division.

Die endgültige Formel hatte das Aussehen:

$$m = \frac{DADC - DOFF}{n+1} * \frac{DCON}{DMZ}$$

mit

| | |
|---|---|
| DADC | = ADCwert |
| DOFF | = ADCoffset |
| DCON | = Maschinenconstante |
| DMZ | = Userparameter |
| n | = Stückzahl |

Alle Werte haben 16bit, das Endergebnis ebenfalls.

Nachdem ich nun diverse Libraries ( u.a. bei Philips durchsucht) habe, erwiesen die sich praktisch alle als unbrauchbar. Entweder zu lang, völlig unübersichtlich, ein Wust von Hilfszellen, ein heilloses Wirrwar in der Parameterübergabe. Es graust die Sau, da für alles und jedes eigene Speicherzellen deklariert wurden und schon mal garnicht klar war, was wann wie wo gebraucht wurde. ☹

Ich kam auf die Idee, daß ein kleiner 'Stack' mit 3 doubles und ein double Workregister völlig ausreicht. Ich habe aber keine Zeit und Lust, ein Forth auf dem 8051 zu entwickeln, also kam ich auf die Idee, die Forth primitives als MACROs zu schreiben, sowie die Grundrechnungsarten, die ich benötige. Kein innerer Interpreter, da ich ein 'subroutine threaded Forth' mache, kein outer Interpreter, da ich da eh' keine Zeit und Resourcen habe

Ein weiteres Problem war, daß ich wenig Lust hatte, auch noch mit dem Stackpointer rumzuwirtschaften. So beschloß ich, daß der Stack als Folge von reservierten, benamsten Zellen im internen Memory reserviert wurde und die Daten dann durch die Gegend verschoben werden. Also TOS, NOS für TopOfStack und NextOfStack usw. Bei den geringen Datenmengen war das für mich tolerierbar und es entspricht viel eher dem Modell des "spring loaded stack' von Leo Brodie in seinen Büchern.

Durch die 'Forth' macros kann man

1. auf geprüfte Funktionen zurückgreifen z.B. UD/MOD
2. die Programme sind leicht zu prüfen
3. es wird sehr wenig Platz im internen RAM benutzt

Für einen Forth Programmierer ist sehr leicht ersichtlich, wie das Programm abläuft.

Einen forthigen Eindruck macht es doch aber ein bisschen. So entstand das MACRO4th.A51, aus dem Auszüge hier angeführt sind. Eine Besonderheit habe ich noch bei den Strichrechnungen, sie hinterlassen das PSW bzw. die Funktion COMPARE liefert einen 16bit Wert auf dem Stack zurück mit den obersten Byte der Subtraction und dem 2. Byte als PSW (Prozessorstatuswort).

Der Erfolg gibt mir recht, das System ist sehr schnell geworden, jedenfalls merkt man nichts in der Display Ausgabe, die in einer riesigen langen Schleife alles nacheinander abarbeitet.

Die Meßwerte werden per 10msec Interrupt Service (u.a. auch ADC-pacer) sowie Zählpulse per Interrupt ermittelt, mit dem MACRO4th verwurstet und ausgegeben.

**Beispiel Source der Routine, die den Display Wert nach Dwert schreibt und obendrein Werte mit DAC ausgibt:**

```
; ------------------------------------------
; LAD2val       convert ADC value to Wert%

LAD2val:

; Dat16 = sum of 16 conversions w/ 12bit
; DAof  = offset fuer ADC

;          (Dat16-DAof) * Dcon
; m% =     --------------------
;          (n+1)          * DMzig

  ifetch Dat16H          ; get 16 samples
```

```
  ifetch DaofH          ; 0...999
  literal 6             ; build offset *16 *4 =*64
  lshift                ; 0...1023 (03FFh) -> 0FFC0h
  NMINUS                ; sub offset*64
  jnc Lad2v11           ; NC: no borrow!

     DROP
     literal 0          ; correct underflow

Lad2v11:
  ifetch DconH
  UMmul
  ifetch DGcigH
  UMmul
  UdslashMOD            ; ( udiv udor -- udQ udR )
  DDROP
  DROP                  ; ( -- uQ )
  DUP
  istore DwertH

; show Dwert on DAC0 as 12bit
    ifetch DFo0H
    UMmul
    DROP
    MOV   DAC0H,TOS     ; output
    MOV   DAC0L,TOS+1   ;  DAC0
    DROP
RET
```

Beispiel Auszüge aus MACRO4th.asm Definitionen...

```
; Mit doubles meine ich Forthmaessig 32bit Zellen.
; Also im Prinzip sowas wie ein hp16C :-)
;
; Die Macros sind so geschrieben, dass sie nur beim
; 1. Aufruf sich selber als Code ausfuehren (also
; den 'body' erzeugen), jeder weitere Aufruf wird
; nur als 'CALL body' abgelegt.
; Das sind beim 8051 ueblicherweise 2byte und erst
; bei Ueberschreitung von 2k Grenzen werden es
; 3bytes.
;
; ACHTUNG!Nminus Nplus Dminus Dplus geben das Cyreg
;         im PSW zurueck, bei richtigen Forth ist
;         das undefiniert.
;         So ist es aber praktischer. Da es eh nicht
;         ANS konform ist, erlaube ich mir das!
;
; ---------
; Leider koennen wg. der Einschraenkung fuer
; Assembler Labels bzw. Mnemonics der 8031 CPU die
; Standard Worte nicht immer verwendet werden. In
; der Beschreibung werden deshalb die neuen Namen
; sowie die Standardnamen angegeben.
; ---------
;       Funktionen:
;
; MacroName                             Forth
; DUP       ( n -- n n )
; DDUP      ( d -- d d )                2DUP
; DROP      ( n -- )
; DDROP     ( d -- )                    2DROP
; S2D       ( n -- d )                  S>D
; C2N       ( c -- n )                  C>N
; NABS      ( n -- n )                  ABS
; DABS      ( d -- d )
; Nplus     ( n n -- n )                +
; Ninc      ( n -- n )                  1+
; Dplus     ( d d -- d )                D+
; Nminus    ( n n -- n )                -
; Dminus    ( d d -- d )                D-
```

```
; MacroName                             Forth
;
; UMmul     ( u1 u2 -- ud )             UM*
; UDMmul    ( ud u -- ud )              UDM*
; UDslashMOD ( udiv udor -- udQ udR )   UD/MOD
; NEGATE    ( n -- n )
; DNEGATE   ( d -- d )
; NIP       ( n1 n2 -- n2 )
; TUCK      ( n1 n2 -- n2 n1 n2 )
; NSWAP     ( n1 n2 -- n2 n1 )          SWAP (swap is a
;                                       machine code)
; DSWAP     ( d1 d2 -- d2 d1 )          2SWAP
; ROT       ( n1 n2 n3 -- n2 n3 n1 )
; AND       ( u1 u2 -- u )
; OR        ( u1 u2 -- u )
; OVER      ( n1 n2 -- n1 n2 n1 )
; RSHIFT    ( u c -- u2 )
; LSHIFT    ( u c -- u2 )
; LITERAL   ( -- n )
; DLITERAL  ( -- d )
; ifetch    ( "addr" -- n )     \ fetch from
;                               \ internal addr
; istore    ( "addr" n -- )     \ store to
;                               \ internal addr
; i2fetch   ( "addr" -- d )     \ double fetch
;                               \ from internal addr
; i2store   ( "addr" d -- )     \ double store to
;                               \ internal addr
; COMPARE   ( n1 n2 -- ? )      \ ?= 15..8 MSB of
;                               \ subb, 7..0=PSW

; ===========================================
 SEGMENT DATA
 ORG 050h

; NOTAM: this 'Stack' field must contain >2 doubles,
; not less. Otherwise D- D+ TUCK NIP ... must be
; changed!!!

TOS:
S0:     DS 1
S1:     DS 1
NOS:
S2:     DS 1
S3:     DS 1

D2H:
S4:     DS 1
S5:     DS 1

D2L:
S6:     DS 1
S7:     DS 1

S8:     DS 1
S9:     DS 1
S10:    DS 1
S11:    DS 1
EOS:            ; only for MOVEdup... calc.
sl equ $-S0     ; stack len bytes
ns equ sl/2     ; number of n's on stack

; MOVEdrop from,to,nchar NOS -> TOS direction (a+)
; MOVEdup  from,to,nchar TOS -> NOS direction (a-)

Work:
W0:     DS 1  ; work
W1:     DS 1
W2:     DS 1
W3:     DS 1
```

```
; MOVEdup  from,to,nchar TOS -> NOS direction (a-)
MOVEdup   MACRO  from,to,nc
 ifndef   LMOVEdup
          sjmp LMOVEdup9
LMOVEdup  LABEL $
          MOV  A,@R0
          MOV  @R1,A
          DEC  R0
          DEC  R1
          DJNZ B,LMOVEdup
          RET
LMOVEdup9:
 endif
          MOV  R0,#from
          MOV  R1,#to
          MOV  B,#nc
          CALL LMOVEdup
 ENDM
; ----------------------------------------[snipp]
DUP       MACRO
 ifndef   LDUP
          sjmp Ldup9
LDUP LABEL $
  MOVEdup EOS-3,EOS-1,s1-2
          RET
Ldup9:
 endif
          CALL LDUP               ; ( n -- n n )
 ENDM
 ; -----------------
DDUP      MACRO
 ifndef   LDDUP
          sjmp LDdup9
LDDUP LABEL $
  MOVEdup EOS-5,EOS-1,s1-4
          RET
LDdup9:
 endif
          CALL LDDUP              ; ( d -- d d )
 ENDM
 ;
  ----------------------------------------[snipp]
```

```
; UDMmul ( ud u -- ud )
UDMmul    MACRO
 ifndef   LUDMm
          sjmp LUDMm9
LUDMm     LABEL $
          DUP      ; ( uHL u -- L H u u )
          ROT      ; ( -- L u u H )
          UMmul    ; ( -- L u udHU )
          DROP     ; ( -- L u HU )
          ROT      ; ( -- u HU L )
          ROT      ; ( -- HU L u )
          UMmul    ; ( -- HU udLU )
          ROT      ; ( -- udLU HU )
          Nplus    ; ( -- ud )
          RET
LUDMm9:
 endif
          CALL LUDMm
 ENDM
 ; ----------------
```

## Danksagung

Die Sources sind für den public domain assembler 'AS' von Alfred Arnold CCAC (Computer Club Aachen).
http://john.ccac.rwth-aachen.de:8000/
Ich benutze den letzten unter TurboPascal mit der Version 1.41r7.
AS (auch 'Alfsembler') kennt sehr viele verschiede CPUs.

Vielen Dank an Dr. Alfred Arnold, Leo Brodie und natürlich Chuck Moore sowie die Leser, die bis hierhin tapfer durchgehalten haben.☺

Tschüß Wolfgang Allinger

# Gehaltvolles

**zusammengestellt und übertragen von Fred Behringer**

# VIJGEBLAADJE der HCC Forth-gebruikersgroep, Niederlande

## Nr. 27, August 2001

### De belofte voor 2001

**Willem Ouwerkerk <w.ouwerkerk@kader.hobby.nl>**

In diesem dritten Teil des holländischen Roboter-Bauprojekts (Ushi, die an der Tischkante entlang fährt, ohne runterzufallen) wird das BIOS (Basis-I/O-System) des Roboter-Bausteins besprochen. Das Wort MOTOR wird näher erklärt. Servomotoren werden per Impulsbreitenmodulation angesteuert: 1,55 ms (schnell rückwärts) über 1,7 ms (Stop) bis 1,85 ms (schnell vorwärts). Zur Erinnerung: Das gesamte Projekt wurde seit einiger Zeit in den Vijgeblaadjes vorgestellt und beschrieben. Die Bauteile können von Conrad bezogen werden. Eine erste Materialliste (von mechanischen Modellbauteilen) unter Angabe von Conrad-Bestellnummern stand im Vijgeblaadje 24.

### Forth MANY TIMES

**De Schoolmeester <a.nijhof@kader.hobby.nl>**

"Forth hat keine Syntax", sagt der Schulmeister, "und arbeitet sich interpretierend Wort für Wort durch die Forthzeile durch." Wichtiges Instrument dabei ist >IN. Durch zwischenzeitliche Manipulation an >IN während des Interpretiervorgangs kann man dem Forth-Interpreter beispielsweise vorgaukeln, daß das in Wirklichkeit gerade schon abgearbeitete Wort noch gar nicht abgearbeitet ist. So gelangt man zu den Worten MANY und TIMES.

# Tower forever – die zweite

**Miachael Kalus, Adolf Krüger**

*Die Forth Gesellschaft verfügt in den beiden Mitgliedern Adolf Krüger (Hardwarespezialist) und Michael Kalus (sehr informierter Hardwarelaie) über ein Team, das es sich zur Aufgabe gemacht hat, spezielle Fragestellungen zur Hardware der Lego® RCX Komponenten zu erforschen. Der geneigte Leser mag sich noch an die evolutionär (d.h. durch beherztes Trail-and-Error) entwickelte Lösung zur dauernden Empfangsbereitschaft eines Lego® RCX Infrarot-Sendeturmes aus der vorigen Ausgabe erinnern (vgl. Fred Behringer: ASM2COM über Turbo-Forth: Warum meldet sich der RCX nicht?Vierte Dimension 1/2001 Seite 21 und Martin Bitter, Fred Behringer: Den Lego-Transmitter überlisten – mit und ohne Forth Vierte Dimension 3/2001 Seite 40).*

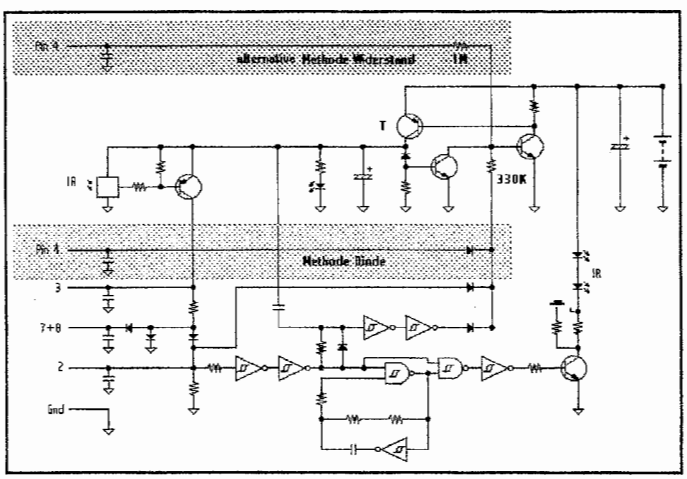
*Hier zeigen uns nun Wissende, wie es noch besser geht!*

*Mehr davon!* *MB*

Hallo Martin,

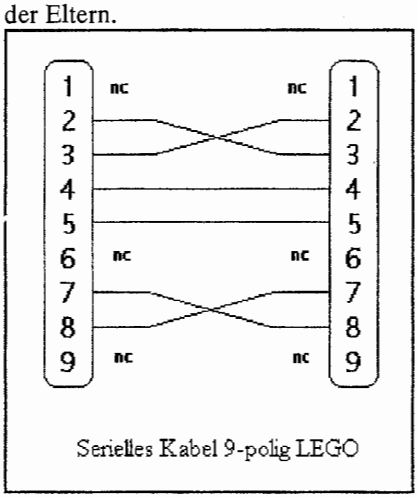
Adolf und ich haben heute eine Lösung ersonnen für deine Frage: "Wie kann der LEGO IR-Turm dauerhaft empfangsbereit gehalten werden?"

In der Anlage ein Bild, das zeigt wie die Operation ausgeführt wird, der Schaltplan des LEGO-Turm und die Kabelbelegung nach LEGO-Art.




Einkommende Daten erzeugen positive Spannung hinter dem Widerstand R. Dadurch wird die +9V Versorgung eingeschaltet, die ganze Schaltung aktiviert. Ein retriggerbares Monoflop startet. Mit jedem Datenbit wird nachgetriggert. Bleiben die Daten aus kippt der Monoflopausgang nach 5 Sekunden auf low und der Transistor T schließt die 9V-Versorgung, die Schaltung ist wieder weitgehend stromlos.

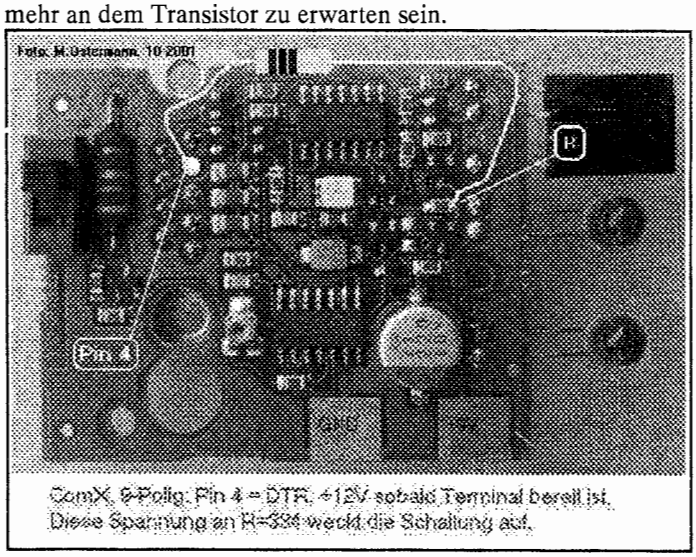
So ist der Turm eigentlich immer abgeschaltet sobald keiner mehr damit spielt: Wenn gar kein serielles Kabel angeschlossen wurde oder der Anwender (Kinder) alles einfach angestöpselt lassen und weggehen. Das schont die Batterie und die Nerven der Eltern.




Serielles Kabel 9-polig LEGO

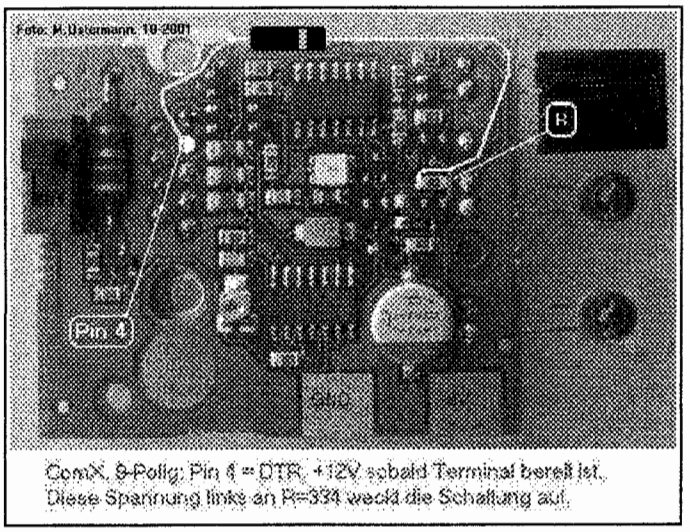
Das serielle Kabel von LEGO enthält nicht alle Leitungen der seriellen Schnittstelle, die Auswahl an steuernden Leitungen ist daher begrenzt. Auf der Suche nach einer geeigneten Leitung fanden wir DTR, RTS und CTS. Letztere sind auf der Turmplatine verbunden und werden wohl von Terminals benutzt um die Anwesenheit des „Modems Turm" festzustellen. So bleibt nur DTR. Mein Hyperterminal z.B. ist so nett DTR zu bedienen: Die aufgebaute Verbindung wird mit H-Pegel signalisiert.

Das läßt sich nutzen: Sehr hochohmig an den Punkt A angekoppelt hält DTR die Schaltung aktiv. Wir haben 100K bis 1M Ohm ausprobiert, ging alles. Ab >470K dürfte kein Schaden mehr an dem Transistor zu erwarten sein.




ComX, 9-Polig: Pin 4 = DTR, +12V sobald Terminal bereit ist. Diese Spannung an R=33K weckt die Schaltung auf.

Im Internet fand ich dann noch die Lego-Site von Smientki und den Turmschaltplan von Shirakawa. Auch dort beschäftigte




man sich mit der gleichen Frage – Lösung dort: Widerstand und Diode in Serie von Pin4 an eben den selben Widerstand, den wir auch herausgefunden hatten. Wir meinen: Eine Diode rüber zur gleichen Stelle wohin auch die Datenpulse gelangen reicht auch, der Widerstand dort hat 330K, das ist hochohmig genug.

Sobald das Terminal die Verbindung abbaut geht DTR auf low und nach 5 sec schaltet der Turm wieder wie gewohnt ab. Wenn das serielle Kabel abgezogen wird ebenso. So ist die Batterie also weiterhin geschützt davor, einfach vergessen, sich fix zu entladen.

*Bitte Beachten: Sie verlieren die Garantie des Herstellers. Sie werkeln auf eigene Gefahr. Tun Sie das nicht! Die Gefahr die Platine zu zerstören ist besonders mechanisch gegeben. Die Lötpads von SMD-Bauteilen sind sehr klein, nur aufgeklebt und gehen daher leicht ab schon bei geringer Zugbelastung. Drähte und Bauteile müssen zugfrei verlötet und zugentlastet befestigt werden – **die Pads und SMD's halten das NICHT!***

Wir haben dünne Drähte genommen und wie gezeigt angelötet, auf die Unterseite der Platine geführt und daran den Widerstand bzw. die Diode angelötet und das Bauteil dort mit 2-Komponenten-Kleber befestigt.

Das ganze geht natürlich auch, wenn statt DTR die +9V von der Platine selbst genommen und über einen hochohmige Widerstand an die beschriebene Stelle gelegt werden. Dann ist der Turm immer aktiv, auch wenn die Verbindung beendet oder unterbrochen wird. Es müßte dann wohl ein Schalter nach außen geführt werden. Wir bevorzugen unsere einfache Lösung.

Michael.

PS Siehe auch:
http://baserv.uci.kun.nl/~smientki/Lego_Knex/Lego_electronica/IR_tower/IR_tower.

# Die Forthbriefmarke - der Prototyp

Hans Eckes

Auf der Forthtagung 2000 (und wohl auch auf der davor) wurde in den Raum gestellt, daß es doch schön wäre, wenn es auch eine Forthbriefmarke gäbe. Zu Gesprächen über irgendwelche Details ist es dann nicht mehr gekommen. Das Thema ist interessant und sollte nicht wieder versanden.

Hier nun ein Diskussionsbeitrag in Hardware. Die angepeilte Zielgruppe ist jeder, der nicht nur Forth in die Hand nimmt, sondern auch einen Lötkolben. Kleine Aufgaben mit Messen/Steuern/Regeln sind genauso möglich, wie auch z.B. das Ansteuern von Modellbau- Servos oder anderen Aufgaben zur Pulsbreitenerzeugung.

## Der Prozessor

Der Prozessor der Forthbriefmarke ist der PSC1000 von Patriot. Er wurde in der VD bereits vorgestellt, mal mit negativem, mal mit positivem Unterton. Fakt ist, der Prozessor ist bereits eine32-Bit-Stackmaschine, man muss ihm nicht erst eine überstülpen. Damit kommt er den Bedürfnissen für Forth so nahe, wie derzeit kein anderer Prozessor. Etliche Forth-Primitives gibt es bereits als Assembler-Befehle, viele Forth-Kommandos lassen sich aus einigen wenigen Assembler-Befehlen zusammenbauen.

Die Wortbreite ist 32 Bit, die Instruktionen sind in der Regel 8 Bit breit (ausgenommen Literals, Sprünge, Unterprogrammaufrufe). Üblicherweise lädt sich der Prozessor bei 32 Bit Datenbusbreite 4 Befehle auf einmal und holt, noch während er diese abarbeitet, die nächsten Befehle, so dass er bei schnellem Speicher nahtlos weiterarbeiten kann. Bei der Forthbriefmarke gibt es aus Platzgründen nur einen 8 Bit breiten Datenbus, deshalb ist für jedes Byte ein Speicherzugriff nötig.

Der Prozessor erlaubt bis zu 4 Speichergruppen, jede mit einem eigenen Timing und Busbreite. Die Forthbriefmarke kann daher verschiedene Timings für den Zugriff auf Boot/FlashROM, die I/O-Ports und den Arbeitsspeicher verwenden.

Die Forthbriefmarke verwendet derzeit einen 40 MHz Oszillator, der Prozessor verdoppelt diesen Takt noch mal und

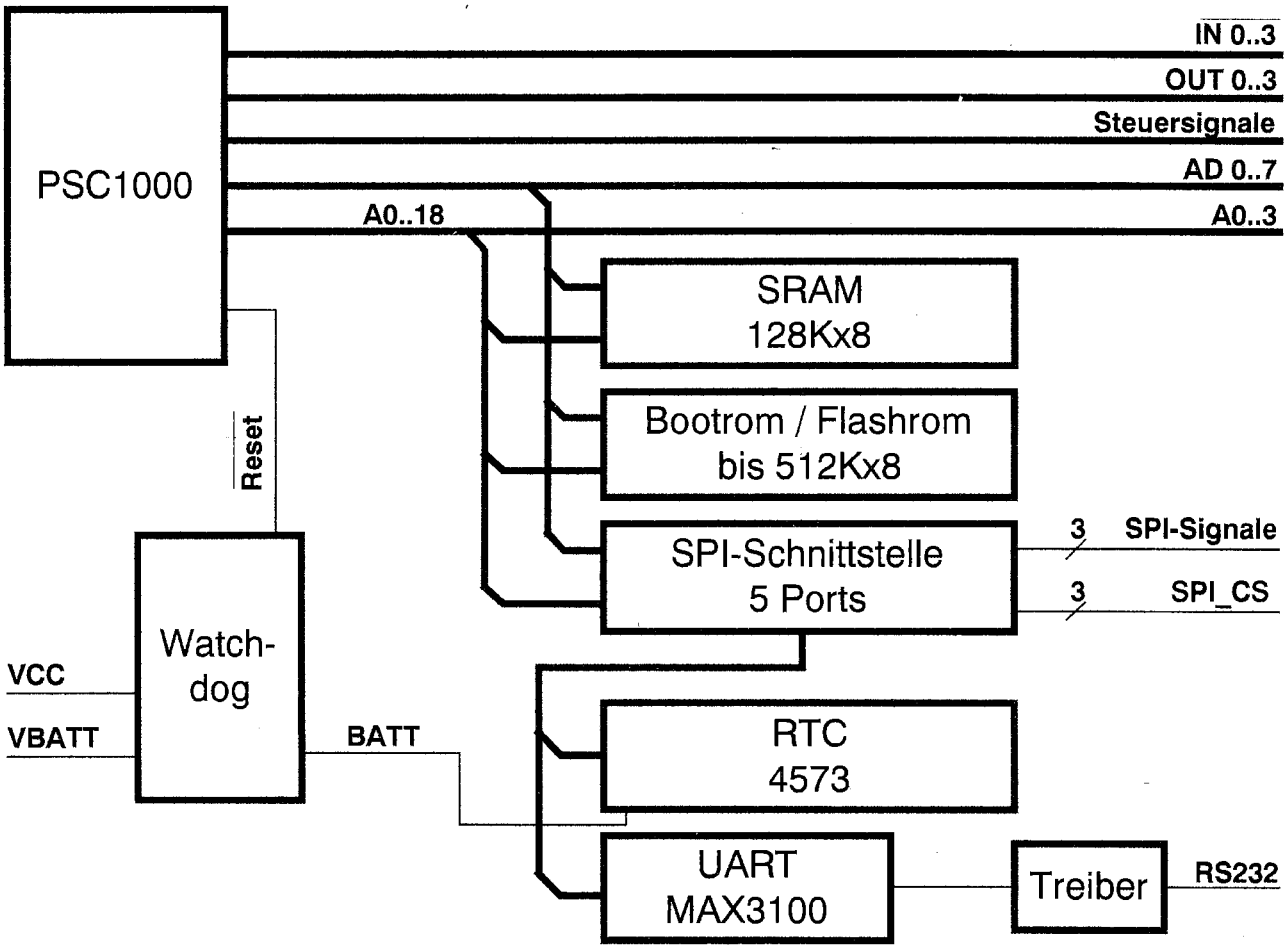
arbeitet dann intern mit 80 MHz. Im praktischen Betrieb wird damit z.B. eine FFT aus einem Zeitsignal mit 1024 Stützpunkten in 0,14 Sekunden berechnet.

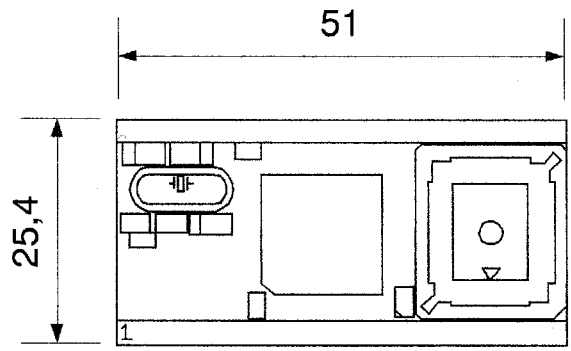
Der Zweizeiler:

```
2VARIABLE   SUMME
: TEST          (   --   )
  1000000  0  DO  SUMME  2@  I  DUP
                  M*  D+ SUMME  2!
           LOOP  ;
```

benötigt für seine 1 Million Durchläufe 4,8 Sekunden, im Vergleich dazu benötigt ein DX2-50 Notebook mit URF386 immerhin 3,4 Sekunden. Man beachte, dass mit 64-Bit-Zahlen gearbeitet wird, und für eine Multiplikation auf dem PSC1000 32 Takte benötigt werden. Der Zweizeiler ist also keineswegs ein besonders günstiger "Benchmark" für den PSC1000.

## Blockschaltbild Forthbriefmarke



Diese Geschwindigkeit ist erst mal vorläufig, die nächste Entwicklungsstufe soll den PSC1000A mit 3,3V Spannungsversorgung und 100 MHz internem Takt verwenden.

## Details zur Forthbriefmarke

Die äußere Form sollte die einer Aufsteckplatine sein, wie ein großes IC. Das Ziel für die Entwicklung war viel Forth auf möglichst wenig Fläche. Das beißt sich und führt dann zu Kompromissen wie z.B. die Busbreite oder die Art des UARTs.

Die Forthbriefmarke sollte mit ihrer Umgebung Kontakt (im wörtlichen und elektrischen Sinne) aufnehmen können. Das sind dann eine RS232-Schnittstelle, eine SPI-Schnittstelle mit freien Chipselects, ein paar Ein- und Ausgänge, ein paar Adressen (4 Bit) und ein Byte Datenbus sowie auch gleich die Steuerleitungen für externe Bustreiber und Adresslatches und -decoder und natürlich noch Pins für die Stromversorgung.

Da sind dann 2x20 = 40 Pins zusammengekommen, damit ist bei einem Rastermaß von 2,54 mm die Länge von 51 mm rausgekommen. Der Prozessor selbst hat mit den Pins eine Grundfläche von 17x17 mm, eine Rastermaßbreite von 20 mm hat zusammen mit den Stiftleisten an den Seiten da nicht mehr gepasst, deshalb wurden es 22,8 mm mit einem Außenmaß von 25,4 mm.

Damit war die Größe der Leiterplatte festgelegt: 51x25,4 mm.

Das Minimum an Bauteilen war schon festgelegt durch die Kontakte, die nach "draußen" gehen:

Der Prozessor, ein FlashROM im Sockel für das Booten dazu, ein RAM, ein UART für die RS232-Schnittstelle, ein Treiber für den RS232-Pegel, eine SPI-Schnittstelle mit mehreren Chipselects, ein Oszillator für den Prozessor, sicherheitshalber noch ein Watchdog und last but not least ein programmierbarer Logikbaustein, der den Kram zusammenhält. Und weil dann immer noch etwas Platz auf der Platine übrig war, kam auch eine Realtime Clock dazu.

Das alles wurde dann zu einer 4-Lagen-Multilayer Platine, die auch tatsächlich nicht wesentlich größer ist als eine Briefmarke.

Der Logikbaustein, der alles zusammenhält, ist ein EPM7032 von Altera. Die Entwicklungssoftware dazu gibt es kostenlos bei Altera. Was der Baustein tun soll, lässt sich in einen Schaltplan zeichnen, es ist ein Simulator dabei, und zum Programmieren benötigt man nur einen Parallelport und einen Adapter. Jeder, der plant solche Bausteine einzusetzen, kann hier ganz ohne HDL-Kenntnisse oder Programmiergerät zum Ziel kommen.

Zurück zu den technischen Details:

Die Platine arbeitet mit 5 V bei etwa 200 mA.

Der Prozessor wird mit 40 MHz getaktet, intern arbeitet er mit 80 MHz.

Das RAM ist 128 KByte groß, die Zugriffszeit beträgt 15 ns. Es ist ein richtiger Stromfresser, eine Batteriepufferung von Daten entfällt damit.

Das Flashrom enthält das Forth für das Booten sowie auch die Applikation. Die unteren 64K sind reserviert für das Forth, alles was darüber liegt, dient zum Abspeichern der Applikation bzw. von sonstigen Daten. Es kann onboard programmiert werden. Der Schreibzugriff auf die unteren 64K wird von der Decodierlogik verhindert, man kann sich den Bootvorgang also nicht zerstören.

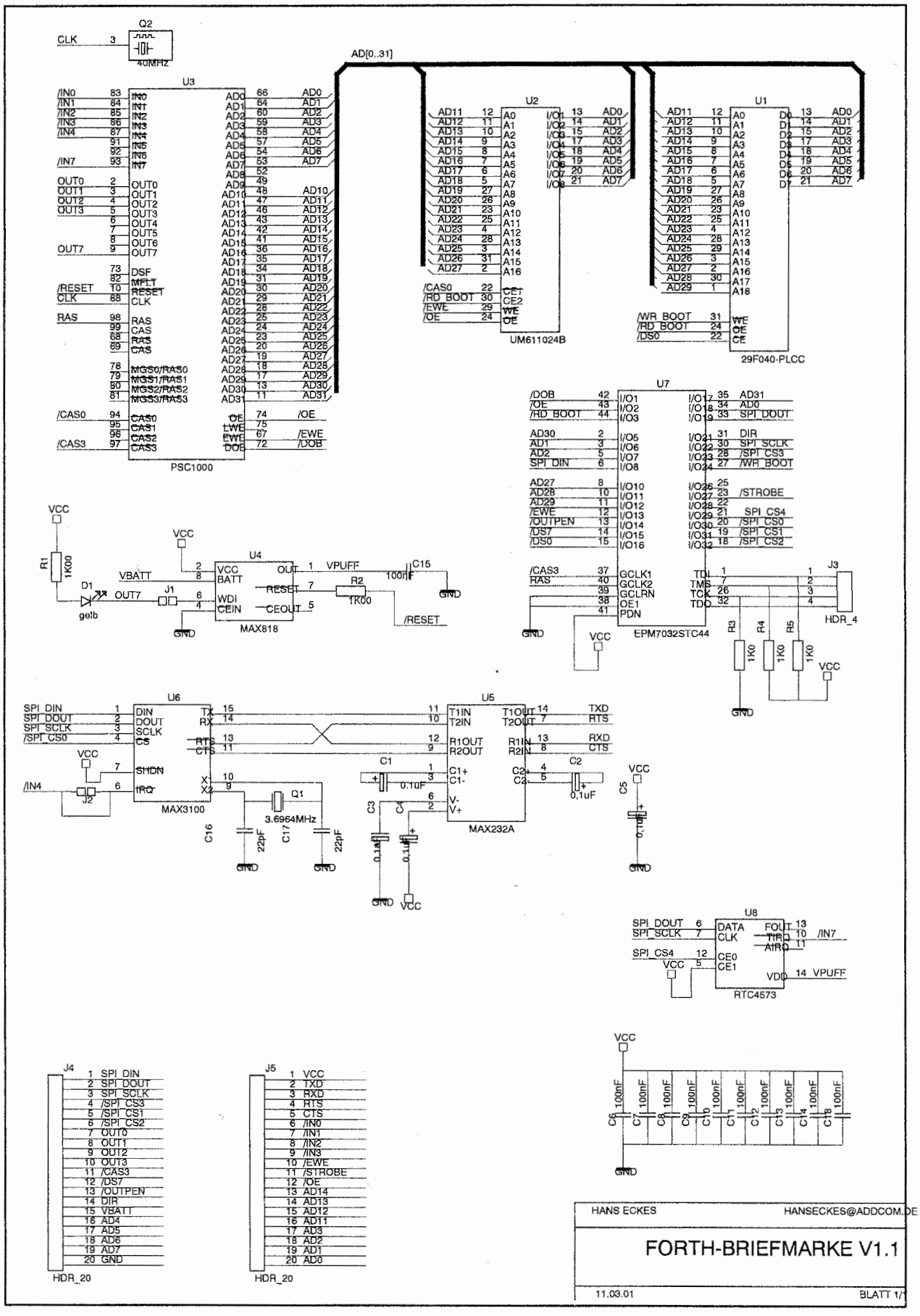
Der Logikbaustein 7032 realisiert auch die SPI-Schnittstelle. Diese spricht den UART für die RS232-Schnittstelle an, die Realtime Clock, und bietet noch 3 freie Chipselectleitungen, die an die Stiftleiste geführt werden. Damit lassen sich 3 SPI-Bausteine ohne weitere Decodierlogik anschließen.

Der UART ist der MAX3100, er bietet einen Empfangs-FIFO, praktisch beliebige Baudraten bis 230 kBaud (die Forthbriefmarke verwendet standardmäßig 115,2 kBaud), kann einen Interrupt auslösen und bietet die Möglichkeit zu einem RTS/CTS-Handshake (der allerdings derzeit vom Forth nicht unterstützt wird).

Zu den 8 Daten- und 4 Adressleitungen gehen noch die Steuersignale für Bustreiber, Adresslatches und -decodierer zu den Stiftleisten. Eine Erweiterung der I/O-Möglichkeiten kann sich damit auf das Wesentliche konzentrieren und muss nicht erst aus den Steuerleitungen des Prozessors die Signale extrahieren. Mit den 4 Adressleitungen kann man 16 Adressen ansprechen. Das klingt nicht nach viel, aber wenn man z.B. für Steuerungsaufgaben 32 Eingänge und 32 Ausgänge benötigt, hat man erst die Hälfte der I/O-Adressen verbraucht. Im Zweifelsfall könnte man noch mit den 4 Prozessorausgängen weitere Adressleitungen erzeugen.

Die 4 Prozessorausgänge OUT0..3 sind unabhängig vom Daten/Adressbus und lassen sich genauso gut als einfache Ausgänge wie z.B. auch zur Pulsbreitenerzeugung verwenden.

Die 4 Prozessoreingänge IN0..3 kann man z.B. zum Auslösen von Interrupts verwenden.


Q2
CLK
40MHz
U3
PSC1000
AD[0..31]
U2
UM611024B
U1
29F040-PLCC
U7
EPM7032STC44
U4
MAX818
VCC
GND
/RESET
/OE
/EWE
/DOB
U6
MAX3100
U5
MAX232A
Q1
3.6964MHz
U8
RTC4573
J1
J2
J3
HDR_4
J4
HDR_20
J5
HDR_20
HANS ECKES
HANSECKES@ADDCOM.DE
FORTH-BRIEFMARKE V1.1
11.03.01
BLATT 1/1

Das war ein Überblick über die Hardware des Prototypen. Für Interessierte gibt es an dieser Stelle auch gleich den Schaltplan dazu.

Mit recht überschaubarem Aufwand sind Varianten denkbar, die z.B. durch Einsatz eines langsameren RAMs und einer anderen Decodierlogik den Stromverbrauch drastisch verringern.

Oder durch Einsatz eines größeren EPLDs softwareunabhängige Funktionen realisieren.

Oder durch Tausch der RS232- gegen eine RS485-Schnittstelle den Betrieb im Verbund mit anderen Forthbriefmarken ermöglichen. Die Grenzen werden an dieser Stelle nur durch die Aufgabenstellung gezogen.

## Das Forth zur Forthbriefmarke

Das Forth zur Briefmarke ist nicht extra für die Forthbriefmarke geschrieben worden. Es wird vielmehr das Forth vom "großen Bruder", dem MiniModul verwendet.

Das MiniModul kommt seit einiger Zeit in der industriellen Meßtechnik zum Einsatz und hat sich bei verschiedenen Aufgabenstellungen bewährt.

Andersherum gesagt, ohne das Forth des MiniModuls hätte es keine Forthbriefmarke gegeben (zumindest wohl keine mit dem PSC1000).

Der Hauptanteil an Arbeit, nämlich das Forth, war ja damit schon fertig, es waren lediglich Anpaßarbeiten für die neue onboard-Hardware nötig.

Der UART ist ein anderer, die Realtime Clock auch, ebenso der Mechanismus zum Speichern der Applikation. Der Umstieg von 32 Bit Busbreite auf 8 Bit war durch Setzen eines einzigen Konfigurations-Bits erledigt.

Das Forth ist ein 32-Bit F83-Forth mit Assembler, Disassembler und Multitasker. Es ist in Wortschatz und Verhalten dem 386UR-Forth von LMI nachempfunden.

Der Interpreter läuft auf der Forthbriefmarke, die Softwareentwicklung geschieht daher interaktiv. Der Quelltext kann in Fließtext oder Screens (Entweder, Oder bzw. Gemischt) geschrieben werden. Er wird dann über die serielle Schnittstelle zur Forthbriefmarke gesendet und dort compiliert.

Es wird kein gefädelter Code erzeugt, sondern gleich Assembler-Code.

Die neu dazugekommenen Wörter lassen sich dann im Flashrom abspeichern, so dass die Forthbriefmarke beim nächsten Booten diese Wörter wieder vom Flashrom in den Arbeitsspeicher holt und dann die Applikation automatisch startet (Debugmöglichkeit ist natürlich vorhanden).

Durch diesen Mechanismus gibt es ein (!) Standardbootrom auf dem dann beliebige Applikationen aufsetzen.

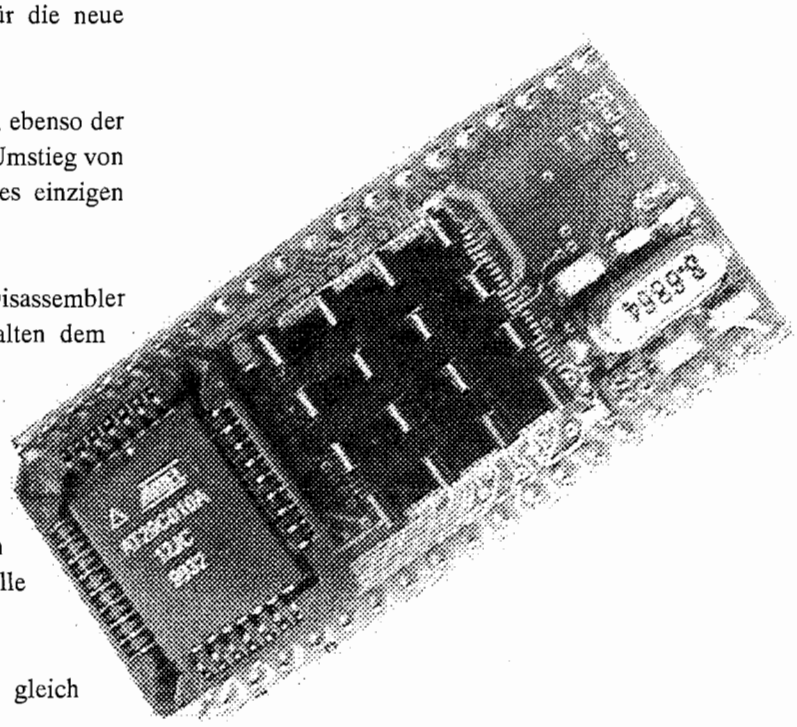
Ein sehr schönes Terminalprogramm für aktuelle Betriebssysteme, das den Download von Fließtext unterstützt, gibt es von der Firma Kartscher Elektronik (www.kartscher-elektronik.de).

Von mir gibt es lediglich ein Terminalprogramm für DOS, ebenfalls für Fließtext sowie für Screens.

Das Forth selbst ist komplett in PSC1000-Assembler geschrieben und nicht crosscompiliert. Änderungen und Erweiterungen lassen sich trotzdem sehr einfach testen, der Interpreter läuft sogar im PSC1000-Simulator von Patriot. Wer so tief einsteigen möchte, kann von mir den Quelltext für das Forth der Forthbriefmarke bekommen (den dazu nötigen Assembler gibt es bei Patriot), sowie das Prozessor-Handbuch als .pdf-Datei.

Ein Handbuch zur Forthbriefmarke wird in der nächsten Zeit (Sommer 2001) entstehen. Ein Evalboard mit 16 Eingängen, 16 Ausgängen, 16 Chipselects, dem Datenbus und der SPI-Schnittstelle an Pfostensteckern gibt es bereits. Fragen bitte an:

hanseckes@addcom.de.



# QUARTUS Forth
# 1. Erfahrungen

**Dipl.-Ing. Wolfgang Allinger**
**Brander Weg 6**
**42699 Solingen**

*Zusammenfassung:*

*QUARTUS (lat. 4th) ist ein ANS kompatibles 16bit Forth für Palm PDAs als shareware. Leider mit einigen gewöhnungsbedürftigen Besonderheiten, die durch die Eigenarten der Palm PDAs sowie des OS begründet sind.*

Das Herz der Palm PDAs ist der 'Dragonball' aus der Mot(z)orola 68k Familie. Als Anzeige dient ein 160x160 pixel Touchscreen LCD Display, als Eingabe ein 'Kratzstift' sowie eine besondere berührungsempfindliche Fläche. Weiterhin sind einige Tasten, eine serielle RS232 Schnittstelle sowie ein Irda Interface vorhanden.

Ich habe die Palms schon längere Zeit beobachtet, aber lange Zeit keine Anwendung darauf gesehen, die mich irgendwie begeistert hätte. Es war für mich ein Yuppie oder Manager-Nintendo™ ☺, was ich mir nicht antun wollte. Zum Termine vergessen, reicht mein Terminkalender völlig aus.

Irgendwann habe ich aber erkannt, daß so ein Teil auch gut als Bediener-Terminal für Meßgeräte geeignet ist. Viele Meßgeräte haben eine rudimentäre Eingabe und Anzeige, die zwar mehr oder weniger gut für die jeweilige Meßaufgabe geeignet ist, aber die eigentliche Bedienung ist schon mühsam.

Besonders tragisch wird's dann, wenn da viele User-Parameter in vielen Menüs vergraben sind und/oder auch noch wichtige Daten enthalten sind. Im Servicefall (Gerät muß gewechselt werden) weiß üblicherweise keine Sau, was in den Geräten an wichtigen Daten vergraben ist. Das neue Gerät enthält latürnich völlig andere Daten. Und schon ist der Ärger da. Viele neumodische Geräte haben zwar einen µC an Bord, manchmal ist sogar die serielle Schnittstelle rausgeführt. Aber wer will schon mit einem bleischweren Schlepptop in einer Raffinerie rumkraxeln. Im Ernstfall ist die Batterie leer, BSOD (blue screen of death) sind auch nicht jedermanns Sache. So kam ich auf die Idee, eine Service Bedieneroberfläche auf einem Palm zu programmieren. Der Palm kann locker in der Hemdentasche mitgeführt werden. Und solange hauptsächlich Daten angezeigt und gespeichert, aber wenig verändert werden müssen, ist die gewöhnungsbedürftige Eingabe nicht so nachteilig.

Auf der Suche nach einer Entwicklungsumgebung stolpert man über ein Riesenpaket 'Code-Warrior' der auch schon mal Bloat-Warrior genannt wird. Obendrein auch noch (un)ziemlich teuer. Wer meine C-Allergie kennt, weiß, wie mirs gegangen ist, als ich das Quartus Forth für den Palm fand.

Quartus Forth (QF) ist eine (fast vollständige) 16bit Implementierung eines ANS Forth. Ich bin vom F-PC stark verwöhnt, daher ist das QF schon ein übler Rückschritt ☹ Viele Merkwürdigkeiten und Unzulänglichkeiten hängen auch stark mit dem Palm und der meist unprofessionellen Handhabung der Dokumentation und Revisionen/Versionen der einzelnen Programme, Dokumentationen etc. So gut wie nie wird vermerkt, wer was wann warum geändert hat und welche Version denn nun neuer oder älter ist. Auch das Geheimnis der aufsteigenden Versionsnummern ist nicht überall bekannt ☹ Das Sytem ist gewaltig aufgebläht, es sind weit über 800 System Aufrufe aus dem QF vorhanden.

Also das **positive vorab**: QF wird recht gut in einem speziellen Forum unterstützt, besonders der Entwickler und Vertreiber Neal Bridges ist i.a. recht schnell. Jedoch Änderungen, die in die Libraries einfließen sind quälend langsam. Scheint auch ein Grund zu sein, warum seit kurzem ein WIKI Server für QF aufgesetzt wurde.

So und nun mein Gemecker: ( Q =QF, P=Palm Problem)

- Q mageres knochentrockenes System
- Q Test von Grafik Ausgaben sehr schwierig, man kommt kaum in den Interpreter zurück
- Q Testhilfen sind sehr mager, z.B. SEE (MORE fehlt darin!) rast durch, man sieht nichts
- P PALM OS sehr fehlerhaft
- P Seriell I/O sehr lahm
- P/Q Änderungen schlecht dokumentiert
- P POSE (Palm OS Emulator) hat ein paar Macken
- P Hotsync zwischen POSE und PC erstmal nur über Nullmoden und COM1/COM2 !
- P Netsync ist eine gutes Geheimnis
- Q Grundlagen Beispiele in QF sehr mager bzw. zu einfach
- Rsrc (resourcen) Editor ist unbedingt erforderlich (freeware)

- Q Symmetrische Division (kein floored!)
- Q Kein Hypertext
- Q Library fehlerhaft
- P 10msec Raster vom OS, feiner geht nichts
- P memo Länge maximal ~4000 byte
- P doppelte Memo Einträge machen extremen Ärger
- Sehr gute Ansätze für newbies sind leider nicht weitergeführt worden (Steve Donahue...)

Wichtige Hilfen/ Applicationen in QF:

ALL (Allinger), NAB (Neal Bridges)

*.txt (memo Dateien)

| | | |
|---|---|---|
| $NUMS | numerisches formatieren | ALL |
| 0prelude | einleitende Gesänge | ALL |
| 0project | alles um ein Projekt anzufangen | ALL |
| ABORT" | | NAB/ALL |
| ALLs-Lib v0.05 | meine lib | ALL |
| hexnums | hex formatter | ALL |
| quithooks | visual stack (ungetestet) | WIKI |
| redefine | warning | |
| seeA | verbessertes SEE | ALL |
| sio | SIO Funktionen | ALL |
| WAITutil | wait a little | ALL |
| WinA | testen in windows | NAB/ALL |

## Wichtige weitere Informationen

Einige wichtige Zusatz Programme, Dokus, Beispiele etc.

http://palmgear.com hier gibt's zuverlässig viel Software. Auch Bezahlung per Kreditkarte ist problemlos.

http://Kristopher_d_johnson.tripod.com/mtask.html "_d_"!!! Multitasker

http://www.anomaly.org/wade/pilot4th viele Feinheiten

http://home.t-online.de/home/Erwin.Schomburg/howtog.htm Guter Einstieg

Tschüß Wolfgang Allinger

# Entscheidungstabellen-Syntax in FORTH

**Klaus.Zobawa@t-online.de**

1. "...außerdem gibt es keine Unterprogramme, Hauptprogramme, Dienst- oder Verwaltungsprogramme, die jeweils unterschiedlich aufzurufen wären. In FORTH ist alles und jedes ein WORT."

2. "FORTH ist eine Umgebung zur Entwicklung anwendungsorientierter Sprachen, in denen dann die Anwendung beschrieben wird."

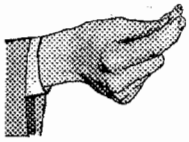
3. "Ziel ist es Konditionalausdrücke / Steuerstrukturen zu eliminieren."

4. "Von zwei adäquaten Lösungen ist immer die einfachere richtig."

( Zitiert aus Leo Brodie's, Thinking FORTH ).

## Motivation:

Mit diesen Marschbefehlen im Tornister machte ich mich im Rahmen einer Neuentwicklung auf den Weg. Ziel dabei war es, eines mehrerer Hardwaremodule eines chirurgischen Gerätes mit eigener Intelligenz, in Form eines Hitachi H8-Derivats, auszurüsten. Möglich machte dies erst die Portierung der Entwicklungsumgebung FieldForth der Fa. FORTecH. Nachdem die Rostocker ein lauffähiges System geliefert hatten wurde der bereits erstellte Entwurf implementiert. Als erstes wurden Timer gebaut und der I²C-Bus in Betrieb genommen. Zur Steuerung des I²C-Controllers war eine State-Machine erforderlich. /!\ Frage: Ist es nicht möglich eine anwendungsorientierte Sprache zur Implementierung von State-Machines zu entwickeln, ganz im Sinne von Leo Brodie ? Also den Compiler um Sprachelemente erweitern, die STM's unterstützen. OK läuft ( Dies ist aber eine andere Geschichte und soll zu gegebener Zeit erzählt werden ). Zur Bearbeitung von einzelnen Prozessen wurde eine Eventqueue erstellt, auf deren Basis ein minimalinvasives Multitasking System entstand ( dies ist eine andere Geschichte ... ).



# Leo Brodie´s Telefontarif Ermittlung:

Neben weiteren Erweiterungen wie Datenstrukturen, Codeoptimierungen ( ...ein andermal erzählt werden) und Zahlenbasishandling, entstand als jüngstes Kind die Syntax zur Implementierung von Entscheidungstabellen. Hier beginnt der eigentliche Aufsatz.

**Entscheidungstabellen:**

Forthler kennen das einprägsame Beispiel der Implementierung einer Telefonkostenbestimmung aus Loe Brodie's Thinking Forth. Er vergleicht die Implementierung in Konditionalausdrücken mit der Darstellung einer Entscheidungstabelle. Nach einer Vereinfachung der Decision Table ( kurz DT ) erfolgt die Implementierung in einer FORTH typischen Weise.

Entscheidungstabellen dienen herkömmlich als Entwurfsmittel, um komplizierte Zusammenhänge in überschaubarer Weise darzustellen. Sie stellen eine eigene formale Sprache dar, die auch zur Dokumentation von Software verwendet wird. Bei der Implementierung sind dem Softwareentwickler alle Freiheitsgrade gegeben, die ihm seine Programmiersprache ( bei FORTH mehr, bei anderen weniger ) bietet. Leo Brodie erläutert sehr ausführlich die möglichen Varianten, von der Verschachtelung von IF/ELSE-Konstrukten, über Tabellen mit entsprechenden Zugriffsmechanismen oder auch Berechnung. Wie die Praxis zeigt, besteht die häufigste Variante aus IF/ELSE Verschachtelungen, wobei in der Regel auf eine Entscheidungstabelle als Entwurf verzichtet wird. Jeder kennt dieses von Leo Brodie erwähnte Gefühl, irgendwann keine Lust mehr darauf zu haben.

Alternativ dazu bieten sich eindimensionale Tabellen mit entsprechenden Zugriffsfunktionen an, um Entscheidungstabellen unterschiedlicher Dimension zu implementieren. Wie Leo Brodies Beispiel zeigt sind auch bei dieser Implementierungsweise problemspezifische Lösungen zu erarbeiten. Die Tabelleninhalte lassen sich relativ einfach aus der bereits vereinfachten Entscheidungstabellen entnehmen. Für den Zugriff und die Berechnung des Telefontarifs sind weitere Worte notwendig, die das Entnehmen der Tabellenwerte und deren Berechnung zu einer Gesamtgebühr realisieren. Dies ist der aufwendigere und damit fehleranfällige Teil. Kommen weitere Faktoren hinzu, muß der gesamte Zugriffsmechanismus und die Berechnung überarbeitet werden.

Wie oben erwähnt bietet FORTH die Möglichkeit sich selbst erweitern zu lassen. Darunter verstehe ich, FORTH dazu zu benutzen, eine Sprache zu entwickeln, in der dann die Anwendung, sprich die Entscheidungstabelle, realisiert wird. Was liegt nun näher als die Tabelle selbst als Sprache anzusehen, also eine Tabelle als Sourcecode, die von FORTH kompiliert und zur Laufzeit ausgeführt wird und letztendlich in FORTH selbst definiert wird ?

Wie Eingangs erwähnt ist Alles und Jedes in FORTH ein Wort - alles außer Blank und Line Feed. Somit können alle Zeichen und Symbole des ASCII-Zeichensatz verwendet werden. Auch die des erweiterten ASCII-Codes, somit auch ┌>> und ┼┼ oder ┐┐ . Stichwort Pseudographik. Vor langer langer Zeit, als die Rechner noch PC-XT hießen "malte" man Tabellen, Liniengraphiken etc. mit Hilfe des erweiterten ASCII-Zeichensatzes. Macht heute keiner mehr, jedenfalls nicht um Dokumente zu erstellen. Damit lassen sich aber ( falls ein passender Zeichensatz wie OEM-Fixed oder MS-LineDraw vorhanden ist ) Entscheidungstabellen nicht nur entwerfen, sondern auch direkt implementieren. Entwurf = Implementierung = Dokumentation. Nicht mehr der Tabellenzugriff wird über Wort XY realisiert, sondern:

**die Tabelle wird ausgeführt.**

Sie wird in FORTH typischer Weise als Wort definiert und entsprechend aufgerufen.

Als Eingabe in die Tabelle müssen die Eingabeparameter übergeben werden, die Tabelle liefert den Ausgabewert. Die Implementierungsebene entfällt, so daß man automatisch in Entscheidungstabellen denkt und nicht mehr von der Implementierung her.

## Leo Brodie´s Telefontarif Ermittlung:

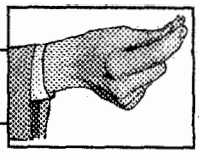
(die jetzt folgenden Beispiele stammen aus:
Leo Brodie: In FORTH denken, Hanser 1986
Seite 51 – 58 )

### Entscheidungstabelle:

| | voller Tarif | | ermäßigter Tarif | | niedriger Tarif | |
|---|---|---|---|---|---|---|
| | erste Minute | folge Minute | erste Minute | folge Minute | erste Minute | folge Minute |
| Direkt - wahl | 0,30 + 0,12/ 100 km | 0,20 + 0,12/ 100 km | 0,22 + 0,10/ 100 km | 0,15 + 0,10/ 100 km | 0,12 + 0,06/ 100 km | 0,09 + 0,06/ 100 km |
| Ver- mitt- lung | 1,20 + 0,12/ 100 km | 0,20 + 0,12/ 100 km | 1,12 + 0,10/ 100 km | 0,15 + 0,10/ 100 km | 1,02 + 0,06/ 100 km | 0,09 + 0,06/ 100 km |

### Strukturierte Sprache

Eine strukturierte menschliche Sprache (Englisch und/oder Deutsch o.a.) ist eine Art strukturierter Pseudocode, in dem sich unsere Tarifdarstellung ungefähr so lesen würde:



```
IF voller Tarif
   IF Direktwahl
      IF erste--Minute
           ,30 + ,12/100km
      ELSE ( weitere--Minuten)
           ,20 + ,12/100km
      ENDIF
   ELSE ( Amt)
      IF erste--Minute
             1.20+ ,12/100km
      ELSE        ( weitere--Minuten)
           ,20 + ,12/100km
      ENDIF
   ENDIF
ELSE ( nicht-voller--Tarif)
   IF ermaessigter-Tarif
    IF Direktwahl
      IF erste-Minute
           ,22 + ,10/100km
      ELSE ( weitere-Minuten)
           ,15 + ,10/100km
      ENDIF
    ELSE ( Amt )
      IF erste-Minute
          1,12 + ,10/100km
      ELSE ( weitere-Minuten )
          ,15   + ,10/100km
      ENDIF
    ENDIF
   ENDIF
ELSE ( niedriger-Tarif)
   IF Direktwahl
      IF erste-Minute
           ,12 + ,06/100km
      ELSE ( weitere-Minuten )
           ,09 + ,06/100km
      ENDIF
   ELSE ( Amt)
      IF erste-Minute
           1,02 + ,06/100km
      ELSE  ( weitere-Minuten)
      ,09 + ,06/100km
     ENDIF
   ENDIF
ENDIF
```

„Das ist doch alles sehr, sehr verwickelt. Es ist nur schwer lesbar, noch schwerer zu warten und am schwersten zu schreiben. Und noch dazu ist es für die Implementierung ganz und gar wertlos. Ich habe gar keine Lust, darauf noch weiter einzugehen." („In FORTH denken" Seite 53)

## Fortherweiterung Entscheidungstabellen:

Umsetzung der Telefontabelle mit Hilfe der Entscheidungstabellen-Syntax:

Das Wort D-DTv- ( lies: Definitionword-DesicionTable-Eingangsvar,- ) definiert die Entscheidungstabelle, wobei die beiden angehängten Kleinbuchstaben v- das Parameterhandling

```
: T-30/12
          d 30    d 12
          ;

: T-120/12
          d 120   d 12
          ;
```

```
D-DTv-: TARIF-TAB
```

| >> TEL-TARIF | >> TEL-WAHL | >> TEL-MIN | >> TEL-PREIS |
|---|---|---|---|
| >> VOLL-TARIF | >> DIREKT | >> ERSTE | T-30/12 |
| VOLL-TARIF | DIREKT | >> FOLGE | T-20/12 |
| VOLL-TARIF | >> VERMITTL | ERSTE | T-120/12 |
| VOLL-TARIF | VERMITTL | FOLGE | T-20/12 |
| >> ERM-TARIF | DIREKT | ERSTE | T-22/10 |
| ERM-TARIF | DIREKT | FOLGE | T-15/10 |
| ERM-TARIF | VERMITTL | ERSTE | T-112/10 |
| ERM-TARIF | VERMITTL | FOLGE | T-15/10 |
| >> NIED-TARIF | DIREKT | ERSTE | T-12/6 |
| NIED-TARIF | DIREKT | FOLGE | T-9/6 |
| NIED-TARIF | VERMITTL | ERSTE | T-102/6 |
| NIED-TARIF | VERMITTL | FOLGE | T-9/6 |

festlegen. Für diese Tabelle werden Eingangsparameter als Variablen ( v ) und die Ausgangswerte , falls vorhanden ( - ) , über den Stack übergeben.

In der obersten Zeile sind die Eingangsparameter und Ausgangsparameter definiert. Der Doppelpfeil >> deutet in der Tabellensyntax immer auf eine Definition einer Variablen oder Konstanten, abhängig vom Kontext, hin. TEL-TARIF, TEL-WAHL und TEL-MIN werden als Variablen definiert, in denen die Eingabeparameter an die Tabelle übergeben werden. Als Eingabeparameter sind die in den Eingabespalten definierten Konstanten zu verwenden. Dies sind VOLL-TARIF, ERM-TARIF und NIED-TARIF in der ersten Spalte, der TEL-TARIF Eingabespalte, DIREKT und VERMITTL in der TEL-WAHL Spalte und ERSTE, FOLGE in TEL-MIN. Die Eingabespalten werden von den Ausgabespalten durch eine Doppellinie getrennt. In der Ausgabespalte TEL-PREIS werden ausführbare Worte eingetragen, die zuvor definiert sein müssen. Um die Tabelle auszuführen, müssen die Eingabevariablen initialisiert werden.

Wird z.B.

VOLL-TARIF TEL-TARIF !

VERMITTL TEL-WAHL !

ERSTE TEL-MIN !

ausgeführt und anschließend die Tabelle mit

TARIF-TAB

aufgerufen, so wird

T-120/12

ausgeführt und es werden damit die beiden Dezimalwerte 120 und 12 auf den Stack gepackt.

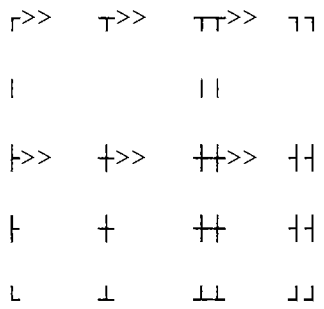
Alternativ zur Parameterübergabe über Variablen, können durch Verwendung von D-DTs- die Eingangswerte über den Stack übergeben werden:

VOLL-TARIF VERMITTL ERSTE TARIF-TAB

( Die Reihenfolge der Eingabeparameter spielt dabei keine Rolle )

## Tabellensyntax:

## Tabellenworte:

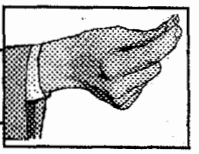


## Tabellendefinitionsworte:

D-DTv-:
D-DTvv:
D-DTs-:
D-DTsv:

Die Entscheidungstabelle besteht generell aus einem Eingabefeld und einem Ausgabenfeld. Das Eingabefeld wird durch einfache Linien beschrieben, das Ausgabefeld durch Doppellinien. Die Anzahl der Eingangsparameter legt die Dimension der Tabelle fest. Eine DT hat min. zwei Eingabewerte - ist somit min. zweidimensional. Einen Sonderfall stellen Tabellen mit einem Eingangswert da. Sie entsprechen einem herkömmlichen Array. Kennzeichnend für Eingangsparameter ist die Anzahl der möglichen Werte die er annehmen kann. Auch hier ist die kleinste sinnvolle Größe zwei. Parameter die nicht min. zwei Werte annehmen, bleiben in der DT unberücksichtigt. In der Literatur wird immer dann von einer vollständigen Entscheidungstabelle gesprochen, wenn alle kombinatorischen Möglichkeiten der Eingabeparameterwerte berücksichtigt werden. Die hier beschrieben DT-Syntax basiert auf der vollständigen Darstellung. Vereinfachung wird in der jetzigen Ausbaustufe nicht unterstützt. In den einzelnen Spalten des Eingabefeldes werden als "Name" die Parameterwerte eingetragen. Sie stellen Konstanten dar, die definiert sein müssen. Die Definition erfolgt in jedem Fall durch die Markierung mit dem Doppelpfeil ( >> ), der dem Konstantenwort vorausgeht. Eine bereits auf diese Art definierte Konstante kann in einer nachfolgenden Tabelle mit IDENTISCHEN /!\ Eingabeparametern verwendet werden, ohne neu mit >> definiert werden zu müssen.

Das Ausgabefeld besteht aus einer oder mehreren Spalten. In den Ausgabespalten sind ausführbare Worte einzutragen. Dies können also Worte, Variablen oder Konstanten sein, die bereits vorher definiert wurden. Die Ausgabe-Worte einer Zeile, bei mehreren Spalten, werden von links nach rechts nacheinander ausgeführt.



Die 1. Zeile der DT definiert Eingabe- und Ausgabeparameter als Variablen. Es können beliebige, bereits vorher definierte Variablen verwendet werden. In der 2. Zeile wird der Tabellenname mit Hilfe des Definitionswortes D-DTxx definiert. In den drauffolgenden Zeilen wird die vollständige Tabelle mit Eingangswerten und den zugehörigen Ausgabeworten eingetragen.

## Ausführung der Tabelle:

Die Ausführung der DT zur Laufzeit wird durch Aufruf des Tabellennamens erreicht. Es müssen, wie bei jedem Wort üblich, die Eingabeparameter vorliegen. Je nach Tabellenart erfolgt die Übergabe über die Eingabevariablen ( D-DTv- ) oder über den Stack ( D-DTs- ). Falls das ausgeführte Ausgabewort Werte liefert, werden diese auf dem Stack ( D-DTv- ) oder in der Ausgabevariablen ( D-DTvv ) übergeben. Dabei ist zu beachten, daß pro Ausgabespalte eine Ausgabevariable definiert werden kann, die auch nur einen Wert aufnimmt. werden mehrere Parameter erzeugt, müssen diese auf dem Stack übergeben werden. Auch bei einer Variabelenausgabe wird zur Laufzeit zunächst der Stack als Zwischenspeicher verwendet, bevor die Eintragung in die Ausgabevariablen erfolgt.

## Tabellenvarianten:

In der Praxis können Entscheidungstabellen in den unterschiedlichsten Varianten auftreten, wovon in der bisherigen Ausbaustufe die wichtigsten unterstützt werden.

Folgende Merkmale werden unterschieden:

- Art der Parameterübergabe ( Ein- und Ausgabeparameter über Variable oder Stack )

| | |
|---|---|
| Eingabe durch Eingabevariablen: | D-DTvx |
| Eingabe über Stack: | D-DTsx |
| Ausgabe durch Ausgabevariablen: | D-DTxv |
| Ausgabe über Stack | D-DTx- |

- Anzahl der Eingabeparameter und Ausgabeparameter:

  Für jede Eingabe wird eine Spalte angelegt. Sonderfall: Ein Eingangswert

  Für jede Ausgangsgröße wird eine Spalte angelegt.

- Definition von Eingabeparameter in Form von Eingangsvariablen:

  Eingangsvariablen werden durch X>> in der ersten Zeile im Eingabefeld definiert.
  X>> darf nur dann entfallen, wenn eine bereits zuvor definierte Variable verwendet wird.
  Die Definition ist auch dann erforderlich, wenn sie nicht zur Parameterübergabe herangezogen wird.

- Definition von Eingangskonstanten

  Eingabekonstanten werden durch X>> definiert.
  X>> darf nur dann entfallen, wenn eine zuvor definierte Konstante verwendet wird. Erfolgt diese Definition außerhalb der DT, muß sicher-gestellt sein, daß es sich bei der Konstanten um eine Definition handelt, die im IDENTISCHEN Eingabefeld erzeugt wurde.

- Definition von Ausgabeparameter in Form von Ausgangsvariablen:

  Ausgangsvariablen werden durch X>> in der ersten Zeile im Ausgabefeld definiert.
  X>> darf nur dann entfallen, wenn eine bereits zuvor definierte Variable verwendet wird. Die Definition ist auch dann erforderlich, wenn sie nicht zur Parameterübergabe herangezogen wird.

- Definition von Ausgangskonstanten

  Ausgabekonstanten werden durch Xc>> definiert.
  Xc>> darf nur dann entfallen ,wenn eine bereits vorher definiertes Wort, Konstante oder Variable verwendet wird. Als Ausgabeparameter wird beim Ausführen der DT der Konstantenwert geliefert

- Definition von Ausgangsvariablen

  Ausgabevariablen werden durch Xv>> definiert.
  Xv>> darf nur dann entfallen ,wenn eine bereits vorher definiertes Wort, Konstante oder Variable verwendet wird.
  Als Ausgabeparameter wird beim Ausführen der DT die Adresse der Variablen geliefert

## Implementierung der Entscheidungstabellensyntax:

Der FORTH Compiler arbeitet inkrementell und erlaubt die Ausführung von FORTH-Worten unmittelbar ( IMMEDIATE ) während des Kompilierens. Liniengraphik Symbole wie ¬ sind als IMMEDIATE definiert und werden zur Compilerzeit ausgeführt. Damit läßt sich die Umsetzung der Tabellengraphik in FORTH Code auf relativ einfache Weise realisieren.

Beim kompilieren der Tabelle werden die Variablen im RAM-Bereich angelegt und im ROM-Bereich zunächst die Konstanten und Anschließend die PFA der Ausgangsworte nachei-

nander ins Dictionary eingetragen. Zwischen jedem Ausgangswort wird ein Exit kompiliert, so daß durch einfaches CALL ein, oder bei mehreren Ausgangsspalten auch entsprechend mehrere Worte, aufgerufen werden. Die Eingangskonstanten werden so berechnet, daß deren Addition direkt das zugehörige Ausgabewort adressiert. Zur Laufzeit werden lediglich die übergebenen Eingangsparameter ( Konstanten ) aufsummiert ( daher ist die Reihenfolge der Übergabe beliebig ), die Summe zum Anfang der Ausgabetabelle hinzuaddiert ( die EXIT' sind dabei schon berücksichtigt ) und ein CALL mit dieser Adresse ausgeführt. Beim H8/300 beträgt die Ausführungszeit für eine Tabellenzeile zwischen 2µs und 4 µs ( je nach Parameterübergabe ), bei einer Taktfrequenz von 16 MHz. Dies entspricht 16 einfachen Assemblerbefehlen bzw. 5-8 Befehlen mit 2 bis drei Taktzyklen. Dabei wurden zentrale Elemente in Assembler optimiert. Unabhängig von Tabellengröße und Eingangskombination bleibt die Laufzeit konstant.

---

*Henry Vinerts unser langjähriger 'Auslandskorrespondent' aus dem Silicon Valley sendet uns, zuverlässig wie immer, drei Berichte über die Treffen SVFIG (Silicon Valley Chapter of the Forth Interest Group)*

*Herzlichen Dank an Thomas Beierlein für seine stete Übersetzungsarbeit. Hier sei Henry Vinerts selbst zitiert, der bei anderer Gelegenheit meinte: „I have never corresponded directly with Dr. Beierlein, but I certainly would like to let him know that he impresses me with the fine translations of my hard-to-understand American slang."*

*MB*

# From the big Teich ...

**Henry Vinerts**

**- Juni**

Lieber Martin,
zuallererst möchte ich mich entschuldigen, daß ich das Junitreffen der Silicon Valley Forth Interest Group übersprungen habe. Meine Verbindung zum Schachspiel zieht mich von Forth fort. Falls Du möchtest, kannst Du den Beweis unter http://www.chessdryad.com/photos/photos_17.htm nachprüfen. Der alte Bursche im blauen Hemd auf dem dritten Foto dieser Seite "Ihr stets Ergebener". Ich tauche auch auf den Fotos auf Photos_19.htm mit meinem Schachfreund Hans auf.

Wie ich herausbekam, war auf dem Juni Treffen auch ein Sprecher von National Instruments mit einer Präsentation von LabVIEW "Real-Time". Für alle Interessierten gibt es mehr Informationen auf folgender Webseite: http://digital.ni.com/productpages/nilabviewrt.nsf/main?readform.

Ich glaube nicht, daß Dr. Ting zum Juni-Treffen in der Stadt war, daher wurde wahrscheinlich ansonst ziemlich wenig mit technischem Inhalt den Anwesenden präsentiert.

Das Juli-Treffen brachte an die 20 von uns zurück zu einem Morgen voll mit Dr. Ting's "Zeigen und Berichten" über seine Arbeit mit ATMEL und XILINX Boards und eForth und Sprach-Synthesizern. Da gibt es eine recht gute Chance für eines der Boards an Bord eines Satelliten zu kommen und seinen Anteil an der Weltraumerkundung zu leisten.

Die jüngste Arbeit von Ting dreht sich um Sprach-Synthese, hauptsächlich für die chinesischen Märkte, in denen es eine Menge Anwendungen für Cell Phones, Taschenrechner, handheld Spiele usw. gibt. Wie Du weißt, kann die chinesische Sprache nicht so einfach mit westlichen Tönen simuliert werden, vor allem aufgrund der Tonhöhenänderungen (der Modulation - Inflektion) der Vokale. Ein Wort wie z.B. 'Ma' kann 'Mutter', 'Hanf', 'Pferd' oder 'Fluch' bedeuten, abhängig davon welche der vier Inflektionen für den Vokal 'a' benutzt wird. Ich glaube nicht, daß die Welt einen besser geeigneten Mann zur Entwicklung des Sprach-Synthese Algorithmus und der Software finden konnte, als einen zweisprachigen chinesischen PhD Chemiker, der außerdem noch einer der weltbesten Forth-Experten ist.

Ich möchte auch noch sagen, daß meiner Meinung nach, die SVFIG ohne Dr. Ting und George Perry wie ein paar verlorene Kinder auf der Suche nach einem Zirkus wäre. Diese beiden Burschen gaben uns die Führerschaft, die diese Gruppe brauchte, zumindest während der letzten 10 Jahre die ich dabei war. Ting hat uns nicht nur etwas Technisches beizubringen, wann immer die Gelegenheit es erfordert, sondern er (und George) kümmern sich ständig um die Gruppe und sorgen dafür daß der Kaffee oder und die Donuts oder Bagels für die ersten Ankömmlinge bereitstehen. George ist der talentierteste, gewillte und unermüdliche Vorsitzende und Zeremonienmeister der Gruppe den ich mir jemals vorstellen könnte. Natürlich muß er oft den harschen Schulleiter spielen um seine 'Schüler' zu den Vorlesungen zu sammeln, da Spielen, Plaudern und Schwatzen für die Mehrheit von uns anziehender zu sein scheint.

Obwohl sogar Chuck Moore uns den ganzen Tag mit seiner Anwesenheit ehrte, war der Höhepunkt des Tages ein weiteres

dieser Wundervollen Barbecues, dank Mr. und Mrs. Ting (Mrs. Ting kommt normalerweise nicht zum Treffen, aber Sie hatte die schmackhaftesten Steaks fürs Grillen gefunden. Ich bin sicher, die Forth Gruppe ist dank Ihrer Vorsorge viel glücklicher.) Chuck hielt dieses mal keinen Vortrag, aber er ließ seine Webadresse an der Tafel: www.colorForth.com, so daß jeder der es will beobachten kann womit er sich zur Zeit beschäftigt.

John Hall hatte zwei Macintosh Laptops mitgebracht - den Titanium G4 und den iMac, wobei er davon ausging, uns ca. 15 Minuten darüber zu berichten. Stattdessen hielt unser Interesse an den Macs John, sogar nach dem Essen, für über zwei Stunden auf der Bühne, bis es schließlich Zeit war nachhause zu gehen. Es kann sein, daß ich vorher nicht erwähnt habe, das John für Apple arbeitet. Seine Hauptverantwortung liegt bei dem Forth-basierten Open Firmware, welches einen Gerätebaum während des Bootens des Computers aufbaut. Dieses Open Firmware ist ein Abkömmling von Mitch Bradleys Open Boot Prom für Sun Microsystems. Es wird in den Macintosh Produkten seit über 5 Jahren eingesetzt, beginnend mit den iBooks. Soweit ich weiß, ist John noch der amtierende Präsident der FIG, aber ich kam nicht dazu ihn nach Neuigkeiten an dieser Front zu fragen.

Ansonst, geht alles wie gehabt weiter, außer daß das Durchschnittsalter der SVFIG Teilnehmer sich jedes Jahr um eins erhöht.

Auf Wiederhoehren until next time,

Henry

– **August**

Hallo, Martin,
das August-Treffen der Silicon Valley Forth Interest Group schien mir in mehr als einer Hinsicht etwas unüblich zu sein.

Zuallererst wegen dem Terminkonflikt mit dem Cogswell College, welches uns seit Juli 1996 beherbergt hatte. Das Treffen war dieses mal am dritten anstatt wie sonst dem vierten Samstag im Monat.

Als zweites enthielt der Zeitplan des Treffens, den wir mit der Post oder per e-mail erhalten, überhaupt keine vorgesehenen Redner. Ich erwartete daher das nur wenige Teilnehmer erscheinen würden und nahm nicht an, daß ich die gesamten sechs Stunden dort verbringen würde (Ich habe mich bei beidem geirrt!).

Drittens, waren da, obwohl die übliche Anzahl Leute erschien, mindestens 4 oder 5 Leute, einschließlich einer jungen Dame, die ich auf vorherigen Treffen weder gesehen noch gesprochen hatte.

Und zu guter Letzt, _war_ Dr. Ting da, obwohl er nicht mal als Redner aufgeführt war! Und er führte uns alle zu "Ting's Frühstücks Buffet" welches neben dem üblichen Kaffee und den Donuts dieses mal auch Pizza und in chinesischem Tee hartkochte Eier bot! (Es war dann so, daß Dr. Ting ungefähr ein und eine halbe Stunde über die Herausforderungen sprach, chinesische Zeichen auf einem Chip unterzubringen. Dies ist sein neuestes Projekt.)

Das Thema des Treffens drehte sich um Forth unter Unix oder Linux. Obwohl sich niemand für Vorträge gemeldet hatte, war jeder eingeladen sein Wissen einzubringen. Und einige sachkundige Personen waren auch gekommen um mitzumachen.

Charley Shattuck von AMR Forth in Sacramento begann die Präsentationen. Deine Landsleute hören vielleicht gern, daß er dazu GForth (von Ertl, Paysan, Wilke?) unter Linux über eine serielle Schnittstelle nutzt. Er berichtete, daß er auch BigForth probiert hatte - es war aber zu groß für seine Zwecke und wahrscheinlich auch zu schwer zu verstehen.

Gary (sein Nachname ist irgendwo in meinen früheren Berichten) berichtete uns über seinen Besuch in Mexico City, der in Verbindung mit einem seiner letzten Wartungs-Aufträge von Forth, Inc. (das ist Elizabeth Rather's Firma). Forth, Inc. hat sich etwas verkleinert und ist an einen anderen Ort Los Angeles umgezogen. Aber Forth läuft in Mexico -- eine Reihe von 166 MHz 486s, die einen Produktionsprozeß in einer Owens-Corning Glasfaser-Fertigung steuern.

John Sokol war das erste mal bei uns. Er ist ein Unix Spezialist und an einer Reihe von Projekten beteiligt über die er uns berichtete: Chuck Moore's Processoren, Jeff Fox's tragbare Computer, Jef Raskin's Wiedergeburt der Canon Cat, etc.

# From the big Teich ...

Ron Hochsprung (Du solltest hören, wie er seinen Namen betont um es für die Amerikaner einfacher zu machen) arbeitet für Apple (zusammen mit John Hall), hauptsächlich mit dem "Open Firmware" welches komplett in Forth geschrieben ist. Es basiert auf Wil Baden's ThisForth und auf F-code und kurbelt (Ich erinnere mich an ein Wort wie "Kurbelstange" aus meinen Tagen in Deutschland; das ist es was Open Firmware in meiner Vorstellung ist) die meisten Macintosh's in dieser Welt an.

Nun zum Höhepunkt des Treffens -- das war nach Dr. Ting's Rede am Nachmittag -- die junge Dame, deren Namen ich nicht verstanden habe, erzählte uns wie sie durch eines unserer Mitglieder auf Forth gebracht wurde, als sie nach der besten Sprache suchte, die ihr bei der Entwicklung einer übers Netz benutzbaren "Lern-Maschine" helfen konnte. Diese Maschine "würde einem den Geist öffnen, wenn jeder andere verrückt würde". Unglücklicherweise war unsere Zeit zu Ende, da wir den Raum im College nur bis 16 Uhr bekommen hatten. Hoffentlich kommt sie das nächste mal wieder, um ihre Geschichte zu Ende zu bringen.

Nun, Du siehst, da gibt es eine Menge Unterhaltung und Weiterbildung, die nach wie vor die Forth'er vom Silicon Valley forthers zu den treffen zieht. Der Hauptgrund diese Zeilen zu schreiben ist für mich, damit ihr wisst, daß Forth hier nach wie vor gesprochen wird und daß ungeachtet der Sprache, die Forth-Programmierer rund um die Welt wohl aus dem selben Holz geschnitzt sind.

Keep smiling!

Henry

– **September**

Lieber Martin,
die Silicon Valley Forth Interest Group traf sich gerade elf Tage nachdem die Welt durch den terroristischen Anschlag auf die U.S.A. erschüttert wurde. Ich denke, ohne Dr. Ting hätte sich das Treffen Gesprächen über die Weltprobleme und andere bedrückende Themen gewidmet, aber der "gute, alte" Ting übernahm sofort nach den Tradition gewordenen Kaffee, Tee und Gebäck, und fesselte etwa 15 bis 18 von uns für 2 Stunden mit seinem Bericht, wie er Forth zu Kindern in Taiwan brachte. Wie Ihr seht, können die taiwanesischen Hauptschüler nicht einmal mit dem Erlernen des Programmierens beginnen, bevor sie nicht die Klassenstufen erreicht haben in denen Englisch gelehrt wird. Ting hat sich schließlich Win32Forth zugewendet, seine Meldungen in Chinesische Grafik übersetzt und arbeitet an der Eingabe und Übersetzung chinesischer Zeichen. Er sagt daß Forth die beste Programmiersprache für Chinesen ist. Die Zeichen sind sehr mit Forth verwandt, erweiterbar; elementare Zeichen werden kombiniert um andere zu bilden. (Ich glaube er sagte, daß es 55401 Zeichen in einem der gegenwärtigen Systeme gibt, aber man benötigt _nur_ ungefähr 600 Basisformen).

Falls jemand von Euch weiß wie man eine Soundkarte mit Win32Forth ansteuert, dies ist ein Gebiet auf dem Ting Rat sucht. Er sagte, daß er mit der Zeit gut genug mit Win32Forth bekannt wurde, so daß er sogar eventuell ein Handbuch für es schreiben könnte.

Nach der Mittagspause gab es einen informellen Informationsaustausch, das übliche Wortgeplänkel. einige politische Ansichten, ein wenig über Linux X-Windows, Listen mit interessanten Web-Sites, usw.... Schließlich, ohne irgendeinen geplanten Redner, stellte es sich heraus, daß Dr. Ting wieder den Rest des Tages mit seinem niemals endenden Vorrat an technischen Informationen füllen würde. Ich bedauere, daß ich das Treffen eher verlassen mußte. Damit muß mein September-Bericht an dieser Stelle enden.

Bis zum nächsten mal dann,

Henry

# Gehaltvolles

**zusammengestellt und übertragen von Fred Behringer**

**FORTHWRITE der FIG UK, Großbritannien**

**Nr. 112 Juli 2001**

Graeme Dunbar ist neuer Bibliotheksverwalter. Diese Aufgabe hat er von Sylvia Hainsworth übernommen, die mit ihrem Mann in den Ruhestand nach Spanien verzogen ist. Die University of Teeside hat die Forthwrite abboniert.

## 2 Forth News
**Dave Abrahams <d.j.abrahams@cwcom.net>**

64-bit Forths; Ficl v2.06; eForth; Quest32; IRE-2001-Workshop über Java Virtual Machines; ANS-Forth-Internationalisierung; Machine Forth und Color Forth; ProForth VXF 3.4; ProForth VXF freie Prüfversion; RTX 2000; Shboom-Chip; TpForth 3.2; Forth und super-skalare Prozessoren; Chuck Moore hat jetzt eine eigene Website: http://www.mindspring.com/~chipchuck

## 5 Forth for NEAR Spacecraft
**Joe Anderson <jia@jia.abel.co.uk>**

"Near-Earth Asteroid Rendezvous". Eine Raumsonde wurde 1996 ausgesandt, um im vergangenen Jahr Eros auszukundschaften. Der Autor und Berichterstatter hat viel E-Mail-Informationen von John R. Hayes von der Johns Hopkins University eingebaut: http://www.jhuapl.edu/ .

Die Umstände waren vorteilhaft, man entschied sich, die Raumsonde auf Eros aufzusetzen. Die Software mußte schnell umprogrammiert und den neuen, eigentlich nicht vorgesehen gewesenen Gegebenheiten angepaßt werden. "In Forth ging das glänzend."

Wo hat John R. Hayes die Forth-Programmierer herbekommen, wo man doch in seiner Universitäts-Umgebung Forth gegenüber Unverständnis zeigt? Er hat sie ganz schnell selbst eingewiesen. "In Forth geht das."

"Wer sich für logische Zusammenhänge und Elektronik interessiert, ist für Forth geeignet. Wer sich nur oberflächlich für das interessiert, was er per Programm der Maschine abfordern möchte, soll ruhig bei C oder C++ bleiben."

## 10 Chairman's Message
**jeremy.fowell@btinternet.com**

Jeremy stellt sich als neuer Vorsitzender (bei unseren englischen Forth-Freunden gibt es derer nur einen) vor. Nachfolger von Chris Hainsworth, der nach Spanien in den Ruhestand gegangen ist. Mit Forth hat Jeremy kurz nach 1980 die erste Bekanntschaft gemacht. Zur englischen Forth Interest Group ist er 1987 gestoßen. Er hält es mit Chuck Moore: "We need to beat the drum" (um Forth bekannt zu machen, müssen wir die Trommeln rühren - Klappern gehört zum Handwerk).

## 11 F11-UK
**jeremy.fowell@btinternet.com**

PC mit DOS, Pygmy-Forth-Compiler, Motorolas HC11, 47 engl. Pfund, 4 Pfund Porto, 25 US-Dollar für Registrierung von Pygmy-Forth.

## 12 "Quikwriter" Proposal

Jenny Brien arbeitet ehrenamtlich in einer Organisation mit, die sich um Menschen kümmert, die ihre Finger nicht normal bewegen können. Es geht um geeignete Computer-Eingabemöglichkeiten. Es sind verschiedene Joystick-Lösungen vorgeschlagen worden, die zum Teil auf das F11-Hardwareprojekt der FIG UK zurückgreifen. 3 Seiten geballter Aktivität der englischen Forth-Freunde.

## 15 euroForth 2001

23. bis 26. November 2001 auf Schloß Dagstuhl bei Saarbrücken. Als Ehrengast wird Chuck Moore erwartet.

## 16 Forth-Gesellschaft - Tagung 2001
**Chris Jakeman <cjakeman@bigfoot.com>**

Chris, er war "Special Speaker" auf unserer Tagung, berichtet. Er ist voll des Lobes. Wo hat er nur die ganzen Informationen her? Immerhin besteht ja seit dem historischen Turmbau eine gewisse Sprachverwirrung zwischen den Menschen. Wie ist er an die Informationen über die Aktivitäten von Adolf Krüger und Johannes Reilhofer gekommen? Und über den Drachen weiß er auch zu berichten. Und über Ulli Hoffmann als neuem Direktor. Und am Roboter-Wettbewerb hat er sogar selbst teilgenommen. Und seinen Lesern schlägt er vor, in der FIG UK ähnlich umfangreiche Jahrestagungen einzurichten. Fürs nächste Jahr, zum Eingewöhnen, hätte er gern ein paar der englischen Forth-Freunde dazu animiert, uns in der Münchener Umgebung einen Besuch abzustatten.

## 19 The FIG UK Awards of 2000

Die Preisträger von 1999 waren Alan Wenham und Jeremy Fowell. Diesmal hat Keith Matthews den Allgemeinen Forth-Preis für seine stetige Arbeit hinter den Kulissen als Schatzmeister bekommen. John Tasgal bekam den Forthwrite-Preis

für seinen ausgezeichneten (auch im Internet erhältlichen) Artikel über Chuck Moores Machine Forth und Color Forth.

### 20 Arithmetized Logic in Forth - An Introduction
**Fred Behringer und Chris Jakeman**

Chris Jakemans Aufbereitung einer Kurzfassung des Tagungsvortrags des Rezensenten.

### 25 Buchbesprechung: Extreme Mindstorms - An Advanced Guide to LEGO Mindstorms
**David Abrahams <d.j.abrahams@cwcom.net>**

David ist Fachmann in Automatisierungstechnik und Software-Engineering durch und durch. Er beschreibt das Vier-Autoren-Buch auf sieben (!) Seiten und lobt es sehr. Mit Ralph Hempel, der in diesem Buch das Kapitel über pbForth bestreitet, hat er per E-Mail ein Interview geführt, das er wiedergibt. Der Rezensent (hat natürlich das Buch auch und) schließt sich dem Höchstlob an.

### 32 Deutsche Forth-Gesellschaft

Diesmal wieder unsere Werbeanzeige.

### 33 JenX - A very simple XML parser
**Jenny Brien <jennybrien@bmallard.swinternet.co.uk>**

XML ist das Zauberwort des Augenblicks. XML-Kennwörter ("tags") in einem Parser so interpretieren, als ob es Forth-Worte wären. Zusammenarbeit mit Leo Wong. Wir werden mehr hören.

### 36 Three Free Forths and an OS too!
**Paul Bennett <PEB@amleth.demon.co.uk>**

Der Autor arbeitet seit Weihnachten 2000 daran, von den Microsoft-Produkten wegzukommen. Er hat sich für FreeBSD entschieden. Er möchte nicht unbedingt andere überzeugen. "Es ist nicht jedermanns Sache, auf ein anderes Betriebssystem umzusteigen." Das Betriebssystem BSD enthält FICL als Bootstrap-Lader. Außerdem betrachtet der Autor PFE und GForth.

### 40 Letters

3 Briefe: Einer von Paul Bennett als Antwort auf Andrew Holts Behauptung "Die beste Zeile Code ist die, die ein anderer geschrieben hat". Ein Brief von John Matthews, der nach einer größeren Zahl von R65F12 (6502 mit Forth-Kern) sucht. Ein Brief von Douglas Neale, der das Projekt eines SQL-Paketes in Forth ins Leben rufen möchte.

---

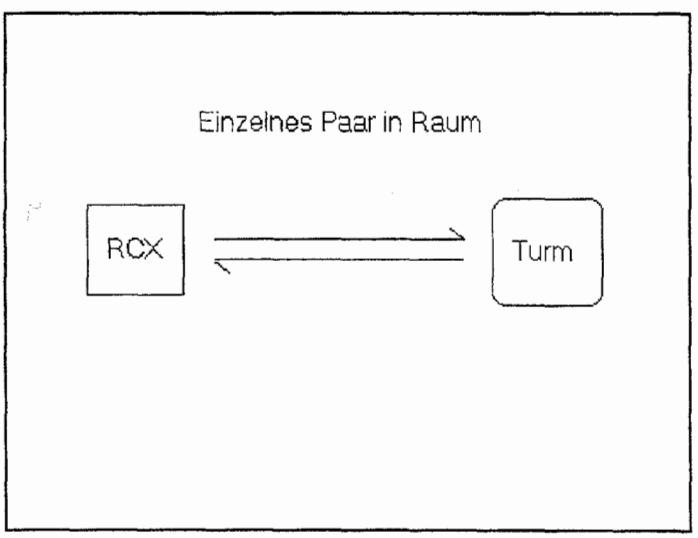
# Zur LEGO Mindstorm Infrarot Datenübertragung.

## Untersuchung der Hardware im IR-Turm und im RCX 1.0

Michael Kalus, Adolf Krüger

Ziel der Untersuchung war es herauszufinden ob die Hardware es ohne Änderungen zulässt mehrere LEGO RCX damit multiplex zu betreiben. Ist es damit möglich, an 2 und mehr RCXse, die gleichzeitig in einem Raum sind, gleichzeitig und unabhängig voneinander von einem oder sogar mehreren Türmen aus Daten zu senden und auch zu empfangen? Oder die RCXse sich gegenseitig "unterhalten" zu lassen?

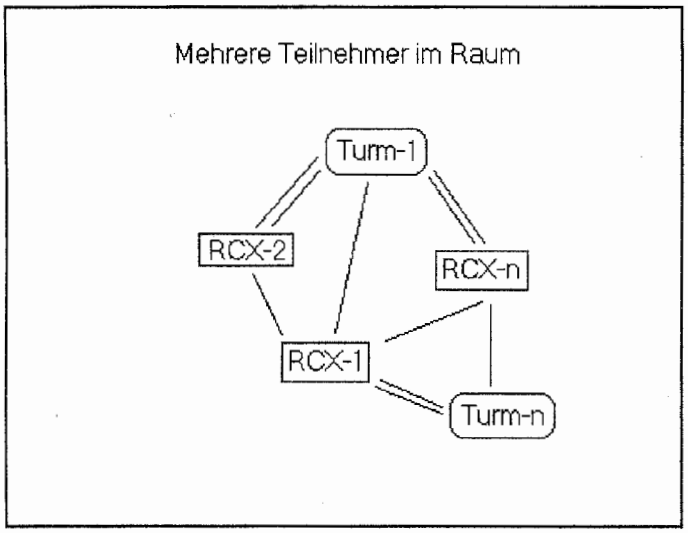
Die Hardware war ja von LEGO wohl eigentlich nur gedacht für ein Paar in einem Raum gleichzeitig, und das zeitmultiplex., also schön einer nach dem anderen, und das ohne Protokoll, ohne Absprache, sozusagen nicht interaktiv sondern klassisch im Batch. Entwurf am PC, compilieren, downloaden in den RCX, probieren ob's geht, sonst: Versuchs noch mal, ich weiß du kannst es besser! Also die klassische Herausforderung für jeden der Forth kennen gelernt hat.




Im Schulbetrieb hingegen, bei Programmierertreffen und bei den Robotikern hat sich genau das als sehr lästig herausgestellt. In dieser Not wurde natürlich sofort das mobile Ortsmultiplexverfahren entwickelt, bei dem durch übergestülpte Pappkartons die gültige Anordnung wieder hergestellt wird - elegant, da getreu nach dem Motto "keep it simpel" geschneidert. Aber es taucht dann natürlich gleich die Frage auf warum denn dann nicht via Kabel übertragen wird? Was soll diese IR-Geschichte, wenn es sowieso gemeinsam nicht geht - fängt doch da der Spaß erst richtig an. Ich sehe schließlich schon das LEGO Bat-

telgame vor meinem geistigen Auge. Ihr vermutlich auch. Also ran an die Frage: Forth, kannst du das besser?

Fakt ist: In der vorgesehenen einfachen IR Übertragungsart stören sich die Quellen gegenseitig. Eine multiplexe Anordnung gleichzeitig in einem Raum ist in der vorhandenen Firmware nicht vorgesehen. Aber wird es mit einer anderen Software gehen oder macht die vorgegebene Hardware das unmöglich, und wenn, was müsste alles geändert werden?




## Wie funktioniert der Turm?

Angeschlossen an einen COMx des PC mit dem konventionellen 9-poligen D-sub Stecker kann via Terminal dem RCX ein Image geschickt werden, das sieht seine Firmware vor. Ist das pbForth so einmal geladen, lassen sich interaktiv Zeichen austauschen über die IR-Schnitstelle mit dem RCX Baustein. Doch geschieht dies ohne jedes Protokoll für eine Verständigung: Via Terminal wird der blanke Bitstrom an den IR-Turm geschickt und die IR-Lichtblitze werden in den umgebenden Raum abgegeben. Jeder im Raum anwesende RCX-Klotz kann damit machen was er will: Schlafen, empfangen oder auch dazwischenquatschen - und dann ist der IR-Datenstrom im Raum nur noch Müll und von niemandem mehr zu entziffern. Übrigens auch ohne Forth ist das so beim Mindstorm, seine Firmware ist so.

Gemacht wird das so: Sowohl im Turm wie im RCX sind je ein IR-Sender und ein Empfänger untergebracht, die üblichen IR-Fernsteuerung wie beim Fernseher. Da hat es der Modellbau besser, da gibt es wenigstens Funkfrequenzbänder so das mehrere Teilnehmer gleichzeitig ihre Modelle fahren oder fliegen lassen können.

Öffnet man den Turm, ist darin eine kleine Platine mit 2 Stück IC 74LS132 zu je 4 Schmitt-Triggern, einigen Transistoren, Widerstände, Kondensatoren und oben am Rand rechts 2 IR Sende-Dioden und links ein IR-Empfängerbaustein. Als Stromversorgung dient eine 9V-Blockbatterie.

Die IR-Dioden arbeiten als einfache IR-Blitzer mit einer Blitzfrequenz von 38,5KHz, und können von der Datenleitung der seriellen Schnittstelle aus ein oder aus geschaltet werden. Dabei bedeutet ein 'Null-Bit' das die IR-Blitzefrequenz abgestrahlt wird, eine 'Eins' bedeutet IR-Stille - oder besser gesagt Dunkelheit.

Der IR-Empfänger setzt eine solche IR-Blitze-an-aus-Folge wieder um in eine entsprechende Bitfolge.

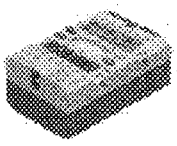
Er braucht 150us um zu beschließen das nun eine passende IR-Frequenz im Raum vorhanden ist und schaltet dann seinen Datenausgang auf 'low' um. Und er braucht gut 250us bis er merkt das nun keine solche IR-Frequenz mehr da ist bevor die Flanke zum Pegelwechsel auf 'high' kommt.

Die verdrahtete Logik im Empfänger bewirkt also, das die Blitze zu Pulsen integriert werden, wirksam ist dabei zusätzlich eine lichtabhängige Verstärkerschaltung und ein Frequenzfilter. Hinzu kommt, das eine Stromsparlogik eingebaut ist - ein nachtriggerbares Monoflop hält die Schaltung noch 5 Sekunden in Betrieb nach dem letzten gesendeten Bit, dann wird abgeschaltet. Geweckt wird mit dem nächsten Bit das gesendet wird. Es gibt keine eigene Selektleitung dafür - die Aktivität der Platine ist gekoppelt an den Datenstrom. Die Anwesenheit von IR-Licht, das empfangen werden könnte, kann von der Schaltung nicht erkannt werden - die Logik wird davon nicht geweckt. Das liegt so fest, kann nicht durch Programm beeinflusst werden.

Wir haben nun untersucht wie sich diese Hardware auf die Datenrate auswirkt. Denn soll ein Multiplex-Betrieb zwischen den Teilnehmern in der Praxis noch zufrieden stellen, wird die erzielbare Datenrate zum wichtigsten Faktor. Wir haben als untere Grenze mal 300 Bit/s angenommen für die niedrigste Rate tatsächlich übermittelter Daten die gemultiplext zwischen den Teilnehmern noch garantiert werden sollte.

Schaut man sich daraufhin an was der IR-Sender im Turm noch abstrahlt wenn die rohe Datenrate zum Turm hin raufgesetzt wird, ergibt sich folgendes Bild.

Bei 2400 Bit/s bilden 15 IR-Blitze 1 Bit Information 'Null', die IR-Blitzfrequenz ist 38,5KHz. Erhöht werden konnte bis 19200 Bit/s, damit konnte so gerade noch korrekt gesendet werden. Dann formten allerdings nur noch 2 IR-Blitze eine 'Null'. Gemessen wurden ein 54us Intervall für diese 2 Blitze, das ließ sich am Oszilloskop noch schön zeigen. Da weiter oben im Text aber schon dargestellt wurde das allein 150us gebraucht





werden bis bei größerer Entfernung (1m) das Signal erkannt wird, und 250us um es zu beenden, landen wir wieder bei realistischen 2400 Bit/s welche der Empfänger mit machen würde.

Im IR-Empfänger des Turmes sah es trotzdem bei unserem Modell bis 4800bit/s noch gut aus, die Bits wurden doch noch richtig erkannt. Allerdings hatte die logische 'Eins' schon nur noch 1/3 der normalen Pulsbreite von der bei 2400 Bit/s. Bei noch höheren Datenraten sorgte der Filter nun dafür, das die 'Einsen' immer schmaler und kleiner wurden und dann nicht mehr erkannt werden konnten. Die Signale gingen in den 'Nullen' unter.

Bei 9600 Bit/s konnten nur noch zwei 'Einsen' in Folge, also ein breiterer Puls - wieder als logisches Bit 'Eins' erkannt. werden. So gab z.B. die Folge 01010101 (HEX 55) im Empfänger nur noch eine 00000000 Folge (HEX 00) und erst 00110011 00110011 (2xHEX33 in Folge) verschmolz zu 01010101 sodass wieder HEX55 erschien - wenn man Glück hatte und sich gerade richtig synchronisierte. Bei dieser Datenrate schrumpfte also der unterscheidbare Zeichenvorrat schon auf die Codes mit vollen 11-Pulsen zusammen. Aus dem Vorrat von 12 Zeichen (0000 bis 1111) wurden dann nur noch 3 Bitfolgen von Null unterschieden: 3, 6 oder 7 und 11 oder 12. Dabei wurde die 7 mal als 1, mal als 3 interpretiert; die 12 unsicher mal als 1 oder 3 oder 5 oder 9. Der Zeichenvorrat schrumpfte so auf grad mal 4/12 zusammen.

Das hat zur Folge das bei einer rohen Datenrate von 9600 bit/s die Anzahl von Teilnehmern, die noch unterschieden werden können, schon auf 2 sinkt, wenn ein Codemultiplex-Verfahren eingesetzt werden soll.. Denn 2 Zeichen pro Teilnehmer sollten gegeben sein um deren 'Nullen' und 'Einsen' darstellen zu können. Und das sind nun mal doch zu wenig Teilnehmer.

Bleibt man hingegen bei 2400 Bit/s, läßt sich der ganze Zeichenvorrat an 8-Bit-Folgen nutzen, die Nettorate übertragener Information im Codemultiplex sinkt aber auf 1/8 von 2400, es bliebe noch 200 Bit/s netto - zu wenig nach unseren Kriterien.

Fazit: Der Turm ist nicht brauchbar für das Multiplexverfahren. Die Hardware der Empfängerseite schafft nur mühsam eine rohe Datenrate von etwa 4800 bit/s - vielleicht so gerade noch genug für einen Codemultiplexer. Noch störender aber ist der sehr unerfreuliche Tatbestand, das die Stromsparschaltung immer wieder ein nachtriggern des Turmes erforderlich macht, was zusätzlich Zeit und Dekodieraufwand kostet und die Datenrate weiter senkt.

Übrigens: Der IR-Empfänger im Turm liegt auf der gleichen Platine unmittelbar neben den IR-Sendedioden und wird daher stark angestrahlt - das gesendete Signal 'überspricht' voll in den Empfangsteil, und schickt so ein lokales Echo an das Terminal. Ein Terminalprogramm braucht deshalb keine eigene Simulation des lokalen Echos – diese Option muss abgeschaltet werden. Andererseits ist ein Echo im Terminal kein Zeichen dafür das das Forth im RCX arbeitet.

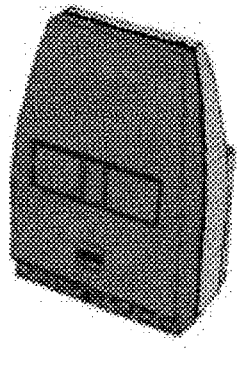
Als nächstes haben wir untersucht, ob denn zumindest der RCX selbst mit seiner IR-Schnittstelle geeignet sein könnte.

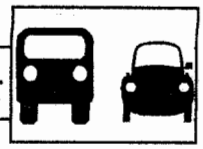
Im LEGO-Klotz fanden wir die gleichen 2 IR-Sendedioden und den gleichen IR-Empfänger wie im Turm (TS OP1138). Doch anders als dort erzeugt hier der Prozessor selber die 37,5KHz IR-Blitzfrequenz über einen Timer-Pin. Diese geht auf eine Anordnung von 4(?) Transistoren und dort wird der Datenstrom von der seriellen Schnittstelle des H8/300 dazugemischt und immer über die 2 IR-Dioden abgestrahlt. Auch hier liegt der Empfänger gleich nebenan und das Signal 'überspricht' klar und deutlich, kann also wenn man möchte vom Controller auch sofort wieder gelesen werden. Diese Anordnung ist also zumindest teilweise unter der Kontolle des Programmierers. Sowohl die IR-Blitzfrequenz als auch die Rate serieller Daten ist programmierbar. Wiederum nicht beeinflussbar ist auch hier der Empfangsfilter. Gemessen mit einem Frequenzgenerator zeigte sich am Oszilloskop eine Durchlässigkeit für starke IR-Signale aus unmittelbarer Nähe im Bereich 21 bis 47 KHz. Darunter und darüber gab es zufällige Pegelwechsel oder weiter außerhalb dann gar keine mehr. Die minimale Blitzpausenlänge betrug auch hier 250us und legt so eine Grenze bei knapp unter 4800 Bit/s fest.

Fazit: Auch im RCX ist durch die Wahl der IR-Empfangseinrichtung der Weg zu einem Codemultiplex praktisch verbaut. Allerdings kann prozessorseitig sofort erkannt werden wenn die 'Luft nicht rein ist', andere dazwischen senden und so eigene Botschaften zerstört wurden. Oder es kann auf diese Art bösartig dazwischen gesendet werden um andere zu stören - Battelgame!

Übrigens hatten wir 2 Türme und RCXse im Test. Dank der Forthgesellschaft! Sie lieferten beide die gleichen Ergebnisse.

Ende





# Die Seite für den Umsteiger

## CODE-Definitionen ohne CODE und END-CODE

**Fred Behringer <behringe@mathematik.tu-muenchen.de>**

In Forth gibt es zwei Grundkategorien von Worten, Colon-Definitionen und CODE-Definitionen. (VARIABLEn und dergleichen erweitern die Möglichkeiten.) CODE-Definitionen sind die Grundworte (bei irgendwelchen Elementen muß ja mal angefangen werden), die das, was sie zu sagen haben, unmittelbar in Maschinenbefehlen ausdrücken. Sie werden normalerweise über einen im Forthsystem (Forth ist Sprache und gleichzeitig auch System) eingebetteten Assembler konstruiert. Colon-Definitionen sind einfach nur zusammengesetzte Forth-Worte, nichts anderes. Sie können CODE-Definitionen enthalten, oder auch andere zusammengesetzte Forth-Worte. Das folgende Forth-Wort ist eine CODE-Definition.

```
CODE XXX
   7777 #          AX MOV
                   AX PUSH
   NEXT
   END-CODE
```

**CODE** baut (unwichtige Einzelheiten lasse ich weg) den Kopf eines Forth-Wortes auf, das von anderen Forth-Worten aufgerufen werden kann, und schaltet in das Vokabular **ASSEMBLER**. Die dann folgenden Maschinenbefehle werden in das aufzubauende Forth-Wort, hier **XXX**, gelegt. Sie leisten beim späteren Aufruf von **XXX** das Folgende: Die unmittelbar zu verarbeitende (**#**) Zahl **7777** wird per **MOV** in das Maschinenregister **AX** gelegt, der Inhalt des Registers **AX** wird per **PUSH** auf den Datenstack gelegt und es wird zum inneren Interpreter (**NEXT**) zurückgesprungen. **END-CODE** schaltet schließlich wieder zum Vokabular **FORTH** zurück.

Die allereinfachste Möglichkeit, diesen Maschinencode in eine Colon-Definition einzubetten, wäre folgende:

```
: YYY XXX ;
```

Schön ist, daß man jetzt die von **XXX** auf den Stack gelegte Zahl **7777** mit nur einem einzigen zusätzlichen Federstrich auf dem Bildschirm ausgeben lassen kann:

```
: YYY XXX . ;
```

In der **CODE**-Definition, als Maschinenbefehle, wäre das Anzeigenlassen eine ziemlich aufwendige Sache. Unschön ist, daß man eine weitere Wortbezeichnung braucht, nämlich **YYY**.

Es geht schöner:

```
: ZZZ
  ASM[
   7777 # AX MOV AX
   PUSH NEXT
  ]FORTH
  . ;
```

Das ist das, was in anderen Sprachen als "Inline-Code" bezeichnet wird, Inline-Code par excellence! Und das High-Level-Forth-Anzeige-Wort "." ist auch noch gleich mit eingebaut. (Ich spreche von Turbo-Forth. Nicht jedes Forth-System bietet **ASM[** und **]FORTH** . Turbo-Forth kann von mir bezogen werden. Eine E-Mail-Anfrage genügt. Das Programm (26K) kommt dann als Attachment.) Man bedenke, wie umständlich es beispielsweise in den ersten Turbo-Pascal-Versionen war, Inline-Code zu erzeugen! Und die unverständlichen Beschränkungen, mit denen man sich beispielsweise beim Einschieben von Code-Teilen in einem OCCAM2-Programm herumschlagen mußte! In Forth ist "Inline-Code" so selbstverständlich, daß man dafür nicht einmal einen eigenen Begriff braucht! Und nichts ist verboten! Man kann den Forth-Assembler aus dem Stand heraus nach allen Richtungen hin verbiegen und dem eigenen Geschmack anpassen, sogar mitten im Assembliervorgang! "Let us exercise our constitutional right to assemble", fordert Julian Noble in seinem kürzlichen Lehr-Artikel "A Call to Assembly 1/3" in der Forthwrite 113. Vielleicht ein bißchen hoch gegriffen, aber was habe ich selbst an mehreren Stellen immer wieder gesagt?: Ich "mißbrauche" Forth gern zum schnellen ungehinderten Assemblieren.

Der vorliegende Artikel soll ein Lehrstück für Umsteiger sein. (Der Einsteiger wird sich kaum um Inline-Code scheren.) Dem Umsteiger werde ich gerecht, wenn ich kurz erkläre, was **ASM[** und **]FORTH** tun. Gleichzeitig wird die Eleganz aufgezeigt, mit der Forth solche Dinge erledigt. (In der Mathematik werden ab und zu komplizierte und undurchsichtige Berechnungen mit dreifachen Indizes oben und unten benötigt. In einer Lehrbuch-Darstellung oder in einem Forschungsbericht sind sie aber ver-

```
: ASM[ ( -- )
  BASE @ CONTEXT @   \ Basis und erstes Suchvokabular aufbewahren
  HERE 6 + ,         \ Adresse der "cfa" des Maschinencodes
  COMPILE BRANCH     \ High-Level-Sprungbefehl hineincompilieren
  HERE               \ Wird unter ]FORTH als Speicheradresse benötigt
  0 ,                \ Platz halten ( Inhalt siehe ]FORTH )
  HERE 2+ ,          \ Wirkt wie cfa einer CODE-Definition ohne Wortkopf
  ASSEMBLER          \ In das Vokabular ASSEMBLER schalten
  [COMPILE] [ ;      \ In den Interpretierzustand schalten
  IMMEDIATE

: ]FORTH ( -- )      \ Wieder nach FORTH schalten
  HERE SWAP !        \ Adresse, wo es weitergeht, an BRANCH aus ASM[ hängen
  CONTEXT ! BASE !   \ Suchvokabular und Basis wiederherstellen
  [COMPILE] ] ;      \ Wieder in den Compiliermodus schalten
```

pönt. "Einfach" und "schön" sollen die Formeln sein - und dabei doch die ganze Welt umspannen. Nun, in Forth geht es uns ebenso - mir jedenfalls.)

## Wirkungsweise von ASM[ und ]FORTH :

Die aufzubauende Colon-Definition (ich bezeichne sie ab jetzt mit **XXX**) enthält, wie jede andere Colon-Definition auch, eine Liste von 16-Bit-Plätzen ("Zellen") mit den Codefeld-Adressen (cfa) der darin nacheinander abzuarbeitenden Forth-Worte. Im Maschinenregister SI wird der Forth-Instruction-Pointer (IP) festgehalten. Der jeweilige Wert zeigt immer auf den cfa-Eintrag des als nächstes abzuarbeitenden Forth-Wortes der Liste. Wir mögen uns auf "Eintrag 1" (= IP-Wert) befinden und betrachten die nachfolgenden Einträge 2, 3, 4 und 5. Das Immediate-Wort **ASM[** legt bei Aufbau (beim Compilieren) von **XXX** die Adresse von Eintrag 4 in Eintrag 1. Dort steht eine Adresse (sie wird in der Zeile "**HERE 2+ ,**" erzeugt), die als cfa einer "**CODE**-Definition ohne Wortkopf" aufgefaßt werden kann, eben des einzuschiebenden Maschinencodes. In Eintrag 4 wiederum steht die Adresse von Eintrag 5 und in Eintrag 5 steht der erste Befehl des einzuschiebenden Maschinencodes. Beim späteren Abarbeiten von **XXX** ruft also das in Eintrag 1 liegende "Wort" das in Eintrag 4 liegende Wort auf und die in Eintrag 4 liegende cfa verhält sich wie die cfa einer **CODE**-Definition (eben des einzuschiebenden Maschinencodes ab Eintrag 5). Ist letztere (während des späteren Aufrufs von **XXX**) abgearbeitet, so geht der Instruction-Pointer IP in der Listenhierarchie wieder bis auf das "Wort" in Eintrag 1 zurück und führt als nächstes das Wort in Eintrag 2 aus. Dort wurde aber von **ASM[** beim Aufbau von **XXX** der **BRANCH**-Befehl hingelegt, der einen (High-Level-)Sprung an eine Stelle ausführt, die in Eintrag 3 steht und die beim Abschluß des Aufbaus von **XXX** von **]FORTH** erzeugt wurde. Der **BRANCH**-Sprung geht an genau die erste Stelle hinter dem Maschinencode, dort, wo es mit dem High-Level-Forthcode weitergeht.

### Und nun noch etwas für den fortgeschrittenen Umsteiger

Maschinencode in eine vorgegebene Hochsprache einbetten konnte man schon immer. Irgendwie. Auch wenn es in Forth viel einfacher und eleganter geht, es ist nichts Neues. Was aber, wenn Sie beispielsweise eine Turbo-Pascal-Funktion (als Quellcode) in ein Assembler-Programm einbetten wollen? Geht nicht? In Forth ist so etwas eine Kleinigkeit. Ich ändere das obige Wortpaar **ASM[** und **]FORTH** ein klein wenig ab, und schon habe ich das Ziel erreicht. Im folgenden Beispiel wird in einer **CODE**-Definition der Maschinencode kurz unterbrochen, um den Stackinhalt in High-Level-Forth auf dem Bildschirm auszugeben, und dann wieder aufgenommen. Das ist vergleichbar mit einem Unterprogramm-Aufruf in Maschinencode, der einen String ausgeben soll. In High-Level-Forth steht die String-Ausgabe eines Stackwertes mit dem Wort "." bereits gut ausgetestet zur Verfügung - und das nutze ich in diesem Beispiel aus.

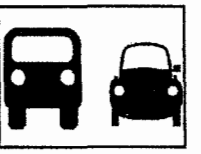
```
CR .( Einbettung von FORTH-Code in CODE-Definitionen )
ONLY FORTH DEFINITIONS ALSO ASSEMBLER ALSO
HEX

: ASM]FORTH ( -- )              \ Übergang von Codeteil zu FORTH-Teil
     [ ASSEMBLER ]              \ Damit # nicht aus FORTH genommen wird.
     HERE 5 + # W MOV           \ Wo geht es weiter
     0 [W] JMP                  \ mit der FORTH-Wort-Adressen-Liste ?
     AVOC @ CONTEXT !           \ CONTEXT wiederherstellen
     ['] : @ ,                  \ NEST hineincompilieren
     [COMPILE] ] ;              \ und Überleitung zur Wortliste

: FORTH[ASM ( -- )              \ Übergang von FORTH-Teil zu Codeteil
     HERE 2+ ,                  \ cfa und pfa füllen ( Wort ohne Header
     HERE 2+ ,                  \ als CODE-Definition aufbauen).
     [COMPILE] [                \ In den Interpreter-Modus schalten
     CONTEXT @ AVOC !           \ High-Level-Vokabular aufbewahren
     ASSEMBLER                  \ Ins Vokabular ASSEMBLER schalten
     0 [RP] IP MOV              \ Nächste Adresse => Instruction-Pointer.
     RP INC RP INC              \ RStack von Rückkehradresse befreien.
     ; IMMEDIATE                \ Ausführung bei Compilation des neuen
 Wortes

CR .( Beenden einer CODE-Definition als Colon-Definition )
ONLY FORTH ALSO ASSEMBLER ALSO DEFINITIONS
HEX

: C;FORTH ( -- )                \ Assemblerversion von ASM]FORTH
     [COMPILE] ASM]FORTH        \ Gleiche Funktion wie ASM]FORTH
     !CSP                       \ Stacküberwachung einschalten,
     [COMPILE] ; ; IMMEDIATE    \ damit sie hier nicht stört.

ONLY FORTH ALSO DEFINITIONS
```

```
CODE XXX
   AX POP
   AX INC
   ASM]FORTH . FORTH[ASM
   AX DEC
   AX DEC
NEXT END-CODE
```

Die Erklärungen laufen ähnlich wie beim obigen Wortpaar **ASM[** und **]FORTH**. Ich beschränke mich auf die Kommentare im folgenden Listing. (Man beachte, daß ich mit dem eigentlichen Maschinencode in diesem Beispiel keinen tieferen Sinn verknüpfe. Er sollte lediglich zur Demonstration dienen.)

### Von Assembler nach Forth und zurück - beliebig hin und her

Interessant an diesem Listing dürfte sein, daß ich in ein und demselben aufzubauenden Wort, sei es eine **CODE**-Definition, sei es eine Colon-Definition, beliebig oft von Forth nach Assembler und wieder zurück hin- und herschalten kann. (Man beachte, daß es dabei auch nicht unbedingt



das Vokabular **FORTH** zu sein braucht, das anfangs als **CONTEXT** oder/und **CURRENT** vorgegeben wird.) Man probiere einfach mal die folgenden Worte der Reihe nach aus. Ich habe versucht, alle Kombinationsmöglichkeiten zu erfassen. Die genaue Syntax kann aus diesen Beispielen abgelesen werden.

```
ONLY FORTH ALSO DEFINITIONS
HEX

CODE XX1 7777 # CX MOV CX PUSH
     ASM]FORTH . 5555
     FORTH[ASM AX POP AX INC AX PUSH
     ASM]FORTH . ;

: XX2 3333
     FORTH[ASM CX POP 1111 # CX SUB CX PUSH
     ASM]FORTH 2* .
     FORTH[ASM NEXT END-CODE

CODE XX3 4444 # AX MOV AX PUSH
     ASM]FORTH 1- 2*
     FORTH[ASM AX POP AX DEC AX PUSH C;FORTH

2 BASE !
: XX4 10101010 01010101
     FORTH[ASM BX POP DX POP BX DX OR DX PUSH
     ASM]FORTH 2 BASE ! ." Ergebnis = " . HEX ;
HEX

Und hier noch etwas komplizierter:

: XX5 ( n -- )
     BEGIN
       FORTH[ASM AX POP 6 # BX MOV AX SHR
                 U< IF BX MUL 4 # AX ADD THEN
                 AX PUSH
       ASM]FORTH DUP DUP U. 2 < IF DROP EXIT THEN
     AGAIN ;
```

Frage an den Programmierer: Was macht das Programm? Frage an den mathematisch Interessierten: Gibt es Zahlen, nach deren Eingabe das Programm in aller Ewigkeit weiterarbeitet, ohne je zu einem Ende zu kommen? Vom Verfahren her gesehen? Von der beschränkten Registerlänge her gesehen? Kann es Zyklen geben? Man probiere das Programm beispielsweise mit 41 oder 313 aus.

Der Wechsel von Sprachebene zu Sprachebene wird im Rahmen von F83 sehr gut im "grünen Buch" von Zech beschrieben. Es gibt viele Möglichkeiten. Interessant ist, daß in Forth schon ein Anfänger, einmal zum Nachdenken angestoßen, sofort in der Lage ist, sich beliebig viele mögliche und unmögliche Möglichkeiten auszudenken und sie, und das ist das Einzigartige an Forth, auch tatsächlich ohne Umschweife selbst gleich praktisch umzusetzen. Ich hatte 1992 "meinen Zech fleißig studiert", war bemüht, ein völlig symmetrisches Werkzeug (zum beliebig oftmaligen Hin- und Herschalten) aufzubauen, und brachte das Ganze dann in mein Transputer-Forth-System F-TP 1.00 (liegt auf taygeta) ein. Inzwischen ist es, wie im wissenschaftlichen Alltagsleben auch, soweit, daß ich nicht mehr genau weiß, welche Ideen von mir stammen, und welche von anderen. Forth ist für mich zum Handwerkzeug geworden, das ich mir ständig neu zurechtfeile.

Meine Namensgebung ist mnemotechnisch wie folgt zu verstehen: Das ] in **ASM]FORTH** deutet darauf hin, daß ab jetzt in den Compiliermodus (High-Level-Forth) geschaltet wird. Das [ in **FORTH[ASM** deutet darauf hin, daß ab jetzt (zum Assemblieren) in den Interpretiermodus geschaltet wird.

Alle Programmteile aus dem vorliegenden Artikel habe ich nicht nur in Turbo-Forth, sondern auch unter ZF getestet: Geht wunderbar.

# Hardcode
# Assembler 'brute force' für den Lego-RCX

**Martin Bitter**

Um die wesentlichen Forth-Worte zum Zugriff auf den Speicher zu lehren, habe ich meinen Schülern aufgetragen, eine 'Karte' des RCX-Displays zu erstellen. Diese 'Karte' kann in der Folge dazu benutzt werden, einzelne Segmente des Displays gezielt anzusteuern. Ziel ist es, einfache Textmeldungen auf dem RCX-Display auszugeben.

Für die, die einen solchen RCX-Legobaustein nicht kennen: Das ist ein ca. 8cm x 5cm x 3cm großer 'Legostein' in dem sich sechs Batterien und eine Platine mit einiger Elektronik den Platz gerecht teilen. Der 'Stein' ist also autark – völlig drahtlos ( vgl. Seite 26). Die Elektronik besteht aus einem Hitachi H8/300 $\mu$P, einer Infrarot Sende-Empfang Einrichtung, 3 bidirektionalen, 9V PWM Ausgängen, drei vielfach konfigurierbaren Eingängen, vier Tastern (Schalter), einem Summer und einem Display. Das alles über Treiber abgesichert und recht 'narrensicher'. An die Ein- und Ausgänge können die üblichen Lego-Kabel angesteckt werden. Kinder können also recht gefahrlos damit programmieren lernen, elektrische Anschlüsse werden nur gesteckt. Weitere Werkzeuge oder gar Elektrokenntnisse sind nicht von Nöten.

Programmieren kann man diesen RCX-Stein mit der mitgelieferten visuellen Lego-Software, einer im Vergleich dazu wesentlich erweiterten Version von VISI-Lab für den Schulbe-

darf (ca. 250 DM) oder mit verschiedenen freien Sprachen, darunter pbForth von Ralph Hempel. Ralph spricht mit pbForth viele der Firmwarefunktionen an, die werksseitig von Lego eingebaut sind. Im Grunde genommen eine Art Call-Mechanismus. Das beschränkt den Zugriff auf die Hardware-Peripherie auf die von Lego vorgesehenen Möglichkeiten: Ziffernanzeige, Statusbalken, rotierende(r) Kreis(segmente), etc.

Einschränkungen durch Softwaremechanismen? Das klingt nach Beschränkung persönlicher Freiheit! Hier hilft Forth!

Insgesamt verfügt das Display über 72 Segmente, die in der Regel einzeln ansteuerbar sind. Sie sind bit-gesteuert, d.h. durch Setzen oder Löschen eines einzelnen Bits im Speicher wird das jeweilige Segment ein- oder ausgeschaltet. Diese Bits verteilen sich mit einigen Lücken über die 10 Bytes von Hex 0xEF43 bis hex 0xEF4D. Eine Systematik der Zuordnung der Adressen (Nummern) der einzelnen Segmente und ihrer räumlichen Anordnung auf dem Display haben wir bis jetzt noch nicht entdecken können.

Ein ANSI Forth wie pbForth bietet nicht unbedingt die Möglichkeit des bitweisen Zugriffs. Nun kann man zwar einzelne Bits über das Verodern bzw. Verunden mit entsprechenden Masken in Hochsprache setzen oder löschen, ohne benachbarte Bits ungewollt zu beeinflussen.. Aber andererseits ist ein solcher Zugriff im Repertoire fast jeden $\mu$P enthalten, so auch in den Maschinenbefehlen des H8/300. Warum also nicht die entsprechenden Maschinenbefehle nutzbar machen?

Es gibt keinen in pbForth integrierten Assembler oder gar die Möglichkeit, In-Line-Assembler-Code zu schreiben wie Fred Behringer es in seinem Artikel (Seite 29) vorschlägt. – dazu ist der Speicherplatz im RCX zu kostbar. Besser wäre es Codestücke auf dem PC zu assemblieren und diese als fertige Worte in den RCX zu laden.

Bis ein solches 'Feature' auf unterschiedlichen PC-Platformen vorliegt mag in Einzelfällen das 'Hardcodieren' helfen: Dabei werden per Komma ' , ' die jeweiligen Maschinenbefehle des $\mu$Ps direkt in den Speicher geschrieben.

Alles was man braucht: a) einen einfachen Befehl, der einen Wortheader erzeugt, b) das entsprechende Pendant um ein Wort zu beenden und c) ein wenig Kenntnisse [ ;-) ] über die Maschinenbefehle.

a) und b) sind recht schnell durch 'dumpen' von Primitives erledigt und führen im Falle des pbForth V1.12 zu:

```
: CODE ( -- )    \ leitet Assemblerdefinition ein
  CREATE HERE 6 - TO HERE ;
```

und zu:

```
: NEXT ( -- )     \ beendet Assemblerdefinition
  6D40 , 5900 , ;
```

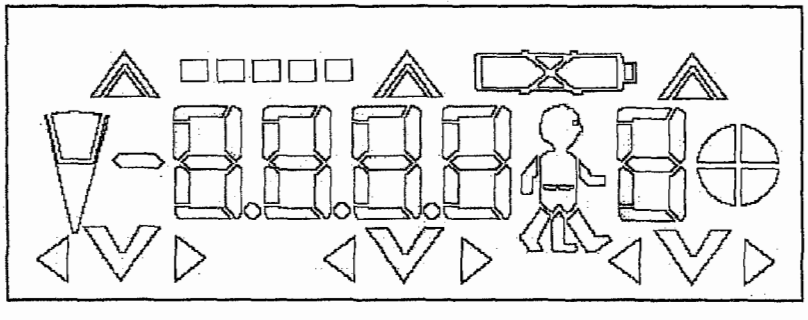
Dabei erzeugt CODE via CREATE einen Header und setzt HERE so zurück, dass anschließend direkt dahinter mit ' , ' der (hoffentlich) ausführbare Maschinencode abgelegt werden kann. NEXT schreibt in Maschinensprache zuerst einen Befehl (6d40), der den Forth-Instruction-Pointer (Register ER4) in das Register R0 kopiert, danach den Befehl, zu dieser Adresse zu 'jumpen' (5900).

c) ist das, was mir am schwersten fällt: Ich wurde fündig bei http://www.hitachisemiconductor.com über die 'parametric search (Programming Manual:H8/300) . (Der Besuch bei Hitachi ist immer wieder kurzweilig, denn regelmäßig wird das Site-Layout geändert 'gebookmarkte' Adressen sind ungültig und die Klickorgie kann von Neuem beginnen. Jetzt auch mit Java! Qui bono?) Hilfreich ist ebenfalls ein Blick in das Manual von Egmont Woitzels H8 Assembler (s.h. Innentitel Anzeige FORTecH).

```
CODE BSET ( bit# addr -- ) \ (Bytezugriff!)
  6D71 ,          \ MOV.W @ER7+,R1 (NOS nach R1 mit Increment)
  0C91 ,          \ Kopie von R1l nach R1h (wg. Beschränkung auf
                  \                                  High-Byte?)
  7D60 , 6010 ,   \ BSET R6,@ER1
  6D76 ,          \ MOV.W  @rDSP+,rTOS ( das ist pop)
NEXT

CODE BCLR ( bit# addr -- ) \ (Bytezugriff!)
  6D71 ,          \ MOV.W @ER7+,R1 (NOS nach R1 mit Increment)
  0C91 ,          \ Kopie von R1l nach R1h (s.o.)
  7D60 , 6210 ,   \ BSET R6,@ER1
  6D76 ,          \ MOV.W  @rDSP+,rTOS ( das ist pop)
NEXT
```

Diese Worte sind für die Schüler zur Zeit noch in einem anderen Highlevelword versteckt, aber mir und später den Schülern erlauben sie komfortablen Zugriff auf jedes einzelne Bit.

Jetzt haben wir nur noch ein Problem: Wieviele Buchstaben können mit einer 7-Segmentanzeige dargestellt werden.?



## Forth-Gruppen regional

**Hamburg** **Küstenforth**
**Klaus Schleisiek**
Tel. 040 / 375008-13 g
kschleisiek@send.de
Treffen jeden 4. Freitag im Monat
16:30 Uhr, Fa. SEND, Stubbenhuk 10
20459 Hamburg

**Moers** **Friederich Prinz**
**Tel.: 02841-58398 (p) (Q)**
(Bitte den Anrufbeantworter nutzen !)
(Besucher: Bitte anmelden !)
Treffen: (fast) jeden Samstag,
14:00 Uhr, **MALZ, Donaustraße 1**
**47443** 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, **China Restaurant XIANG**
**Morungerstraße 8**
**München-Pasing**

## µP-Controller Verleih

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

## Gruppengründungen, Kontakte

**Fachbezogen 8051 ... (Forth statt Basic, e-Forth)**
Thomas Prinz
Tel.: 06271-2830 (p)

## Forth-Hilfe für Ratsuchende

**Forth allgemein**
**Jörg Plewe**
Tel.: 0208-49 70 68 (p)

Jörg Staben
Tel.: 02173-75708 (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,<br>TDS9092),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 |
| Forth-Mailbox (KBBS) | 0431-533 98 98 (8 N 1)<br>Sysop Holger Petersen<br>hp@kbbs.org<br>Tel.: 0431-533 98 96 (p) bis 22:00<br>Fax : 0431-533 98 97<br>Helsinkistraße 52<br>24109 Kiel |

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 Arbeistzeiten
g = geschäftlich

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

# Vorankündigungen

## euroForth '01

The 17th EuroForth conference

Schloss Dagstuhl, near Saarbrücken, Germany.

23rd to 26th November, 2001

Wie dem Ankündigungstext zu entnehmen ist, ist die Konferenzsprache englisch.
Wer Näheres erfahren will, kann dies unter http://dec.bournemouth.ac.uk/forth/euro/ef01.html tun.

---

# Jahrestagung der Forth Gesellschaft e.V
# 2002
# in Garmisch-Partenkirchen

(man sieht: Ich setzte das 'fetter' als ... )

Organisiert und geplant wird diese Tagung von dem darin erfahrenen Heinz Schnitter, dem Fred Behringer dabei ein klein wenig zur Hand geht.

Tagungsort ist diesmal ein Hotel in besonderer Lage – es ist nur über eine Bergseilbahn zu erreichen (Rechnertransport soll dennoch kein Problem sein.)

Auch die Termine stehen schon fest: vom Freitag, den 19. bis Sonntag, den 21. April 2002, ein optionaler 'freier' Tag wird am Donnerstag, den 18. April angeboten.