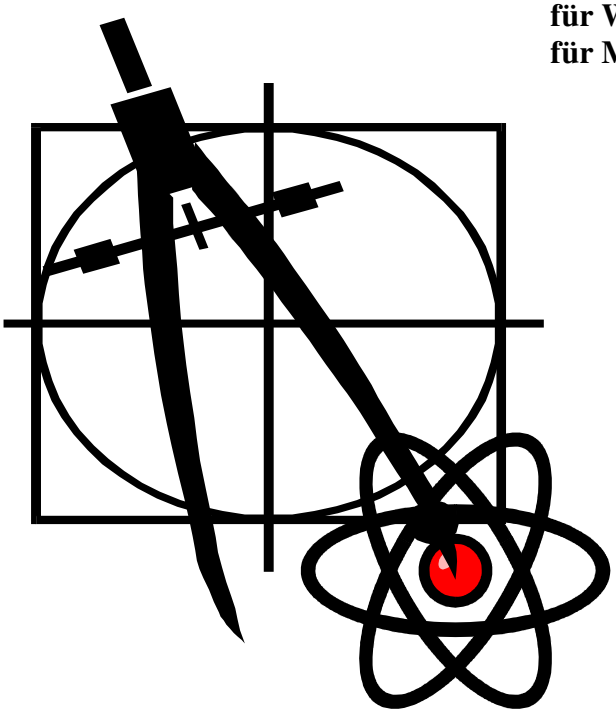
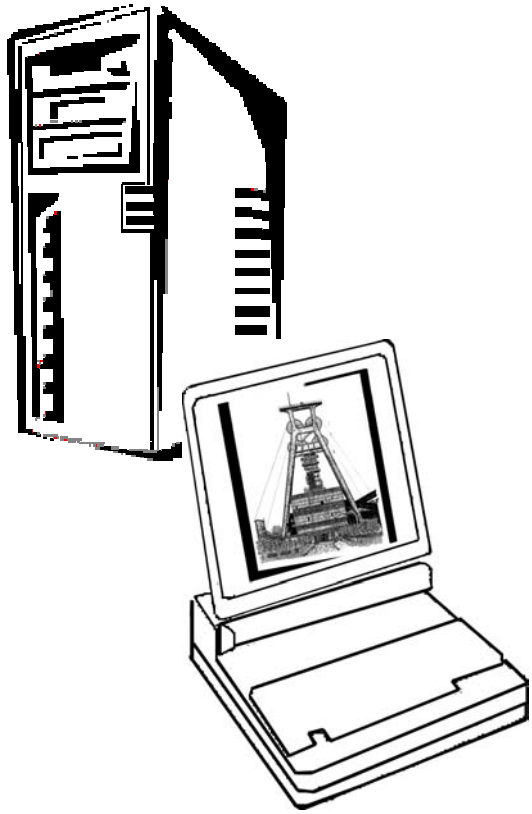
# VIERTE DIMENSION

2 / 2005

## Das FORTH-Magazin


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





### In dieser Ausgabe:




21. Jahrgang - **Ausg. 2/2005** - 4,00 €

Organ der FORTH-Gesellschaft e.V.

# Dienstleistungen und Produkte fördernder Mitglieder des Vereins

# Inhalt



In der nächsten Ausgabe finden Sie voraussichtlich:

- Interessantes aus der Bastelstube von Rafael Deliano
- Was immer SIE uns schicken
- Berichte von der Tagung der Forthgesellschaft

# Editorial


**IMPRESSUM**

**Name der Zeitschrift**

**Vierte Dimension**

**Herausgeberin**

Forth-Gesellschaft e.V.
Postfach 19 02 25
80602 München
Tel.: (0 89) 1 23 47 84
E-Mail:
***SECRETARY@FORTH-EV.DE***
***DIREKTORIUM@FORTH-EV.DE***

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

**Redaktion & Layout**

Friederich Prinz
Haspelstrasse 6 d
47443 Moers
Tel.: (0 28 41) 5 83 98
E-Mail: ***VD@FORTH-EV.DE***

**Anzeigenverwaltung**

Büro der Herausgeberin

**Redaktionsschluß**

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

**Erscheinungsweise**

1 Ausgabe / Quartal

**Einzelpreis**

4,00 € + Porto u. Verpackung




Liebe Leser,

Diese Ausgabe der Vierten Dimension erreicht Sie spät, viel zu spät. Der Grund hierfür ist, daß die VD zusammen mit ihrem Editor umgezogen ist. So ein Standortwechsel ist immer aufwändig. Wenn man ihn aber neben den Anforderungen eines anspruchsvollen Berufs gestalten, organisieren und realisieren muß, dann bleiben keine Ressourcen für „andere Dinge“. Ich bitte Sie für die lange Wartezeit um Entschuldigung.

Bei einem Umzug gehen immer auch Dinge kaputt und verloren. Ich hoffe, daß es mir gelungen ist, nichts von dem zu verlieren, was die Autoren der VD mir in der Umzugszeit geschickt haben. Falls Sie aber doch einen Ihrer Beiträge oder Hinweise vermissen, bitte ich Sie, mich kurzfristig darauf anzusprechen.

Min Moore ist tot. Wie geht man mit einer solchen Nachricht um? Ich weiß von Charles Moore zu wenig, als daß ich ihm mein Beileid kund tun könnte. Und die Existenz seiner Frau habe ich erst wahrgenommen, als mich die Nachricht von ihrem plötzlichen Tod erreichte. Die Menschen, die Min Moore kennen lernen durften, haben mir von einer überaus lebensbejahenden, spontanen und fröhlichen Persönlichkeit erzählt. Ich denke, ich hätte sie gerne kennen gelernt. Aber dann wären mir diese Zeilen vermutlich noch viel schwerer gefallen.

Auch das Leben einer Gemeinschaft wie der unseren ist ein ständiges Kommen und Gehen. Immer wieder besonders angenehm ist es, wenn ich im Auftrag der Forthgesellschaft im Editorial unserer Zeitschrift **neue Mitglieder** begrüßen darf. **Rolf Lauer** aus 63936 Schneeberg ist Mitglied der Gesellschaft geworden. Und **Jörg Krumböck** aus München hat sogar schon eine lokale Forthgruppe für sich gefunden. Sie sind uns sehr willkommen.

Ihr

*Friederich Prinz*

**Quelltext-Service**

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

Die Forthgesellschaft wird durch ihr Direktorium vertreten:

| | | |
|---|---|---|
| Prof. Dr. | Fred Behringer | Kontakte: Direktorium@Forth-ev.de |
| Dr. | Ulrich Hoffmann | |
| Dipl. Inf. | Bernd Paysan | |

**Auszug aus Mails von Rolf Schöne mit Rolf Lauer:**

Lieber Herr Lauer,
...
Eine kurze Frage habe ich noch: Zuerst las ich "VtX Forth" und konnte damit nichts anfangen. Ich hätte wohl eher "VFX Forth" lesen müssen. Das macht Sinn. Stephen Pelc und MPE - ist es das?

Und noch eine Frage: Wie sind Sie auf uns aufmerksam geworden?

Herzlichen Gruß

Rolf Schöne

==========

Guten Morgen Herr Schöne,

ja genau, Stephen Pelc. Leider ist sein Handbuch nicht proportional zur Güte des Systems. Anyway, man kann nicht alles haben :-) Ich gehe mal davon aus, daß die meisten Mitglieder wohl ausschließlich Freeware einsetzen, und es keinen weiteren Benutzer von VFX Forth gibt. Ist das Richtig ?

Wie bin ich auf den Forth-ev aufmerksam geworden ? Ehrlich gesagt, ich weiß es nicht mehr. Der Forth-ev ist mir schon sehr lange ein Begriff und vor 3 oder 4 Jahren wollte ich schon mal Mitglied werden. Aber dann kam mal wieder etwas dazwischen und die Arbeit fraß mich auf und ich vergaß es wieder. Letzte Woche dachte ich dann: Wenn alle so träge sind wie ich, ist der Verein irgendwann dicht. Dann füllte ich das Formular aus und schickte es weg. Auf jeden Fall freue ich mich diesen Schritt endlich getan zu haben!

Mit den allerbesten Grüßen aus dem Odenwald

Rolf Lauer

Betreff: **VD als PDF im Internet?**
Von: Michael Kalus <michael.kalus@onlinehome.de>

Am 13.10.2004 um 20:00 schrieb Friederich Prinz:

> Hmm - wieviele Leute würden potentiell mit einem 56 kBit
> Modem an einer 10 MByte Datei verzweifeln?

Fragen wir die Mitglieder: Also Leute, wie ist das? Wer braucht die gesammelten VDs aller Zeiten als PDF und komprimiert so klein es überhaupt geht - oder ist es auch ok, wenn die 'wie sie sind' auf der Homepage zur Verfügung gestellt werden? Also um 1-3 MB pro Stück, selten mehr, wenige Hefte bis 10 MB?

Michael

*Antworten bitte an die Redaktion der VD und an den Webmaster der FG!*

**Auszug aus Mails von Rolf Schöne mit Jörg Krumböck:**

Seit einiger Zeit nehme ich an den Forth-Treffen in München beim Gran Sasso teil. Da ich mich selbst auch mit Forth beschäftigt habe und mich diese Sprache auch weiterhin interessiert, bin ich zur Unterstützung des Forth-Vereins beigetreten. Und hier ein kleiner Beitrag für die "Leute"-Rubrik:

: Jörg Krumböck,
Dipl.-Ing. Elektro- und Informationstechnik
Über Daniel Ciesinger bin ich in meinem Studium zu Forth gekommen und habe mit ihm zusammen diese Sprache erforscht.
Beruflich entwickle ich SW für DSPs und die Audio-Signalverarbeitung, vorrangig in C. Forth kam zwar noch nicht zum Einsatz hat aber meinen Horizont in Sachen Programmierung sehr erweitert.
Meine Seite www.krumboeck.de ist leider noch eine Baustelle, werde sie aber noch fertig stellen. Dabei wird mir evt. Forth helfen ...
;

Fred Behringer zum Artikel von B. Hodson (siehe Seite 8)

Der Autor betont an mehreren Stellen, dass sein Programmierprinzip mit einigen wenigen Worten erklärt ist. Das Geheimnis liegt in der Erzeugung eines numerischen Bytecodes, der durch seine Beziehungen zwischen den Zahlen ein Abbild der Struktur des Programmiersystems liefert. Aufgrund dieser numerischen Beziehungen wird es möglich, einen sehr kleinen Compiler (als Zulieferer zum virtuellen Prozessor) zu erzeugen. Über den genauen Vorgang, sagt der Autor, braucht der Anwendungsentwickler nichts zu wissen. Er braucht sich nur die paar Regeln des Programmierprinzips zu merken. Alles andere macht der (sehr kleine) Compiler. Ich meine, das ist zu wenig. Wir wollen etwas über den genauen Hergang wissen.
Das Bytecode-Prinzip ist bekannt. Das Einschalten eines virtuellen Prozessors gab es schon immer. Zumindest der elitären Gruppe von Enthusiasten, genannt "Forthler", ist es zur Genüge bekannt. Beim JAVA-Bytecode erfuhr ein größeres Publikum vom "Prinzip des Zwischencodes". Knuth schreibt in seinen Büchern vom virtuellen Prozessor. Bei den FPGA-Projekten spart man sich das "Virtuelle" und baut gleich den wahren Prozessor nach den Wünschen der geeignet konstruierten Sprachprinzipien. Vor Jahren habe ich in Transputer-Forth einen Disassembler für den JAVA-Bytecode geschrieben - und hatte meine Freude daran.
Was wir wissen wollen, ist, wie Hodson sein numerisches Beschreibungssytem aufbaut. Wir möchten einen Beispiels-Compiler sehen, an dem man sich orientieren kann. Einen Compiler zum Nachbauen.
Der vorliegende Artikel ist gut für jemanden, der mit dem Gedanken spielt, für seine Firma oder Institution die "Software" von Hodson käuflich zu erwerben.
Wir in der FG wollen lernen. Wir brauchen mehr. Wir brauchen einen zweiten Artikel, in dem das steht, was wir eigentlich wissen wollen.

Fred Behringer

# Gehaltvolles

zusammengestellt und übertragen
von Fred Behringer

## VIJGEBLAADJE der HCC Forth-gebruikersgroep, Niederlande

### Nr. 48, Februar 2005

#### Forth vanaf de grond 6
#### Ron Minke

Sechster Teil des Versuchs, ein AVR-Forth "von der Pike auf" auf die Beine zu stellen. Am Ende des Artikels steht "wird fortgesetzt". Besprochen und erläutert wird die Frage, ob indirekt gefädelt, direkt gefädelt oder unterprogramm-gefädelt. Erklärt werden NEXT, DOCOL und EXIT. Im AVR-Prozessor kann ausführbarer Code einzig und allein im FLASH liegen, im RAM nur Daten. Damit verbietet sich hier eine direkt gefädelte Forth-Version. Die **Annahme 9** muss also heißen: Für das AVR-Forth wird indirekte Fädelung gewählt.

#### Rückblick auf das Jahr 2004
#### Willem Ouwerkerk und Albert Nijhof

**Deutschland:** Besuch bei den östlichen Nachbarn auf Fehmarn in Deutschland und Teilnahme am Stand des deutschen Forth-Clubs auf dem Linuxtag in Karlsruhe.

**Tingel-Tangel:** Technische Überholung der beiden Tingel-Tangels und Software-Update für 2005 geplant.

**AVR-ByteForth:** Version 2.07 erschienen.

**Ushi:** Gerard van der Sel hat über ein Infrarot-Interface ein ATS an Ushi angeschlossen.

**Mais:** Für das Camel-MaisForth (6809-Projekt) hat Albert Nijhof einen völlig neuen Metacompiler geschrieben.

**Egelwerkboek:** Das überarbeitete PDF steht auf der Website der niederländischen Forth-Freunde.

**ANSI-Forth-Kurs:** Hier steht folgende Meldung: "Die Programmiersprache Forth" wurde von Fred Behringer, Vorstandsmitglied der deutschen Forthgruppe, vom Niederländischen ins Deutsche übersetzt. Siehe: http://www.mv-buchhandel.de/mv-buchhandel.htm

**Festplatteninterface:** Mark Hartjes hat an eine ATS-Platine ein Festplatten-Interface angeschlossen.

**Platinen:** Unsere Sammlung wurde um drei neue Platinen ergänzt.

**Allgemeine Mitgliederversammlung 2005**

Die diesjährige allgemeine Mitgliederversammlung der HCC-Forth-gg findet unter der unten stehenden neuen Treff-Adresse am 9. April von 13.30 bis 14.30 statt:

**Aus der Tagesordnung:**
5. Vorstandswechsel: Albert van der Horst und Dick Willemse können ab-, aber auch wiedergewählt werden.

**Adresse der Redaktion des Vijgeblaadjes:** a.nijhof@kader.hobby.nl

**Adressänderung:**

Ort der Treffen ist ab jetzt: Boslaan 3, 3722AB Bilthoven
(siehe Karte auf der Website: http://www.forth.hccnet.nl/nieuws).

### Nr. 49, April 2005

#### Forth vanaf de grond 7
#### Ron Minke

Siebenter Teil des Versuchs, ein AVR-Forth "von der Pike auf" auf die Beine zu stellen. Am Ende des Artikels steht "wird fortgesetzt". Besprochen und erläutert wird die Frage der Stacks und der Speicheraufteilung. Das ganze Gebilde besteht aus den folgenden Bausteinen: ATmega162, 74HC573, RAM 32KB, Resetschalter, dazu ein Pufferkondensator und ein Pull-up-Widerstand, MAX 232, Oszillator 8 MHz - mehr nicht! Für ein "Standard-Forth" für Experimentierzwecke reicht ein Datenstack von 32 Zellen (à 16 Bit) und ein ebenso großer Returnstack. Es wird ein FIG-Forth-Modell aus dem Jahre 1982 gewählt, da dieses überall in der Literatur gut beschrieben ist. Der ATmega162 enthält eine serielle Anschlussmöglichkeit in Form eines eingebauten UARTs, 1024 Bytes an RAM und 8 KB Flash. CODE-Definitionen werden (neben dem eigentlichen Forth-Kern) in den Flash-Speicher gelegt (Code kann beim ATmega162 nur dort ausgeführt werden, nicht im RAM!), die zugehörigen Headers (und Links!) werden beim Booten ins RAM übertragen. Beim Booten wird auch der Grundstock an Colon-Definitionen ins RAM gebracht. Alle benutzererzeugten Colon-Definitionen gehen ins RAM. Der Benutzer kann nachträglich leider keine CODE-Definitionen mehr einbringen. Ein Meta-Compiler ist vorgesehen.

#### Mus2, een betere bril voor Ushi
#### Willem Ouwerkerk

Ushi, der Roboter, den wir auf der Forth-Tagung 2004 auf Fehmarn bewundern durften, bekommt eine bessere Brille. Mus (1), der Mini-Ultra-Schall-Abstandsmesser, erwies sich auf Ushi als zu störanfällig. Der Autor dachte über eine Weiterentwicklung nach. Mus 2 hat neben einem besseren Sender auch einen 12-Volt-Generator "an Bord". Dahinter steckt die Idee,

dass bei höherer Sendeleistung der Empfänger robuster aufgebaut werden kann. Behandelt wird die Frage "Wie kriegt man solch eine neue Platine zum Laufen?".

**Für Maurits**
**Albert Nijhof**

Een Forthprogrammeur te Amsterdam Noord
(hij werd betaald, helaas, per woord)
at 's zondags gehakt,
want Forth is compact,
maar,
sinds hij voor C gekozen heeft,
dineert hij wekelijks met kreeft
in restaurant "De Zeevaarderspoort".

Übertragungsversuch:

Ein Programmierer aus Amsterdam-Nord,
(Bezahlung bekam er für Forth nur pro Wort),
aß sonntags Gehackt-
es, denn Forth ist kompakt.
Doch seit er nun C sein Arbeitsfeld nannte,
tut Hummer er speisen im Restaurante
"De Zeevaarderspoort".

---

Liebe Direktoren, lieber Fritz,

Thomi Dammann schickte einen Hinweis an den Secretary. Sendet ihn bitte weiter an Leute, von denen ihr annehmt, dass sie sich dafür interessieren und engagieren wollen. Fritz, bringst du bitte einen Hinweis auf die angegebenen Links in der nächsten VD? Thomi Dammann ist unter

**Thomi35@bigfoot.com erreichbar.**

Herzlichen Gruß
Rolf

= = = = = = = = = = = = = = = = = = = = = = = = =

Liebe Forth'ler,

zu der Frage, was man tun kann, habe ich gerade eine E-Mail vom FFII erhalten, die ich Euch weiterleite: Es gibt immer mehr regionale (landesweite) Verbände, in denen sich KMUs vereinigen, um gegen Software-Patente vorzugehen.

Liebe Grüße
Thomi (Dammann)

Betreff:
[ffii] Pressemitteilung zum Start der Initiative baden-württembergischer Unternehmen gegen Patentierbarkeit von Software Datum: Tue, 12 Apr 2005 00:31:46 +0200 Von: Marco Schulze <mail@bw-gegen-softwarepatente.de> An: neues@ffii.org

*weiter: Seite - 8 -*

Lieber Friederich!

Ich merke gerade, daß ich dir seit letztem November, kurz nach dem Forth-Tag, nicht geschrieben habe. Es wird Zeit, dir für die Vierte Dimension 1/2005 zu danken (die in meinem Briefkasten vor zwei Tagen eintraf) und meine Freude auszudrücken, daß du die SVFIG Bilder in diese Ausgabe aufnehmen konntest. Dank deiner Bemühungen war das Heft weder dünner noch weniger interessant als die vorhergehenden.
Ich nehme an, daß ich bis zum März Zeit habe einen weiteren Bericht zu schreiben. Ich werde es also kurz machen.
Ich möchte die traurigen Neuigkeiten allen Freunden von Chuck Moore auf Eurer Seite des Atlantik mitteilen. Chuck verlor seine Frau am 11. Januar. Sie starb unerwartet in Phoenix, Arizona. Die zweite Trauerfeier wird am 27. Februar in Half Moon Bay in Kaliforniern stattfinden.
Ein paar Worte über das SVFIG-Treffen am 11. Dezember 2004, ehe ich es vergesse. Wie zu dieser Jahreszeit zu erwarten war, kam nur eine kleine Gruppe von ca. 10 Personen zum Treffen. Dr. Ting, wie immer voller Ideen, füllte den Morgen mit Orgelmusik von Bach (in F#, falls ich so sagen darf, und ich bin sicher, daß Dr. Ting eine Anregung aus Dr. Behringers Kommentar über das "Fis" auf Seite 28 der VD1/05 bekommen wird). Wie Ihr seht, ist die Zeit für Dr. Ting gekommen, seine Bach-Aufzeichnungen vom alten 4.77 MHz PC auf die neueren Computer, die Windows und MIDI-Files nutzen, zu übertragen.
Ich muß meine alten Berichte überprüfen, um zu sehen, ob das ursprüngliche Projekt in FPC geschrieben war - die neue Version ist in F# geschrieben, welches seine Antwort auf Win32Forth ist. Wieder einmal sind die technischen Details zu kompliziert für mich, um sie zu beschreiben. Jedoch, wie üblich, wenn ich Dr. Ting zuhöre, lerne ich immer etwas Neues. Ich lernte, dass es in der chinesischen Musik keine Harmonien gibt und das um 1900 die Chinesen eine vereinfachte Musiknotation erfunden hatten, die heute noch in Gebrauch ist. Und obwohl meine beiden Eltern Musiker sind, weiß ich viel weniger über Musik, als die meisten Forther, die ich in der SVFIG traf.
Wie ich schon zuvor gesagt habe, werde ich weiter zu den SVFIG-Treffen gehen, um mich weiterzubilden und zu unterhalten. Wo sonst kann man so viel kostenlos über Computerwissenschaften, Mathematik, Chemie, Physik, Medizin, Musik, Philosophie usw. lernen - und das alles in einem einzigen Klassenzimmmer?
Der Rest des Tages war im wesentlichen Unterhaltung, da kein anderer Vortragender vorhanden war. Die Unterhaltung betraf eine ganze Menge verschiedener Sachen, neben Kevin Apperts Witzen und spontanen Wortspielen und geistreichen Bemerkungen aller anderen. Im Wesentlichen kann das ganze in die Gruppen "Einführungen, Ankündigungen, Gerüchte, Plauderei" eingeteilt werden, wie es seit vielen Jahren auf unseren Treffen nach dem Mittag üblich ist, wobei es oft auch länger andauert und manches Mal mehr Enthusiasmus als die schläfrig machenden Vorträge am Nachmittag hervorbringt. So endete das letzte SVFIG-Treffen für 2004. Ich werde mit 2005 das nächste Mal fortfahren.
Mit besten Grüßen

*Henry*

# Initiative gegen Softwarepatente

PRESSEMITTEILUNG 2005 April 11

Neue Unternehmerinitiative gegen Brüsseler Pläne zur Einführung von Softwarepatenten

Daß sich in Brüssel mit Softwarepatenten eine Existenzgefährdung von historischer Größenordnung zusammenbraut, bemerken nun zunehmend mehr mittelständische Unternehmen. Um sich dagegen zu verteidigen, haben seit Oktober letzten Jahres mehr als 250 Firmen Initiativen in sechs Bundesländern gestartet. Am heutigen Montag geht in Baden-Württemberg die siebte Initiative online. Noch rechtzeitig vor der zweiten Lesung im Europaparlament Anfang Juni formieren die Betroffenen Widerstand gegen die von der EU-Kommission vorangetriebene und vom Ministerrat akzeptierte Richtlinien-Version zur Einführung von Softwarepatenten.

Die Unternehmen setzen sich dafür ein, daß die Vorstellungen des EU-Parlamentes umgesetzt werden, welches sich in der ersten Lesung Ende September 2003 mit einem klaren Nein gegen die Patentierbarkeit von Software ausgesprochen hat. Der Ministerrat jedoch hat in seinem Gegenentwurf vom Mai 2004 sämtliche relevanten Vorschläge des Parlamentes gestrichen. Am 07. März dieses Jahres wurde dieser kontroverse Gegenentwurf dann unter Verletzung der eigenen Regeln und gegen den expliziten Willen sowohl mehrerer Regierungen als auch nationaler Parlamente (darunter auch des Deutschen Bundestages) beschlossen.

Wenn es dem Europäischen Parlament nicht gelingt, in seiner zweiten Lesung - voraussichtlich am 2. Juni - die bereits im Jahr 2003 beschlossenen Änderungen erneut durchzusetzen, werden mittelständische Unternehmen in Deutschland und Europa zu Freiwild für die größtenteils US-amerikanischen Schwergewichte in der Software-Branche.

Da es für das EU-Parlament auf Grund mehrerer prozeduraler Hürden sehr schwierig wird, sich in einer zweiten Lesung gegen den Ministerrat durchzusetzen, sehen die Unternehmen somit akuten Handlungsbedarf. Sie wollen sich daher nun persönlich an die Abgeordneten wenden. Die Initiativen dienen dabei als Sprachrohr, da viele der klassischen Branchenverbände, wie z. B. die Bitkom, gesteuert von Großkonzernen wie Siemens und SAP, gegen die eigenen mittelständischen Mitglieder lobbyieren.

Weitere Informationen zur Initiative: http://bw-gegen-softwarepatente.de

URL dieser Mitteilung: http://www.bw-gegen-softwarepatente.de/docs/2005_04_11_Pressemitteilung.pdf

---

Initiative baden-württembergischer Unternehmen gegen Patenttierbarkeit von Software - Marco Schulze, NightLabs GmbH, Rehlingstr. 6 d, 79100 Freiburg, Germany.

# EINE NEUE ART

# DER

# COMPUTERPROGRAMMIERUNG

von

**BERNARD A. HODSON**

bernard@genetix.ca

**Übersetzt von Fred Behringer**

Zusammenfassung

Die Industrie wird heutzutage von zahlreichen Problemen geplagt: Unsichere Betriebssysteme, Viren, Würmer, Spams, Diebstahl von Kennwörtern, Eindringen in persönliche Systeme, Abfangen von Funkdaten, Abhören von Satellitendaten, Hacker und so weiter und so fort. Die Kosten, die der Industrie allein durch Spams verursacht wurden, sind hoch und Viren hatten verheerende Auswirkungen auf die Geschäftswelt, ja, sie trieben einzelne Unternehmen gar in den Ruin. Die Sicherheitsbedrohungen von einzelnen Personen, von Unternehmen und Ländern wachsen an. Es wird höchste Zeit, dass wir wirksame Lösungen ins Auge fassen und uns an Lösungen orientieren, die den größtmöglichen Erfolg versprechen.

Der vorliegende Artikel beschreibt die Möglichkeit einer solchen Lösung. Er umreißt ein Programmierparadigma, das zu einem Standard entwickelt werden könnte. Erfolgreiche Anwendung fand es bereits auf verschiedenen Computerebenen, vom Großrechner über Mikrocomputer bis hin zu 8-Bit-RISC-Chips für Smart-Cards und eingebettete Systeme. Das vorzuschlagende Paradigma ist ein Standard, das auf allen Stufen der Programmierung angewendet werden kann und eine beträchtliche Flexibilität in Hinsicht auf die Anpassung an Benutzerwünschen bietet. Man könnte mit ihm so ziemlich alle der eingangs erwähnten Probleme ausräumen und es liefert Systeme, die so klein sind, dass das gesamte System auf jedem Computer und Server verschlüsselt werden könnte. Es würde sämtliche Betriebssysteme vereinfachen oder gar überflüssig machen.

Das Paradigma verwendet eine erweiterbare Sprache, die auf jedem beliebigen Computer in einen Byte-String verwandelt werden kann. Der Byte-String ist völlig unabhängig vom Zielcomputer der jeweiligen Anwendung. Mithilfe der für das Paradigma festgelegten Regeln kann jede in der erweiterbaren Sprache geschriebene Anwendung über einen einfachen Compiler verarbeitet werden. Ja, die Regeln sind so einfach, dass man Anwendungen entwickeln kann, die gar keinen Compiler benötigen. Man kann zur Verarbeitung durch das Laufzeitsystem

| | | |
|---|---|---|
| Anwendungszeile 1<br>Anwendungszeile 2<br>usw. | Der Compiler ist sehr klein | Bytecode-String (Code 0 bis 255)<br>z.B.: 2^*8aj~/2k{90>x |

Bild 1

auch gleich von vornherein einen Byte-String erzeugen. Für weitergehende Fragen zur Beschaffung eines solchen einfachen Compilers und einer rudimentären erweiterbaren Sprache zum Ausprobieren dieser Programmierphase ist der Autor unter bernard@genetix.ca erreichbar.

Das Laufzeitsystem arbeitet den erzeugten Bytecode-String ab. Das geschieht über ein doppeltnumerisches System, welches jedes Element, das in der Anwendung benötigt wird, eindeutig kennzeichnet. Auf diese Weise lässt sich das Laufzeitsystem des virtuellen Prozessors sehr klein halten, etwa drei- oder vier tausend Bytes in 8-Bit-RISC-Chips, welche typische Smart-Card- oder Embedded-Systems-Anwendungen enthalten, bis hin zu sieben- oder achttausend Bytes für einen Microcomputer mit einfacher Grafik, und ein bisschen mehr an Bytes, wenn Video-Bilder und andere anspruchsvolle Anwendungen verarbeitet werden sollen. Das zum Einsatz kommende numerische Codierungssystem identifiziert eindeutig jede auszuführende Aktivität und sorgt für schnell laufende Anwendungen. Die numerischen Codes des Paradigmas sind eindeutig und erlauben das nachträgliche Einfügen neuer Möglichkeiten, ohne das bisher schon Entwickelte zu beeinflussen.

## COMPILIEREN ODER NICHT COMPILIEREN, DAS IST DIE FRAGE

Der Autor hat Erfahrung in der Entwicklung von Compilern und auch eines JAVA-Laufzeitsystems. Fortran und Cobol (wie auch C, C++, PL1 und andere Compiler) erzeugen eine Maschinensprachstruktur, die von einer Bibliothek von Unterprogrammen Gebrauch macht. In JAVA wird ein Bytecode-String erzeugt, der eine Bibliothek von Methoden und ähnlichen Strukturen nötig macht. Die Bibliotheken für beide Programmierarten neigen dazu, recht groß zu werden. Fortran und Cobol haben nur recht beschränkte Möglichkeiten, und JAVA verwendet einen überreichen und umständlichen Wortschatz. Selbst das allgegenwärtige ganz einfache ‘Hallo Welt’-Programm benötigt in JAVA eine Riesenmenge an Methoden und Ressourcen.

Der eigentliche Compiler für das Paradigma des vorliegenden Artikels ist sehr klein und kann dem Laufzeitsystem wahlweise auch vorangestellt werden. Er nimmt (da er mit dem Laufzeitsystem gemeinsame Routinen verwendet) nur ein paar Hundert zusätzliche Bytes in Anspruch. Dabei werden dem System statt der Bytecodes die Sprachelemente zugeführt und das System erzeugt zunächst die Bytecodes, bevor es diese dann abarbeitet. Eine solche Betriebsart ist besonders für sicherheitskritische Anwendungen (bei denen der Compiler UND die Anwendung jedes Mal überprüft werden müssen, wenn eines der beiden abgeändert wird) nützlich. Aus Gründen der Ausgeglichenheit des Restes der vorliegenden Arbeit wollen wir das Compilieren jedoch als schon erledigt betrachten und annehmen, dass dem System ein String von Bytecodes präsentiert wird. Der Compiliervorgang wird in Bild 1 gezeigt (Der Compiler wandelt Sprachelemente in Bytecodes um).

Alle Elemente des Laufzeitsystems sind statisch. Der einzige Teil des Paradigmas, der als variabel betrachtet werden kann, ist der erzeugte Bytestrom. Er ändert sich von Anwendung zu Anwendung.

## SPRACHELEMENTE

Jede Anwendung besteht aus einem String von Sprachelementen, die mit Parametern, wie Zahlen oder Namen von Variablen, verknüpft sind. Mit Ausnahme von numerischen Daten und festen Werten werden alle Sprachelemente und Variablen in jeweils ein einziges Byte umgewandelt. Jedem Sprachelement ist eine Zahl zugeordnet, die momentan von 0 bis 255 läuft. Von diesen Zahlen wird im Augenblick jedoch nur ein Bruchteil verwendet. Es ist unwahrscheinlich, dass jemals mehr als 256 benötigt werden. Aber selbst wenn, kann der Bereich bis nahe an 65536 heran erweitert werden, ohne dass das zuvor Entwickelte in irgendeiner Weise beeinflusst wird.

Ein paar typische Beispiele von Sprachelementen:

```
looping 1 1 100 adr grt
screen ^hello world to^ name
bitmap
arith alpha = beta + gamma / delta + 13
```

Die hier gezeigten Sprachelemente, denen ein numerischer Code, wie 3,5,+,* , zugeordnet wird, sind ‘looping’, ‘screen’, ‘bitmap’ and ‘arith’.

Für das Element ‘looping’ stellen die Zahlen 1 1 100 Schleifenparameter dar, die von 1 bis 100 mit der Schrittweite 1 laufen. Die Symbole adr und grt legen fest, was beim Ergebnis true oder false geschehen soll. Solch ein Sprachausdruck kann beispielsweise die Bytecode-Sequenz 2(1(1(d25 zur Folge haben.

Die (1 zeigt an, dass ein numerischer Wert in sein binäres Äquivalent, hier eine 1, umgewandelt wurde. Die 25 bedeutet, dass in Abhängigkeit vom Ergebnis der Schleifenarithmetik das zweite oder fünfte mit Namen versehene Sprachelement übertragen werden soll (das wird während des Compiliervorgangs automatisch erledigt). Andere Sprachelemente liefern andere Formen der Schleifensteuerung.

Das Element ‘screen’ könnte die Bytecodefolge 511 nach sich

ziehen, die anzeigt, dass der erste numerische Wert auf den Bildschirm gesetzt werden soll, gefolgt von der ersten Variablen, die dann wohl den Namen derjenigen Person enthält, an die die Nachricht geht. Wie schon erwähnt, wird es nur wenige Anwendungen mit mehr als 256 Variablennamen geben. Während dies die Grenze für ein anfängliches System ist, können Erweiterungen auf nahezu 65536 Variablennamen eingerichtet werden, ohne das zuvor Entwickelte zu stören.

Das Element 'bitmap' könnte den aus einem einzigen Byte bestehenden Bytecode + nach sich ziehen, welcher im Laufzeitsystem eine Folge von Vorgängen auslösen würde, die den Namen des Bitmap-Bildes verlangen, das erzeugt werden soll.

Das letzte Sprachelement 'arith' könnte den Bytecode *47+6/3+%51~13 erzeugen, in welchem die 4. Variable zum Ergebnis hat, dass zur 7. Variablen die 6. hinzuaddiert, durch die 3. Variable geteilt und schließlich 13 hinzuaddiert wird.
Hier bedeutet das %, dass das, was kommt, eine Gleitkommazahl von der Länge 5 mit einem positiven Vorzeichen und dem Wert 13,0 ist. Die jeweiligen Zahlen sind auch hier wieder eine Funktion des Compilers und der Programmierer braucht sich um die Codesequenzen nicht zu kümmern.

Auf den ersten Blick mag die Struktur kompliziert erscheinen, die Umwandlung in Zahlencodes wird jedoch von dem äußerst kleinen Compiler vorgenommen und die Codes werden vom Laufzeitsystem in einfachster Weise abgearbeitet. Das System der Codierung braucht dem Programmierer nicht speziell bekannt zu sein. Das Laufzeitsystem entnimmt dem Bytecode-Strom alles Erforderliche und tut genau das, was es tun soll.

Und noch etwas ist wichtig: Ein böswillig in den Bytestrom gesetzter gefälschter Bytecode würde die Anwendung wahrscheinlich abstürzen lassen. Für empfindlichere Anwendungen könnte am Ende der Bytecodes eine Prüfsumme eingefügt werden, die für den Gesamtwert aller Bytes steht und die eine mehr oder weniger große Sicherheit gegenüber Hacker- und Virusaktivitäten bietet. Die Summe wäre dann am Anfang der Anwendung zu überprüfen.

## DER PROZESSOR FÜR DIE VIRTUELLE MASCHINE – DAS LAUFZEITSYSTEM

Das Laufzeitsystem besteht aus etwa 30 kleinen Modulen in Maschinensprache, deren genaue Zahl vom eingebauten Funktionsumfang abhängt. Die meisten dieser Module sind voneinander unabhängig, so dass sich die Größe des virtuellen Prozessors (VP) je nach den gewünschten Systemanforderungen reduzieren lässt (Smart-Card-Anwendungen beispielsweise werden kaum die Grafik- oder Bitmap-Module benötigen). Aber selbst ohne das wäre der VP für die meisten Systemanforderungen sehr klein, typisch etwa 4 bis 10 Kilobyte, je nach eingebautem Funktionsumfang.

Auf den VP kann über einen numerischen Code innerhalb des statischen Teils der Software (der in Abschnitten weiter unten beschrieben werden soll) zugegriffen werden.

Die meisten Module benötigen nur ein paar Bytes an Maschinencode. Eine Ausnahme machen nur solche Module wie Bitmap und Gleitkommaroutinen fürs Addieren, Subtrahieren, Multiplizieren, Dividieren und Austesten von Gleitkommazahlen (welche ähnlich wie im IEEE-Format, aber genauer aufgebaut sind). Es ist die Art der numerischen Codierung des statischen Teils, die einen solch kleinen VP möglich macht. Die Codierung erlaubt es dem VP zudem, sowohl die gewünschten Module als auch die zugehörigen Parameter direkt anzusprechen.

Die meisten Module beschäftigen sich mit dem Datentransfer in Bezug auf direkte oder indirekte Adressen und mit binären arithmetischen und logischen Operationen. Das stellte sich bei Durchsicht des vom Compiler erzeugten Codes in vielen Anwendungen aus dem Wirtschaftsbereich als das Einzige heraus, was wirklich benötigt wurde.

## SPRACHELEMENTE UND INTERNE ELEMENTE

Zusätzlich zum VP gibt es noch einen statischen Abschnitt (was den Hauptgrund dafür darstellt, dass solch ein kleines, aber von der Funktion her mächtiges System gebaut werden kann), der aus einer Folge von Zahlen besteht, die mit den einzelnen Sprachelementen verknüpft sind (für jedes Element eine), und einem System von numerischen Strings interner Elemente, die die Module im VP ergänzen. Sowohl die Sprachelemente als auch die die VP-Module ergänzenden internen Elemente sind das Ergebnis einer Untersuchung existierender Anwendungen aus dem Wirtschaftsleben. Beide können ergänzt und erweitert werden, ohne das zuvor Entwickelte in Frage zu stellen. Das lässt eine kontrollierte Entwicklung des Programmierkonzepts zu.

Der Anwendungsprogrammierer braucht die interne Struktur der Elemente nicht zu kennen. Dafür interessiert sich allein die ganz kleine Schar von Leuten, die mit Systemerweiterungen beschäftigt sind. Die allgemeine Form der Elemente soll jedoch kurz erläutert werden (siehe Bild 2).

Name1 A, B, C, m, D, E, F, G, name2, n, H, I, o, name7, endit

Bild 2: Typische Struktur eines Elementes

Jedes Element, ob Sprachelement oder internes Element, wie beispielsweise Name1 im Bild, erhält ein mnemonisches Kürzel zugewiesen, welches die Funktion des betreffenden Elements wiedergibt.

A,B,…. stellen eindeutig gegebene mnemonische Kürzel dar, die VP-Module mit geeigneten Parametern, wie 'screen' oder 'looping', aufrufen.

m,n… kennzeichnet eine Verzweigung in Abhängigkeit davon, ob das Ergebnis der vorausgegangenen Aktion ‘true’ oder ‘false’ war.

name2, name7…. beziehen sich auf andere interne Elemente-Strings, die hier verwendet werden sollen. Das aufgerufene Element braucht kein Sprachelement zu sein. Gerade diese Eigenschaft trägt auch wesentlich zur Kompaktheit des Systems bei. Es können mehrere Schichten von internen Elementen angesprochen werden, bevor das System zu dem VP-Modul zurückkehrt, das dem Aufruf eines anderen internen Moduls folgt.

*endit* ist eine spezielle Funktion, die das Ende eines Elements anzeigt. *Endit* braucht nicht unbedingt am Ende des Elements zu stehen, es markiert den logischen Schlusspunkt des Elements.

Die verschiedenen Posten A, m, namex werden vom Systementwickler festgelegt. Das System selbst hat die Aufgabe, die Bytecodes zu verarbeiten, die vom Compiliervorgang geliefert werden. Wie bereits erwähnt, können die Bytecodes auf jedem beliebigen System mit einem geeigneten Compiler, ja, sogar von Hand erzeugt werden.

## STRUKTUR DES NUMERISCHEN CODES

Der numerische Code ist eine Zahl zwischen 0 und 65535. Zahlen oberhalb 65500 sind für Kontrollfunktionen, wie endit und error, und für Beendigungsvorgänge reserviert. Zahlen unterhalb 512 deuten logische Übergänge innerhalb des Elementes an, während sich Zahlen unterhalb 32768, aber oberhalb 512 auf Stellen beziehen, an denen Namen von internen Elementen liegen. Was die internen Elemente betrifft, steht nicht zu erwarten, dass sie jemals mehr als 32768 Bytes benötigen. Im Moment erfordern sie nur etwa ein Viertel dieser Zahl. Falls das Konzept als Industrie-Standard akzeptiert wird, ist es vielleicht doch vorstellbar, dass die Anzahl der internen Elemente 32768 überschreitet. Es wurde aber eine Strategie entwickelt, mit der man, falls diese Situation jemals auftreten sollte, eine geordnete Aufstockung vornehmen kann.

Die Zahlen zwischen 32768 und 65500 beziehen sich auf die Verwendung von Modulen innerhalb des VPs. Der eine Teil der Zahl zeigt das spezielle Modul an, das verwendet werden soll, während der Rest der Zahl eindeutig die Stelle kennzeichnet, von der die Parameter geholt werden sollen. Es darf noch einmal betont werden, dass diese Zahlen schon auf der Compilierstufe aus den Sprach-Ausdrücken der Anwendung heraus zugeordnet werden und dem Anwendungsentwickler nicht bekannt zu sein brauchen.

## MEHRFACHANWENDUNGEN

Die meisten Anwendungen sind, was die Bytecode-Struktur betrifft, verhältnismäßig kompakt und es können viele Anwendungen gleichzeitig im Speicher gehalten werden, selbst auf Smart-Cards mit ihrem begrenzten Aufnahmevermögen, und auch in eingebetteten Systemen. In einer frühen Entwicklungsstudie war es ein komplettes Krankenhaus-Informationssystem, ein Online-Rechnungswesen und ein Wirtschaftskreditbearbeitungssystem, die alle dieselbe VP-Software verwendeten, jedes System mit einem eigenen Anwendungs-String.

Verwendet man Mehrfachanwendungen mit derselben Software, so braucht man eine Kontrolle über die Anwendungsnamen, um doppelte und nicht-eindeutige Namensgebungen zu vermeiden. Doch das bekommt man verhältnismäßig leicht in den Griff. Wenn man will, kann man den Mehrfachanwendungen auch ein, zwei oder drei Prioritätsebenen zuordnen. In einer solchen Umgebung werden alle Anwendungen mit Priorität 1 ein Stückchen weit abgearbeitet, dann eine mit Priorität 2, und das auf Prioritätsebene 2 solange, bis alle Anwendungen der Priorität 2 einmal drangekommen sind, und dann eine Anwendung der Priorität 3. Diese Round-Robin-Priorität stellt sicher, dass keine der Anwendungen im Gesamtablauf zu kurz kommt. Erreicht wird das durch einen einfachen Roll-in-roll-out-Prozess von Variablen-Daten innerhalb der Anwendung, der auch den Stack-Prozess einschließt, welcher die Mehrschicht-Verarbeitung der internen Elemente und der Sprachelemente steuert.

Eine andere nützliche Eigenschaft besteht darin, dass die Sprachelemente in jeder Sprache der Welt gefasst sein können, sogar von Anwendung zu Anwendung innerhalb einer Mehrfachanwendung verschieden. Ja, selbst in einer einzigen Anwendung kann man mit Hilfe von Synonymen mehr als nur eine natürliche Sprache verwenden. Die VP-Verarbeitung wird bei Verwendung von Synonymen etwas langsamer, das aber auch nur in unbedeutendem Maße.

## VERSCHLÜSSELUNG

Der VP und auch die numerischen Elemente sind recht klein und die Elemente könnten nach irgendeinem der gebräuchlichen Algorithmen unmittelbar verschlüsselt werden. Die Entschlüsselung bräuchte erst dann zu erfolgen, wenn sie benötigt wird. Dies würde einen geringen, aber ständigen zeitlichen Mehraufwand bedeuten. Eine andere Möglichkeit wäre, die Elemente in verschlüsselter Form zu speichern und sie erst bei Beginn des jeweiligen Laufes zu entschlüsseln. Für zeitunabhängige Anwendungen wäre das eine vernünftige Strategie. Für kontinuierlich ablaufende Anwendungen, wie sie beispielsweise bei der Überwachung von Pipelines oder Kernenergieanlagen vorkommen, wäre das wahrscheinlich weniger geeignet.

Für Leute mit geringerem Sicherheitsbedürfnis könnten diverse Prüfzeichen eingebaut werden, sowohl für den VP wie auch für die Elementabschnitte. Diese Prüfsummen können nötigenfalls in jedem Lauf oder in zufällig verteilten Intervallen abgefragt werden. Es ist unwahrscheinlich, dass irgendwelche Angriffe auf ein Element oder den VP unentdeckt bleiben würden.



ENTWICKLUNGSSTRATEGIE

Der vorgeschlagene Programmierstandard stellt den Höhepunkt von Jahren der Entwicklung dar, das meiste davon als Zulieferung zu den herkömmlichen Verfahren. Der jetzt vorgeschlagene Standard ist im Wesentlichen eine Tabelle von Zahlen mit einer kleinen Anzahl von zugeordneten Modulen, die die Zahlen entziffern und das ausführen, was bisher von Computerbefehlen erledigt wurde. Die Module werden sehr langsam in dem Maße anwachsen, wie die Konzepte von der Industrie akzeptiert werden. Bei den numerischen Elementen wird es mit steigendem Funktionsumfangs etwas schneller geschehen. Mit der hier skizzierten Verfahrensweise könnte das alles als eine einzige, unverrückbare Technik betrachtet werden, die allen jetzigen und späteren Verwendungsansprüchen genügt, von der Nanotechnik auf Molekularebene bis hin zu den mathematischen Expansionsbestrebungen der stärksten Supercomputer.

Die bereits früher verwendeten Konzepte lieferten erfolgreiche Anwendungen auf Großcomputern, auf Computern von mittlerer Größe und auf Mikrocomputern, aber auch bei Mikro-Controllern mit RISC-Chips.

Es wird noch eine Reihe von Jahren vergehen, bevor sich diese Konzepte in der Industrie durchgesetzt haben, aber durchsetzen werden sie sich ganz bestimmt. In erster Linie sollten sie an die Mikro-Controller für Smart-Cards und eingebettete Systeme angepasst werden. Diese stellen über 90% aller installierten Computersysteme dar, werden aber nicht von monopolistischen Softwarefirmen beherrscht. Auch sind die betreffenden Prozessoren untereinander nur schwach verbunden.

Man müsste dann einfache Netzwerke einrichten, die auf diese Konzepte aufbauen. (Die Verbindung und Steuerung von Hochgeschwindigkeitsnetzwerken war Bestandteil früherer Entwicklungen.) Das käme besonders für die Verwendung von Smart-Cards für die verschiedensten Zwecke infrage, so im Gesundheitswesen, bei finanziellen Transaktionen, bei der Personenerkennung und Ähnlichem. Damit würden die Konzepte in den Chips auf den Plastikkarten, in den Kartenlesern und in den Servern, die das Netzwerk steuern, Einzug halten.

Gleichzeitig sollte die numerische Vorgehensweise auf dem Gebiet der eingebetteten Systeme eingeführt werden, und das von speziellen Industriezweigen wie der Kraftfahrzeugtechnik und der Luftfahrt.

Sobald die Methode dann auf Mikro-Controller-Ebene Fuß gefasst hat, könnte sie auf größeren Systemen eingesetzt werden, zunächst einmal, indem sie sich in die dortigen Betriebssysteme integriert, dann aber auch, indem sie diese ersetzt. Die eigentlichen Betriebssysteme werden ja dann überflüssig. Durchzuführen wäre das am besten wieder von der Industrie (Server, Grafik, Video etc.). Aber nach erfolgreicher Einführung in der Mikro-Controller-Welt wird dann auch die Industrie bereit sein, sich anzuschließen.

Um das alles von den PC-verbundenen Industrie-Gruppierungen, die ja das Eigentum an Software hochhalten, ohne ein großes Durcheinander erreicht zu bekommen, wäre es nützlich, eine Arbeitsgruppe zu bilden, die eine geordnete Entwicklung des numerischen Verfahrens überwacht.

FREI VON VIREN, WÜRMERN UND KENNWORTDIEBSTAHL

Einer der Gründe, warum Würmer und Viren immer noch existieren, liegt darin, dass die gängigen Softwaremethoden auf Computersprachen beruhen, die eine riesige Infrastruktur erfordern. Die Komplexität der verwendeten Betriebssysteme macht es unmöglich, dafür zu garantieren, dass keine Sicherheitslöcher bestehen. Diese Schlupflöcher werden dann von übelgesinnten Zeitgenossen ausgenützt, um Programme in die Welt zu setzen, die Millionen von Benutzersystemen zum Erliegen bringen und die sich zu einem nationalen Sicherheitsrisiko ausweiten können. Solange diese Verwundbarkeit nicht beseitigt wird, ist kein Anwender, keine Firma und kein Land vor bösartigen Angriffen auf ihre lebenswichtigen Computereinrichtungen sicher.

Das Bedürfnis, von dieser Problemvielfalt fortzukommen, ist einer der Gründe, weshalb man zu dem numerischen Verfahren übergehen sollte. Die numerische Vorgehensweise liefert:

1. Einen sehr kleinen VP, der statisch ist, und der, wenn Prüfsummen eingeschaltet werden, keinen schädlichen Code eindringen lässt.

2. Einen sehr effizienten VP, der erforderlichenfalls verschlüsselt werden kann.

3. Ein statisches System von Elementen, die eine Folge von Anwendungen numerisch in einer Form beschreiben, wo jede Zahl innerhalb eines Elements direkt auf den verlangten VP-Prozess weist.

4. Ein statisches System von numerischen Elementen, die über Prüfsummen verifiziert werden können.

5. Selbst in dem unwahrscheinlichen Fall, dass eine falsche Zahl in ein numerisches Element eingebracht wurde, ohne die Prüfsumme anzutasten, würde das Anwendungsprogramm wegen der empfindlichen Beziehungen zwischen den einzelnen Zahlen innerhalb der Elementstruktur sofort aussteigen.

SICHERHEIT

Der einzige Prozessteil, in welchem das numerische System nicht die volle Kontrolle hat, ist der vom Compiler erzeugte Bytecode-Strom. Aber selbst dieser hat keinen Zugang zu den numerischen Elementen oder zum kleinen VP-Maschinencode. Und auch beim Verarbeiten von Mehrfachanwendungen kön-

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

**HCC-Forth-gebruikersgroep.**

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

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

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

nen Daten der einen Anwendung keine Daten von anderen Anwendungen zerstören, vielleicht mit einer Ausnahme: Es gibt Anwendungen, die sich Daten teilen. In solchen Fällen muss der Anwendungsentwickler die Sache klären.

FEHLERFREI

Es ist zwar noch keine Analyse durchgeführt worden, man kann aber davon ausgehen, dass der VP so klein ist, dass seine Fehlerfreiheit nachgewiesen werden kann. Ähnliches gilt für jedes einzelne numerische Element. Numerische Elemente und der VP sind statisch. Wenn man sie einmal als fehlerfrei erkannt hat, gibt es so gut wie keinen Grund, sie abermals zu testen.

WEITERGEHENDE FRAGEN?

Eine Demonstration der hier genannten Verfahren kann auf individueller Basis arrangiert werden. Auch die Erstellung von Dokumentationen zur Weitergabe an andere käme infrage. Wahrscheinlich werden die Dinge aber eher verstanden werden, wenn der Autor in einer Vorführung vor Einzelnen oder in Gruppen in gedrängter Form die Architektur seines Systems erklärt. Wenn er erklärt, wie man damit wirksam die Probleme der heutigen Informationstechniken bekämpfen kann, die jährlich über 46 Milliarden Dollar kosten und ständig wachsen, und wie man der ungeheuren Verschwendung an Betriebssystemmitteln begegnen kann. Weitergehende Fragen zum vorgeschlagenen Programmierverfahren beantwortet der Autor gern unter

**bernard@genetix.ca.**

*Einen Folgebeitrag diskutiert Fred Behringer auf Wunsch der VD bereits mit Bernard Hodson.*
*Hat Ihnen der vorliegende Beitrag gefallen oder sogar genutzt? Bytecodes und deren Interpreter sind für „den Forther" nicht Neues. Das gilt auch für die von Hodson erkannten Potentiale. Wollen Sie trotzdem mehr lesen?*
*Sagen Sie dem Autor und der VD Ihre Meinung!*

*fep*

FIG UK

(Englische Forth-Gesellschaft)

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

Wenden Sie sich an:

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

# Die Web-Seite der FG in Kürze

(Kurzeinführung in geeklog)

(Aufruf zur Mitarbeit)

von **Michael Kalus**

Die Pflege einer Website ist, wenn sie aktuell bleiben soll, eine mühsame Aufgabe. Das haben mit uns viele andere schon erfahren müssen - und das Weblog erfunden. Ein Weblog füttert den Browser, wie es für schnellen Seitenaufbau wünschenswert ist, mit rein statischen Html-Seiten. Aber diese werden erst in dem Moment auf dem Server erzeugt, in dem der Browser sie anfordert.

Damit sind die Seiten immer aktuell, sogar aktueller als eine Zeitung es sein kann. Denn die Beiträge werden in eine Datenbank eingegeben, mit Erscheinungsdatum und Datum, wann sie wieder verschwinden sollen. Der Text kann dort als „plain old text“ abgelegt werden. Das ganze Layout erfolgt erst, wenn die Seite aufgerufen wird, alle Schriften, Bilder, Tabellen, Rahmen und solche Elemente der Gestaltung wie Hintergründe werden erst in dem Moment von einem ‚stylesheet’ aus generiert, der Text wird automatisch aus der Datenbank hinzu gegeben.

Geeklog ist so ein Weblog, und es ist freie Software. Eine prima Arbeit ist da veröffentlicht worden.

Die Seiteninhalte sind in Kategorien gegliedert. Beidseits der Sachgebiete bilden Blöcke einen Rahmen, in den Verknüpfungen als Elemente der Navigation gesetzt werden. Im Kopf der Seite bleibt eine Navigationsleiste dauerhaft stehen. Neben den Sachgebieten gibt es schon das fertige Modul eines Kalenders, aus dem heraus automatisch Ereignisanzeigen generiert werden. Umfragen, kommentierbare Link-Listen, die Möglichkeit eigene Steuerblöcke zu erstellen und Dateien thematisch geordnet zum Herunterladen ablegen zu können, das gehört ebenfalls zur Ausstattung. Geeklog läuft unter vielen verschiedenen Betriebssystemen und benötigt PHP4 und MySQL.

Das besondere dabei: Man kann vom Browser aus Beiträge hinzufügen und Kommentare zu vorhandenen Beiträgen schreiben.

Geeklog kann, was unsere Homepage will:

Nachrichten in Kategorien ordnen.
Ereignisse verwalten.
Dokumente ablegen.
Datei Service ordnen.
Links online verwalten.
Viele Administratoren können mitmachen.
User können Beiträge verfassen.
Themen sind umschaltbar.

Geeklog passt gut zu dem, was wir schon auf der bisherigen Homepage hatten. Unsere Inhalte können dort klar geordnet dargeboten werden. Und: Es ist teamfähig! Damit gelingt es uns vielleicht, den Stillstand zu überwinden, der sich inzwischen über unsere Site gelegt hatte.

Derzeit stelle ich die Inhalte der alten Homepage in das Geeklogsystem ein. Danach kann es losgehen, unsere Homepage frisch und informativ zu halten und einen guten Service zu bieten.

Komm ins Team!

*Mka*


Nun sind Drachen trotz ihrer allgemein bekannten Flugtauglichkeit auch nicht für Leichtfüßigkeit, Grazie und besondere Anmut bekannt, aber swappende Gnus sind sicher eine separate Erwähnung wert. Mehr darüber gibt es natürlich bei

**http://www.jwdt.com/~paysan/gforth.html**

zu lesen.

*fep*

*2005 – Tagung in Sachsen – Prinz; Hoffmann, Bitter*
*Photos: J.Wilke*

# Just in Time

## Optometer P9710 auslesen

**Claus Vogt**

0177-295 24 17
Yorckstr.75
D-10965 Berlin

„Hast Du vielleicht mal was mit der seriellen Schnittstelle gemacht? Kannst Du vielleicht? Muss aber schnell gehen?“ Hatte ich, konnte ich, war aber lange her. In zwei Stunden sollte das Versuchsfahrzeug rausfahren. Was tun? In Visual C++ die API-Dokumentation lesen und wohl gar verstehen: Unmöglich in der Zeit. Also griff ich zu meinem Lieblingswerkzeug MS Excel.

### Excel liest COM

Irgendwo hatte ich mal gelesen, dass seit WindowsXX beim Öffnen von Dateien nicht mehr auf Spezialgeräte wie ‚COM1’ zugegriffen werden darf. Aber probieren kann man es ja mal. So wuchs mit ein wenig Herumtesten die nebenstehende VBA-Lösung heran.

Die Aufgabe war prinzipiell erfüllt: Es sollte dem Messgerät per COM1 die Zeichenkette ‚MV’ gefolgt von einem Linefeed (Ascii 10) gesendet werden. Seine Antwort, so was ähnliches wie ‚5.7122E+02’ sollte in eine Datei geschrieben werden, möglichst mit Ort und Datum in jeder Zeile.

Schade, dass sich das Programm nicht abbrechen lässt. Aber ich habe partout keine VBA-Anweisung gefunden, die eine Taste einliest und eine Messagebox alle 10 Sekunden schien mir benutzerfeindlich. Na, mit <Ctrl>+<Break> wird man schon wieder rauskommen. Und auf dem Bildschirm möchte ich gerne auch was sehen, aber da gibt es diese Debug.Print-Anweisung.

Beim Testen mit dem Hyperterminal hat Excel auch brav kommuniziert. Aber wie stelle ich per Excel die Kommunikationsparameter ein, wenn der Hilfetext schon von COM1 nichts weiß? Der Hilfetext fing sowieso immer mehr an zu spinnen. Dass er bei jedem zweiten Tastendruck MS Project installieren wollte, kannte ich schon. Ist halt eine mies installierte Krücke, an der ich hier arbeite. Plötzlich will die Hilfe aber auch nicht mehr nach ‚get’ oder so was suchen.

Mit dem realen Messgerät ging erst mal gar nichts. Vielleicht die falschen Kommunikationsparameter? Aber sowohl Hyperterminal, als auch die Systemsteuerung melden 9600-8N1 ohne Handshake. Warum macht das Hyperterminal eigentlich eine allgemeine Schutzverletzung, wenn ich es mit COM1 starte? Na, vielleicht hat mein Excel die Schnittstelle in Benutzung, ja, tatsächlich. Aber irgendwann spinnt Hyperterminal auf beiden Schnittstellen und mein Programm ist schon zu. Na ja, also neu starten. Dabei sind schon eineinhalb Stunden rum und nur noch eine halbe Stunde übrig.

### Boot tut gut

Die Begeisterung für Excel lässt nach. Während der Rechner durch Neustart meine und seine Zeit verplempert, trifft eine kleine aber nicht unwesentliche Ergänzung der Aufgabenbeschreibung ein. Der Kollege hatte einige andere Kollegen erfolglos angeklingelt, ob sie vielleicht das passende Programm in der Schublade haben. Jetzt hat er wieder Zeit und erinnert sich: Der Laptop im Versuchsfahrzeug hat gar kein Excel installiert. Also tritt Plan B in Kraft.

### Term4

Vor Jahrenden, so ca. 1990 hatte ich mal ein volksForth-Programm zur Kommunikation über die serielle Schnittstelle geschrieben. Term4 hieß es und ins Internet hatte ich es auch mal gesteckt: http://home.arcor.de/clv/download/term4_22.zip. Damals hatte ich es so gestrickt, dass man schnell mal ins Forth wechseln und ein paar Zeilen schreiben konnte. Das Wegspeichern zusammen mit den Kommunikationseinstellungen konnte es sowieso. Und in Dateien mitprotokollieren konnte es sowieso. Na ja, probieren kann man es ja mal.

Nach etwas Rumprobieren konnte ich auch wieder Forth und wusste wieder in etwa, was diese Worte ‚rx’, ‚rx?’ und ‚?rx’ so machten.

So wuchs dann eine Art Quelltext. Gottseidank ließen sich die paar Zeilen auch ohne Editor eingeben, denn den Editor hätte ich nicht gerne benutzt. Vor dem habe ich großen Respekt. Den darf man nie ohne Betriebsanleitung und Sicherheitsbeauftragten starten.

Nach einigen gescheiterten Versuchen ergab sich ungefähr folgender Quelltext:

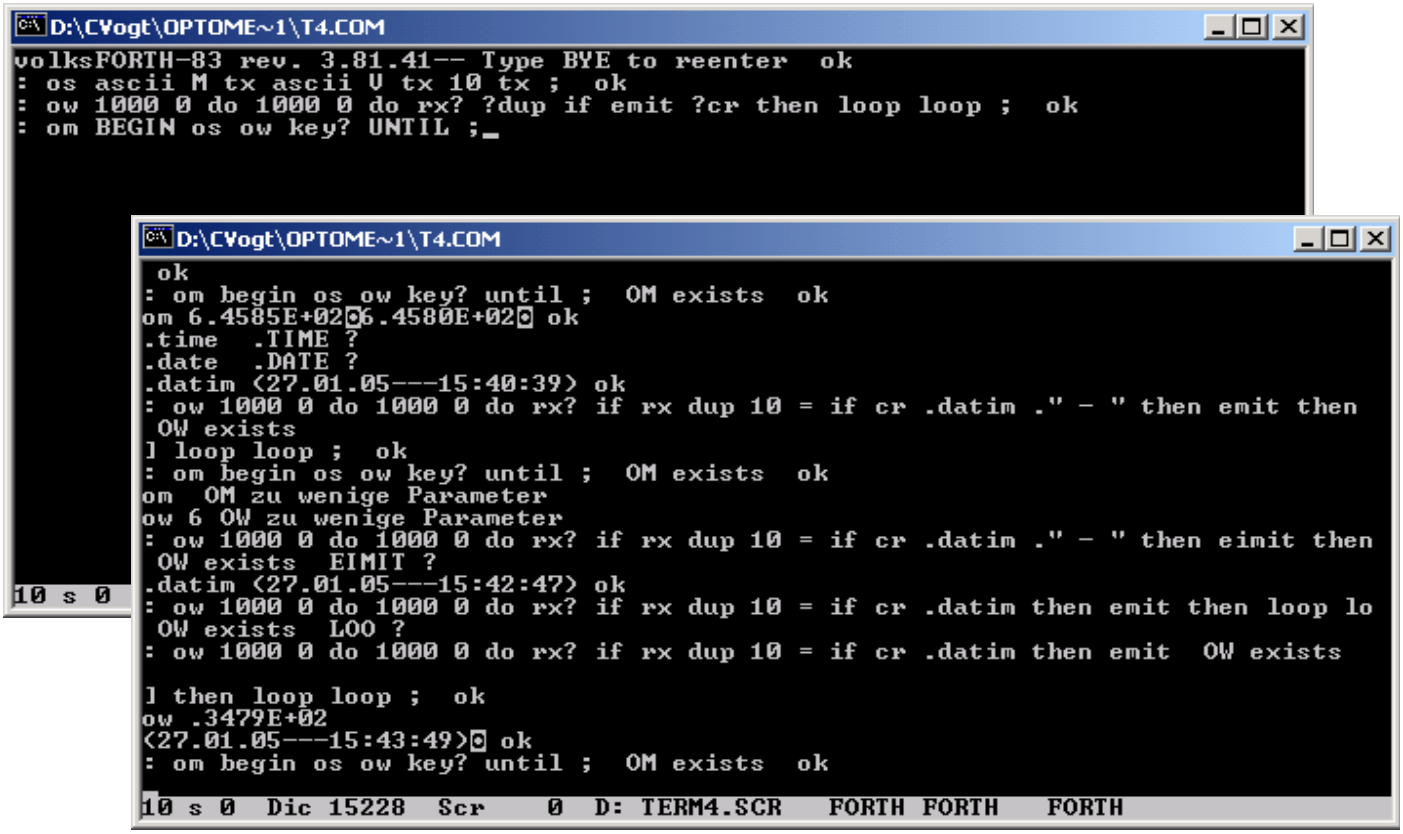
```
: os ascii M tx ascii V tx 10 tx ;
: ow 1000 0 do 1000 0 do rx? if rx
  dup 10 = if cr .datim then emit
  then loop loop ;
: om begin os ow key? until ;
```

Wie man leicht erkennt, ist der Quelltext selbstdokumentierend. Was man in der Programmiersprache ausdrücken kann, soll man ja nicht durch Kommentare verwässern. Das liest sich auch ganz prima, wenn man sprechende Namen wie ‚os’, ‚ow’, ‚om’ verwendet, die ja ganz offenbar für Optometer ‚senden’, , warten’ und ‚main’ stehen.

Ob das wirklich ganz genau der Quelltext ist, weiß ich nicht. Denn einen Quelltext gibt es nicht. Ein paar Screenshots blieben mir, aus denen ich den eventuellen Quelltext restauriert habe. Das Fahrzeug ist inzwischen abgedampft, und einen Disassembler suchen und benutzen mag ich sowieso nicht.

# Optometer P9710 auslesen

Schade, dass mein Testen der Warteschleife nicht mit abgelichtet ist. Das Warte-Wort hieß im ersten Versuch ‚w' und weil ich drei Versuche brauchte, bis es ungefähr haargenau 10 Sekunden wartete, hieß es am Ende ‚www'. Warten, warten, warten ...




Es blieben sogar ein paar Minuten, um das Programm ein bisschen zu verzuckern. Ein kleines Batchfile namens Start.bat sorgt dafür, dass der Kollege auch in großer Hektik noch rein und raus kommt und sein File parametrieren kann:

```
t4 protocol start.out om endterm
```

‚t4.com' heißt hierbei das gesicherte Terminal-Forth-System mit om-Programm. Das ‚protocol start.out' öffnet das Mitschreiben in Datei und ‚endterm' beendet nach Rückkehr aus dem om-Programm das umschließende Terminalprogramm. Eine berechtigte Frage galt es noch zu klären: Wird bei jedem Start das File überschrieben? Aber das wusste ich noch, dass mein Terminalprogramm brav hinten anhängt, sogar unter Angabe von Zeit und Grund der Wiedereröffnungsschließung:

```
opened(27.01.05---15:52:44)
5.7158E+02
(27.01.05---15:52:45)
5.7122E+02
(27.01.05---15:52:50)
5.7116E+02
(27.01.05---15:52:55)
closed(27.01.05---15:52:59)
```

Zwischen den ‚opened'- und ‚closed'-Zeilen, den Datumsangaben und einem bislang unerforschten Gewirr von Sonderzeichen (die im Ausdruck sicher zu interessanten Effekten führen), sehen wir ganz deutlich Dinge wie ‚5.7122E+02'. Auf diese Dinge hatte der Kollege es abgesehen. Obwohl ich nicht weiß, was so interessant an der Helligkeit in kalten Büros der Schefenacker Vision Systems GmbH oder gar der Dämmerung auf langweiligen schwäbischen Rush-Hour-Straßen sein soll.

**Fazit**

Während ich diese Zeilen schreibe, hat das Programm bereits mehrere hundert Euro gespart, weil jetzt nur ein Testfahrer im Auto sitzt. Der zweite hätte sonst die Helligkeit händisch mitschreiben müssen (oder immer MV einhacken). Wenn ich das vorher gewusst hätte, hätte ich vielleicht eine hinreichende Entschädigung für die Hilfskasse Hartz-IV-geschädigter Testfahrer verlangen sollen.

Dass ein 15 Jahre altes Forth-Programm trotz Windows 2000 immer noch die Ports der seriellen Schnittstelle lesen und schreiben kann, widerspricht zwar der NT-Philosophie und Dokumentation, ist aber sehr praktisch.

Nach langer Zeit kann man sich tatsächlich noch an manche Forth-Worte erinnern, sofern sprechende Namen gewählt wurden.
Und das Terminalprogrämmchen ist erstaunlich praktisch für kleine Schnellschüsse.

*Claus Vogt*
Schefenacker Vision Systems GmbH, 2005

# Windows-Programmierung mit SwiftForth und MySQL

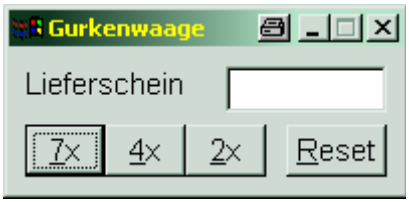
## Teil 3

**Stefan Schmiedl**

<s@xss.de>

### Was bisher geschah ...

In den beiden ersten Teilen habe ich eine Möglichkeit beschrieben, die libmysql.dll unter SwiftForth zu verwenden und eine Batch-Applikation beschrieben, mit der Datenübernahmen aus einem anderen Programm vorgenommen werden. In diesem letzten Teil der Serie ist das Thema eine „richtige“ Windows-Applikation. Es dürfte die kleinste GUI-Applikation sein, die ich bisher geschrieben habe ...




Um die durchgängige Rückverfolgbarkeit der eingelegten Gurke sicherzustellen, werden die Sammelbehälter mit den vorsortierten Gurken bei der Anlieferung mit der Nummer des Lieferscheins und einem Zeitstempel versehen. Diese Daten sollen automatisch auf einem Etikettendrucker ausgegeben werden, wobei die benötigte Anzahl an Etiketten von der angelieferten Gurkenmenge abhängt. Die aktuelle Lieferscheinnummer wird automatisch aus der Protokolldatei der Waagenüberwachung übernommen, kann aber „für Nachzügler“ manuell überschrieben werden.

### Dialoge mit SwiftForth

Dialoge wie der gezeigte können in SwiftForth bequem im Quelltext definiert werden. Zunächst werden numerische Konstanten festgelegt, über die Komponenten im Dialogfenster identifiziert werden. Die Bestandteile in `windowstyle` sorgen für das gezeigte Aussehen des Fensters und die ursprüngliche Platzierung in der Bildschirmmitte.

```
100 enum LIEFERSCHEIN__
    enum LIEFERSCHEIN.BOX
    enum 0.BUTTON
    enum 1.BUTTON
    enum 2.BUTTON
    enum RESET.BUTTON
    enum CHECK.TIMER
drop

WS_POPUP WS_SYSMENU or WS_CAPTION or
  WS_BORDER or DS_CENTER or
  WS_MINIMIZEBOX or
constant WINDOWSTYLE
```

```
dialog GURKENWAAGE-DIALOG
[modeless " Gurkenwaage" 0 0 98 34
  (style windowstyle)
  (font 11, Arial) ]
 [ltext " Lieferschein"
  lieferschein__     4   4  50  12  ]
 [edittext
  lieferschein.box 54   4  40  11
  (-style WS_TABSTOP) ]
 [pushbutton " &7x"
  2.button           4  18  20  12  ]
 [pushbutton " &4x"
  1.button          24  18  20  12  ]
 [pushbutton " &2x"
  0.button          44  18  20  12  ]
 [pushbutton " &Reset"
  reset.button      69  18  25  12  ]
end-dialog
```

Dialog können „modal“ oder „modeless“ sein. Die erste Kategorie wartet das Schließen des Dialogs ab, bevor der normale Ablauf fortgesetzt wird, letztere lässt den Dialog „nebenher“ mitlaufen. Die Koordinatenangaben werden in „Dialog units“ vorgenommen, die von der Größe der eingesetzten Schrift abhängen. Der Dialog-Compiler erlaubt es auch, einzelne Elemente individuell zu gestalten, wie am Beispiel der `liefer-schein.box` deutlich wird: Sie kann nicht mit der Tab-Taste angesprungen werden. Im SwiftForth-Handbuch gibt es ein komplettes Kapitel zu diesem Thema.

Neben dieser Definition gibt es noch ein bisschen mehr Verwaltungkram. Da hätten wir die Nummer des aktuellen Lieferscheins, die Zahl der Etiketten, die jeweils gedruckt werden soll, sowie den Filedescriptor und die Größe der von der Waagenüberwachung geschriebenen Protokolldatei.

```
0 value LS
create (COPIES)  2 , 4 , 7 ,
0 value TARGET
2variable TARGET-SIZE

throw#
  s" Standarddrucker nicht gefunden!"
    >throw enum IOR-NOPRINTER
  s" Fehler beim Drucken!"
    >throw enum IOR-NOPRINTING
  s" Problem beim Dateizugriff"
    >throw enum IOR-FILE
to throw#
```

Neben einigen Fehlermeldungen verwende ich noch „digitales Schmierpapier“:

```
0 value #scratch
0 value (scratch)

: >$      ( n1 n2 : buffer -- n3 )
  create
  swap cell+ cell+ dup , over , +
  does>  ( -- addr n )
  2@ (scratch) + swap ;

: INIT-SCRATCH  ( -- )
  #scratch allocate to (scratch) ;
```

```
: FREE-SCRATCH  ( -- )
  (scratch) free  0 to (scratch) ;

0 20  >$ $DATE
  20  >$ $TIME
  16  >$ $LOCALTIME
  20  >$ $DOCINFO
  10  >$ $LS
to #SCRATCH
```

### Ausdrucksstark

Beim Drucken unter Windows werden Dokumente in eine Warteschlange gestellt, die jeweils ein oder mehrere zu drukkende Seiten enthalten. Jede Seite führt dabei die Anweisungen mit sich, die dann vom geräteabhängigen Treiber realisiert werden. Die im Folgenden beschriebenen Worte kapseln die Windows-API-spezifischen Details, so dass eine ziemlich übersichtliche Beschreibung des zu druckenden Inhalts möglich wird.

```
package PRINTING

0 value PDC
variable OLDFONT  variable OLDPEN

: TWIP-MAP  ( -- )
  pdc MM_TWIPS SetMapMode drop ;

: (DOCINFO)  ( z -- addr )
  >r $docinfo over ~!+  r> !+
  0 !+  0 !+  0 !+  drop ;
```

Windows erlaubt es, „bequeme“ Koordinaten zu verwenden und rechnet diese je nach MapMode in interne Koordinaten um. Der hier gewählte Modus `MM_TWIPS` arbeitet mit Zwanzigstel-Punkten als Basiseinheit. In der `DocInfo`-Struktur wird unter anderem der Name des Druckauftrags abgelegt, der in der Druckerwarteschlange zu sehen ist.

```
public

: PRINT-ERROR?  ( n -- )
  0 <= ior-noprinting ?throw ;

defer PRINT-DC
defer PRINT-GDIUP
defer PRINT-GDIDOWN
```

Die Worte print-gdiup und print-gdidown werden in der Applikation entsprechend den Anforderungen definiert.

```
private

: DOD  ( ... n -- )
  0 ?do  DeleteObject drop  loop ;

: [PRINTER  ( -- )
  print-dc to pdc  print-gdiup ;
: PRINTER]  ( -- )
  pdc DeleteDC drop
  print-gdidown dod ;

: [DOC  ( z --  )
  pdc swap (docinfo) StartDoc
  print-error? ;


: DOC]  ( -- )
  pdc EndDoc print-error? ;
```

Beim Arbeiten mit GDI-Objekten fährt man am sichersten, wenn man den „alten“ Zustand nach dem Arbeiten mit den eigenen Einstellungen wieder herstellt. Dabei werden alle GDI-Objekte einheitlich behandelt, wobei hier nur Schriften und Zeichenstifte von Interesse sind.

```
: [GDI  ( obj old -- )
  pdc rot SelectObject swap ! ;
: ]GDI[ ( obj -- )
  pdc swap SelectObject drop ;
: GDI]  ( old -- )
  @ ]gdi[ ;
```

Das Schwierige an `CreateFont` war, sich bei den zwölf nicht notwendigen Argumenten nicht zu verzählen, die zwischen der Schriftgröße und dem Schriftnamen kommen.

```
public

: CREATE-FONT  ( z n -- font )
  swap >r 20 *
  0 0 0 0  0 0 0 0  0 0 0 0
  r> CreateFont ;
```

Die folgenden Worte klammern (logische) Blöcke beim Ausdruck:

```
: [JOB   ( z -- )      [printer [doc ;
: JOB]   ( -- )        doc] printer] ;

: [PAGE   ( -- )
  pdc StartPage print-error?
  twip-map ;
: PAGE]   ( -- )
  pdc EndPage print-error? ;
: ]PAGE[  ( -- )
  page] [page ;

: [FONT  ( font -- )  oldfont [gdi ;
: FONT]  ( -- )       oldfont gdi] ;
: ]FONT[ ( font -- )  ]gdi[ ;

: [PEN   ( pen -- )   oldpen [gdi ;
: PEN]   ( -- )       oldpen gdi] ;
: ]PEN[  ( pen -- )   ]gdi[ ;
```

Nun noch ein paar Worte zum Ausrichten von Text und Ziehen von Linien.

```
private

: ...   ( x y z ta -- )
  TA_BASELINE or pdc swap
  SetTextAlign drop
  >r pdc -rot -20 * swap 20 * swap r>
  zcount TextOut drop ;
```

```
public

: <...    ( x y z -- ) TA_LEFT ... ;
: <..>    ( x y z -- ) TA_CENTER ... ;
: ...>    ( x y z -- ) TA_RIGHT ... ;

: __    ( x y w -- )  locals| w y x |
  pdc x 20 * y -20 * 0 MoveToEx drop
  pdc x w + 20 * y -20 * LineTo drop ;

end-package
```

### Etikettendruck

Mit diesem Package in der Hinterhand wird der Druck der Etiketten schön einfach. Doch zunächst brauche ich noch den darzustellenden Inhalt:

```
: TIME?  ( -- addr )
  $localtime drop dup
  GetLocalTime drop ;

: DATE$ ( z -- addr )
  >r LOCALE_SYSTEM_DEFAULT 0 time?
  r> $date over >r GetDateFormat
  drop r> ;

: TIME$ ( z -- addr )
  >r LOCALE_SYSTEM_DEFAULT 0 time?
  r> $time over >r GetTimeFormat
  drop r> ;
```

Die Ausgabe erfolgt auf dem Drucker „Etiketten“ mit den Schriften Arial Größe 20 pt und 36 pt.

```
0 value FONT1
0 value FONT2

:noname ( -- dc )
  z" WINSPOOL" z" Etiketten" 0 0
  CreateDC ;
is PRINT-DC

:noname  ( -- )
  z" Arial" dup
  36 create-font to font1
  20 create-font to font2 ;
is PRINT-GDIUP

:noname  ( -- )  font1 font2 2 ;
is PRINT-GDIDOWN
```

Die Lieferscheinnummer wird groß geschrieben, Datum und Uhrzeit kommen kleiner darunter.

```
: (PRINT-LABEL)  ( z -- z )
  [page
    font1 [font  9 50 third <...
    font2 ]font[
      9 80 z" dd.MM.yyyy" date$ <...
      9 100 z" HH:mm" time$ <...
    font]
  page] ;
```

### Beobachter

Die Waagenüberwachung schreibt die aktuellen Informationen (vor allem die Lieferscheinnummer) in einer Datei mit. Ein Job für `tail -f`, aber unter Windows ... dabei gibt's auch gleich eine Überraschung: Das eingebaute `open-file` von SwiftForth öffnet Dateien immer exklusiv, selbst dann, wenn nur gelesen wird. Wegen mangelnder Faktorisierung muss ich mich also selber mit der API-Funktion `CreateFile` befassen:

```
: OPEN-KOPF  ( -- fid )
  Z" daten\kopf.dat"
  GENERIC_READ
  FILE_SHARE_READ FILE_SHARE_WRITE or
  0 OPEN_EXISTING
  FILE_ATTRIBUTE_NORMAL
    FILE_FLAG_RANDOM_ACCESS or
  0  CreateFile dup
  INVALID_HANDLE_VALUE =
    ior-file ?throw ;

: @SIZE  ( -- )
  target file-size drop
  target-size 2! ;

: INIT-CHECK  ( -- )
  open-kopf to target  @size ;

: SHUTDOWN-CHECK  ( -- )
  target close-file drop
  0 to target ;
```

Richtige Arbeit muss mein Programm nur dann leisten, wenn sich die Größe der Datei geändert hat. In diesem Fall wird die im Dialog stehende Lieferscheinnummer entsprechend oft ausgedruckt. Das leere Etikett am Ende sorgt dafür, dass die Abreißkante des Druckers das richtige Etikett abreißt.

```
: CHANGED?  ( -- b )
  target-size 2@
  @size target-size 2@  d= not ;

: CURRENT-LS  ( -- z )
  ls WM_GETTEXT $ls swap dup>r
  SendMessage drop r> ;

: COPIES  ( i -- n )
  cells (copies) + @ ;

: PRINT-LABELS  ( n -- )
  current-ls dup [job
    swap copies 0 ?do
      dup (print-label)
    loop drop
    [page page]
  job] ;
```

Im Regelfall wird die Nummer allerdings nicht manuell eingegeben, sondern über einen Zeitgeber aus der Datei geholt. Dazu werden die entsprechenden 6 Byte ausgelesen und als Nullterminierter String gespeichert.

```
: LAST-RECORD  ( -- )
  target file-size drop
  274 s>d d- target
  reposition-file drop ;

: >ZSTR  ( a n -- z )
  over + 0 swap c! ;

: GET-ID  ( -- z )
  $ls drop 6 2dup target read-file
  2drop >zstr ;

: LAST-ID  ( -- z )
  last-record get-id ;
```

### Dialogaktionen

Nachdem die aktuelle Lieferscheinnummer jetzt bekannt ist, kann sie auch angezeigt werden. Ebenso wie beim Auslesen oben wird dafür eine Botschaft an das Eingabefeld geschickt.

```
: SHOW-LAST-ID  ( -- )
  ls WM_SETTEXT 0 last-id
  SendMessage drop ;
```

Beim Starten der Applikation wird die Protokolldatei der Waagenüberwachung geöffnet, Schmierpapier bereit gelegt und dafür gesorgt, dass alle drei Sekunden der Zustand der Datei überprüft wird.

```
: INIT-DIALOG  ( hwnd -- )
  init-check init-scratch dup
  lieferschein.box GetDlgItem to ls
  show-last-id
  check.timer 3 sec 0 SetTimer drop ;

: DESTROY-DIALOG  ( hwnd -- )
  free-scratch shutdown-check
  check.timer KillTimer drop ;
```

Ändert sich die Lieferscheinnummer in der Datei, werden automatisch sieben neue Etiketten gedruckt (das hat dem Mann an der Waage besonders gut gefallen!).

```
: PRINT-NEW  ( -- )
  show-last-id  2 print-labels ;

: >FRONT  ( hwnd -- )
  HWND_TOPMOST 0 0 0 0
  SWP_NOMOVE SWP_NOSIZE or
  SWP_SHOWWINDOW or SetWindowPos
  drop ;
```

Das Applikationsfenster schwebt im Vordergrund, selbst dann, wenn es nicht aktiv ist. So ist der Etikettendruck immer nur einen Mausklick entfernt, auch wenn gerade in der Waagenüberwachung etwas eingetragen wird.

```
: BOOT  ( -- )
  hwnd dup >front init-dialog ;

: SHUTDOWN  ( -- )
  hwnd destroy-dialog drop
  0 PostQuitMessage drop ;
```

Diese beiden Worte richten den Dialog ein, bzw. beenden die Applikation. Nun bleibt nur noch, die Aktionen mit dem Dialog zu verdrahten.
Zunächst lege ich die Aktionen der Schaltflächen fest:

```
[switch %GURKENWAAGE drop ( id -- )
  2.button    run: 2 print-labels ;
  1.button    run: 1 print-labels ;
  0.button    run: 0 print-labels ;
  reset.button run: show-last-id ;
switch]
```

Die erste Anlaufstelle für Botschaften stellt allerdings die folgende Liste dar:

```
[switch SWITCHBOARD
  zero ( msg -- res )
  WM_COMMAND    run:
    wparam loword %gurkenwaage true ;
  WM_TIMER      run:
    changed? if hwnd >front print-new
    then true ;
  WM_INITDIALOG run: boot 0 ;
  WM_CLOSE      run: shutdown true ;
switch]
```

Hier werden die von der Applikation empfangenen Botschaften vorsortiert bzw. gleich bearbeitet. Beim Öffnen des Dialogs wird `boot` aufgerufen, beim Schließen des Fensters `shutdown`. Beim Auslösen eines Timer-Ereignisses sorge ich dafür, dass das Fenster sichtbar ist und der erste Satz Etiketten gedruckt wird. Ereignisse, die die Kontrollelemente betreffen, werden in der anderen Liste verarbeitet.
Damit sind nun alle Zutaten für eine Callback-Funktion bereit gestellt, die von Windows aufgerufen wird, um Applikationsereignisse und -botschaften zu behandeln. Das Wort msg enthält dabei im niederwertigen Wort die identifizierende Nummer.

```
:noname ( -- res )
  msg loword switchboard ;
4 cb: GURKENWAAGE-CALLBACK
```

Zusammen mit der Dialogbeschreibung legt sie das Verhalten der Applikation fest:

```
: MAIN-WINDOW  ( dlg cb -- wnd )
  locals| cb dlg |
  hinst dlg 0 cb 0
  CreateDialogIndirectParam ;

: GURKENWAAGE  ( -- wnd )
  gurkenwaage-dialog
  gurkenwaage-callback main-window ;
```

### Verpackung

Um eine selbstständige EXE-Datei zu erhalten, muss noch etwas Verwaltungskram erledigt werden, allerdings etwas mehr als beim letzten Mal, denn jetzt muss noch der „Main Event Loop“ dazu, der dafür sorgt, dass die Anwendung offen bleibt, bis sie vom Benutzer geschlossen wird.

```
: GO ( -- wnd )
  gurkenwaage
  dup dlgactive !
  dup SW_SHOWNORMAL ShowWindow drop ;

:noname  ( -- )
 go locals| wnd |
 begin
  winmsg 0 0 0 GetMessage drop
  wnd winmsg IsDialogMessage not  if
   WINMSG TranslateMessage DROP
   WINMSG DispatchMessage DROP
  then
 msg@ WM_QUIT = until
 winmsg 2 cells + @ ExitProcess ;
'main !

program GW.EXE
```

### Fazit

Der Einstieg nach über zehn-jähriger Forth-Abstinenz war relativ schmerzlos; „blöde“ Fehler sind natürlich passiert, aber auch nicht auffallend mehr oder weniger als mit anderen Sprachen. Die Applikationen werden täglich eingesetzt und laufen schnell und stabil, mittlerweile schon seit fast einem Jahr.
Als es einmal beim Ausdrucken Probleme gab, konnte ich guten Gewissens behaupten, dass das Problem nicht in meinem Code liegt (nichts drin, was *so* schiefgehen könnte), und das Verschwinden des Problems nach Tauschen des Druckerkabels bestätigte dies.
Schließlich war vor kurzem eine nachträgliche Änderung vorzunehmen, die ich ohne Schwierigkeiten einpflegen konnte, d. h. ich konnte meinen Code auch nach knapp acht Monaten immer noch verstehen.

Seit einiger Zeit verfolge ich mit großem Interesse die Entwicklung von RetroForth (http://www.retroforth.org), einem minimalistischen Nicht-ANSI-Forth, mit dem sich *wirklich* kleine Applikationen erzeugen lassen. So habe ich (unter Linux, macht aber nichts) einen Präprozessor für XSL-FO geschrieben, der im ELF-Format mit gewaltigen 9 kB zu Buche schlägt, wovon gut 1 kB Textbausteine sein dürften. Faszinierend.

*Liebe Leser,*

*Haben Sie alle drei Teile dieser Arbeit von Stefan Schmiedl gelesen? Hat Ihnen seine Arbeit gefallen oder sogar helfen können? Oder wäre es für Sie persönlich besser gewesen, wenn alle drei Teile in einer einzigen Ausgabe der VD erschienen wären?*
*Bitte schreiben Sie uns Ihre Meinung.*

*fep*

# MicroCore anwenden

## Report über eine zutiefst befriedigende Instantiierung

**kschleisiek @ send.de**
**www.microcore.org**

Vortrag des Autors auf der euroForth Konferenz,
sowie auf der
Jahrestagung der Forthgesellschaft
2005 in Sachsen.

## Zusammenfassung

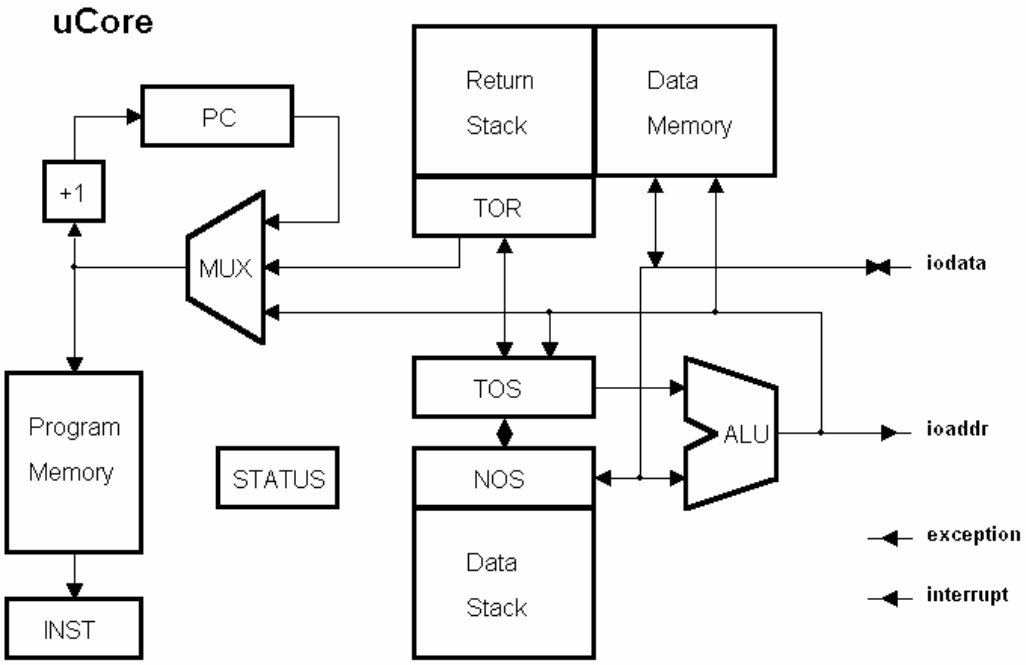
Vor drei Jahren habe ich die "MicroCore"-Architektur vorgestellt, einen Harvard-Prozessor mit zwei Stacks, der Forth zur Richtschnur für den Instruktionssatz nimmt und Architekturinnovationen des Transputers genutzt hat. Damals existierte MicroCore, oder kurz "uCore", nur als simulierter VHDL-Code.

Seitdem haben wir uCore genutzt, um unser neuestes seismisches Vierkanal-Datenaufzeichnungsgerät für den Betrieb auf dem Meeresboden zu realisieren. Dieses Projekt hat über alle Erwartungen die Richtigkeit des Konzepts erwiesen. Mehrfach hat sich herausgestellt, dass die Softwareentwicklung schneller als geplant war - ein in der Industrie eher unübliches Phänomen.

Das liegt hauptsächlich am Exception-Mechanismus von uCore, der ungefähr die Hälfte der Interrupts ersetzt und die Software-Komplexität erheblich reduziert hat. Aber nicht nur das: Einige wenige Spezialinstruktionen haben die Software-Komplexität weiter verkleinert und gleichzeitig die Stromaufnahme reduziert.

Zusammengefasst: Die volle Kontrolle aller Aspekte des Systementwurfs, nämlich Prozessorarchitektur, Betriebssystem und Anwendungsprogrammierung erlaubt es nicht nur, das Hardware/Software-Interface im Hinblick auf Einfachheit, Eleganz und Stromeffizienz zu optimieren, sondern gleichzeitig werden die Kosten für den zukünftigen Einsatz verbesserter Hardware - als "Softwareportierung" seit jeher gefürchtet - drastisch reduziert, weil der Instruktionssatz des "Soft Core"-Prozessors auch bei Änderung der zugrundeliegenden FPGA-Architektur gleich bleibt.

# 1 uCore Architektur




# 2 Entwicklungsbord "uCore100"

Als erster Schritt, uCore in realen Anwendungen zu nutzen, wurde ein (durch die Forth-Gesellschaft gesponsertes) Entwicklungsbord "uCore100" realisiert, das einen XC2S200 FPGA, 512k Programm- und 256k x 32 Daten-RAM (je 10 ns), 2 M x 16 Flashspeicher, ein frei nutzbares USB-Port und viele freie I/O Anschlüsse besitzt, die auf einer doppelreihigen Steckleiste und einem 96-poligen Steckverbinder herausgeführt sind. Mit einem 24 MHz-Oszillator führt uCore jede 80 ns eine Instruktion aus.

uCore100 wurde benutzt, um uCore das erste Mal zum Leben zu erwecken, und um den Debugger zu entwickeln. Zuerst wurde das "Umbilical Interface" des Debuggers an ein Druckerport des PCs (Centronics) angeschlossen, das ein schnelles paralleles Laden des Programmspeichers mit asynchronem Handshake in beiden Richtungen erlaubte, und damit wurde auch die Software-Entwicklungsumgebung (Debugger) auf dem PC entwickelt. Später wurde als Vorbereitung der Migration auf die Hardware des neuen Datenloggers eine serielle UART (COM Schnittstelle) als Umbilical Interface genutzt, mit 32-Bit Daten als Bytesequenz ("Start"-Byte gefolgt von vier Datenbytes), gefolgt von einem Handshake-Byte in der Gegenrichtung als Signal für das Ende jeder einzelnen Transaktion.

Später hat sich dann herausgestellt, dass dieses Konzept sogar in der tatsächlichen Anwendung beibehalten werden kann, indem in zwei verschiedenen Modi gearbeitet wird: Im "Anwendungsmodus" wird die UART wie üblich für KEY und EMIT genutzt. Im "Debugmodus" wird die UART genutzt, um 32-Bit-Werte mit dem Debugger auszutauschen, die bis zu KEY und EMIT durchgereicht werden, wenn die Bits 31...8 auf Null gesetzt sind.

## 2.1 Softwareentwicklungsumgebung

Die Entwicklungsumgebung (der Debugger) läuft unter Linux und ist für einfache Portierung in GCC geschrieben. Ein Forth Cross-Compiler mit uCorespezifischen Befehlen dient als "Makroassembler" und wird auf Gforth oder Win32For geladen und mit einem dazu passenden Disassembler kann der erzeugte Code untersucht werden. Der Cross-Compiler kann zur Simulationsunterstützung VHDL-Code erzeugen, als auch binären Objektcode und ein Symbolfile für den interaktiven Debugger.

Das Umbilical Interface des Debuggers wird durch passende Hardware auf dem Target-System unterstützt, so dass der Programmspeicher mit oder ohne Prozessorreset initialisiert werden kann. Einer der Einzyklus-Userinstruktionen von uCore wird als Breakpoint-Instruktion benutzt, die beim Single-Stepping unter der Kontrolle des Debuggers durch den Code "geschoben" wird. Auf uCore selber beobachtet ein konventioneller Debug-Monitor das Umbilical-Interface und wartet von dort auf eine Adresse, die dann angesprungen wird. Ist uCore mit der Abarbeitung fertig, wird ein "Fertig-Signal" oder ein Fehlercode an den PC zurückgeschickt.

# 3 Geolon-MCS Systemarchitektur

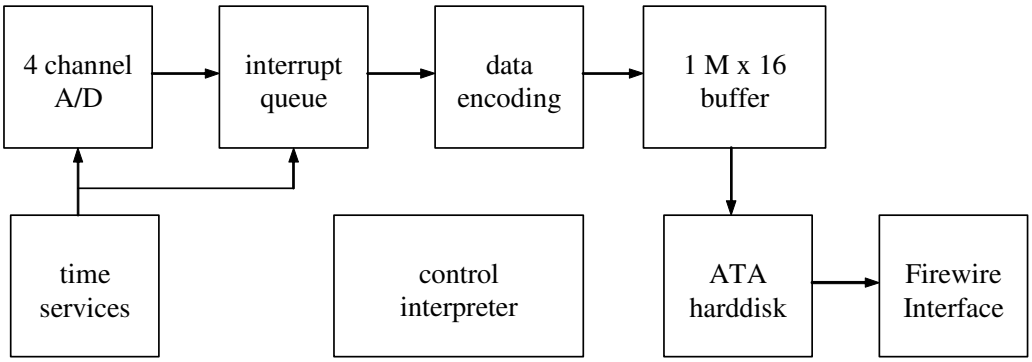
Geolon-MCS ist ein Vierkanal (Hydrophone und 3-Komponenten-Geophone) "Seismocorder", der in autonomen seismischen Registriersystemen auf dem Meeresboden eingesetzt wird. Sein A/D-Wandler erzeugt 24-Bit-Zahlen mit einem Signal-Rauschabstand von 130 dB bei einem Stromverbrauch von 500 mW und die Daten werden auf Festplatten der Autoindustrie gespeichert und über ein Firewire-Interface ausgelesen.

Die Hardware ist unterteilt in

- Energiemanagement zur Erzeugung von +/- 2.7V, 3.3V und 5V aus einem 6-15V Batterieeingang
- Digitale Steuerung mit 256x8 Programm-RAM, 256x16 Daten und Return-Stack RAM, 1Mx16 Datenpuffer RAM, einem RS232 Interface für interaktive Konfigurierung, ein ATA Festplatteninterface und ein Firewire-Interface.
- A/D-Wandler mit differentiellen Vorverstärkern für das Hydrophone und die Geophone, Sigma-Delta Modulatoren, 24-Bit digitalen Filtern, Referenzspannungserzeugung und analoge Stromversorgungsaufbereitung.

- Hochpräziser mikroprozessorgesteuerter Quarzoszillator, der sowohl einen 1-pps Zeitimpuls als auch ein 12,288 MHz Taktsignal erzeugt.

Ein unerwarteter, aber eigentlich offensichtlicher Vorteil der Nutzung eines Soft-Cores im FPGA: Es gibt keinen Prozessorchip, der Platz auf dem PC-Bord verbraucht oder verdrahtet werden muss und deshalb hat die gesamte Digitalelektronik auf ein einziges Bord statt wie bisher auf zwei Bords gepasst.




Datenflussarchitektur des Geolon-MCS

## 3.1 uCore Implementation

Bevor uCore auf die Hardware des Geolon-MCS portiert wurde, wurde eine Prototyp auf dem uCore100-Prototypingboard erstellt, d.h. es wurden nur 16 Bits des Datenspeichers genutzt und das Umbilical Interface wurde neu implementiert auf der Basis einer seriellen RS232-Schnittstelle statt des parallelen Centronics-Ports. Darüber hinaus wurden beide Stacks achtfach nebeneinander angelegt und unser bewährter kooperativer Multitasker mit sowohl priorisierter als auch gleichberechtigter Zuteilung wurde auf "uCore-Forth" portiert. Glücklicherweise meldete sich bereits der erste Implementationsversuch von uCore auf der neuen Hardware beim Debugger.

Eigenschaften der uCore Instantiierung auf Geolon-MCS:

| | |
|---|---|
| Datenpfadbreite | 32 Bits |
| Programmspeicher | 256k externes RAM, 2k interner "Cache" |
| Datenspeicher | 128k externes 55ns RAM, 1k interner "Cache". Wird für Variablen und den Returnstack genutzt. Das externe RAM ist physikalisch nur 16 Bit breit und deshalb verbraucht jedes Fetch und Store zwei Zyklen. |
| Datenpuffer | 2MB externes 55 ns RAM 16 Bit breit, so dass DMA-Zugriff mit der 16 Bit IDE-Schnittstelle möglich ist. |
| Taktfrequenz | 12.288 MHz = 160 ns Instruktionszyklus |
| RTOS | Kooperativer Multitasker mit 8 physikalischen Daten- und Returnstack-bereichen und einem Taskwechsel in 12 usec. |
| FPGA | XC2S200E "Spartan 2" Technologie, 35% davon werden von uCore verbraucht. |

## 3.2 Booting

Ein weiterer Vorteil der Softcore-Herangehensweise ist die Freiheit, jede Bootstrategie realisieren zu können. Auf Grund der Hardwaremöglichkeiten des MCS wurde folgender Bootprozess realisiert:

Sobald die Konfiguration des FPGA durch den externen seriellen Speicher beendet ist, ist ein internes Flip-Flop "boot" auf den Wert '1' vorbesetzt. (Unglücklicherweise scheint eine solche Vorbesetzung von verschiedenen Synthesizern unterschiedlich gehandhabt zu werden.)
Solange "boot" auf '1' gesetzt ist, wird ein winziges Bootprogramm in den Programmspeicherbereich eingeblendet, das die Harddisk einschaltet und eine feste Anzahl von Sektoren von der Platte in den "normalen" Programmspeicher kopiert. Wenn dies erledigt ist, wird ein Sprung zur Adresse Null ausgeführt (dies ist die Adresse des Reset-Vektors), wodurch das "boot" Flip-Flop auf '0' gesetzt wird und die Ausführung des Programms beginnt, das von der Platte gelesen wurde. Das Boot-Flip-Flop wird durch einen Reset des Prozessors nicht mehr verändert: Es bleibt auf '0', wenn es einmal durch den Sprung auf die Adresse Null zurückgesetzt wurde. Dieses Bootprogramm ist 120 Bytes lang und es wird bei der Synthese im FPGA fest "verdrahtet".

## 3.3 FPGA Technologieprobleme

Die Auswahl der FPGA-Technologie war nicht einfach. Die Nutzung von Xilinx FPGAs war vorgegeben, da wir die verschiedenen Softwaretools dafür kannten (im Rückblick jedoch erscheint dies als unglückliche Einschränkung auf der Suche nach geeigneteren FPGAs). Zunächst schien die neue Spartan-3-Architektur die selbstverständliche Wahl zu sein, da diese

Multiplikationszellen enthält, die es uns erlaubt hätten, eine 32 x 32 -> 64 Bit Multiplikation als 1-Zyklus-Instruktion zu realisieren. ABER: Spartan 3 wird in einem 90 nm Prozess gefertigt, der zu einem dramatischen Anstieg des statischen Stromverbrauchs im Vergleich mit älteren Technologien führt, so dass das FPGA noch vor dem Anlegen eines Taktsignals schon 200 mW verbraucht hätte. Deshalb haben wir uns letztlich für das FPGA XC2S200E entschieden, mit dem die gesamte Steuerung des MCS mit 40 mW Energieeinsatz erledigt wird.

# 4 Exceptions / Interrupts

Das interessanteste Konzept, das uCore vom Transputer übernommen hat, ist der "Exception Mechanismus" und davon haben wir vorteilhaftesten Gebrauch gemacht. Zuerst möchte ich versuchen, klar zu machen, worum es sich dabei überhaupt handelt, indem ich es mit dem altbekannten Interrupt vergleiche:

| | |
|---|---|
| Interrupt: | Es hat ein Ereignis stattgefunden, das NICHT von der Software erwartet wurde. |
| Exception: | Es hat ein Ereignis NICHT stattgefunden, das von der Software erwartet wurde. |

Und so benutzt die Debugger-Schnittstelle den Exception-Mechanismus: Es gibt ein DEBUG_REGISTER im FPGA, das als Ein- und Ausgang der Umbilical-Schnittstelle zum Host verwendet wird. Wenn uCore sonst nichts zu tun hat, dann macht es DEBUG_REGISTER @. Und alles ist gut, wenn von der Umbilical-Schnittstelle ein Wert in diesem Register abgelegt wurde.

Aber was passiert, wenn dort (noch) nichts hinterlassen wurde, das geholt werden kann? Oder, um den obigen Satz zu wiederholen, wenn das "Umbilical-Ereignis" NICHT stattgefunden hat, das von dem "@" vorausgesetzt wird?

Die Adressdekodierelektronik des DEBUG_REGISTER (Memory-Mapped) im FPGA "weiß", ob ein neues Element im Register ist, oder nicht. Wenn nicht, dann wird der Exception-Eingang von uCore während der Ausführung der @-Instruktion aktiviert. Dadurch bleiben alle Register in uCore in ihrem vorherigen Zustand erhalten mit Ausnahme des Instruktions-Registers (IR), dem die "Exception Aufruf Instruktion" zugeführt wird, wodurch dann im folgenden Zyklus ein Unterprogrammaufruf der Exception-Behandlungsroutine erfolgt.

Dadurch verhält sich der Prozessor so, als ob das "@" niemals stattgefunden hätte. Wenn die Ausführung der Exception-Behandlungsroutine beginnt, dann liegt wegen des stattgefundenen Unterprogrammaufrufs auf dem Return-Stack eine Programmadresse eine Instruktion hinter der Instruktion, die die Exception ausgelöst hat. Deshalb ist die einfachste Form der Exception-Behandlungsroutine die Phrase "`R> 1- >R EXIT`", so dass versucht wird, die @-Instruktion, die die Exception ausgelöst hatte, in einer sehr kurzen Schleife wieder auszuführen. Deshalb wird der Prozessor an dieser Stelle blockiert, bis endlich durch die Umbilical-Schnittstelle ein neuer Wert im DEBUG_REGISTER verfügbar wird. Keine Abfrage von Status-Flags und all den IFs und WHILEs, an die wir uns alle gewöhnt haben bei der Programmierung von Echtzeitsystemen, die den Code verschandeln und ihn nur zu oft obskur und schwierig zu warten machen.

Es wird inzwischen nicht mehr überraschen, dass in einem Multitasking-System die Exception-Behandlungsroutine unter Anderem aus einen Aufruf von PAUSE eines kooperativen Multitaskers besteht, so dass die Task, die einen frischen Wert im DEBUG_REGISTER erwartet, erst einmal schlafen gelegt wird und zeitweise etwas anderes erledigt wird.

Die Nutzung des Exception-Mechanismus für die Umbilical-Schnittstelle war seine erste Anwendung und als das System auch nach mehreren Stunden immer noch weiterlief, obwohl auch noch alle 10 Millisekunden ein Timer-Interrupt stattfand, entstand Zutrauen, dass wir es "richtig" hinbekommen hatten.

## 4.1 Zeitdienste

Üblicherweise wird ein periodischer Interrupt (z.B. jede 10 Millisekunden) genutzt um die Variable TIMER hochzuzählen, die als Fundament für Zeitdienste genutzt wird, um z.B. die folgenden Worte zu definieren:

```
: ahead    ( ticks -- time.ahead )
  Timer @ + ;

: timeout? ( ticks -- ticks f )
  dup Timer @ - 0< ;

: continue ( ticks -- )
  BEGIN  pause timeout? UNTIL drop ;

: sleep    ( ticks -- )
  ahead continue ;
```

Im MCS haben wir das mit Hilfe des Exception-Mechanismus noch eleganter hinbekommen und zudem auch noch eine Interruptquelle eliminiert: Das Register TIMER wird alle 30 Mikrosekunden hochgezählt (weil: Wieso nicht? PAUSE selber braucht auch nur 12 usec!).

`TIMER @` funktioniert wie gewohnt: Es legt den aktuellen Inhalt von TIMER auf den Stack. Aber `TIMER !` funktioniert ziemlich anders: Wenn der "time.ahead" Wert auf dem Stack (im NOS) größer als der aktuelle Inhalt von `TIMER` ist, dann wird eine Exception ausgelöst, wodurch die Programmausführung nicht über die versuchte "!"-Instruktion hinaus fortgeführt wird. Auf Grundlage dieses Hardware-Mechanismus lässt sich `CONTINUE` sehr viel einfacher realisieren als vorher und es ist sogar sinnvoll, es als Makro zu definieren:

```
: continue ( ticks -- )     Timer ! ;
```

## 4.2 Semaphore

Es hat nicht lange gedauert, bis wir gemerkt hatten, dass der Exception-Mechanismus dazu genutzt werden kann, "richtige" Semaphore in Hardware zu realisieren. Dazu wurde das Register SEMAPHORES angelegt und jedes einzelne Bit dort kann als ein Semaphore benutzt werden, die die Software mit spezifischen Ereignissen synchronisieren.

Wenn man z.B. `#sema_ide Semaphores !` ausführt, dann wird solange eine Exception ausgelöst, wie das #sema_ide Bit von `Semaphores` gesetzt ist. Andernfalls wird der Code danach weiter ausgeführt.

Bis heute gibt es folgende Semaphore im MCS:

**`1 Constant #sema_ide`**
wird gesetzt, wenn ein Kommando im HDD Kommandoregister gespeichert wird, und es wird rückgesetzt, wenn das HDD-Interface an seiner Interrupt-Leitung die Beendigung eines ATA-Kommandos anzeigt. Dadurch wird ein weiterer Interrupt ersetzt.

**`2 Constant #sema_adc`**
wird gesetzt, wenn ein Kommando an das SPI-Interface des A/D-Konverters abgesetzt wird, oder wenn man durch diesen Semaphor "hindurchgelaufen" ist mit der Phrase `#sema_adc Semphores !`. Es wird rückgesetzt, wenn der A/D-Konverter bereit ist, das nächste Kommando zu empfangen. Dies wird durch automatisches Pollen des A/D-Konverters durch das FPGA realisiert.

**`4 Constant #sema_reset`**
ist gesetzt, solange das externe Reset aktiv ist und es wird rückgesetzt, wenn die externe Resetleitung wieder inaktiv geworden ist. Dadurch können alle Peripherieeinheiten (A/D-Konverter, Firewire-Schnittstelle usw.) programmgesteuert zurückgesetzt werden und der Prozessor wartet auf diesen Semaphor und beginnt dann mit der Neuinitialisierung. Dieser Semaphor wurde als Fehlerbehebung für ein fehlerhaft funktionierendes Peripheriechip eingebaut.

**`8 Constant #sema_buf`**
wird gesetzt, wenn der Datenpuffer voll ist und wird dadurch wieder rückgesetzt, dass der Datenpufferpointer an den Anfang des Pufferbereichs zurückgesetzt wird. Mehr dazu unter "Datenkodierung und -speicherung".

## 4.3 Ereignisse mit Zeitüberwachung

```
7 POP USR Opcode: event
   ( ticks semaphor.bit -- ticks )
```

Das ist die Definition der "User"-Instruktion EVENT im Crosscompiler (ich bin aber noch auf der Suche nach einem besseren Namen!). EVENT kombiniert beide oben beschriebenen Mechanismen: Zeitdienste und Semaphore. Es wartet auf den Semaphor `semaphor.bit` aber nur solange, bis `ticks` vergangen sind. Das heißt, dass eine Exception ausgelöst wird, solange beide Bedingungen noch nicht eingetreten sind: `semaphor.bit` in `Semaphores` ist gesetzt und the Zeit `ticks` ist noch nicht erreicht.

Wenn `semaphor.bit` in `Semaphores` nicht länger gesetzt ist, wird die Programmausführung hinter EVENT fortgesetzt und das Carry-Bit zurückgesetzt. Wenn dagegen die `ticks` erreicht sind, wird mit gesetztem Carry-Bit hinter EVENT fortgefahren. Dadurch kann mit einem nachfolgenden branch on carry (einer uCore Instruktion) zwischen diesen beiden Fällen unterschieden werden. EVENT wird folgendermaßen benutzt:

```
50 ms ahead #sema_adc event drop  carry IF
adc_error THEN  ...
```

Neuigkeiten vom Mikrocontrollerverleih

Neu im Mikrocontrollerverleih sind 6 MicroCore-Boards, hergestellt von Send GmbH/Klaus Schleisiek und finanziert von der Forth-Gesellschaft. Das MicroCore Kit umfasst das MicroCore Board mit XILINX Spartan FPGA (MicroCore ist vorkonfiguriert), Netzteil, USB Slave- und Master-Anschlüsse, Parallelkabel zum Anschluss an einen PC, CD-ROM mit Softwareentwicklungskit inkl. Debugger (Linux) und einem Handbuch.

Mit diesem Board kann die interessante MicroCore (Havard CPU mit Forth als Maschinensprache, siehe http://www.microcore.org) getestet werden. In der Zukunft sind noch weitere FPGA-Kerne geplant (z.B. Bernd Paysans b16 CPU), die per Software in das Board eingespielt werden können.

Auf der CD befindet sich ein Video, welches den Aufbau und Anschluss an einen PC erklärt, so das schnelle Ergebnisse möglich sind. Die kommende Ausgabe der Forth Knoppix Linux CD-ROM wird die MicroCore-Entwicklungssoftware vorkonfiguriert mitbringen, so das die Benutzung des Boards nach dem Plug-and-Play-Prinzips möglich ist.

**Carsten Strotmann**
mailto:mikrocontrollerverleih@forth-ev.de
mailto:mcv@forth-ev.de

## Forth-Tagung 2006

(zum Vormerken)

Die Forth-Tagung 2006 wird im Ruhrgebiet sein.

Traditionell wird auf der Forth-Tagung jemand 'ausgeguckt' der den Tagungsort für das kommende Jahr finden soll. Gewünscht war etwas in der Mitte Deutschlands. Nun, ich hatte es übernommen, mich umzuschauen. Und ich fand etwas weiter westlich, bei mir in der Nähe, ein passendes Tagungshotel.

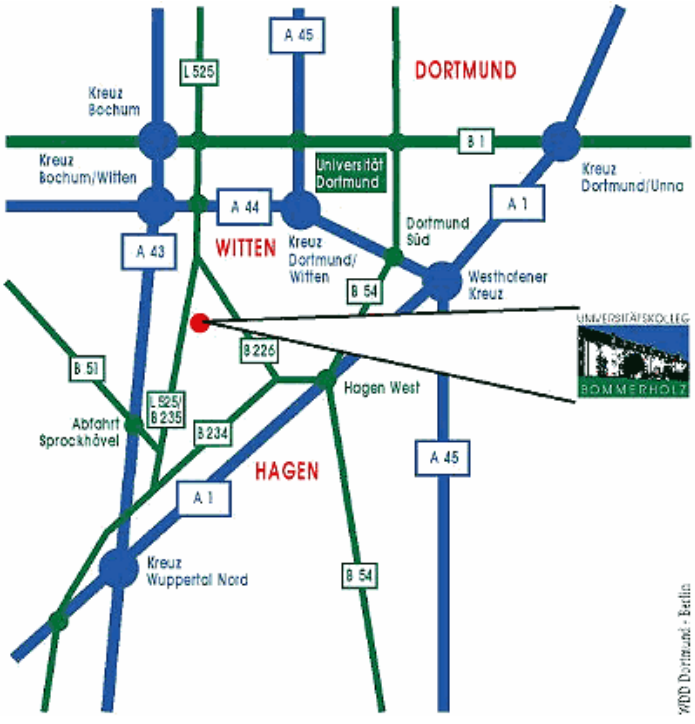
Am südlichen Rand des Ruhrgebietes, da, wo es ins bergische Land hinauf geht, liegt Witten. Dort, im Ortsteil Bommerholz, liegt versteckt im Wald das Universitätskolleg Bommerholz. Es dient als Lehr- und Weiterbildungstätte der Universität Dortmund und steht auch anderen für Tagungen, Kolloquien, Seminare und kleine Kongresse offen. Mit seinen Gästezimmern auf Hotelniveau eignet es sich bestens auch für mehrtägige Veranstaltungen. Und die Ausstattung ist für unsere Zwecke optimal: WLAN, Beamer... Abends eine Kneipe im Haus - alles da. Ich freue mich auf das Treffen 2006.

Termin: Wochenende vom 12. - 14. Mai 2006
Anreise ab Donnerstag möglich.

Tagungsanschrift:

UNIVERSITÄTSKOLLEG BOMMERHOLZ
Bommerholzer Str. 60
58456 Witten-Bommerholz

Tel.: (0 23 02) 39 60
Fax.: (0 23 02) 39 63 20
Organisation: TFG2006@onlinehome.de
(Michael Kalus)




# Gehaltvolles

zusammengestellt und übertragen
von Fred Behringer

### VIJGEBLAADJE der HCC Forth-gebruikersgroep, Niederlande

### Nr. 50, Juni 2005

Das 50. Heft der "Notausgabe" des vor 8 Jahren mangels Beteiligung eingegangenen Vijgeblads, der mit der Vierten Dimension vergleichbaren Großaufmachung. Aus der Notausgabe wurde eine recht lebendige und lesenswerte Erscheinung. Wir, die deutsche Forth-Gesellschaft, gratulieren unseren Forth-Nachbarn für ihren Mut, die immer schwerer werdenden Zeiten der Aldi-Discount-Supermaschinen durchzustehen! Acht Forthler der ersten Stunde wurden gebeten, etwas darüber zu schreiben, wie sie zu Forth gekommen sind. Sehr interessant, muss ich wirklich sagen! Wir sollten uns überlegen, ob wir nicht einmal gelegentlich etwas Vergleichbares auf die Beine stellen können! Können könnten wir das bestimmt, aber wollen müsste man das eben sollen.

#### 1 Het begin
#### Dick Willemse

Seit 1986 dabei. Anfangs Sinclair QL mit zunächst QBASIC. Dann "een rare taal", eine selten-seltsame Sprache, genannt Forth - viel schneller als QBASIC...

#### 2 Zo begon het
#### Cos Haak

1982 las ich einen Artikel über Forth. Diese befremdliche Sprache hatte einen Stack. Das kannte ich von den HP-Rechenmaschinen her. Unter dem Eindruck von Forth schuf ich mir auf der TI 5 einen Stack und begann, in RPN zu programmieren. Seit Vijgeblad 2 bin ich Mitglied. 1985 kaufte ich mir einen ZX Spectrum. 1986 kam der PC. CHForth (ANS) schuf ich später.

#### 3 Forth in een drukke tijd
#### Albert van der Horst

Meine erste Bekanntschaft mit Forth schloss ich auf der Universität. Forth, eine Sprache auf der PDP-11, die kompakter war als Assembler und obendrein noch schneller, da die Unterprogramme "gefädelt" waren und somit keine Zeit für Aufruf und Rückkehr verbrauchten. Ich kaufte einen Nascom. Mit Kassetten für den Philips Datenrecorder mit 2400 Baud. Das Forth auf dem Nascom war ein FIG-Forth-System, das in klassischer Art auf einer anderen Maschine assembliert worden war.

### 4 Mijn Forth verhaal
### Paul Wiegmans

Meine Forth-Geschichte begann 1988 mit einem VC-20 und dann dem C64. Aber eigentlich erst 1994, als ich mich auf die Entwicklung von Micro-Controller-Systemen verlegte. Im Jahre 2000 gewann ich den zweiten Preis in einem von der Zeitschrift Elektor ausgeschriebenen Roboterwettbewerb. Wenn es um Micro-Controller geht, ist für mich Forth die erste Wahl.

### 5 Hoe een knutselaar bij Forth kwam
### Willem Ouwerkerk

Gebastelt habe ich schon immer gern und Roboter waren von Anfang an meine Leidenschaft. Zuerst mit diskreten Logikbausteinen. Dann kam der VC-20 mit 3 KB RAM und vielen schönen I/O-Möglichkeiten. 32 KB RAM selbst zusammengebastelt. Etwas BASIC gelernt. Im Ausverkauf ein Forth-ROM-Modul für den VC-20 gekauft (FIG-Forth). ... Zugang zur HCC-Forth-gebruikersgroep gefunden. Mein vorlautes Mundwerk schleuderte mich in den Vorstand. Mit Lennart Benschop zusammen einige Jahre das Vijgeblad herausgegeben. Forth für B+, Bamboe, Tingel-Tangel, Transputer-Forth, ByteForth für 8051 und AVR. ...

### 6 Mijn Forthverleden
### Albert Nijhof

Meine ersten Programmiererfahrungen habe ich mit einem programmierbaren Taschenrechner von Texas Instruments gemacht. Dann kam ein Pearcom (6502), ein holländischer Apple-II-Klon. In Apple-BASIC konnte man Maschinencode über REM-Anweisungen einbinden. Keine Assembler-Mnemonics. 1980 erschien die berühmte BYTE-Ausgabe, die ganz der Sprache Forth gewidmet war. Ich besuchte den holländischen Forth-Club. Keiner arbeitete mit Apple. Inzwischen gab es Floppys. Die Geschichte wiederholte sich: Es kam für mich ein ATARI ST (68000) mit inzwischen meinem eigenen Forth. Und wieder beim Club (1990). Keiner arbeitetet mit ATARI. Alles nur DOS und Windows. Ahnung hatte ich wenig. Und so bin ich in den Vorstand gewählt worden. Seit 2001 arbeite auch ich mit Windows. Mein letzter Forth-Beitrag war MaisForth (6809).

### 7 Forth vanaf het (bijna) begin
### Ron Minke

Es begann vor 25 Jahren. Mit der Septemberausgabe des "Forth"-BYTE-Heftes, das ich an meiner Arbeitsstelle zu Gesicht bekam. Dann ein Micro-Controller-Bausatz (6502), das dem KIM1, dem AIM65 oder dem Elektuur Junior ähnlich war. Ein ganzes Kilobyte RAM, LED-Display und Hex-Eingabe. FIG-Forth-Listing besorgt und eingetippt. Ein EPROM nach dem anderen (2 K) gebrannt. Dann kam mitten der Achtziger der 8051 und an meiner Arbeitsstelle ein Assembler-Entwicklungssystem. Viel zu umständlich. Sollte das nicht schneller gehen? Ja, es ging. Mein 6502-Forth an den 8051 angepasst. Nur den Kern. Der Rest war ja High-Level. Indirekte Fädelung ersetzt durch Unterprogramm-Fädelung. Das ging schneller. Dann kam ANS-Forth. 1997 der AVR von Atmel. Mein 8051-Forth nach AVR umgeschrieben. Das und meine Familie beschäftigt mich heute noch. Forth lebt!

### 8 Electronica en Forth
### Ben Koehorst

Ich war Elektronik-Amateur. Im Radio-Bulletin las ich etwas über Mikroprozessoren. Der Gedanke der Programmierbarkeit faszinierte mich. Ich wurde Mitglied der taufrisch-jungen HCC, folgte dem TELEAC-Mikroprozessor-Kurs (8080-Assembler), kaufte einen Heathkit-Computer-Bausatz und begann ... Mit BASIC war alles viel zu umständlich, mit Assembler zu mühsam. Ich wollte mehr Macht über meine Maschine haben! Dann kam 1980 das BYTE-Heft über Forth. Das war's, mein Enthusiasmus war geweckt. Forth konnte man nach Belieben als höhere oder auch als niedere Sprache einsetzen. Forth-Bücher gab es kaum. Ich begann zu experimentieren. Resultat: Mein eigenes Forth, BCK-Forth. Dann kam, mit Pieter Surie zusammen, die Entwicklung eines neuen Forth-Systems in MASM für den 8080. Ende der Neunziger passte ich mich schließlich den ANS-Bemühungen an. Mein Sohn lacht über mich und programmiert einen Flugsimulator in 3 Monaten - in Java. Die simple Ansteuerung eines einfachen Micro-Controllers kriegt er aber ohne den letzten Schrei der allerschnellsten und allerteuersten Supermaschine nicht hin.

### Niet slim

Een Forthprogrammeur te Heeze
placht het Vijgeblaadje slecht te lezen.
Zijn programma's waren pover ...
Zijn stack liep steeds over ...
Lezer, laat dit een les voor u wezen!

**Albert Nijhof**

### Nicht besonders schlau

Ein Forthprogrammierer zu Heeze(n)
fand Zeit nicht,
das Vijgeblaadje zu lesen.
Sein Programm war nie doll:
Der Stack war stets voll ...
Dem Leser
sei dies eine Warnung gewesen!

**Übertragungsversuch des Rezensenten**

# Projekt: VolksForth Wiederbelebung

Ein Aufruf von

**Carsten Strotmann**

VolksForth, das Forth-System der Forth-Gesellschaft, hält mindestens seit Mitte der 90 Jahre einen Dornröschenschlaf. Die letzten Änderungen, welche in der VD abgedruckt wurden, stammen aus den frühen 90er Jahren. Danach sind viele Forth-Benutzer umgestiegen auf neue Rechnergenerationen (Windows, Linux/Unix, Apple Mac, OS/2) und das VolksForth für 8086 Rechner (MS-DOS), 8080/Z80 CP/M, Atari ST und C=64 war nicht mehr interessant. Zu Unrecht, wie ich meine.

Auch heute gibt es Leute, die sich mit Rechnern beschäftigen, für die VolksForth ideal ist. VolksForth beinhaltet für die Zeit, in der es entstanden ist und weiterentwickelt wurde, viele interessante und innovative Konzepte, welche sich nicht in anderen Forth-Systemen aus dieser Zeit finden.
Es ist heute sehr schwer für Neu-Einsteiger und Personen, welche die Forth Gesellschaft nicht kennen, Informationen über VolksForth zu bekommen. Im Internet, dort, wo man sonst alles finden kann, gibt es nur wenig Informationen (Das UltraForth auf den Seiten von Claus Vogt, VolksForth als ARC-Archiv auf FTP-Servern und Verweise auf VolksForth in der Dokumentation von BigForth. Ein Totschweigen, das VolksForth so nicht verdient.

*Hinweise: Auf* ***http://www.stcarchiv.de/stc1987/03_forth.php*** *ist zumindest der Anfang eines Einführungskurses von Peter Monadjemi zu lesen (ST-Computer, die VD für den ATARI ;-). Auf* ***http://www.complang.tuwien.ac.at/forth/old-faq/more-books.html*** *findet sich ein Hinweis auf MacCabes Forthbuch mit dem Hinweis, daß dies auch für Volksforth geeignet ist und in der Übersetzung von Monadjemi verfügbar ist. Zudem lassen sich dort Buchtitel zu Forth finden, die ich gar nicht kenne.*

*fep*

Im Gegenteil, es gibt auch heute noch Bedarf an einem Forth für kleine Systeme, welches im Quellcode vorliegt und gut dokumentiert ist. Auf der Forth-Tagung dieses Jahres, in Krögis, habe ich die Idee einer VolksForth Wiederbelebung vorgestellt. Und diese ist auf ein positives Echo gestoßen. Diese Wiederbelebung wird in vier Schritten durchgeführt werden:

Schritt eins: Sichtung und Bewahrung. In diesem Abschnitt befindet sich das Projekt zur Zeit. Es wird alles Verfügbare zum Thema VolksForth gesammelt und in elektronische Form gebracht. Dazu zählt:

* Sichern von (alten) Disketten in Disketten-Abbilder oder in Dateien, welche auf modernen Rechnern gespeichert und verwaltet werden können.
* Zusammentragen der vorhandenen Handbücher und System-Dokumentation .
* Wenn Dokumentation nur in gedruckter Form vorhanden ist, so wird diese gescannt, um sie so elektronisch verwalten und bewahren zu können.
* gescannte Handbücher (aktuellste Version) werden per OCR in elektronische Dokumente überführt, um die Handbücher einfacher anpassen zu können.

Die so gesicherte Dokumentation, Handbücher und die Software wird über die (neue) Forth-Gesellschaft-Webseite frei zur Verfügung gestellt. Hierbei ist zu diskutieren, das VolksForth und auch die Dokumentation unter eine aktuelle OpenSource-Lizenz zu stellen.

In dieser ersten Phase bin ich von Martin Bitter (Sicherung der C=64 und Atari ST Disketten, Bereitstellung des C=64 UltraForth Handbuches), Heinz Schnitter (Informationen über das VolksForth und KK-Forth zum Super8 Mikrocontroller Kit), Ulrich Hoffmann (VolksForth auf CP/M mit yaze-Emulator) und Klaus Kohl (Handbücher und Software aus dem Forth-Versand) tatkräftig unterstützt worden.

Ein Etappenziel für Schritt eins ist der LinuxTag 2005. Auf dem LinuxTag-Stand der Forthgesellschaft wird es wieder, wie schon in den letzten Jahren, die neue Forth-Knoppix-Linux CD geben. Und auf dieser CD sollen alle VolksForth Varianten, ausführbar in Emulatoren, zusammen mit der Dokumentation verfügbar sein.

Schritt zwei: den Meta-Compile Prozess des VolksForth verstehen und wiederbeleben. Ich muß gestehen, daß ich vor diesen Meta-Compilern Respekt habe. Ziel ist es, VolksForth aus den Kernel-Quellen für alle Plattformen über den Meta-Compiler neu erstellen zu können. Sobald dies möglich ist, werden schon veröffentliche Verbesserungen in eine Version (3.9x) einfliessen und das VolksForth Vokabular wird dort, wo es Sinn macht, den aktuellen Forth-Systemen näher gebracht (Klaus Schleisiek wünscht sich eine moderne File-Schnittstelle im VolksForth, andere Verbesserungen wurden schon in der VD beschrieben, sind aber nie in die VolksForth "Distribution" eingeflossen). Meine Planung sieht vor, daß dieser Schritt Ende 2005 abgeschlossen ist.

Schritt drei: Portierung auf neue Plattformen. Es mag verwunderlich scheinen, aber ich habe Anfragen nach neuen Forth-Portierungen auf bisher nicht unterstützte (alte) Plattformen:

* 6502 VolksForth (UltraForth): Apple II, Atari XL, Commodore PET
* 8086 VolksForth: Atari Portfolio (IBM "fastkompatibler" Palmtop mit 40x16 Zeilen Display und 128 KB Speicher)
* Übersetzung der Screens und der Dokumentation ins Englische, damit VolksForth nicht nur auf den deutschen Sprachraum beschränkt bleibt.

*Ich hätte das Volksforth gerne für meinen PDA, ausgestattet mit einem Intel PXA255 (400 MHz) und Windows Mobile 2003 SE als Betriebssystem!* *fep*

Schritt vier: behutsame Weiterentwicklung und Aktualisierung, basierend auf den Rückmeldungen der Benutzer.

Die Geschwindigkeit, in der diese Schritte begonnen und abgeschlossen werden, hängt ganz direkt von der Anzahl der Mitarbeiter zusammen. Die VolksForth Wiederbelebung ist ein Hobby-Projekt, und alle Beteiligten arbeiten soviel an dem Projekt, wie es die übrige Arbeit und die Familie zulassen. Zum Glück besteht keine Gefahr, daß VolksForth veraltet.

Wer Interesse an einer Mitarbeit hat, ist sehr willkommen. Es gibt viel zu tun, und besonders als Forth-Einsteiger kann man einiges lernen.

Volksforth ist wieder da!

*Carsten Strotmann*

Hallo Carsten,

das Volks4th wird von mir immer noch eingesetzt in der Datenbankanalyse von Flugdaten. Zur Zeit bekomme ich pro Gerät und Minute über 800 Parameter auf die Festplatte, die der Auswertung harren. Und was hilft mir dabei? Natürlich eine blockorientierte Datenbank, in Anlehnung an die Adressdatenbank im "Starting Forth", geschrieben mit dem Volks4th. Wichtig dabei ist der schnelle Bildinhaltwechsel in der DOS-Box, für eine schnelle, visuelle Sichtung der Daten, die schönen Suchtools und die gewisse Kompatibilität zu Excel und demnächst Mathcad.
Ich habe Ullis neuen Patch schon eingebaut und muss mich nicht mehr „schäl“ anschauen lassen, ob der NT4 (und höher) Fehlermeldung bezüglich *direct* beim Start des Programms.

Ansonsten ist hier das volks4th.dok, wonach Du suchst, im Anhang und eine Dokumentation zum Delta-t Forth.

*Karsten Roederer*

## Die Top 20 Hersteller von 8 Bit Controllern 2003

Ab und zu veröffentlichen Marktforschungsunternehmen [1] entsprechende Statistiken. Platz 1 wird traditionell von Motorola, die sich jetzt in Freescale umbenannt haben, belegt. 68HC05 und 68HC11 laufen zwar nominell zugunsten des 68HC08 aus, die Oldtimer werden aber wohl immer noch die Haupteinnahmequelle sein.
Nach Meinung anderer Erbsenzähler war Microchip knapp daran, Motorola einzuholen. Die Fusion von Hitachi und Mitsubishi zu Renesas hat aber für eine neue Nummer 2 gesorgt, die allerdings das Problem hat, daß nun alle 8 und 16 Bit Produktlinien doppelt existieren. Die Kunden dürfen nun zitternd die Portfoliobereinigung erwarten. Da Mitsubishi im 16 Bit Bereich stärker war, wird wohl der H8 das Rennen machen.
Microchip hat jetzt insgesamt 6 Familien im 8 Bit Bereich, und ist am erfolgreichsten wohl immer noch mit sehr kleinen Prozessoren wie dem klassischen PIC16xx. Da sind die Margen geringer, deshalb in $-Statistiken immer hinten, für Stückzahlen sähe es wohl anders aus.
Atmel dürfte zwar im Herzen der Hobbyisten mit dem AVR einen Spitzenplatz einnehmen, ist aber einstweilen sogar einen Platz abgerutscht. Trotzdem war das Wachstum mit +36% gut und der Aufstieg in den letzten Jahren kontinuierlich.
Infineon ist bei 8 Bit zwar nicht unter den Top 10. Aber bei Controllern mit dem 16 Bit 80166/167 wohl um so erfolgreicher. Im wichtigen Bereich Halbleiter für automotive in Europa hinter Freescale auf Platz 2, aber vor ST .
Dito taucht TI in dieser Statistik gar nicht auf, weil der lowend MSP430 als 16 Bit gezählt wird, obwohl er in Preis und Anwendung mehr mit highend 8 Bit konkurriert.
Europäer können sich mit Blick aufs Tabellenende noch über Micronas (Ex-ITT Freiburg) freuen. Ältere Semester wird warm ums Herz, wenn sie Zilog entdecken. Ansonsten tummeln sich in der unteren Hälfte vermehrt Taiwaner und Koreaner, die den Japanern zunehmend das Wasser abgraben. Wer glaubt, es sind zuwenig Europäer aufgeführt, soll mal die Amerikaner abzählen.
Haben solche Statistiken für den Anwender Nutzen? Wenn man langfristige Verfügbarkeit von Bauteilen bevorzugt, sind Marktanteile und damit Stückzahlen durchaus von Bedeutung. Teile die viele „design ins“ haben, kann der Halbleiterhersteller über Jahre hinweg produzieren, weil er Kunden hat, die kaufen. Da sind dann auch nach Abkündigung die Chancen besser, daß man noch Bestände bei Brokern oder z.B. www.us-bid.com bekommt bzw. eigene Überbestände dort los wird. Selbst Hersteller der Grösse Motorolas bauen Flops, wie z.B. die 68HC16-Serie, die recht schnell vom Markt verschwindet.
Für kleine Hersteller sind Versuchsballons der Regelfall: Erst viel bunte Werbung, dann nicht lieferfähig und bald darauf abgekündigt.
[1] www.isupply.com

*Rafael Deliano*

*Mit dem Hinweis auf die vorhergehende Seite, bzw. auf meinen Wunsch, ein Volks-Forth für meinen PDA haben zu wollen, stelle ich hier die Frage, was das denn überhaupt für ein Prozessor ist, dieser PXA 255. Mit welchen Werkzeugen läßt sich dafür Software entwickeln (natürlich unter WinCE, bzw. WinMob 2003)? Wie gut kann ein Forth in dieser CPU sein? Läßt sich auf dem PC für den PDA entwickeln? Wo kann ich entsprechende Werkzeuge bekommen? Welcher Forther hat bereits Erfahrungen mit diesem oder anderen PDAs?*

*Glückauf*
*Friederich Prinz*

| 2003 | 2002 | Hersteller | % | Mio $ |
|---|---|---|---|---|
| 1 | 1 | Frescale | 16,1 | 833 |
| 2 | 6,9 | Renesas | 10,5 | 544 |
| 3 | 2 | Microchip | 10,3 | 534 |
| 4 | 3 | NEC | 9,2 | 476 |
| 5 | 7 | ST | 7,8 | 405 |
| 6 | 5 | Atmel | 7,5 | 389 |
| 7 | 4 | Toshiba | 5,5 | 283 |
| 8 | 10 | Philips | 5,0 | 261 |
| 9 | 8 | Matsushita | 4,8 | 250 |
| 10 | 12 | Samsung | 2,1 | 111 |
| 11 | 16 | Sony | 2,1 | 109 |
| 12 | 11 | Fujitsu | 2,1 | 107 |
| 13 | 13 | Infineon | 2,0 | 102 |
| 14 | 15 | Hynix | 1,9 | 99 |
| 15 | 14 | Sanyo | 1,8 | 95 |
| 16 | 21 | Holtek | 1,1 | 55 |
| 17 | 20 | Micronas | 1,0 | 54 |
| 18 | 18 | Zilog | 1,0 | 52 |
| 19 | 22 | Kawasaki | 1,0 | 50 |
| 20 | 19 | Winbond | 0,9 | 46 |

# Bessere CRCs

Rafael Deliano

Die Einführung in [1] befaßte sich hauptsächlich mit der Implementierung. Wie aber in [2] schon angedeutet, gibt es qualitative Unterschiede bei den Polynomen.

Fehlerkorrigierende Codes werden üblicherweise durch die Längenangabe ( n,k ) charakterisiert. Dabei ist k die Zahl der Datenbits und n die Zahl der Datenbits plus Prüfbits (Bild 1). CRCs sind gekürzte zyklische Codes. Das Datenwort k hat bei CRCs zudem variable Länge, da die Länge des Datenpakets meist recht willkürlich festgelegt wird. Man kann aber formal annehmen, daß es die maximale Länge hat und die oberen Bits einfach alle den Wert 0 haben. Die natürliche Länge n (Tabelle 1) gibt damit an, wieviel Bit ein CRC minimal abhängig von der Länge des Datenpakets haben sollte.
Zur Charakterisierung der Sicherheit ist die „undetected error probability“ $P_{ue}$ von Interesse. Also die Zahl der Fehler, die die CRC nicht erkennt. Es gibt dafür einen simplen Grenzwert, der nur von der Länge der CRC abhängt (Tabelle 2). Der gilt aber nur bei langen Datenpaketen. Die Qualität der Polynome sieht man, wenn man die Annäherung an den Grenzwert im Datenwort bestimmt, z.B. für die übliche CRC-12 (Bild 2 [4]) und eine verbesserte 12 Bit CRC (Bild 3 [4]). Überschwinger, die den Grenzwert überschreiten, sind offensichtlich unerwünscht. Die horizontale Achse, die die BER (bit error rate) des Kanals darstellt, ist hier nicht wie üblich logarithmisch sondern linear dargestellt und spreizt damit den Bereich sehr vieler Fehler auf. Der Parameter k gibt die Zahl der Datenbits an.
Kurze Pakete sind mehr gefährdet. Speziell bei Feldbussen sind die Pakete aber kurz und die Umgebung ist verseucht. Allerdings sollte man berücksichtigen, daß die CRC-12 kaum schlechter als eine 11 Bit CRC wird, sie ist also gut brauchbar, nur eben nicht optimal. Nicht nur CRC-12, auch CRC-16 ANSI und CCITT haben diese Überschwinger. Auch wenn CCITT etwas besser ist als ANSI.
Das Problem wurde bereits früh erkannt [3]. Es dauerte aber, bis gezielt nach Polynomen mit monotoner Annäherung gesucht wurde. In [4] wurde eine neue CRC-12 vorgeschlagen. In [5] hat man sich an der ETH Zürich 16 Bit vorgenommen und optimale Polynome passend für verschiedene maximale Längen n bestimmt. Für 24 und 32 Bit [6] war dann ein eigener Spezialcomputer, vermutlich basierend auf FPGAs, nötig (Tab. 4). In Tabelle 3 ist jeweils die empfohlene Paketlänge n und das Polynom in hex angegeben. Die Faktorisierung hat für den Anwender keine Bedeutung, ist aber von Interesse für den Entwurf von CRCs, der in einem späteren Beitrag angesprochen werden soll.
Obwohl man häufig algebraische Ansätze zur Bestimmung der Restfehlerwahrscheinlichkeit findet ([7] , [8] , [9]), scheint nur eine Rechnersimulation nach vorangegangener Bildung von Fehlermodellen praxistaugliche Aussagen zu ermöglichen.

[1] emb 3 S. 6 „CRC“
[2] emb 4 S.14 „Sicherheit von Prüfsummen“
[3] Leung-Yan-Cheong , Hellmann
„Concerning a bound on undetected error probability“
IEEE Trans. on Inf. Theory March 1976
[4] Witzke, Leung „ A Comparison on Some Error Detecting CRC Code Standards“; IEEE Trans. on Com. Sept 1985
[5] Castagnoli, Ganz, Graber „Optimum Cylic Redundancy-Check Codeswith 16 Bit Redundnncy“
IEEE Trans. on Com. Jan 1990
[6] Castagnoli, Bräuer, Hermann „Optimization of Cylic Redundancy-Check Codes with 24 and 32 Parity Bits“
IEEE Trans. on Com. June 1993
[7] Münchrath „Datensicherung auf Übertragungswegen mit zyklischen Codes“ Franzis-“Elektronik“ 8/1976
[8] Merchant „CRC-Tests einmal ganz anders betrachtet“
WEKA-“Elektronik“ 23/2003
[9] Merchant „Exakte Bestimung der Restfehlerwahrscheinlichkeit“ WEKA-“Elektronik“ 18/2004

---

Tabelle 1: Natürliche Länge

```
Bit       Byte
CRC-8  n = 2^( 8-1)-1 =   127 =   ca. 15
CRC-12 n = 2^(12-1)-1 =  2047 =  ca. 255
CRC-16 n = 2^(16-1)-1 = 32767 = ca. 4095
```

Tabelle 2: Grenzwert $P_{ue}$

```
CRC-8  Pue = 2^-8  = 3,9 * 10^-3
CRC-12 Pue = 2^-12 = 2,4 * 10^-4
CRC-16 Pue = 2^-16 = 1,5 * 10^-5
```

Tabelle 3: Alte und neue 16 Bit CRC-Polynome

```
Name           n  Polynom   Faktorisierung
C1           151  13D65     EB23 * 3
C2           258  1F29F     65D3 * 3 * 3
IEC TC57     254  15B93     CB * FD * 3 * 3
C3           257  15935     15935
IEEE WG77.1  255  16F63     11D * 177
C4         28658  1011B     9E5 * B * 3 * 3
C5         32767  1A2EB     9E59 * 3
CCITT      32767  11021     F01F * 3       alt
ANSI       32767  18005     8003 * 3       alt
SDLC       32766  1A097     757B * 3 * 3   alt
```

Tabelle 4: Neue 24 Bit CRC-Polynome

```
Name        n  Polynom   Faktorisierung
6.1      4094  15D6DCB   BC9 * BAF * 3 * 3
6.2      4098  17B01BD   4DAAD9 * 3 * 3
MP 8.1   4094  1323009   FE5 * EF9 * 3 * 3
MP 8.2   4094  1401607   AAB * A01 * 3 * 3
5.1      4097  131FF19   131FF19
5.2      4095  15BC4F5   1897 * 19B7
4      388607  1328B63   EE7921 * 3
```

## Forth-Gruppen regional

**Moers**

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

**Mannheim**

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

**München**

**Jens Wilke**
**Tel.: (0 89) - 8 97 68 90**
Treffen: jeden 4. Mittwoch im Monat
**Ristorante Pizzeria Gran Sasso**
**Ebenauer Str. 1**
**80637 München**

**Hamburg**

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

## Gruppengründungen, Kontakte

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

## µP-Controller Verleih

**Carsten Strotmann**
mikrocontrollerverleih@forth-ev.de
mcv@forth-ev.de

## Forth-Hilfe für Ratsuchende

**Jörg Plewe**
Tel.: (**02 08) - 49 70 68** (p)

## Spezielle Fachgebiete

FORTHchips
(FRP 1600, RTX, Novix)

**Klaus Schleisiek-Kern**
Tel.: (**0 40) - 37 50 08 03** (g)

KI, Object Oriented Forth,
Sicherheitskritische
Systeme

**Ulrich Hoffmann**
Tel.: (**0 43 51**) **- 71 22 17** (p)
Fax: **- 71 22 16**

Forth-Vertrieb
vlksFORTH
ultraFORTH
RTX / FG / Super8
KK-FORTH

Ingenieurbüro **Klaus Kohl**
Tel.: (**0 70 44) - 90 87 89** (p)
forth@designin.de

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

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

Hinweise zu den Angaben nach den Telefonnummern:

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

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

Tabelle 5: Neue 32 Bit CRC-Polynome

| Name | n | Polynom | Faktorisierung |
|---|---|---|---|
| 8 | 2046 | 1F1922815 | 465 * 557 * 787 * 3 * 3 |
| MP 8.1 | 1023 | 1404098E2 | 531 * 4C9 * 747 * 3 * 2 |
| MP 8.2 | 1023 | 10884C512 | 42D * 463 * 7B1 * 3 * 2 |
| 6 | 65534 | 1F6ACFB13 | 8100 * C85F * 3 * 3 |
| 5.1 | 65537 | 1A833982B | 1A833982B |
| 5.2 | 65535 | 1572D7285 | 15BAB * 10FEB |
| 4 | 2^31 -1 | 11EDC6F41 | F5B4253F * 3 |
| IEEE-802 | 2^32 -1 | 104C11DB7 | 104C11DB7 |

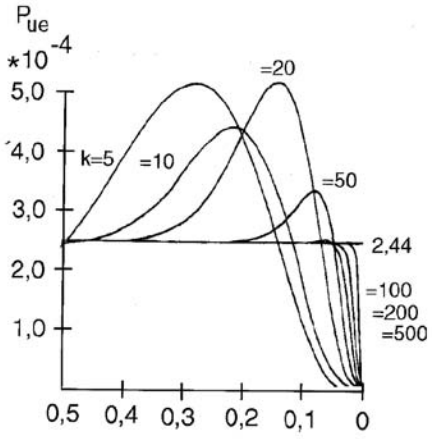



Bild 2: unentdeckte Fehlerwahrscheinlichkeit alter CRC-12

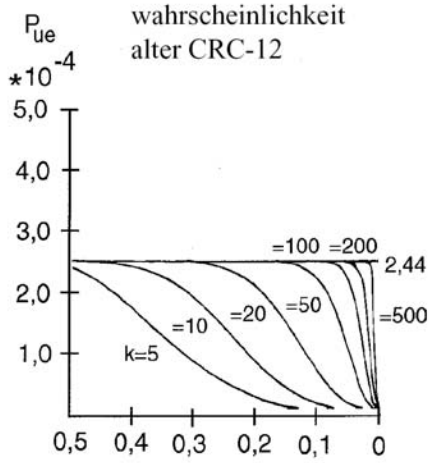



Bild 2: unentdeckte Fehlerwahrscheinlichkeit verbesserte 12 Bit CRC

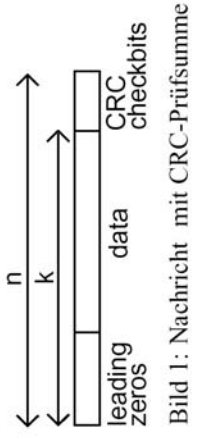
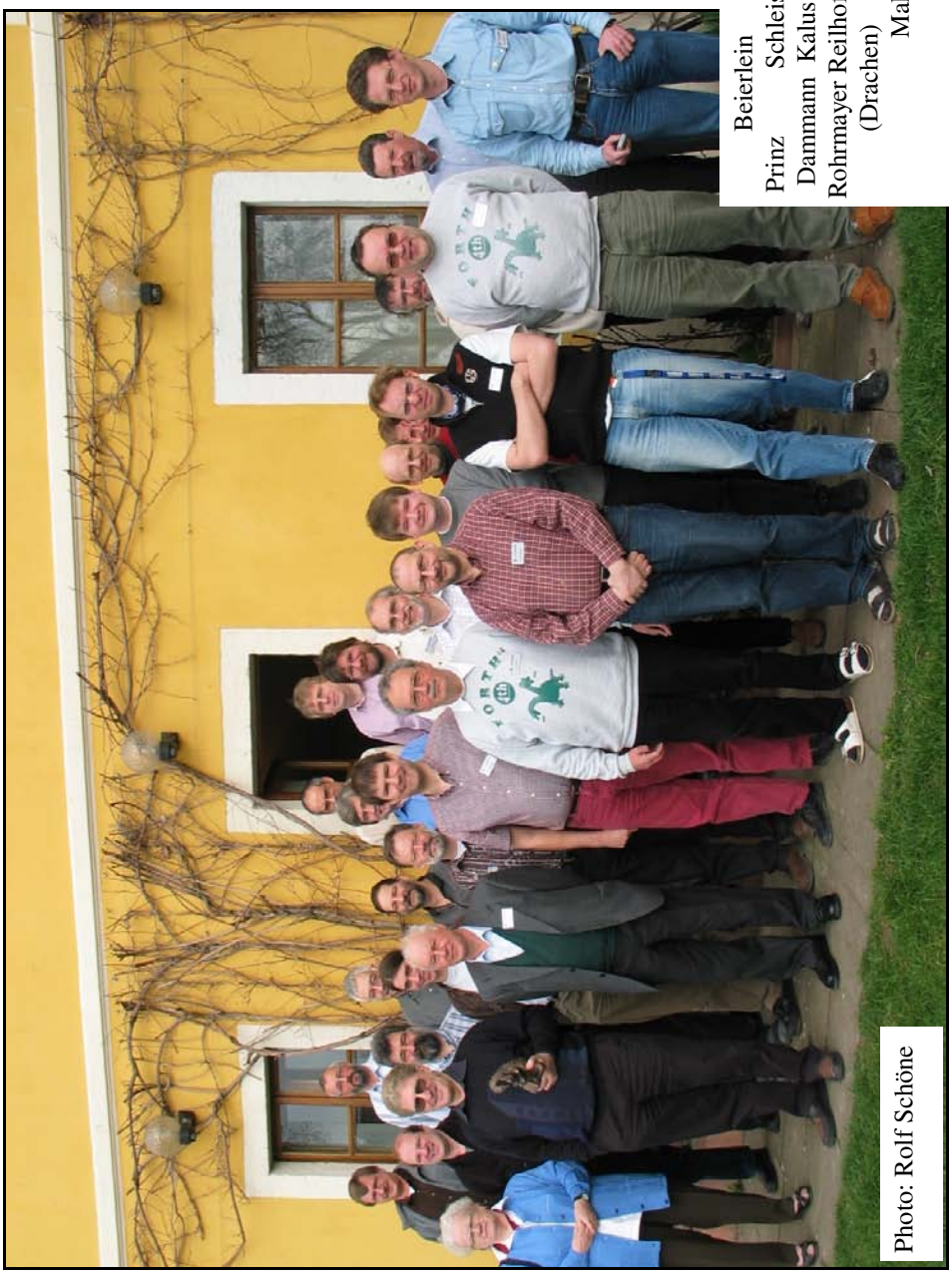



Bild 1: Nachricht mit CRC-Prüfsumme


Johannes Reillhofer gibt in diesem Jahr auf den Drachen Acht!

Meinen Glückwunsch.

Friederich Prinz



Beierlein Martin Pohl Strotmann
Prinz Schleisiek Amend Zobawa Völker Lauer Schöne
Dammann Kalus Malte Krüger Rieger Soll Eckes
Rohrmayer Reilhofer Behringer Hoffmann Schnitter Woitzel Wilke Roederer Paysan
(Drachen)
Malte und Martin heißen Bitter

Photo: Rolf Schöne