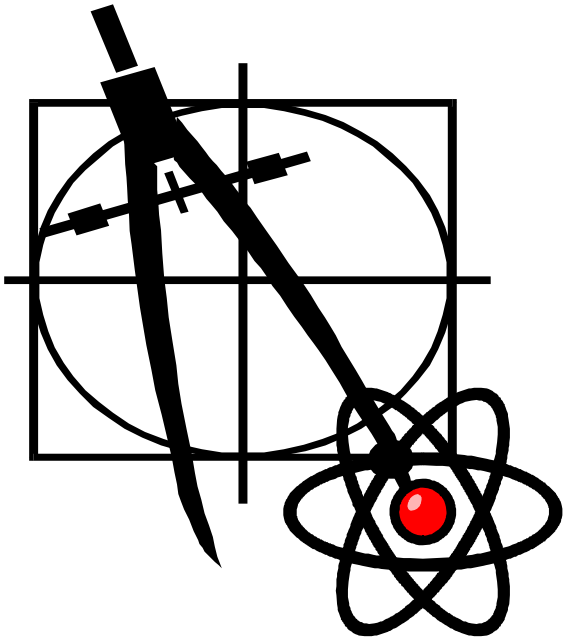
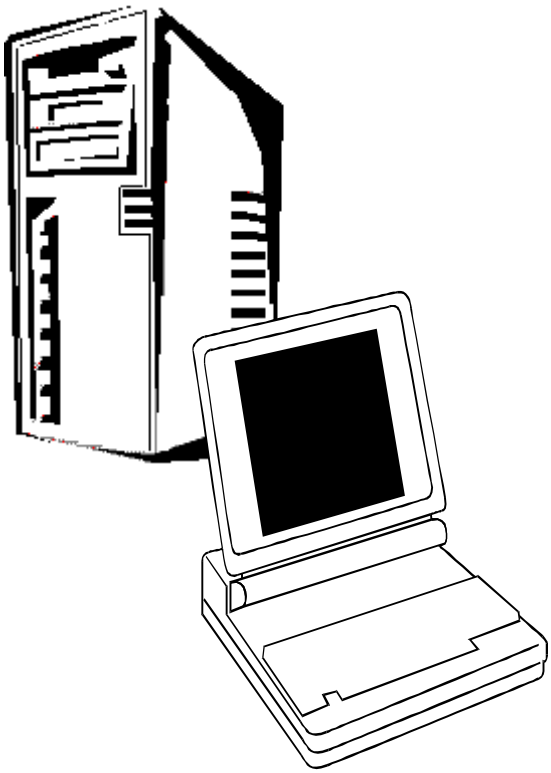
VIERTE DIMENSION

3 / 2000

Das FORTH-Magazin

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





In dieser Ausgabe:

16. Jahrgang - **Ausg. 3/2000** - 10,-

Organ der FORTH-Gesellschaft e.V.

Dienstleistungen und Produkte fördernder Mitglieder des Vereins

Inhalt

In der nächsten Ausgabe finden Sie voraussichtlich:

- Digitale Signaturen und eingebettete Systeme – Ulrich Hoffmanns Vortrag auf der Tagung 2000
- Getriebe-Endkontrollen – Johannes Reillhofers Vortrag auf der Tagung 2000
- UUENCODE / UUDECODE, Implementierung in Forth – Will Baden
- 2XOR und Viererproblem, Rätselauflösungen – Fred Behringer

Editorial

IMPRESSUM

Name der Zeitschrift

Vierte Dimension

Herausgeberin

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

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

Redaktion & Layout

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

Anzeigenverwaltung

Büro der Herausgeberin

Redaktionsschluß 1999

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

Erscheinungsweise

1 Ausgabe / Quartal

Einzelpreis

DM 10,- zzgl. Porto u. Verp.

Liebe Leser,

kunterbunt und gut gefüllt ist die VD dieses Mal wieder geworden – so kunterbunt wie es die Tagung der Gesellschaft in Hamburg war. Wir hoffen, daß Ihnen diese bunte Mischung gefällt, daß „für jeden“ etwas dabei ist. Spieleprogrammierung, Mathematik und die Implementierung eines WEB-Servers – um nur einige Beiträge zu nennen – werden abgerundet von den Leserbriefen eines „eigentlich nicht (mehr) Forthers“ und unseres Freundes aus der FIG Silicon Valley, einem Bericht über die Tagung in Hamburg und den Rezensionen der Periodika der Forther in Großbritannien und den Niederlanden. Alle diese Beiträge sind mit Arbeit verbunden gewesen, für die den jeweiligen Autoren und Übersetzern an dieser Stelle herzlich gedankt sein soll. Von dieser Arbeit lebt die Forthgesellschaft.

Einen Web-Server zu implementieren ist, wie Bernd Paysan schreibt, eine Arbeit für einige Stunden. Diese Arbeit anschließend in eine ansprechende Form zu bringen und sie ausreichend zu beschreiben und zu dokumentieren, damit auch andere Freude daran haben können, benötigt in der Regel in Vielfaches der Zeit, die für die ursprüngliche Arbeit aufgebracht werden mußte. Um so anerkennenswerter ist es, daß Bernd seinen Beitrag in dieser Ausgabe zuvor in Hamburg vorgetragen und zusätzlich auf seiner Homepage als PDF-File zur Verfügung gestellt hat.

Es ist eigentlich unnötig zu erwähnen, daß alle diese Arbeiten in den meisten Fällen von unseren Autoren und Übersetzern neben dem Beruf oder dem Studium, und zusätzlich zu familiärer Inanspruchnahme geleistet werden.

Vielen Dank !

Friederich Prinz

P.S. Eine unerwartete und in mehrfacher Beziehung einmalige Anerkennung für meine Arbeit in der FG und mit der VD habe ich während der Jahresversammlung von Arndt Klingelnberg bekommen. Die Tasse mit Brodies SWAP ist auf beiden Seiten handbemalt und ein echtes Einzelstück. Trotz seiner Unersetzlichkeit ist das Kunstwerk – mit gebührender Vorsicht – seinem ursprünglichen Verwendungszweck entsprechend im Einsatz, vor allem während der Endmontage der VD.

Vielen Dank, Arndt

fep

Quelltext Service

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

Leserbrief von Michael Kalus

Hallo Friederich,

hier hast du einen kleinen Beitrag zur 4D aus meiner Sicht des Forth. Möglicherweise habe ich zu dem Theama schon mal was verfasst, aber nun lobe ich die Gesellschaft als solche in diesem Beitrag besonders.
Und wenn sowas heutzutage auch im WWW zu lesen ist, mag es der Gesellschaft ja etwas nützen.
Pack es hin, wo immer du willst, wenn du den Beitrag gebrauchen kannst.
Herzliche Grüße aus dem Urlaub in Holstein,

Michael Kalus.

Forth **Gesellschaft 2000**

In diesem Jahr werde ich seltsamerweise fünfzig Jahre alt und habe eigentlich seit vielen Jahren nichts mehr mit Forth zu tun in meinem Leben. Trotzdem komme ich jedes Jahr wieder zur Jahrestagung der Forthgesellschaft und höre begeistert zu und wurde eigentlich noch nie enttäuscht von den Vorträgen und Vorführungen dort, wenn ich auch nicht immer jedes Thema gleich gut verstanden habe. Dort traf ich schon viele kompetente Leute, die sich mit Mikroprozessoren, PCs und so ziemlich allem, was sich da drum herum so an Themen anordnen läßt, auskennen. Und was mir dann noch gefallen hat in all den Jahren, war die lockere Art dieser Menschen in der Forthgesellschaft, und auch deren Vielseitigkeit von privaten Interessen. Dadurch gab es neben den eigentlichen technischen Einsichten immer auch sehr anregende Gespräche über Gott und die Welt und jede Menge Witz und Spaß.

Forth kreuzte meinen Weg als ich ein junger Assistenzarzt war. Die Geräte mit Mikroprozessoren hielten Einzug in die Klinik. Fast alles, was bisher so in diesen Geräten wie EKG oder einer Röntgenanlage gewesen war, konnte ich, im Prinzip jedenfalls, verstehen – und nun vergessen. Was da nun kam, war undurchsichtig, und das ärgerte mich. Der Frage, wie es möglich ist, daß so ein Haufen aus Draht und ICs mit mir redet, wollte ich nachgehen. An der Uni im Medizinstudium war das nicht mal am Rande Thema gewesen, doch hatte ich über Freunde bei den Physikern und E-Technikern und auch bei den Psychologen schon mal mit in den Terminalräumen gesessen und war staunend durch die Lochkartenstanzräume gegangen und wurde durch das Rechenzentrum mit all seinen großen Blechschränken geführt, hinter deren Scheiben sich Magnetbänder drehten. Vorbei an tischgroßen Plottern, an denen auf geheimnisvolle Weise Karten und Kurven entstanden, und hin zu mächtigen Druckern, die ganze Zeilen auf einmal mit Typen bedrucken konnten und endlose Berge von grünzeiligem Papier mit Zahlenkolonnen füllten, Papier, das man dann auch gleich darauf kartonweise als Manuskriptpapier mitnehmen durfte, da sie nach dem Ausdruck eigenartigerweise nach kurzer Durchsicht zu Müll erklärt worden waren.

In einem Fortrankurs, den ich dann neugierig belegte, lernte ich Lochstreifen herzustellen, mit denen sich die Lochkartenmaschinen steuern ließen - was nützlich war, um damit originelle Einladungskarten oder Grußkarten herzustellen. Doch gelang es mir nicht, auch nur das einfachste Programm herzustellen. Die Aufgaben damals waren für mich völlig sinnlos. Zahlenreihen in eine Matrize zu setzen und diese zu invertieren und diese dann in vielen Iterationsschritten formatiert auf Papier wieder auszugeben, war nicht meine Welt. Zumal zwischen dem Programmieren mit Bleistift auf dem grünzeiligen Manuskriptpapier und dem Ergebnisausdruck 2 Tage vergingen. Dazwischen lag Coding in einen Lochstreifen, der dann zu Lochkarten wurde, die dann vorne im Rechenzentrum abgegeben und nach eben 2 Tagen hinten in einem Fach wieder abzuholen waren, ergänzt um ein Blatt Papier mit Fehlermeldungen. Na, da war wohl doch nicht alles ganz richtig gewesen. Sie können sich vorstellen, was das für ein Frust gewesen ist.

Nun gut, es sollte ja auch kein Beruf werden, ich war ja nur aus Neugier da in den Kurs geraten. Doch meine Frage konnte dort so nicht beantwortet werden. Das sei halt ein ganzes eigenes Studium und so "mal eben" eben nicht zu vermitteln. So schlummerten meine Fragen dann also weiter im Unterbewußtsein, bis auch die EKG‘s nun so etwas wie Computer geworden waren.

Nun fand ich in einer Computerzeitschrift einen Bausatz mit dem Mikroprozessor 6502. Auf der Platine hatte es ein Tastenfeld für hexadezimale Eingabe und eine 6-stellige Zeile aus 7-Segmentanzeigen, ein bißchen RAM und ein ROM, das einen winzigen Monitor darstellte um hexadezimale Eingaben an die Adressen im RAM zu bringen. Ach ja, und einen Portbaustein natürlich, für die Anzeige und das Tastenfeld und der hatte 8 Bit I/O frei für Experimente. Nachdem dieser Bausatz zusammengelötet war, kam der bange Moment – würde alles gehen, wenn der Strom eingeschaltet wurde? Ja, es ging. Das war ein sehr erhebender Moment, ein Gefühl der Art “So, nun beherrsche ich die Welt, denn ich kann so etwas Kompliziertes bauen!” Die Illusion verflog zwar rasch als es an das Programmieren ging, doch wuchs das Verständnis für Probleme dieser Welt daran gewaltig, brachte mich doch diese Maschine gnadenlos dahin, mir keine Illusionen mehr zu machen, sondern so lange zu forschen und zu streben nach Verständnis der Zusammenhänge, bis alles genau bekannt war – erst dann war diese Maschine bereit zu gehorchen.

Ich kann mich noch gut erinnern, daß es lange Stunden gedauert hatte, bis dann endlich eine LED-Zeile am Port zum ersten Mal korrekt ein Lauflicht abspielte. Damals weckte ich mitten in der Nacht meine Freundin, um mit ihr diesen stolzen Moment zu teilen und zeigte ihr das Wunderwerk. Doch sie hatte dafür nur einen vorwurfsvollen Blick übrig: “Und dafür weckst du mich!?” Sie werden verstehen, daß der Computer natürlich das Rennen gemacht hat und ich heute mit einer anderen Frau verheiratet bin. Nicht das diese nun etwas mehr davon verstünde oder mein Interesse gar mitfühlen

könnte, nein. Aber ich habe sie natürlich nicht auf solche harten Proben der Beziehung gestellt, und so werden wir hoffentlich noch recht lange beisammen sein. Übrigens bekam damals auch meine berufliche Karriere eine andere Richtung, und ich wurde Psychotherapeut. Ich versuche nun meine gründliche Einsicht in menschliche Beziehungskonflikte anderen zu gute kommen zu lassen.

Es dauerte nicht lange und es war klar, daß auch eine hexadezimale Eingabe mühselig ist. Doch war es schon besser als diese Lochkarten-Rechenzentrum-Papier-Zyklen der Programmentwicklung. Da bot sich eine Gelegenheit, einen AIM zu erwerben, ein Board mit einem 6502 darauf, den ich ja nun schon kannte, einer echten vollen ASCII-Tastatur und einer einzeiligen alphanumerischen Leuchtanzeige, einem kleinen Drucker dazu und einer Möglichkeit das Programm in einem Stück RAM Batterie zu puffern. Denn eine 5”-Floppy war unerschwinglich. Und diese Maschine kam mit einem Assembler. Das war schon komfortabler, erlaubte auch schon recht tiefe Einblicke in die Innereien dieses Apparates “Mikrocomputer” und ich kam einer Antwort auf meine Frage näher. Dann bekam ich einen Tip – es gab ein Basic als ROM dafür. Und der Händler erwähnte noch, daß auch Forth zu haben wäre. Also erstand ich gleich beides mit Handbuch und setzte meine Versuche fort. Dabei stellte sich schon nach wenigen Versuchen heraus, daß Forth interaktiv und zum Untersuchen der Maschine ideal war. Keine Adresse blieb mehr unerreichbar, jedes denkbare Manöver konnte ich im Handumdrehen ausführen – toll. Das hätte ich damals gebraucht in der Uni im Rechenzentrum. Nun konnte ich mir schließlich die Frage beantworten, warum so ein Haufen Draht reden kann. Register, Akkumulator, Speicher, Zähler, Ports, Busse wurde mir durchsichtig. Mit Forth als Sonde konnte ich alles untersuchen und sogar das Forth selbst damit verstehen. Nun lag die ganze lange Entwicklung vom Tastendruck zum Bit und von da zum Zeichen auf dem Monitor ausgebreitet vor mir. Das einzige allgemein nützliche Forthprogramm das ich dabei zustande gebracht habe, war ein Disassembler und ein DisForthler, den ich brauchte, um das Forth für den AIM in seine Quelle zurück zu übersetzen, denn auch das wollte ich ja verstehen. Und erstaunlicherweise ging das sogar recht einfach zu machen.

Nun war meine Frage hinreichend beantwortet, und ich habe nichts Wesentliches mehr programmiert. Den AIM habe ich weggegeben, er tat bis vor kurzem Dienst als Steuerung einer Anlage in Wuppertal. Den ersten Mikrocomputer habe ich noch aus nostalgischen Gründen aufgehoben. Und das ich damals bei der Gründung der Forthgesellschaft dabei war, um Hilfe bei der Beantwortung meiner Frage(n) zu bekommen, habe ich nie bereut. Ich bekam immer freizügig Antworten auf alle meine Fragen rund um Computer und sehr viele praktische Hilfe bis heute und begegnete selten Hochmut, sondern Freundschaften und immer Leuten, die wirklich durchblicken. Es ist diese gute Gesellschaft die mich bewogen hat, bis heute Mitglied zu bleiben.

Mka

Buchrezension für VD von **Joachim Merkel**
(J.Merkel@Tbx.Berlinet.de)

Titel	Alan Turing, Enigma
Autor	Andrew Hodges
	[Übers. von Rolf Herken und Eva Lack]
Orig.Titel	Alan Turing: The Enigma, 662 S.
	Berlin, Kammerer & Unverzagt 1989
1. Auflage	Reihe Computerkultur ; Bd. 1
	Wien [...] Springer, 1994,
	Reihe Computerkultur ; Bd. 1
2. Auflage	

Entschlüsselung entscheidet Kriege, war die Überlegung von Fred Behringer am Beginn seines Beitrags zu polyalphabetischen Codes.

Ein Beispiel dafür liefert ein Buch von Andrew Hodges mit dem Titel "Alan Turing, Enigma", das 1989 in deutscher Übersetzung erschien: Eine präzise und erste umfassende Darstellung des Knackens verschiedener deutscher militärischer Verschlüsselungsmaschinen durch den britischen Geheimdienst in Bletchley im 2. Weltkrieg, zuerst mit von Turing mitentwickelten Relaismaschinen und später mit dem "Elektronikgehirn" Colossus.
Ebenso wichtig waren die Dechiffrier-Algorithmen des Chef-Kryptoanalytikers für die Marine-Enigmas, Alan Turings.

Die Erfolge des britischen Geheimdienstes bei der Dechiffrierung des Funkverkehrs der deutschen Wehrmacht und der Marine - die Marine hatte eine noch leistungsfähigere Enigma als das Heer - hält Hodges mit für kriegsentscheidend. Praktisch der Einsatz der gesamten Kriegsmarine und vor allem der U-Boote lag nun offen. Durch die ungeheure Variationsbreite der Enigma bei der Erzeugung polyalphabetischer Schlüssel im Milliardenbereich geblendet, hatte auf deutscher Seite niemand den damit verschlüsselten, gemorsten Funkverkehr für knackbar gehalten.

Was war nun die Sicherheitslücke?

"Seit der Morgendämmerung der Zivilisation hatten die Menschen über Maschinen nachgedacht, aber "On Computable Number" hatte eine präzise mathematische Definition des Konzepts einer 'Maschine' vorgelegt." (S.290) Die Veröffentlichung dieses Buches von Turing (im Alter von 24) erfolgte 1936. Durch seine Beiträge zur mathematischen Beweistheorie verstand er die mathematischen Grundlagen der maschinellen Ver- und Entschlüsselung und den entscheidenden Ansatzpunkt für die Dechiffrier-Algorithmen bei den Enigmas; hatten sie doch eine verblüffende "[...] Analogie mit den formalistischen Konzeptionen der Mathematik, in der Implikationen mechanisch durchlaufen werden sollten [...]. (S.211)

Bei der Marine-Enigma diente zur Verschlüsselung die mit jedem Buchstaben wechselnde Kombination von mit 26 Buchstaben beschrifteten (veränderlich steckbaren)

Ringscheiben auf zuerst 4 (davon 3 aus 8 Typen manuell tauschbaren) Rotoren. Durch den Gebrauch von 10 Paaren elektrischer Steckfelder wurden zudem die Ein- und Ausgabe eines Buchstabens erneut vertauscht. Dieses Steckfeld war der Unterschied der militärischen zur kommerziellen Enigma mit drei Rotoren, die frei verkäuflich war und in Banken eingesetzt wurde. Im Ergebnis waren Milliarden verschiedene Kombinationen des Vertauschs von Buchstabens möglich. Ferner sollte eine doppelte Verschlüsselung der Übermittlung der vom Chiffreur gewählten täglich veränderten Ausgangsstellungsstellung zum Beginn einer Nachricht die Erkennung von Mustern unterbinden. Die Nachrichten an sich wurden meist einfach verschlüsselt.

Durch Vorarbeiten polnischer Mathematiker und ihrem Auffinden regelmäßiger Muster im Ergebnis der Verschlüsselung, sog. Fingerabdrücke, war ein Weg gefunden worden, zunächst die verwendete Rotorstellung bei einem Funkspruch zu ermitteln. Ferner gelang es ihnen, durch Analyse des Funkverkehrs, die Verschaltung der zunächst mit 6-7 Buchstabenpaaren verbunden Steckfelder herauszufinden. Ein weiter Angriffspunkt war die inverse Verschlüsselung der Nachrichten (das heißt, Nachrichten wurden durch dieselbe Maschine entschlüsselt, die auch verschlüsselte) und nicht zuletzt die mangelnde Fähigkeit, einen Buchstaben mit sich selbst zu verschlüsseln.
Bei der weiteren Entschlüsselung in England wurde unter Zuhilfenahme von Wahrscheinlichkeitsrechnung nach Buchstabenfolgen, sog. "Cribs", gesucht, die häufiger vorkommenden Worten entsprachen. Dazu diente eine zeitaufwendige 'bruteforce-attack' mit mechanischen Relaismaschinen mit Spitznamen Bombe, wegen des tickenden Geräusches der Relais. Diese Bomben waren im Grunde ein Nachbau der Funktionen einer Enigma, um sowohl richtige Kombinationen zu finden, als auch falsche auszusondern. Die Identifizierung von sog. "Cribs" wäre nach Hodges nicht möglich gewesen, wenn jede Nachricht doppelt verschlüsselt worden wäre. Auch sorgsamere Chiffrierung hätte das Ergebnis unentschlüsselbar werden lassen. So gelang es nie, die militärischen Überseefunksprüche Hitlerdeutschlands etwa zu Schiffen im Indischen Ozean zu entschlüsseln. Die systembedingten Lücken beim Verschlüsseln durch eine Maschine und ein umfassendes logisches Verständnis ihres Bauprinzips war somit das Einfallstor für die maschinelle Entschlüsselung.

Die Auswertung aller Funkdaten, die Erbeutung von Verschlüsselungsunterlagen und das Zusammenführen aller Erkenntnisse waren zudem nur mit einer industriellen Forschungsorganisation zu bewältigen.
Das Knacken der polyalphabetischen Verschlüsselung des deutschen Funkverkehrs im 2. Weltkrieg blieb bis 1974 ein sorgfältig gehütetes militärisches Geheimnis.

Das Buch enthält darüber hinaus eine einfühlsame Biographie des Computer-Pioniers Alan Turing, wie eine ebenso ausführliche Würdigung seiner wissenschaftlichen Arbeiten.

jm

MiniModul 2M
Eine Prozessorkarte incl. Forth

Hans Eckes
Am Krebsbach 34
85757 Karlsfeld

- Beitrag der Forthtagung 2000; Hamburg -

Seit einigen Jahren gibt es wieder einen Forth-Prozessor, den PSC1000 der Fa. Patriot (www.ptsc.com). Der folgende Artikel soll einen ersten Eindruck vom Prozessor vermitteln, sowie eine PSC1000-Prozessorkarte mit dazu passendem Forth beschreiben.

Was ist der PSC1000?

Der PSC1000 ist ein 32-Bit Stackprozessor. Anders als bei registerorientierten Prozessoren dreht sich hier alles um das Manipulieren der Daten auf den Stacks. Die unteren 18 bzw. 16 Einträge im Daten- bzw. Returnstack werden im Prozessor gehalten. Kommen mehr Einträge dazu, werden die ältesten Einträge in den Speicher ausgelagert, bzw. von dort wieder hereingeholt. Dies geschieht alles automatisch, die Software ist mit der Stackverwaltung nicht beschäftigt.

Etliche Forth-Kommandos existieren aufgrund der Stackarchitektur bereits als Befehle im Prozessor (DUP, SWAP, ROT, DROP, OVER, @, C@, >R, R>, R@, +, -, 0=, AND, OR, XOR, 1+, 1-, 4+, 4-, 2*, 2/, NEGATE, EXECUTE). Viele andere Forth-Kommandos lassen sich aus wenigen Befehlen zusammenstellen.

Da sich der Befehlssatz auf das Arbeiten mit dem Stack konzentriert, blieb die Anzahl der Befehle unter 256. Damit wird pro Befehl nur ein Byte verbraucht. Als 32-Bit Prozessor holt sich der PSC1000 bei jedem Speicherzugriff gleich 4 Befehle, und noch während diese abgearbeitet werden, holt der Prozessor gleich die nächsten 4. Ist der Speicherzugriff schneller als das Abarbeiten der 4 Befehle (was bei schnellem Speicher der Fall ist), kann der Prozessor nahtlos weiterarbeiten.

Die meisten Befehle werden in einem Taktzyklus abgearbeitet, ausgenommen sind Befehle wie Multiplikation (32 Takte), Division (32 Takte), Shiften um mehrere Stellen und einige andere, die 2 Takte benötigen. Bei Unterprogrammaufrufen werden bis zu 4 Bytes benötigt, also ein Befehl pro Speicherzugriff. Für eine Überschlagsrechnung kann man sagen: Bei einem Prozessortakt von 80 MHz, also 12,5 ns und einem Speicherzyklus von 4 Takten, also 50 ns, werden im besten Fall 80 Assembler-MIPS erreicht und im schlechtesten Fall 20 Assembler-MIPS. Wenn man weiter davon ausgeht, daß ein Forth-Wort, welches den Stack manipuliert, im Schnitt aus etwa 4 Assembler-Befehlen zusammengesetzt ist, kann man etwa 5 bis 20 Forth-MIPS ansetzen.

MiniModul 2M

Wer sich jetzt die Frage nach der Rechengeschwindigkeit des MiniModuls stellt, dem ist die Antwort in max. MIPS und geschätzten durchschnittlichen MIPS sicher etwas unkonkret. Der Zeitbedarf des folgenden Einzeilers mit einer Million Durchläufen wurde für das MiniModul und 2 verschieden schnelle PCs mit 80386 UR/FORTH ermittelt:

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

- 1,80 Sekunden für das MiniModul
- 3,40 Sekunden für ein DX2-50 Notebook
- 0,95 Sekunden für einen P166 Desktop
```

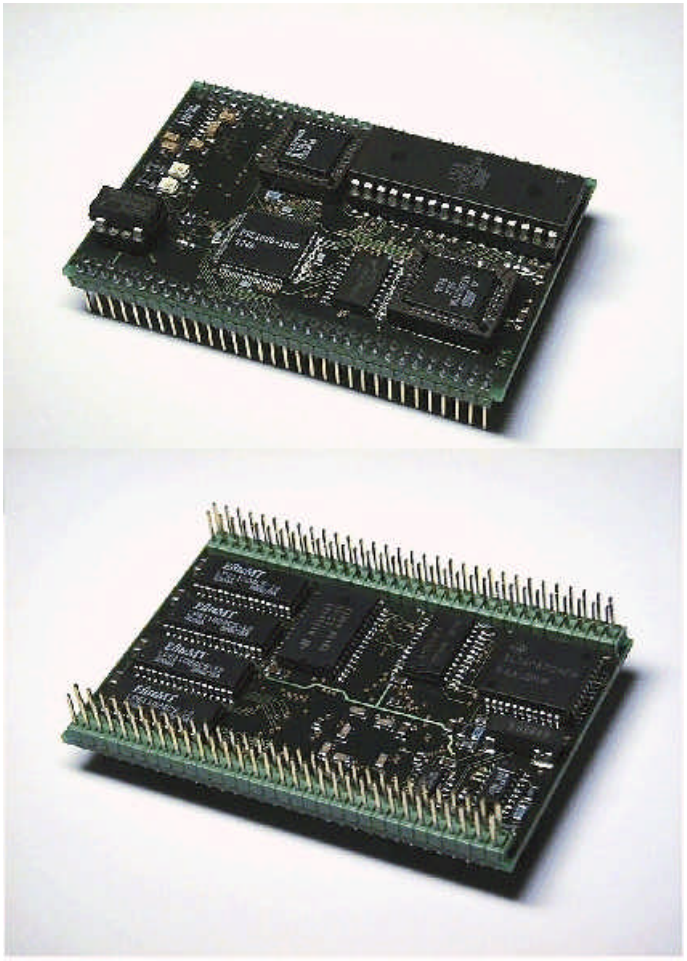
Als "Benchmark" ist die eine Zeile etwas mager, die Richtung dürfte allerdings stimmen.

Der PSC1000 kann die angeschlossenen Bausteine in bis zu 4 Speichergruppen einteilen, jede mit eigenem Timing, einstellbarer Datenbreite von 8 oder 32 Bit; gleichgültig, ob es sich um SRAM oder DRAM handelt. Damit lassen sich problemlos z.B. schnelle SRAMS mit 32 Bit Breite und 15 ns Zugriffszeit neben EPROMs mit 8 Bit und 100 ns oder SRAM neben DRAM betreiben. Beim Zugriff auf die entsprechende Adresse wird automatisch das korrekte Speichertiming eingehalten.

MiniModul 2M - die Prozessorkarte zum PSC1000

Mittlerweile gibt es einige käufliche Prozessorkarten, die den PSC1000 einsetzen, das MiniModul 2M ist eine davon. Im folgenden wird das MiniModul mit einigen Stichworte beschrieben:

Das MiniModul ist eine 6-Lagen-Multilayer Platine mit den Abmessungen 56 x 85 mm und wird mit 5 Volt versorgt. Neben dem Prozessor befinden auf dem Board:
- ein wechselbarer Oszillator mit max. 40 MHz.
  Der Prozessor verdoppelt intern den Takt und läuft dann mit 80 MHz und einer Zykluszeit von 12,5 ns.
- 512 KByte SRAM, 32 Bit breit, 12 ns. Dies ist der Arbeitsspeicher des MiniModuls.
- 512 KByte SRAM, 8 Bit breit, 70 ns, batteriepufferbar. Kann zum Speichern von Daten oder der Applikation verwendet werden.
- bis zu 512 KByte Flashrom, 8 Bit breit, im PLCC-Sockel, onboard programmierbar.
- bis zu 512 KByte Bootrom, 8 Bit breit, im DIL-Sockel, 32Kx8 EPROM bis 512Kx8 Flashrom werden unterstützt.
- Serielle Schnittstelle mit UART 16C550 und MAX 232.
- Realtime clock RTC 72423, batteriepufferbar.
- Watchdog
- alle Prozessorleitungen, 8 Bit Adressbus, 3 freie Chipselects und die serielle Schnittstelle gehen zu den beiden 64-pol. Pfostensteckern.

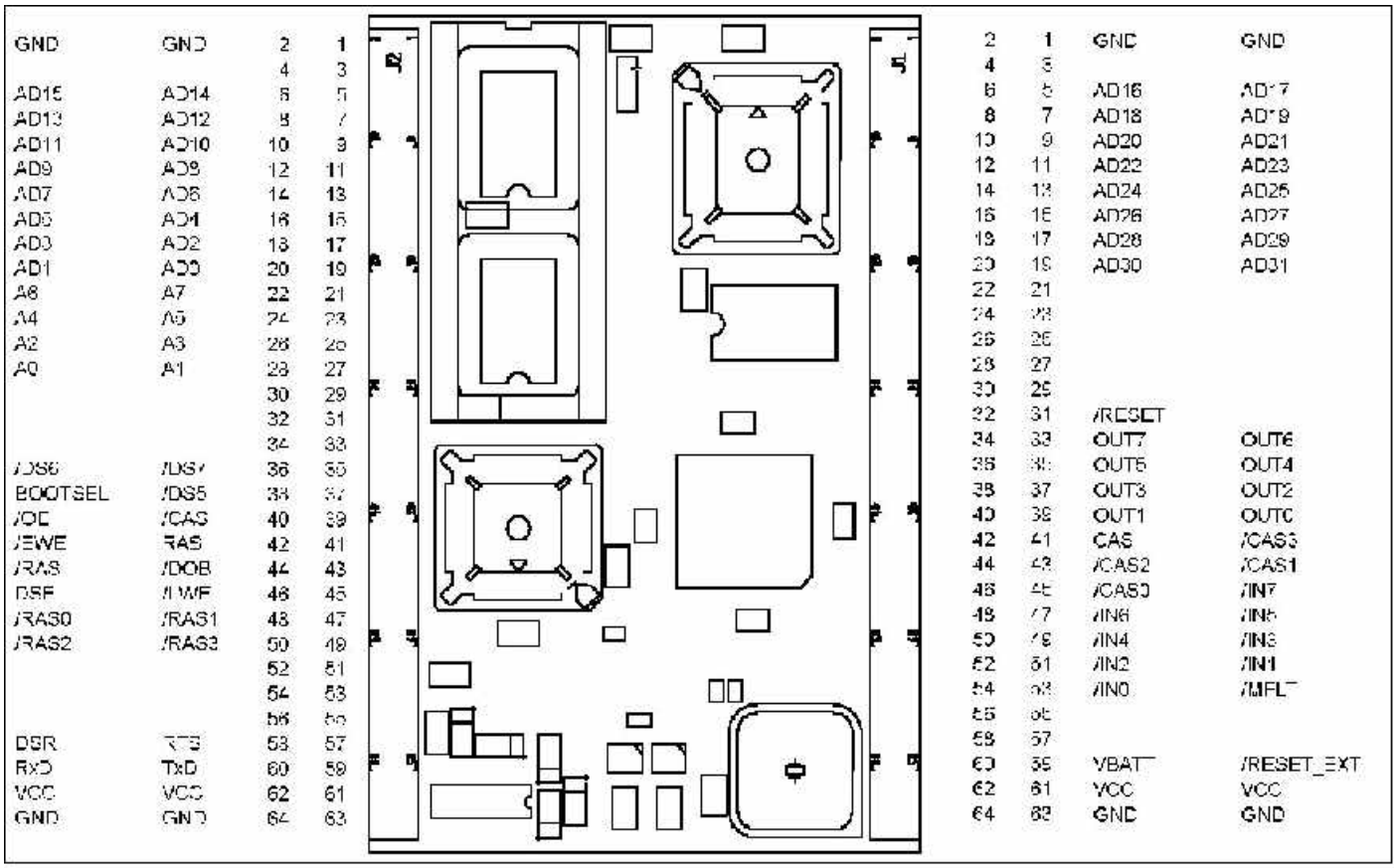


Das MiniModul ist als Aufsteckkarte gebaut (RM 2,54mm) und dafür gedacht, weitere Peripherie entweder direkt anzusprechen oder mittels Treiber eine größere Anzahl von Bausteinen zu bedienen. Das typische Einsatzgebiet ist Messen/Steuern/Regeln; derzeit wird das MiniModul in verschiedenen Geräten der industriellen Meßtechnik eingesetzt.

Das Forth zum MiniModul

Das Forth des MiniModuls entspricht in seinem Funktionsumfang einem 32-Bit Forth83 und ist in seinem Verhalten dem 80386 UR/Forth von LMI nachempfunden. Das Forth ist kpl. in Assembler der Fa. Patriot geschrieben (und in deren Softwareemulator lauffähig) und nicht von irgendwoher crosscompiliert. Beim Portieren von Wörtern in Forth-Quelltext nach PSC1000-Assembler fiel auf, daß die Anzahl der Assembler-Befehle etwa der Anzahl der Forth-Kommandos entspricht.

Neben einem Standardwortschatz beinhaltet es folgende Funktionen:
- Round Robin Multitasker
- PSC1000-Assembler incl. Labels und Vorwärtsreferenzen
- Disassembler
- dynamischer Stringpuffer
- Zugang zu sämtlichen Prozessorregistern, z.B. Ändern des



Bustimings vom Interpreter aus.
- Ansprechen der Realtime Clock
- Programmieren des Flashroms (Atmel und AMD werden unterstützt)
- einfacher Zugang zu den 3 freien Chipselects

Der Interpreter läuft auf dem MiniModul; die Softwareentwicklung geschieht daher interaktiv. Der PC wird lediglich als Terminalprogramm und zum Speichern von Quelltext benötigt. Der Quelltext wird in Screens geschrieben, über die serielle Schnittstelle mit 115,2 kBaud in das MiniModul heruntergeladen und dort compiliert.

Der Compiler erzeugt keinen gefädelten Code sondern direkt Assembler-Befehle bzw. Unterprogrammaufrufe, wenn das zu compilierende Wort länger als 4 Byte ist. Dieser Mechanismus erlaubt es auch, Assembler-Befehle ohne CODE ... END-CODE mitten unter die Forth-Wörter zu streuen und natürlich auch, ganze Wörter kpl. in Assembler zu definieren.

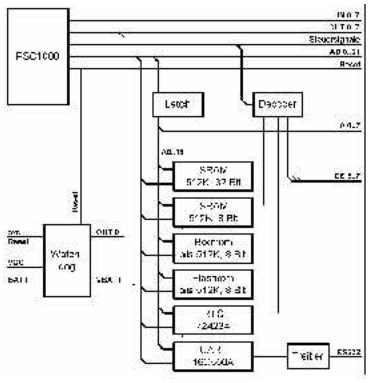
Die neu dazugekommenen Wörter lassen sich aus dem Arbeitsspeicher in das Backupram oder Flashrom abspeichern. Beim nächsten Booten werden sie automatisch in den Arbeitsspeicher geholt und das zuletzt definierte Wort gestartet. Wird dieses Wort (die Applikation) beendet, startet der Interpreter. Zu Debugzwecken lässt sich das Booten abbrechen, so daß sofort der Interpreter läuft.

Der Vorteil: beliebige Applikationen lassen sich auf einem Standard-Bootrom aufsetzen, es muß nicht für jede Änderung im Quelltext ein neues Bootrom erzeugt werden. Wenn später weitere Anforderungen dazukommen, kann alles mit einem neuen Download und nicht durch Wechseln des Bootroms gelöst werden.

Entwicklungsarbeit für die nächste Zeit

Software lebt und wird genaugenommen niemals fertig. Softwareentwicklung ist allerdings kein Selbstzweck, man will ja schließlich etwas damit machen. Das Werkzeug (Forth) ist schon fertig, jetzt kommen die Anwendungen.

Die nächsten Schritte werden sein:
- Anschluß eine LCD-Bildschirms 1/4 VGA oder VGA
- USB-Schnittstelle zum Anschließen von PC-Peripherie
- Ethernet-Schnittstelle incl. zugehöriger Protokolle
- Entwicklung weiterer PSC1000 Prozessorkarten



Jeder, der aus Freude am Entwickeln oder wegen konkreten Bedarfs Interesse an Hardware und Software rund um den PSC1000 hat, ist herzlich eingeladen, an diesem Projekt mitzuarbeiten.

Hans Eckes

Direktorial

Directorial

Liebe Forth-Freunde,

ich danke allen, die mich gewählt haben.

Natürlich ist die Jahresversammlung kein repräsentativer Querschnitt. Die Teilnahme kostet den einzelnen ein Vielfaches des Jahresbeitrags und ist mit Enthusiasmus allein nicht mehr zu rechtfertigen. Ich werde mich aber um das Vertrauen auch der "restlichen" 90% der Mitglieder bemühen.

Mein Konterfei findet sich in der VD 3/99. Ich bin nicht mehr der Jüngste. Das empfinde ich aber nicht unbedingt als Nachteil. Schließlich kann ich mir jetzt eher denn je meine Zeit, die im übrigen immer schneller dahinrast, nach Belieben einteilen. Ich löse Fritz (Friederich Prinz) in seinem Amt als Direktor ab. Fritz, der uns ja als Redakteur, als erfolgreicher Redakteur, erhalten bleibt, hat mir angeboten, mich mit Rat und Tat zu unterstützen. Ich nehme das Angebot mit Freude und Dank an.

Ich habe mich auf den diversen Stufen meiner Ausbildung und später als Ausbildender mit den verschiedensten Fachgebieten beschäftigen dürfen und müssen. Forth gehörte nicht dazu.

Mein Einstehen für Forth hat unter jenen, die den Rechner nur als Werkzeug betrachten und nach dem Techniker schreien, wenn dieser verflixte Kasten wieder mal nicht gleich auf Anhieb geht, Befremden ausgelöst: "Macht man denn das?", "Geht das nicht anders viel besser?". Forth war, ist und bleibt mein Hobby. In meine Beschäftigung mit Forth kann ich aber nachgewiesene Abschlüsse, Erfahrungen oder/und Forschungsarbeiten einfließen lassen als/in: Radiobastler, Hochfrequenzmechaniker (RFT), Physiker (Weltraumstrahlung, Röntgenspektroskopie), Übersetzer (5 Bücher, eng., holl., franz.), Automatische Dokumentation (EURATOM), Mathematiker der angewandten Richtung (Analogrechnen, Hybridrechnen, optimale Skalierung von Differentialgleichungssystemen, Optimierungstheorie linearer, nichtlinearer, dynamischer, diskreter Art, Transporttheorie, Zuverlässigkeitstheorie, Entscheidungstheorie bei mehrfacher Zielsetzung, Wahltheorie, Präferenztheorie, Spieltheorie, Operations Research allgemein, Vektoroptimierung, verallgemeinert konvexe Funktionen, Ausbildung von Studierenden der Mathematik und Informatik mit Nebenfach Wirtschaftswissenschaften und ein paar Dinge mehr), Buchautor (zwei Lehrbücher), diverse Forschungsarbeiten in nationalen und internationalen Fachzeitschriften.

Forth ist ein Fachgebiet wie jedes andere auch. Um Forth weiterzuentwickeln und den modernen Bedürfnissen anzupassen, brauchen wir die Experten. Unser Verein hat sich aber auch in starkem Maße den reinen Enthusiasten verschrieben, die aus anderen Berufssparten kommen und Forth "nur so zum Spaß" betreiben. Und es gibt Übergänge.

Mitglieder, die Forth in ihren eigenen Aufgabenbereichen wirklich "anwenden". Chemiker, Künstler, Mediziner, Lehrer, Bauingenieure, und was sie sonst noch alles sind, nicht zu vergessen unsere Rentner und Studenten. Ich werde versuchen, allen Strömungen gerecht zu werden, und ich werde vor allen Dingen diejenigen nicht aus den Augen lassen, die man als Anfänger, Nachwuchs, Einsteiger, Umsteiger oder wie auch immer bezeichnen kann.

Der Schwerpunkt meiner Bemühungen wird nach wie vor die Stärkung unserer Beziehungen zu anderen Forth-Gruppierungen in der ganzen Welt sein. Das habe ich bisher mit Begeisterung getan, und das werde ich jetzt mit "offiziellem Anstrich" noch besser tun können. Wir leben nicht allein auf dieser Welt. Globalisierung, Zusammenwachsen und Verständigung unter den Völkern darf gerade für uns Forthler kein bloßes Schlagwort bleiben.

Überall herrschen dieselben Probleme. Das Interesse an Forth läßt nach, die Mitgliederzahlen sinken. Echte Verständigungsprobleme gibt es entgegen mancher Unkenrufe eigenartigerweise nicht. Es gibt sie immer dann nicht, wenn man sieht, daß man die Probleme duch Zusammenarbeit besser in den Griff bekommt als einzeln. Das gilt für die Mitglieder unserer Forth-Gesellschaft, einer Interessengemeinschaft im deutschen Sprachraum, das gilt aber auch in bezug auf die vier, mindestens vier, aktiven Forth Interest Groups dieser Erde. Ich werde in meinen Bemühungen um sie nicht nachlassen.

Fred Behringer
behringe@mathematik.tu-muenchen.de

Verlautbarung

Betreff: This year's Dragon Award winner
Von: Fred Behringer
An: c.jakeman@bigfoot.com

Hi Chris,

The SWAP Dragon was handed over to Egmont Woitzel, this year's Dragon Award winner. Dr. Woitzel has gained merits in favour of Forth-Gesellschaft for over ten years. His dedication to Forth reaches even farther back, some years prior to his joining Forth-Gesellschaft.
He is the initiator and organizer of our new Web site.
Can you make a short notice of this in Forthwrite?
Fritz was jumping around with his newly acquired digital camera and has taken some photos of the handing-over ceremony (Egmont and yours truly – and the dragon). Are you interested?
I now know the electorial procedure (Drachenrat) and I was tasting quite a bit of that alcoholic liquid necessary to seal the treaty with the dragon, but I'm afraid I must keep the secret from the other members and from the rest of the world.
Best holiday wishes,

Fred

... Ja, der Swappie, der Swap-Drache, hat dem Drachenrat in Hamburg zu verstehen gegeben, daß er gerne ein Jahr in Rostock verbringen würde. Dem konnte der Rat sich selbstverständlich nicht verschließen. Also verbringt jetzt Egmont Woitzel ein Jahr mit den Drachen ;-)

Über die geheimnisvollen Riten während der Kür des jeweils neuen Preisträgers ist schon Vieles angedeutet worden. Nur wenig hat davon zur Aufklärung beigetragen. Aber Alles davon ist wahr; oder zumindest wahrscheinlich wahr.
Das Photo aus der Ratssitzung wurde unter Einsatz der Leber mit einer versteckten Kamera aufgenommen und gewährt einen leicht verschwommenen Einblick in das dunkelste Treiben des Rates. Zu erkennen ist, daß der SWAP eisern über die Einhaltung der Disziplin des Rates und die Ernsthaftigkeit der Diskussion wacht.

Aber nach getaner Arbeit gestattet der Drache den anwesenden Ratsmitgliedern das Glas auf den neuen Preisträger zu erheben – und sich des Überleben(r)s zu freuen.

von links: Friederich P.; Klaus S.; Ulrich H.; Bernd P. und Fred B.

Photo: K. Röderer

Herzlichen Glückwunsch !

Betreff: Film und Forth
Von: Martin Bitter <mbitter@bigfoot.de>
Foren: de.comp.lang.forth

Hallo Gemeinde,
Zurückgekehrt von der Forth-Tagung (Danke Bernd für den Bericht (s.h. Thread Forth-Tagung)) und aus dem Kurzurlaub, hat sich Malte (mein Sohn) obwohl es gegen Mitternacht war (oder deshalb?) vor den Verdummungskasten gehockt und sich den erstbesten Schund angesehen (Entzug, Erziehungsfehler? ...):
DRIVE - Keiner schägt härter (USA 96) Wie Malte selbst sagt: Blödsinn mit Action. Hier eine Zusammenfassung aus einer Fernsehzeitung: "San Fransisco 2008: Malik gerät in die Schusslinie von Gangstern, die es auf den Kampfexperten Toby abgesehen haben. In dessen Brust ist ein Chip implantiert, der ihm phänomenale Kräfte gibt."
Warum ich das schreibe? An einer Stelle im Film kommt (völlig unlogisch (O-Ton Malte)) eine Online-Umprogrammierung dieses Brustchips vor. Und - jetzt kommts - auf dem Bildschirm flimmerten Codezeilen in Forth vorbei!!!!

==> weiter: Seite 25

Gehaltvolles

zusammengestellt und übertragen
von Fred Behringer

VIJGEBLAADJE der HCC Forth-gebruikersgroep, Niederlande

Nr. 19, April 2000

AF-onturen van een B(+)eginner
Ben Koehorst
<b.c.koehorst@kader.hobby.nl>

Ben, der Schatzmeister des Vereins, berichtet über seine ersten Erfahrungen mit der Inbetriebnahme der AF-Karte und im Umgang mit dem 8052-Forthbuch. Er stellt sich dem Leser als blutiger Anfänger vor. Ist er natürlich nicht. Nur eben bisher keine Ahnung von dieser speziellen Karte, die an COM1 oder COM2 betrieben wird und mit OC535-ANS-Forth vsn 1.12+ läuft. Lustig zu lesen für Leute, die die Karte nur mal schnell "on the fly" ausprobieren wollen und dann doch nicht unbedingt gleich mit ihr zurechtkommen.

Vermogensregeling met Bresenham
Paul Wiegmans

Regelung bei Gleichstrommotoren. Statt Impulsbreitenregelung Anwendung des Bresenham-Algorithmus (den man üblicherweise zum Linienzeichnen verwendet). Die Software ist überraschend einfach. Kurzes Programm in Forth und 8051-Assembler.

Forth-Treffen am 8. April 2000 in der Volkssternwarte Utrecht. Von 10.30 bis 15.00 Uhr. Beantwortung von Anfängerfragen. Vortrag von Albert vd Horst und Coos Haak über Strings in Forth.

Am 15. April war der Tag der offenen Tür der HCC-Abteilung Arnheim, auf dem auch die Forth-gg vertreten war.

Die Redaktion (Albert Nijhof) ist nun auch per E-Mail erreichbar: a.nijhof@kader.hobby.nl

Auch Albert liest, wie er mir in seiner letzten E-Mail sagte, "ohne Schwierigkeiten Deutsch". Schreiben tut er mir auf meine Bitte hin Holländisch, da das natürlich "wesentlich leichter fällt, als sich im aktiven Gebrauch von Deutsch verkünsteln zu müssen". - Wenn man das nur ganz allgemein in Europa (und anderswo) einführen könnte! Mit den deutschen Dialekten geht es ja auch, wie wir auf der Tagung in Hamburg wieder erfahren durften.

(beh).

Forth Interest Group International (FIG USA)

Wollen Sie mit der ganzen Welt verbunden sein und dabei Ihr Englisch perfektionieren ? Amerika ist ein wesentlicher Teil der ganzen Welt. Zumindest, was Forth betrifft. Über tausend Mitglieder aus allen Ländern sind bei uns.
Werden Sie auch ein Mitglied in der Amerikanschen Forth-Gesellschaft (FIG-USA).
Für 45 Dollar im Jahr (Studenten zahlen 18 Dollar) bekommen Sie 6 Hefte unserer Vereinszeitschrift Forth Dimensions und genießen auch sonst verschiedene Vorteile. In den Heften erfahren Sie Forth-Neuigkeiten aus aller Welt, neue Produkte, Literatur, Forth-Ideen, fundiertes Wissen, Artikel auch für Einsteiger, Projekte, Leser-Diskussionen, Quelltexte, Hinweise auf Internet-Verbindungen, kostenlose Forth-Systeme und vieles mehr. (Für Übersee-Porto müssen wir leider noch 15 Dollar hinzurechnen).

Unmittelbare Informationen über uns bekommen Sie, wenn Sie auf der Homepage der Deutschen Forth-Gesellschaft “Links zu anderen Forth-Organisationen” und dann “Forth Interest Group (USA)” anklicken.

Ansonsten bekommen Sie Auskünfte über das amerikanische Forth-Büro:

Forth Interest Group
100 Dolores Street, suite 183
Carmel, California 93923
USA

oder auch vom Redakteuer, Marlin Ouverson, unter der E-mail Adresse:

E-Mail: **office@forth.org** oder
editor@forth.org

Nr. 20, Juni 2000

Robot-arm met (bijv.) vijf motoren
Albert Nijhof <a.nijhof@kader.hobby.nl>

... eine grade Linie zeichnen, mit dem fünfdimensionalen Plotter, den man sich schon immer mal bauen wollte ... Einen Roboter mit fünf Motoren (fünf Freiheitsgrade) zu programmieren, ist nicht gerade leicht. Albert bekam die Aufgabe von Willem gestellt, gibt zu, daß er "von Hardware wenig Ahnung" hat, und versuchte, die Aufgabe in ANS-Forth zu lösen.

Robot-arm, aapassing voor ByteForth
Willem Ouwerkerk <w.ouwerkerk@kader.hobby.nl>

Was machen wir uns da Gedanken über Gedanken über Martin Bitters neue Masche mit den Lego-Robotern und pbForth! Die holländischen Forth-Freunde beschäftigen sich seit geraumer Zeit mit Robotern. (Graeme Dunbar in England auch - aber keiner scheint vom anderen Notiz zu nehmen.) "Das Wort DUP kann nicht interaktiv verwendet werden. ByteForth ist ein Cross-Compiler und DUP kann nur compiliert werden." Was da nicht alles für Probleme auftauchen! VALUE ist "niet aanwezig" (nicht anwesend) in ByteForth ...

FORTHWRITE der FIG UK, Großbritannien

Nr. 106 April 2000

Chris Jakeman begrüßt in seinem Editorial zwei neue Mitglieder, weist auf Dave Pochins Web-Site "Getting Started" hin und macht darauf aufmerksam, daß die Forthwrite jetzt auch von der Homepage der FIG-UK downgeloadet werden kann.

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

Ulrich Hoffmanns Wiki Web Server, "persönliche Daten" über FIG-US-Mitglieder im Netz, Experten am Netz in UK, Europay und Forth-Inc., Bildbetrachter in Win32Forth, Parser für Forth nach C, Forth-OS, VHDL für Forth-Prozessor, DELTA Forth 1.0 (Java-basiert), Hempels pbForth für Lego, hForth für Z80, Holon für Forth auf Java Virtual Machines, iForth, eForth, VFX, RTX 2000, PSC 1000, P32, F21d.

6 The 21st FORML Conference - Forth and the Internet
Federico de Ceballos <fcebcab@ciccp.es>

Der lebendigste Bericht von einer FORML-Konferenz, den ich je gelesen habe. Man hat fast das Gefühl, dabeigewesen zu sein. Drei Zeilen pro Beitrag, aber die sitzen! Auch C.H. Tings Tao-Colon, der kürzeste Forth-Vortrag von Welt, wird in chinesischen Zeichen verständlich gemacht. (Wenn das nichts ist!) Federico fuhr über Madrid, London, Los Angeles nach Monterey. Aus seinem Bericht: Wie stellt Wil Baden fest, ob eine gegebene Maschine "little-endian" arbeitet?

```
: LITTLE-ENDIAN ( -- flag ) BASE C@ 0<> ;
```

Ich kenne Federico vom letzten Jahr von regem E-Mail-Verkehr her. Er ist Mitglied der FIG UK und dort speziell des siebenköpfigen WebForth-Teams. Er hat in dem englischen Projekt "Forth auf Java" den spanischen Teil besorgt. Man müßte Chris Jakeman mal fragen, ob und wie es mit WebForth weitergeht.

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 20 Gulden 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 20 Gulden auf das Konto 525 35 72 der HCC-Forth-gebruikersgroep bei der Postbank Amsterdam. Noch einfacher ist es wahrscheinlich, sich deshalb driekt an unseren Vorsitzenden, Willem Ouwerkerk zu wenden.

11 Did you Know - Novell
Chris Jakeman <cjakeman@bigfoot.com>

Eine interessante Rubrik, in welcher Chris Bemerkenswertes über und um Forth sammelt, wobei er "darauf achtet, Gerüchte und gesicherte Tatsachen auseinanderzuhalten. Das ursprüngliche Novell-Netzwerksystem (1980-81, also VOR dem PC) lief auf dem S-100-Computer (Z80) und war in Forth programmiert."

12 F11-UK, FIG UK Hardware Project
Jeremy Fowell <Jeremy.Fowell@btinternet.com>

Jeremy beendet seine Liste von Dingen, die in diesem (recht lebendigen) Projekt noch getan werden müssen: Glossar für Pymy-HC11 jetzt fertig, 19 A4-Seiten ... Er hat an Motorolas News-Group schreiben wollen unter 68HC11@oakhill-cisc.sps.mot.com. Kam zurück. Kann jemand helfen?

14 32-Bit GCD without Division in ZF and Turbo Forth
Fred Behringer
<behringe@mathematik.tu-muenchen.de>

Ein Beitrag des Rezensenten. Macht Spaß, die Sachen in bestem Sonntagsenglisch veröffentlicht zu sehen. Nur schade, daß auch hier selten "feedback" kommt. Es geht um den größten gemeinsamen Teiler zweier Ganzzahlen. Ganz ohne Division. "Binärer Euklidischer Algorithmus". Nur fortgesetzte Subtraktionen und Bitverschiebungen nach links. Für die 16-Bit-F83-Derivate ZF und Turbo-Forth. Jedoch gleich 32 Bit

breit. Natürlich in Assembler, d.h., als CODE-Definitionen gefaßt. Der ZF-Assembler braucht bloß um zwei Colon-Definitionen erweitert zu werden (nur zwei!), und schon läuft alles über die erweiterten (32-Bit-)Register. War gar nicht so leicht, einen Maschinen-Opcode zu finden, der führende Nullbits elegant beseitigt. BSF (bit search forward) schafft das.

19 Nominations for the FIG UK Awards

Aufruf, Kandidaten vorzuschlagen. Zwei Preise: einen für beste Forth-Leistungen, einen für Leistungen in bezug auf Forthwrite.

20 All you need to know about STATE, IMMEDIATE and POSTPONE
Jack Brien <jackbrien@bmallard.swinternet.co.uk>

Es geht um Compile-Only-Worte und Worte, die von **STATE** abhängen. Jack stellt fünf Regeln auf:

(1) Man versuche nie, sich den cfa-Inhalt von Compile-Only-Worten zu beschaffen oder ihn zu verwenden.
(2) Worte, die mit **POSTPONE** definiert wurden, sind normalerweise Compile-Only-Worte.
(3) Worte, die von **STATE** abhängen und mit **POSTPONE** definiert wurden, sind immer Compile-Only-Worte.

(4) **COMPINT** (siehe Artikel) ist eine abgesicherte Version von **POSTPONE**.
(5) Man vermeide Worte, die von **STATE** abhängen.

Meinung des Rezensenten: Genau wie die ANS-Empfehlungen können solche "Regeln" nur Empfehlungen sein. Wenn man mir die Freiheit nehmen würde, Regel 1 gelegentlich zu durchbrechen, würde ich die Beschäftigung mit Forth aufgeben.

26 Vierte Dimension 1/00
Alan Wenham <101745.3615@compuserve.com>

Alan schafft das wieder prima!

28 An Introduction to Machine Forth
John Tasgal <john@tcl.prestel.co.uk>

Machine Forth ist eine der jüngsten Neuerungen von Chuck Moore. 40 Forth-Worte, die sich eng an die Architektur des verwendeten Prozessors (hier des F21 von Ultra Technology) anschmiegen. Zurück zum "klassischen Forth", um die Fülle von kleineren Ready-to-go-Prozessoren anzubinden. Im nächsten Heft wird der Autor über Color Forth, ebenfalls von Chuck Moore, berichten, das auf Machine Forth aufbaut.

34 Dutch Forth Users Group

Diesmal wieder die Mitglieder-Werbeanzeige der holländischen Forth-Freunde.

35 From the 'Net - Cube Roots
Chris Jakeman <cjakeman@bigfoot.com>

Aus einem Faden von Beiträgen in comp.lang.forth. Paul Bennett wollte "eine gut arbeitende" Methode zur Bestimmung der dritten Wurzel einer positiven Ganzzahl; natürlich nur der größten ganzen Zahl kleiner/gleich dieser Wurzel, was überall stillschweigend angenommen wird. Interessant für den Rezensenten ist Chris' Fußnote, die wieder mal zeigt, daß es nicht reicht, etwas anzubieten, sondern daß man auch jemanden finden muß, der andere auf das Angebot aufmerksam macht. Such is life!

39 Cube Roots Again
Chris Jakeman <cjakeman@bigfoot.com>

```
(1)  0 + 1 =  1
(2)  1 + 3 =  4
(3)  4 + 5 =  9
(4)  9 + 7 = 16
(5) 16 + 9 = 25  usw.
```

Chris kommt nochmal auf diese bestechend einfache Art zu sprechen, die Quadratwurzel aus einer Ganzzahl zu ziehen. Er zeigt, daß Ähnliches auch für dritte Wurzeln gilt und in Forth mit **+LOOP** überaus einfach erledigt werden kann.

40 Letters

Ein Brief (an den Editor) von John Hayhow, der jetzt wieder zu FIG UK gestoßen ist und ein Gemeinschaftsprojekt "Forth für Mikro-Controller" starten möchte. Er erwähnt den PIC16F84, den Atmel 89C51 und fünf andere Controller. Er möchte aus dem bereits Bestehenden das Beste nach eigenen Gesichtspunkten zusammenstellen und ergänzen.

The Way of Stones

Soeren Tiedemann
Freiburg

Viele alte Kulturen wußten um die Geheimnisse dieses aussergewöhnlichen Geduldsspiels. Runa Furtak, wie die alten Germanen es nannten, oder Ishido, unter welchem es Japan eroberte, ist ein würdiger Vertreter unter den Spielen, die 'in einer Minute zu lernen, aber erst in einem ganzen Leben zu meistern' sind. Die Chinesen nannten es Tsi Tao; selbst den Maya war es in dieser Form bekannt. Es diente der Meditation und der Befragung des Orakels.

Das Spielfeld besteht aus 12*8 Feldern. Der Überlieferung nach wurden die Spielsteine (6 verschiedene Farben mit jeweils 6 verschiedenen, individuell bedeutungsschweren Symbolen) von Ihren jeweiligen Besitzern verehrt und gehütet. Manche schrieben den Steinen die Grundlage von Macht zu. So seien Kriege um die Steinsets geführt worden; sie zu erbeuten um die eigene Macht zu steigern, hätte aber viele Reiche ins Unglück gestürzt.
Wie dem auch sei, die Steine überlebten und es liegt an uns, an Ihnen Spaß zu finden.

Das Set besteht aus den 6*6 Steinen, wobei aber jeder Stein doppelt vorkommt. Also sind insgesamt 72 Steine auf 96 Felder zu legen.
Zu Beginn des Spiels werden 6 beliebige Steine auf dem Brett plaziert. In jede Ecke einer, in die Mitte zwei diagonal. Eine Einschränkung besteht nur darin, daß alle 6 Farben und alle 6 Symbole vertreten sein müssen, um einen Spielanfang garantieren zu können. Die restlichen Steine werden verdeckt gehalten.
Der 'Weg der Steine' ist ein Meditationsspiel für einen Spieler. Turniere und Vergleiche werden geführt, indem verschiedene Spieler mit der gleichen Reihenfolge der Steine spielen und die Ergebnisse verglichen werden.

Ziel des Spiels ist es, alle Steine nacheinander vom Stapel auf das Brett anzulegen. Der Stapel sollte geleert werden, was ab mittelmäßiger Spielstufe kein Problem mehr darstellt, gleichzeitig sollen so viele 4-ways wie möglich konstruiert werden. Was ist ein 4-way, und wie sehen die Anlegeregeln aus?

Ein Stein kann maximal 4 Nachbarn haben (die Diagonale zählt nicht!).

- Anlegen an einen Nachbarn: Farbe oder Symbol müssen übereinstimmen.

  Situation: A3 .
  Steine für das freie Feld: Ax oder X3

- Anlegen an zwei Nachbarn: Farbübereinstimmung mit einem, Symbolübereinstimmung mit dem anderen.

  Situation: A3 . B5
  Steine für das freie Mittelfeld: A5 oder B3.

- Anlegen an drei Nachbarn: Farbübereinstimmung mit einem, Symbolübereinstimmung mit beiden anderen
  ODER
  Symbolübereinstimmung mit einem, Farbübereinstimmung mit beiden anderen.

  C5
  Situation: A3 . B5
  Einzig möglicher Stein: A5!

- Legen des Steins als Abschluss in die Mitte von 4 Nachbarn (4-way):
  Farbübereinstimmung mit zweien,
  Symbolübereinstimmung mit den anderen.

  C5
  Situation: C1 . B6
  A6
  Einzig möglicher Stein: C6!

Bei erfolgreich geleertem Stapel zählt die Anzahl der komplettierten 4-ways. Spitzenspieler liegen immer zwischen 10-12 4-ways. Ich spiele momentan 6-8.

Diesem ungewöhnlichem Spiel, so fand ich, gebührt eine ungewöhnliche Lösung. Anschließend folgt der kommentierte 'CORE' (Implementation des Regelwerks) zum Anlegen der Steine in Maschinenforth für MuP21. Es hat mir wirklich Genugtuung verschafft, dieses Spiel in ein paar KB auf meinem MuP21 zum Laufen zu bringen. Ich habe über dieser Programmlösung ebensolange meditiert, als über 6 kompletten Spielen. (Für Esoterikfans ;-)

(MuP21 machine forth examples: The Way of Stones)

```
HEX
0 ORG

'IS Start

0 ',  0 ',  0 ',  0 ',  0 ',  0 ',  0 ',  0 ',
0 ',  0 ',  0 ',  0 ',  0 ',  0 ',  0 ',  0 ',

0 ',  'IS Spielfeld  7F 'ALLOT

0 ',  0 ',  0 ',  0 ',  0 ',  0 ',  0 ',  0 ',
0 ',  0 ',  0 ',  0 ',  0 ',  0 ',  0 ',  0 ',
```

In meinem System arbeitet die Anweisung ORG relativ zu einer Compilations-Adresse, die beliebig gesetzt werden kann. 0 ORG weist also nicht auf die physische Adresse 0 hin, sondern startet mit Offset 0 im vorher bestimmten Target-Bereich. Die Compileranweisung 'IS bildet einen Wortheader im Name-Dictionary mit nachfolgendem Namen, sodaß später die Adresse dieser Stelle mit ' Name auf den Stack geholt werden kann.
Das 12*8 Felder große Spielfeld ist wie folgt angelegt: 12 Felder mit den leeren Umrandungsfeldern zu Beginn und Ende ergeben 14 notwendige Felder. Zur einfacheren Berechnung werden selbstverständlich 16 Felder in x-Richtung gewählt. 8 Reihen mit oberer und unterer Begrenzung ergeben 10 Reihen.
Das Spielfeld besteht also aus 16*10 Feldern. Für jedes Feld wird die Wortlänge von P21 gewählt (20-bit). "Start" zeigt auf den Anfang, "Spielfeld" auf den gültigen Beginn des Bretts.

```
                                                    0 ',
'IS Z23  0 ',     0 ',     0 ',     0 ',
'IS Z21  0 ',  ' Z23 ',     0 ',     0 ',
'IS Z19  0 ',  ' Z23 ',     0 ',  ' Z23 ',
'IS Z17  0 ',  ' Z23 ',  ' Z23 ',     0 ',
'IS Z15  0 ',     0 ',   ' Z23 ',  ' Z23 ',
'IS Z13  0 ',     0 ',   ' Z23 ',     0 ',
'IS Z11  0 ',  ' Z17 ',  ' Z21 ',     0 ',
'IS Z09  0 ',  ' Z13 ',  ' Z17 ',     0 ',
'IS Z07  0 ',  ' Z15 ',  ' Z19 ',  ' Z17 ',    0 ',
'IS Z06  0 ',  ' Z19 ',     0 ',   ' Z21 ',    0 ',
'IS Z05  0 ',  ' Z07 ',  ' Z06 ',  ' Z11 ',
'IS Z03  0 ',  ' Z09 ',  ' Z11 ',     0 ',    0 ',
'IS Z02  0 ',     0 ',   ' Z15 ',  ' Z13 ',    0 ',
'IS Z01  0 ',  ' Z02 ',  ' Z07 ',  ' Z09 ',    0 ',
'IS Z00  0 ',  ' Z01 ',  ' Z05 ',  ' Z03 ',
```

So trivial wie es zunächst aussah, gestaltete sich die Lösung leider nicht. (Die Erfahrung wird leider immer wieder gemacht). Es wurden im Verlauf der Routinen immer mehr Entscheidungen nötig, es wurde immer häßlicher! Schlimmer noch, es sah ganz danach aus, als bräuchte man eine "Vorschau", um zunächst die Anzahl der Nachbarn herauszufinden, bevor die ganzen Entscheidungen überhaupt Sinn machten. Also: kehrt! Ich begann ganz systematisch mir die Zustände klarzumachen, in die man durch verschiedene Nachbarn gelangen konnte. Die Lösung lief also auf einen endlichen Automaten hinaus.

Zunächst begann ich einen Baum anzulegen und alle Pfade systematisch zu verfolgen. Es war richtig umfangreich! Ausgehend von einem Anfangszustand gab es immer die folgenden Möglichkeiten: Der nächste Nachbar paßte weder in Form noch Farbe (bin. 00). Er paßte in Farbe, aber nicht in Form (bin. 01), in Form, aber nicht in Farbe (bin. 10) oder in beiden (bin. 11).
Ausgehend von jedem dieser Zustände waren wieder alle Möglichkeiten für den nächsten Nachbarn zu berücksichtigen. Das Blatt war schnell voll!

Die maximale Anzahl von 4 Nachbarn setzte dem Ganzen Treiben zum Glück ein Ende. Es stachen natürlich sofort die toten Äste hervor. Paßte ein Nachbar weder in Form noch Farbe, so war der zu legende Stein offensichtlich an dieser Stelle falsch. Eine weitere Reduzierung des Graphen ergab sich durch die Tatsache, daß z.B nach Farbübereinstimmung und beim Nächsten nach Formübereinstimmung, der erreichte Zustand natürlich derselbe war, als wenn man erst eine Form und dann eine Farbübereinstimmung entdeckt hätte. Permutationen konnten also auch zusammengelegt werden.
Auch die Tatsache, daß jeder Stein nur zweimal vorkommt ergab Erleichterung. Stimmt z.B in der ersten Ebene ein Nachbar komplett überein, so kann dies ab der zweiten Ebene nie wieder auftreten.

Das System schmolz allmählich wieder auf erträgliche Größe zusammen. Noch zu lösendes Problem war die korrekte Erkennung von Endzuständen. Das Phänomen tritt zum erstenmal bei zwei entdeckten Nachbarn auf. Bei einem Nachbarn lande ich entweder sofort auf einem toten Ast (keine Form oder Farbübereinstimmung, oder ich bin erfolgreich dabei). Bei Zweien kann es ja durchaus passieren, daß für beide eine Form oder Farbübereinstimmung festgestellt wird. Dies wäre ein toter Ast, ja, wenn nicht noch ein dritter Nachbar folgen würde! Denn dann kann noch keine Aussage getroffen werden. Also muß dieser Zustand zunächst noch gültig sein. Steht man aber, nachdem alle Nachbarn getestet wurden immer noch auf diesem Zustand, DANN ist es ein toter Ast. Also spielte für die letztendliche Klärung der letzte Zustand eine entscheidende Rolle. Es wäre sicher möglich gewesen, eine weitere Spalte einzuführen, die signalisiert, ob der aktuelle Zustand ein gültiger Endzustand ist, oder nicht. Ich entschied mich aber für eine andere Lösung. Die Adresse selbst wird als Kriterium herangezogen. Die Tabelle ist so aufgebaut, daß gültige Endzustände auf einer UNGERADEN Adresse liegen. Ungültige Zustände dagegen auf einer GERADEN Adresse. Damit wird auch die Null als "toter Ast" ungültig. Anhand der Numerierung der Tabellenzustände kann man sehen, wann durch einen zusätzlich compilierten Dummy (0) an manchen Stellen die Zustandsadresse von ungerade auf gerade oder umgekehrt wechselt. Die erste Spalte (Offset 00, weder Form noch Farbübereinstimmung) enthält immer die Null (toter Ast). Die nächsten Spalten (Offset 01: Farbübereinstimmung, Offset 2 bzw. bin. 10: Formübereinstimmung, Offset 3 bzw.bin. 11: Farb- sowie Formübereinstimmung) enthalten jeweils die Adresse des nächsten Zustandes.

Es finden also Zustandsübergänge statt. Das war's eigentlich. Fehlt nur noch das Leidige drumherum:

```
': neu              \ A: --       RS: ret  DS: --
   ' Start '# nop a! dup
   dup -or dup dup
   com 2* com nop  \ A: Startadr RS: ret  DS: 1 0 0
   begin
      2* push !a+ dup
      !a+ dup !a+ dup
      !a+ dup !a+ dup
      !a+ dup !a+ dup
      !a+ dup nop nop
   pop -until
   drop drop drop ;'
                    \ A: --       RS: --   DS: --
```

Die Startadresse des Spielfeldes wird als Literal in das Adressregister A geladen. Im Verlauf dieser kleinen Routine sind 2 Datenstackplätze noch frei. Aus diesem Grunde wird die erste 0 mit "dup dup -or" erzeugt, um darunterliegende Werte zu erhalten. Dann wird aus der zweiten 0 die 1 erzeugt, die nun auf dem TOP liegt. 160 wird in dieser Routine als 20*8 aufgefaßt. Nach dem 20-sten Bit-shift nach links ist das Carry gesetzt und die Routine terminiert. In der Schleife werden dann 8 Worte auf Null gesetzt unter Ausnutzung des Autoinkrements des Adressregisters. Das gesamte Spielfeld wird gelöscht. Die Folge "dup !a+ dup !a+" ist vom Timing her etwas ungünstiger.
Dies ist der Grund, warum schon mit duplizierter Null in die Schleife eingestiegen wird.

```
MACRO 4*  2* 2*  END-MACRO
MACRO 8*  4* 2*  END-MACRO
MACRO 4/  2/ 2/  END-MACRO
MACRO 8/  4/ 2/  END-MACRO

': 'xy                 \ A: --  RS: ret  DS: y x
   4* 4*               \ A: --  RS: ret  DS: 16*y x
   -or ' Spielfeld '#  \ A: --  RS: ret  DS: 16y+x
                       \ Spielfeldadr
   '+ nop a! ;'        \ A: Adr RS: --   DS: --
```

Diese Routine wandelt x,y-Koordinaten in die tatsächliche Spielfeldadresse um. Das gültige Spielfeld ist 12 Felder weit. Die x-Koordinate darf nur zulässige Werte von 0 - 11 einschliesslich annehmen. Da y 4 Bits nach links verschoben wird, kann gefahrlos die x-Koordinate hineinge-or-t werden. In diesem Falle hätte es auch ein '+ getan, da das -or im ersten Slot steht. Trotzdem wird hier dieses Detail betont. "+" und "+*" sind die "langsamsten" Instruktionen, sie dürfen nur im ersten Slot stehen, im Zweifel muß das vorhergehende Wort mit "nop's" aufgefüllt werden. (Mein Compilerwort '+ erledigt das sofort)

So, nun ein paar Informationen über die Datenstrukturen. Die Information eines Spielsteines besteht aus seiner Form und seiner Farbe. Da es jeweils 6 verschiedene Möglichkeiten gibt, reichen jeweils 3 Bit aus. Die untersten 3 Bit codieren die Farbe, die nächsten 3 Bit die Form. 14 Bit sind frei. Auf dem Spielfeld werden diese teilweise benutzt, um Informationen über 4-ways zu halten. Auch für einen möglichen Computerspieler bieten sich hier Nutzungsvarianten.
Der nachstehend im Kommentar verwendete Bezeichner "Akt" steht für den zu legenden aktuellen Stein. " 'xy " steht für die Adresse, an die der Stein gelegt werden soll. " 'Zxx " bezeichnet die aktuelle Zustandsadresse in der Tabelle. Die Stackvariable Test enthält in den unteren 3 Bit den Zähler für die Nachbarn (maximal 4), Das 4. Bit codiert die Übereinstimmung der Farbe mit dem letzten getesteten Nachbarn das 5. die Formübereinstimmung. Die sich hier ergebenden Kombinationen (00, 01, 10, 11) werden später als Offset für die Zustandstabelle genutzt. Ist der Offset auf Null zurückgesetzt, wird diese Variable auch nur als "Zhl" bezeichnet. Der Bezeichner " 'N " ist die Adresse eines Nachbarsteines, ausgehend von 'xy. "N" selbst ist der Nachbarstein, an dem der aktuelle getestet wird.

```
': gleich?              \ A: N
                        \ R: ret ret ret
                        \ D: 8/4 F Test Akt 'Zxx 'xy
   push if              \ A: N  RS: ret ret ret 8/4
                        \ DS: F Test Akt 'Zxx 'xy
      drop pop ;'       \ A: N  RS: ret ret
                        \ DS: 8/4 Test Akt 'Zxx 'xy
   then
   drop pop dup push    \ A: N  RS: ret ret ret 8/4
                        \ DS: 8/4 Test Akt 'Zxx 'xy
   2* -or pop ;'        \ A: N  RS: ret ret
                        \ DS: 8/4 Test Akt 'Zxx 'xy
```

Dies ist der Abschluss des Testes auf Form oder Farbgleichheit mit einem Nachbarstein. "gleich?" testet das Flag F, ist dieses 0, so liegt Gleichheit vor. Wurde "gleich?" von Form aus gerufen, so ist das mitgelieferte Literal eine 8. Durch Bitshift nach links ergibt sich 16, (5.tes gesetzt.) Dieses wird in Test eingetragen. (Formübereinstimmung). Geschah der Aufruf von Farbe, so wird das in diesem Fall mitgegebene Literal 4 durch Bit-shift auf 8 erhöht (4.tes gesetzt). Wenn also Gleichheit vorliegt, wird in "Test" die entsprechende Kombination im 4., 5.ten gesetzt und damit der Offset für die Zustandstabelle gebildet.

```
MACRO Form              \ A: N
                        \ R: ret ret
                        \ D: -2 -2 Zhl Akt 'Zxx 'xy
   com 8*               \ A: N  RS: ret ret
                        \ DS: 8 -2 Zhl Akt 'Zxx 'xy
   push 4* com          \ A: N  RS: ret ret 8
                        \ DS: 7 Zhl Akt 'Zxx 'xy
   push over            \ A: N  RS: ret ret 8 7
                        \ DS: Akt Zhl Akt 'Zxx 'xy
   8/ pop and a         \ A: N  RS: ret ret 8
                        \ DS: N Form Zhl Akt 'Zxx 'xy
   8/ -or               \ A: N  RS: ret ret 8
                        \ DS: F Zhl Akt 'Zxx 'xy
   pop CALL gleich?     \ A: N  RS: ret ret
                        \ DS: 8 Test Akt 'Zxx 'xy
END-MACRO
```

Gut zu verfolgen ist hier, wie wichtige Literale (hier die 7 als Maske für die drei Formbits) "on the fly" erzeugt werden, unter Umgehung des Literal-Befehls. Die Form ist in den höheren 3 Bit codiert. Also wird eine Kopie des aktuellen Steins um 3 Bit nach rechts verschoben und mit der Maske werden höhere Informationsbits gelöscht. Auch eine Kopie des Nachbarsteins im Adressregister wird 3 Bits nach rechts geschoben. Da hier aber schon sichergestellt wurde, daß höhere Bits gelöscht sind können die Formen mit "-or" verglichen werden. Daraus folgt: Das Flag zeigt 0 bei Gleichheit!

```
MACRO Farbe             \ A: N
                        \ R: ret ret
                        \ D: 8 Test Akt 'Zxx 'xy
   2/ dup push          \ A: N  RS: ret ret 4
                        \ DS: 4 Test Akt 'Zxx 'xy
   4/ com 4*            \ A: N  RS: ret ret 4
                        \ DS: -8 Test Akt 'Zxx 'xy
   com dup push a       \ A: N  RS: ret ret 4 7
                        \ DS: N 7 Test Akt 'Zxx 'xy
   and nop a! over      \ A: Fb RS: ret ret 4 7
                        \ DS: Akt Test Akt 'Zxx 'xy
   pop and a -or        \ A: Fb RS: ret ret 4
                        \ DS: F Test Akt 'Zxx 'xy
   pop CALL gleich?     \ A: -- RS: ret ret
                        \ DS: 4 Test Akt 'Zxx 'xy
END-MACRO
```

Im Gegensatz zum Formtest müssen hier auch beim Nachbarstein die höheren 3 Bit gelöscht werden. (Enthalten ja noch die Form).

```
MACRO naechster       \ A: --
                      \ R: ret ret
                      \ D: 4 Test Akt 'Zxx 'xy
   4/ com             \ A: -- RS: ret ret
                      \ DS: -2 Test Akt 'Zxx 'xy
   4* com over and    \ A: -- RS: ret ret
                      \ DS: Zhl Test Akt 'Zxx 'xy
   push nop a! over   \ A: Ts RS: ret ret Zhl
                      \ DS: 'Zxx Akt 'Zxx 'xy
   if
      a 8/            \ A: Ts RS: ret ret Zhl
                      \ DS: Offset 'Zxx Akt 'Zxx 'xy
      '+ nop a! push  \ A: Ad RS: ret ret Zhl Akt
                      \ DS: 'Zxx 'xy
      drop @a pop over \ A: Ad RS: ret ret Zhl
                      \ DS: 'Zxx Akt 'Zxx 'xy
   then
   drop pop dup dup   \ A: -- RS: ret ret Dum
                      \ DS: Dum Zhl Akt 'Zxx 'xy
   push
END-MACRO
```

Zu Beginn enthält Test den Zähler in den unteren 3 Bit und den Offset in den darauffolgenden 2. Zunächst wird der Zähler extrahiert. (Damit wird in dieser Kopie auch der Ofset auf 0 zurückgesetzt!) Ist die Zustandsadresse gleich 0, dann wird kein weiterer Zustandsübergang mehr eingeleitet, der Stein kann nicht an diese Adresse gelegt werden, aber die restlichen Nachbarn werden weiter durchlaufen, um den Zähler am Ende gebrauchen zu können. Ist es allerdings noch möglich, den Stein zu plazieren, (Zustandsadresse ungleich 0), dann wird auch der Offset gewonnen, zur derzeitigen Zustandsadresse addiert, und die alte Kopie durch den neuen Inhalt überschrieben. Zum Abschluss werden noch ein paar Dummies erzeugt, um die übergeordneten Strukturen zufriedenzustellen.

```
MACRO weiter          \ A: --
                      \ R: ret ret Dum
                      \ D: Dum Zhl Akt 'Zxx 'xy
   pop drop drop push \ A: --  RS: ret ret Zhl
                      \ DS: Akt 'Zxx 'xy
   push over nop a!   \ A: 'xy RS: ret ret Zhl Akt
                      \ DS: 'Zxx 'xy
   pop pop dup dup    \ A: 'xy RS: ret ret
                      \ DS: x x Zhl Akt 'Zxx 'xy
   -or com 2*         \ A: 'xy RS: ret ret
                      \ DS: -2 Zhl Akt 'Zxx 'xy
END-MACRO
```

Für den neuen Durchgang wird die ursprüngliche Testadresse für den aktuellen Stein wieder in das Adressregister eingetragen. Der nächste Nachbar wird anhand dieser Adresse ermittelt und geholt und zwar so:

```
MACRO links           \ A: 'xy RS: ret
                      \ DS: -2 Zhl Akt 'Zxx 'xy
   2/ dup 2*          \ A: 'xy RS: ret
                      \ DS: -2 -1 Zhl Akt 'Zxx 'xy
END-MACRO

MACRO oben            \ A: 'xy RS: ret
                      \ DS: -2 Zhl Akt 'Zxx 'xy
   dup push           \ A: 'xy RS: ret -2
                      \ DS: -2 Zhl Akt 'Zxx 'xy
   8* pop             \ A: 'xy RS: ret
                      \ DS: -2 -16 Zhl Akt 'Zxx 'xy
END-MACRO

MACRO rechts          \ A: 'xy RS: ret
                      \ DS: -2 Zhl Akt 'Zxx 'xy
   dup push com pop   \ A: 'xy RS: ret
                      \ DS: -2 1 Zhl Akt 'Zxx 'xy
END-MACRO

MACRO unten           \ A: 'xy RS: ret
                      \ DS: -2 Zhl Akt 'Zxx 'xy
   dup push com 2*    \ A: 'xy RS: ret -2
                      \ DS: 2 Zhl Akt 'Zxx 'xy
   8* pop             \ A: 'xy RS: ret
                      \ DS: -2 16 Zhl Akt 'Zxx 'xy
END-MACRO
```

Wie man sieht, wird der entsprechende Offset (-1, -16, 1, 16) auf den Next-of Stack gebracht. Nun braucht er nur noch zum Adressregister addiert werden:

```
MACRO +a              \ A: 'xy RS: ret ret
                      \ DS: -2 Ofs Zhl Akt 'Zxx 'xy
   push a nop nop     \ A: 'xy RS: ret ret -2
                      \ DS: 'xy Ofs Zhl Akt 'Zxx 'xy
   '+ nop a! pop      \ A: 'N  RS: ret ret
                      \ DS: -2 Zhl Akt 'Zxx 'xy
END-MACRO
```

Jetzt haben wir die Adresse eines Nachbarn und prüfen, ob einer vorhanden ist.

```
': besetzt?           \ A: 'N
                      \ R: ret ret ret
                      \ D: -2 Zhl Akt 'Zxx 'xy
   dup push           \ A: 'N RS: ret ret ret -2
                      \ DS: -2 Zhl Akt 'Zxx 'xy
   8* 4* com          \ A: 'N RS: ret ret ret -2
                      \ DS: 63 Zhl Akt 'Zxx 'xy
   @a and pop ;'      \ A: 'N RS: ret ret
                      \ DS: -2 N Zhl Akt 'Zxx 'xy
```

Aus der Adresse des Nachbarsteins wird der Inhalt geholt, nur die unteren 6 Bit sind gefragt, die restlichen Bit werden gelöscht. (Durch "push if" werden wir N auf 0 testen. Wenn N gleich Null ist, gehen wir direkt WEITER zum nächsten Nachbarn. Daher bei WEITER die Dummies! Ansonsten:

```
MACRO Nachbar!        \ A: 'N RS: ret ret -2
                      \ DS: N Zhl Akt 'Zxx 'xy
   a! pop dup push    \ A: N  RS: ret ret -2
                      \ DS: -2 Zhl Akt 'Zxx 'xy
   com                \ A: N  RS: ret ret -2
                      \ DS: 1 Zhl Akt 'Zxx 'xy
   '+ pop dup nop     \ A: N  RS: ret ret
                      \ DS: -2 -2 Zhl Akt 'Zxx 'xy
END-MACRO
```

Wir speichern den gefundenen Nachbarstein im Adressregister, inkrementieren den Zähler und bereiten den Stack für FORM und FARBE vor.

Nun können wir das Puzzle zusammensetzen:

```
': Nachbar?          \ A: 'xy RS: ret ret
                     \ DS: -2 Ofs Zhl Akt 'Zxx 'xy
   +a
   CALL besetzt?
   push if
      Nachbar!
      Form
      Farbe
      naechster
   then weiter ;'    \ A: 'xy RS: ret
                     \ DS: -2 Zhl Akt 'Zxx 'xy
```

Nach dem letzten WEITER sind wir FERTIG.

```
MACRO fertig           \ A: 'xy RS: ret
                       \ DS: -2 Zhl Akt 'Zxx 'xy
   drop push push nop  \ A: 'xy RS: ret Zhl Akt
                       \ DS: 'Zxx 'xy
   a! pop pop a        \ A: 'Zx RS: ret
                       \ DS: 'Zxx Zhl Akt 'xy
   dup 2/ 2* -or       \ A: --  RS: ret
                       \ DS: Flag Zhl Akt 'xy
END-MACRO
```

Hier ist der schlußendliche Test der Zustandsadresse zu finden. Soll eine Zahl auf gerade oder ungerade getestet werden, dann kommt der Test mit:

" dup 2/ 2* -or "

zum Zuge. Ich lösche das unterste Bit der Kopie und vergleiche es mit dem Original. Sind die beiden immer noch gleich, dann produziert das -or eine 0. D.h.: Ist die Testzahl gerade, erhält man als Flag eine Null, ansonsten eine eins! Nun bedeutet "gerade" aber nach Aufbau der Zustandstabelle: kein Endzustand! Also liegt genau das richtige Flag auf dem TOP. Das Flag zeigt an, ob der aktuelle Stein an die gewünschte Stelle gelegt werden kann. Der Zähler enthält die korrekte Anzahl aller Nachbarn an dieser Stelle. Der aktuelle Stein wird erhalten, ebenso die Adresse des Spielfeldes, die gerade getestet wurde.

```
MACRO Nachbarn             \ A: 'xy RS: ret
                           \ DS: -2 0 Akt 'Z00 'xy
   links  CALL Nachbar?
   oben   CALL Nachbar?
   rechts CALL Nachbar?
   unten  CALL Nachbar?
   fertig                  \ A: --  RS: ret
                           \DS: Flag Zhl Akt 'xy
END-MACRO
```

Zu Beginn ist der Zähler natürlich 0 und der Anfangszustand ist Z00.

```
': passt?                  \ A: --  RS: ret
                           \ DS: y x Akt
   CALL 'xy
   push a ' Z00 '# pop     \ A: 'xy RS: ret
                           \ DS: Akt 'Z00 'xy
   dup dup -or com         \ A: 'xy RS: ret
                           \ DS: -1 Akt 'Z00 'xy
   2* CALL besetzt?        \ A: 'xy RS: ret
                           \ DS: -2 F Akt 'Z00 'xy
   push if                 \ A: 'xy RS: ret -2
                           \ DS: F Akt 'Z00 'xy
      drop push drop pop   \ A: 'xy RS: ret -2
                           \ DS: Akt 'xy
      pop 2/ dup com  ;'   \ A: --  RS: --
                           \ DS: 0 -1 Akt 'xy
   then
   pop Nachbarn ;'         \ A: --  RS: --
                           \ DS: Flag Zhl Akt 'xy
```

Jetzt haben wir es geschafft! PASST? ist die höchste Ebene dieses Moduls. "passt?" wird mit x,y-Koordinaten und dem dorthin zu legenden Stein gerufen.
Ein Flag, ob der Stein gelegt werden kann, nebst Zähler, dem aktuellen Stein und der Adressposition wird auf dem Stack zurückgegeben. Es ist zur schnellen Erzeugung von kleinen Literalen (für Masken, zum In- oder Dekrementieren) sinnvoll, ein Literal auf dem Stack als schnelle Variable mitzuführen. In diesem Modul wird die -2 dafür gewählt.

Die Routine testet zunächst, ob der Platz schon belegt ist. In diesem Fall ist das Flag ungleich 0, und die Routine terminiert sofort. Ein negativer Zähler zeigt dies in diesem Fall an. Wird kein Nachbar gefunden, so ist das Flag ebenfalls falsch (Z00 ist gerade) und der Zähler ist 0. In jedem anderen Fall zeigt das Flag wiederum Erfolg oder Mißerfolg an, und der Zähler liefert die Anzahl der Nachbarn um diese Stelle.

Der Programm-Code ist sehr kompakt. Er belegt nur 180 Bytes! (72 Worte) In diesem ganzen Modul findet kein einziger doppelter oder unnötiger Speicherzugriff statt. (Zusätzlich finden selbstverständlich alle nötigen Zugriffe on-page statt.) Alle Variablen werden auf dem Stack gehalten. Die Verwendung von nur 3 initialen Literalbefehlen (natürlich nur außerhalb jeglicher Schleifen) spricht für sich. Sicher ist es aufgefallen, daß dieser Beispielcode kein ... ELSE ... Konstrukt verwendet. Für diese Art der Programmierung hat es sich als unnötig herauskristalliert. Ebenso kommt man bestens mit ganz einfachen Schleifen (BEGIN ... (-)UNTIL) aus.

MuP21 führt Programme dieses Stils extrem schnell aus.

Soeren Tiedemann

Jeff, slowly I got an idea of what you mentioned about having fun with all these small routines and playing around with them.

Yes, this is FORTH!!!

Thanks a lot for your support!

Soeren

Bericht zur Tagung

Forthtagung 2000 in Hamburg

Die diesjährige Forthtagung stand unter dem dunklen Stern der Zeitnot in der sich der Organisator und Einlader der Tagung befand. Alles wirkte, vor allem in der Erinnerung an die Tagung in Oberammergau, ein wenig unorganisiert und *einfach*. Trotzdem war die Tagung für die rund 20 angereisten Forther ein Erfolg und keineswegs unangenehm.

Seit einigen Jahren wird das Angebot, einen Tag früher anzureisen und sich am Tagungsort umzusehen, von einer wachsenden Teilnehmerzahl gerne angenommen. In diesem Jahr haben einige Mitglieder und ihre Begleitungen die Möglichkeit genutzt, sich bei einer Hafenrundfahrt und einem anschließenden Stadtbummel einen ersten Eindruck von Hamburg zu verschaffen.

Der Tagungsgort im Hamburger Stadtteil Rissen firmiert unter der Bezeichnung “Internationales Institut für Politik und Wirtschaft”. Immerhin scheint dort der Club of Rome gelegentlich zu Gast zu sein, was mich allerdings wundern würde, weil neben dem in seiner Art sehr schönen Haupthaus, das leichte Spuren mangelnder Pflege zeigt, ein weniger ansprechendes Gebäudeensemble den Gästen Unterkunft in unangenehm kleinen Zimmern bietet. In den als Doppelzimmer deklarierten Räumen ist der Aufenthalt von zwei Personen eigentlich nur mit Mühe und unter Einhaltung zuvor getroffener Absprachen möglich. Sitzgelegenheiten fehlen dort z.B. ganz, weil die gewöhnlich in diesen Zimmerchen bereitstehende Klappcouch als zweites Bett dienen muß. Daß dieses Möbel zum Schlafen denkbar ungeeignet ist, weil es, je nach Aufstellungsort, entweder zur Wand hin, oder, was ich als schlimmer empfunden habe, von der Wand weg abfällt, verstärkte den Eindruck, in einem Landschulheim untergebracht gewesen zu sein. Daß es des Nachts unangenehm laut im Haus war, ist dann schon beinahe verschmerzbar. Nur um das deutlich zu sagen: Ich habe nichts dagegen, in einem Landschulheim zu nächtigen, aber wo ‚international‘ drauf steht, sollte auch ‚Internationales‘ herausschauen. Nun ja, vielleicht kommt gelegentlich Besuch aus Holland...
Ganz hervorragend war allerdings die Fürsorge, mit der sich

das Personal des Hauses um unser Wohlergehen gesorgt hat ! Gleich nach der Ankunft am Donnerstag war den ersten Teilnehmern ebenso wie den Mitarbeiterinnen des Hauses Rissen klar, daß für die Tagung kaum eine Planung bestand. Aber mit kurzen Absprachen vor Ort konnte alles organisiert werden, was letztlich zum Wohlbefinden nötig ist: Räume zum gemütlichen, ungestörten Beisammensein (die Bundeswehr verkehrt dort auch), ein Getränkeangebot, das weit über das dort übliche Angebot des Bier- und Colaautomaten im Keller hinausgeht, und sogar ein kleines Buffet für die spät Ankommenden wurde bereitgestellt. Während der ganzen Tagung waren das Personal in einer charmant unauffälligen Art erfolgreich bemüht, uns den Aufenthalt angenehm zu gestalten.

In Erinnerung an den doch recht beengten Tagungsraum in Oberammergau war der entsprechende Hörsaal in Hamburg ein echter Fortschritt. Reichlich Platz und dementsprechend gute Luft, Durchlichtprojektor und ‚Beamer‘ sowie weiteres, in Tagungsräumen übliches Material standen ebenso zur Verfügung wie eine ausreichende Anzahl Steckdosen.

Daß die Teilnehmer dem Organisator zunächst nur zwei Vorträge angeboten hatten, ließ vermuten, daß während des Treffens ein ungewöhnlich großer Zeitraum für ‚freie Diskussionen‘ zur Verfügung stehen würde. Unmittelbar nach der Begrüßung darauf angesprochen, fanden sich aber doch weitere Mitglieder zu Vorträgen bereit, so daß letztlich ein insgesamt von Umfang und Inhalt her ansprechendes Programm angeboten wurde.

Zusammenfassung von Bernd Paysan, aus de\comp\lang \forth:

Bernd Paysan: "Ein Web-Server in Forth", ca. 200 Zeilen Code sind ausreichend, um HTTP/1.1 zu implementieren.

Hans Eckes: "Mini-Modul mit PSC1000". Hans hat ein Modul mit PSC1000, ROM, Flash, RAM, UART etc. gebaut, darauf ein Forth portiert, und das alles vorgeführt. Ethernet-Daughterboard gibt's schon, TCP/IP aber (noch?) nicht.
Ulrich Hoffmann stellte digitale Signaturen vor, und zwar im

Zusammenhang mit Controllern, die signierte Messergebnisse verschicken, und nicht, wie sonst üblich, im Zusammenhang mit E-Commerce-Anwendungen.

Thomas Beierlein brachte den obligatorischen Next-Vortrag, mit einem Forth auf noch einem Microcontroller.

Egmont Woizel berichtete von dem einzigen Forth-Projekt, das er letztes Jahr hatte, einer Zeiterfassung vom Open-Network-Forth, das die Taskzeiten aufzeichnet.

Martin Bitter brachte "Lego-RCX mit Forth", Hauptschüler basteln und programmieren Lego-Roboter in Forth. Mit der richtigen Anleitung ist Forth einfach kinderleicht.

Dann gab's noch einige Workshops zum guten Forth-Buch, ActiveX/COM-Komponenten mit Forth, Multitasker 3. Version von Klaus Schleisiek, und die Lego-Bausteine haben wir natürlich auch auseinandergenommen.

Detaillierte Berichte zu den Inhalten der Vorträge sollen hier bewußt nicht aufgeführt werden. Die Hoffnung ist groß, diese Vorträge im Einzelnen in dieser oder einer der nächsten VDs wiederzufinden.

Die Mitgliederversammlung am Sonntag begann, wie immer, mit der Verlautbarung des Drachenrates über diejenige Person, die der Rat für das bis zur nächsten Tagung folgende Jahr als Träger unserer Auszeichnung auserkoren hat. Fred Behringer, bekanntermaßen in 1999 mit dem SWAP beehrt, teilte der Versammlung mit, daß in 2000 Egmont Woitzel insbesondere für seine Arbeit als Webmaster der FG ausgezeichnet wird.

Bereits während der Tagung waren die anwesenden Mitglieder aufgefordert, über die ausgestellten LOGOs des Wettbewerbs der Forthgesellschaft abzustimmen. Von den Mitgliedern, denen die Einreicher der LOGOs nicht bekannt waren, wurde der Vorschlag von Heinz Schnitter mit großer Mehrheit zum LOGO der FG gewählt. Heinz Schnitter wird darum vom Direktorium der Gesellschaft zur Forthtagung 2001 eingeladen ! Das ist der ausgelobte Preis.

Weitere Details der Versammlung können dem Protokoll entnommen werden.

fep

Das Siegerlogo der FG !

von

Heinz Schnitter

Hallo Friedrich,

meine Güte, Du hast schon Deine 4 Jahre gedient! Meine Glückwünsche an Dich für den gut gemachten Job!

Hattet Ihr einen auf der Hamburger Forth-Tagung (außer dem Barkeeper), der nach 1990 mit FORTH begonnen hat? Wie viele, die nach 1980 mit FORTH erstmals in Berührung kamen? Nächstes Jahr feiere ich meine 10 Jahre Teilnahme an den SVFIG Treffen, und ich glaube, ich bin einer der 'Neulinge'.

Aber die 'alten Hasen' kommen weiterhin zu unseren Treffen. Beim April-Treffen waren wir über Al Krever's Erscheinen angenehm überrascht und genossen seinen Vortrag über die verschiedenen FORTH-Projekte, an denen er in den letzten 20 Jahren mitgearbeitet hat. Bei weitem das größte davon war die Installation auf dem internationalen King Khalid Flughafen in Riyadh, Saudi Arabien, mit über 35000 Steuerungsknoten.
Kleinere Projekte sahen FORTH in Schleifmaschinen der Herkules-Werke Siegen, Deutschland, und in Klimasystemen der Saturn-Werke in Troy, Michigan.

Dr. Ting hat seinen FORTH-Compiler für den Analog Devices ADSP2181 fertiggestellt, und er füllte die Pausen zwischen den technischen Präsentationen bis nach dem Mittagessen. Dann kam Skip Carter, um uns alles über die Entwicklung industrieller Anwendungen auf dem Palm Pilot zu erzählen.
Zu dieser Zeit war unsere Gruppe auf ungefähr 23 Leute angewachsen. Hast Du gewußt, daß Neil Bridges ein FORTH für den Palm Pilot entwickelt hat? Es ist scheinbar eine auf dem Netz verfügbar. Da ist genug Platz, Applikationen für den Palm zu schreiben, aber mit Code Warrior und anderen Entwicklungswerkzeugen, nicht mit FORTH.

Auch wenn die meisten SVFIG-Mitglieder Ihr Brot nicht hauptsächlich mit FORTH verdienen, scheint es mir, daß sie es genießen, zu den Treffen zu kommen - um ihre Vertrautheit mit FORTH und den zugehörigen Leuten aufrechtzuerhalten. Und der junge Kerl, der zu Zeiten des FIG Forth Teenager war, David French, war diesen Monat wieder da. Laß uns hoffen, daß die alten Burschen den jungen genügend Ausbildung und Unterhaltung bieten können.

Das ist es für jetzt, Friederich. Ich sende eine Kopie an Fred Behringer, damit er weiß, daß ich bisher nicht aus dem Cyberspace herausgefallen bin.
Glückauf,

Henry

Lieber Friederich,

Ich kann nicht über das ganze SVFIG Treffen vom 26. Februar 2000 berichten, da ich nur die ersten beiden Stunden am Morgen dort sein konnte. Diese waren, wie zur Zeit üblich,

mit Dr. Tings Bericht gefüllt, dieses Mal über ein Motorola 68HC12/11 Evaluationboard. Dr. Richard Haskell von der Oakland Universität in Michigan hatte ein Buch darüber geschrieben und eine Diskette mit einem voll ausgebauten Forthsystem beigefügt. Wir waren an die 16 Personen, die das Buch sahen und Dr. Tings Buchbesprechung hörten.

Am Nachmittag war John Carpenter mit einem Bericht über ein Spiel, welches Rick Van Norman in Swift Forth geschrieben hat, eingeplant.

Dr. Ting war wieder da, um das nächste Treffen am 25. März mit einer detaillierten Beschreibung über das, was er mit einem anderen Chip getan hatte, zu beginnen -- dieses mal den Analog Devices ADSP2181. Dieser hatte es ihm angetan, da er über 80 KB RAM verfügt -- genug um ein voll ausgebautes eForth auf ihm laufen zu lassen. Er hat die Arbeit aber momentan noch nicht beendet.

John Peters schickte die folgende Web-Adresse, auf der man ein Forth in Java online finden kann, welches von Michael Losh geschrieben wurde (die deutschen Übersetzungen sind von Dr. Behringer): http://www.amsystech.com/mlosh/#notes.

Wenn ich mich recht erinnere, hat Chuck Moore vor fünf Jahren seinen F21 Chip entwickelt und Jeff Fox hat seitdem seine Finger mit im Spiel. Dieses Mal brachte Jeff eine Leiterplatte mit einem F21 mit, welches mit einem Videobildschirm gekoppelt war und die Kugel einer Logitech-Maus als Eingabegerät nutzte. Das ist der einzige Chip, den man mit einem Mikrophon, mit Video, einer Maus usw. koppeln kann. Wenn ein Gehäuse für die Leiterplatte fertig ist, haben wir einen "High-Speed PC innerhalb einer Maus". Mehr Informationen gibt es auf Jeff's Web Seite, http://www.Ultratechnology.com.

Der Höhepunkt des Tages war die enthusiastische Präsentation von Forth durch einen Newcomer in der Gruppe, ein junger Bursche namens David Frech.
David war ein Teenager als Fig Forth zuerst auftauchte. Er hat das meiste selbst gelernt und ist heute stolzer Besitzer seines eigenen Forth für Linux.

Friederich, ich bin beeindruckt, wie Du in der Lage bist, die Vierte Dimension zusammenzustellen; und daß Deine Frau die Herzensgüte hat, um den harten Job des Korrekturlesens zu machen. Meine Kopie kam etwas zu spät, um sie Skip Carter auf dem letzten Treffen zu zeigen. Ich wollte Ihm speziell die "Spaß mit Forth" Artikel aus vier verschiedenen Ländern zeigen. Ich konnte jedoch die Anregung geben, daß unsere eigene unpäßliche Forth Dimensions nach einem Weg suchen sollte, ihre Anstrengungen mit den engagierteren Forth Verlegern auf der Welt zu vereinen. Oder, vielleicht, sollte es nur eine Zeitung geben, und Fred Behringer kann alles (außer meinen Yankee Berichten) in alle notwendigen Sprachen übersetzen?
Cheers, *Henry*

Übersetzung: Thomas Beierlein

Alles ist Forth

Jörg Staben
Hagelkreuzstr.23,
40721 HILDEN;

Win32Forth - F95
Forth, Windows, Objekte

Einleitung

Java, Open Boot, Real3D - alles ist Forth, die erfolgreichste Programmiersprache der 90er Jahre. **Win32Forth** ist ein 32Bit-Forth System für Windows NT und Windows´95 und stellt, verglichen mit den oben genannten Spezialisten, einen klassischen Vertreter von Forth dar.

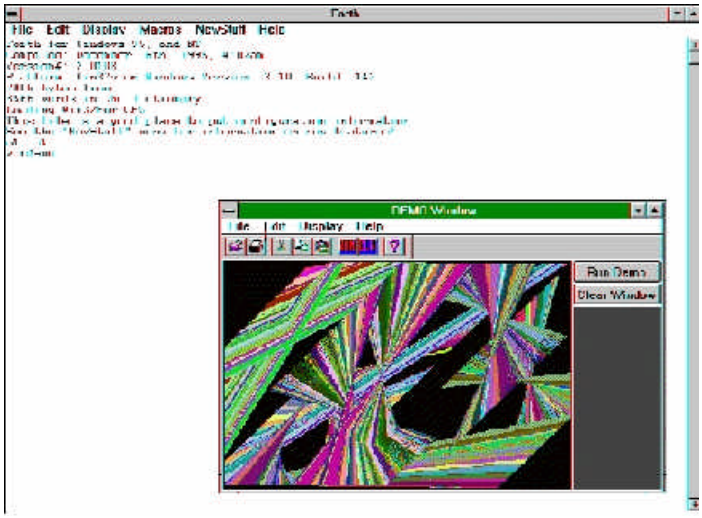
Win32Forth

Win32Forth sollte nach dem Willen seines Schöpfers Tom Zimmer eigentlich kurz und knapp **F-95** heißen, um die direkte Verwandtschaft zu seinem erfolgreichen DOS-Vorgänger F-PC Forth deutlich zu machen.

Dummerweise wollte jeder, nachdem der F-PC Nachfolger angekündigt worden war, das Archiv WIN32FOR.EXE vom Netz holen, und schon hatte **WIN32FORTH** seinen Namen weg.

Was zeichnet Win32Forth aus?

Zuerst einmal ist es - wie auch F-PC - public domain, darf al-



so ohne Einschränkungen benutzt oder verteilt werden. Davon weicht allerdings der interne 486er-Assembler etwas ab, den J.Schneider programmiert und entsprechend dem GNU copyleft zur Verfügung gestellt hat.

Das Win32Forth kann Ihnen als pd-System einen großen

Vorteil anbieten: Alle Quelltexte des gesamten Forth und seiner Dienstprogramme werden mitgeliefert und können mit einem - ebenfalls mitgelieferten MetaCompiler - zum Win32Forth compiliert werden.

Dadurch läßt sich das Win32Forth ändern, verbessern, ganz auf Ihre Wünsche maßschneidern.

Windows und ANSI-Forth

Seit 1983 wurde die Forth-Welt vom informellen Forth-83 Standard regiert. Dieser Standard war zwar über ein Jahrzehnt tragfähig genug für eine Vielzahl von Anwendungen, ließ aber noch viele Fragen offen und wurde den aktuellen Entwicklungen in Forth nicht mehr gerecht. Forth-Prozessoren wie HARRIS RTX bieten ganz andere Möglichkeiten wie Programmausführung während des Compilierens, verlangen aber auch eine dementsprechende Programmierung. All diesem versucht der 1995 vom amerikanischen Institut für Normung ANSI verabschiedete Forth-Standard gerecht zu werden, indem er das Verhalten, nicht aber den Aufbau eines Forth festlegt.

Zur Freude der Forth-Puristen, zum Leid der eingeschworenen F-PC-Anhänger ist Win32For ANSI-kompatibel. Ein entsprechender Konformitätstest findet sich unter den Code-Beispielen; die Abweichungen vom Standard werden dokumentiert - man kann also mit entsprechender Disziplin portable Forth-Programme schreiben oder ANSI-Tools direkt nutzen!

Als ANSI-Forthsystem verfügt Win32For zwangsläufig über die langersehnten lokalen Variablen, die portable Datei-Schnittstelle zum Betriebssystem und über das oft unterbewertete Blockinterface zum Massenspeicher.
Glücklicherweise besteht das Win32Forth nicht nur aus den Wörtern des ANSI-Wortschatzes, denn in einem Forth-Programm für Windows ist nur noch wenig Forth, aber sehr viel Windows zu finden.
Da ist es wichtig, daß eine vollständige Schnittstelle für alle Windows95-Prozeduraufrufe und -Konstanten vorhanden ist und auch die Ankopplung an z.B. das Windows-Hilfesystem bereits fertig ist. Die Parameterübergabe an DLL´s läuft exakt umgekehrt zur Windows API32 Hilfe-Datei, so daß Windows-Aufrufe wirklich einfach durchzuführen sind.

Als Lohn für die Mühen des Umstiegs von F-PC nach Win32For finden Sie in `UTILS` neben vielen liebgewordenen F-PC-Wörtern auch solche Wörter wie `NT?`, `Win95?` und `Win32s?`, die zur Bestimmung der Systemplattform gebraucht werden.

Diese machen dann manchem Nicht-NT-Benutzer deutlich, daß nicht auf jeder Plattform alles zur Verfügung steht: So muß Win32Forth schon unter WindowsNT laufen, damit **Named Pipes**, ein Kommunikationsmechanismus zwischen Applikationen auf einem Host oder auch im Netz, zur Verfügung steht.

Auch das **Broadcasting** als ähnliches Leistungsmerkmal deutet schon an, daß sich Win32Forth an den engagierten Programmierer wendet. Broadcasting ist ein Verfahren, um einzelne Nachrichten an alle Instanzen von Win32Forth auf einem Host zu senden, wobei die einzelnen Win32Forth´e anhand spezieller Teilinformationen entscheiden können, ob sie die Nachricht auswerten müssen.

Fenster als Objekte

Die objektorientierte Programmierung wird mit Begriffen beschrieben wie Abstraktion der Daten (abstraction), Verbergen von Informationen (information hiding), Klassen (classes) und Vererbung (inheritance).
In diesem Ansatz werden Datenstrukturen und die dazugehörigen Operatoren als untrennbare, gekapselte Objekte aufgefaßt, über deren korrekte Handhabung der Compiler wacht. Solche Objekte vererben ihre Eigenschaften und die Zugriffsmethoden. Die interne Realisierung dieser Zusammenhänge bleibt dem Programmierer verborgen, der nicht länger mit bestimmten Operatoren auf Daten direkt zugreift, sondern Nachrichten an Objekte sendet, um ein bestimmtes Verhalten zu erreichen.
OOP ist aber keinesfalls ein Allheilmittel für funktionelle und fehlerfreie Software, sondern erst einmal ein Gedankenansatz, der besonders bei der Entwicklung großer Projekte Vorteile bringen soll.

Weil der Umgang mit Windows für den Programmierer eine recht mühsame Aufgabe ist, ist Win32For objektorientiert!

Das OO-Model des Win32Forth orientiert sich an einer OO-Sprache namens NEON, die auf dem MacIntosh eine gewisse Bekanntheit erlangt hat. Dadurch wird der Umgang mit den alles bestimmenden, gehaßten und geliebten Fenstern deutlich vereinfacht.

So verfügt ein neues Fenster als Objekt bereits über einen Satz von Methoden für die Standard-Aktionen, die so ein Fenster auszuführen hat.
Damit ist ein erstes eigenes Fenster schnell gemacht, das sich dank Forth-Interpreter auch interaktiv ausprobieren läßt:

```
:Object mywindow <Super window
       :M WindowTitle:
          z" My Own Window"
       ;M
;Object

start: mywindow
close: mywindow
```

Hier verhält sich `mywindow` genau wie jedes andere Objekt, das von der Klasse `WINDOW` abgeleitet wird, aber es hat - als Methode - den exclusiven Titel "My Own Window".

In Win32For können Objekte aber auch von anderen Objekten erben; es wird sogar die gleiche Syntax verwendet mit

dem Namen des vererbenden Objektes statt des Namens der vererbenden Klasse.

In diesem Fall dient das vererbende `MYWINDOW` selbst als Klasse:

```
:OBJECT yourwindow <Super mywindow
        :M WindowTitle:
      z“ Your Window"
        ;M
;OBJECT
```

Auch Menüs und Dialog-Boxen sind Klassen, um sie leichter anlegen und verändern zu können. Zudem unterstützt Win32For Anwender-definierte Button- und Menüleisten.

Das ganze OO-Modell wird an Beispielen illustriert, von denen WinView, der hypertextfähige Editor des Win32For das umfangreichste ist.

Schnelligkeit

Win32For ist schnell. Es compiliert blitzschnell, denn nur Schnelligkeit bietet Ihnen bei großen Projekten mit Quelltexten im Megabyte-Bereich noch die Interaktivität, die Forth auszeichnet.

Gerade die Programmierung unter Windows zeigt endlich wieder deutlich den Entwicklungsvorteil, den interpretierendes Forth gegenüber von Compilern wie Visual C++ hat.
Die Schnelligkeit beim Compilieren erzielt Tom Zimmer durch den Einsatz verschiedener Such-Optimierungen, die teilweise auch schon beim F-PC angewendet wurden.

In der Ausführungsgeschwindigkeit dürfte sich Win32For mit Visual BASIC vergleichen lassen, wobei die Schnelligkeit der Programmausführung noch durch den integrierten 486er-Assembler gesteigert werden kann. Dort setzt allerdings der Programmierer die Grenzen, denn optimierten 486er Code zu schreiben, ist nicht mehr trivial.

Speicher

Ein einziger linearer 32Bit-Adreßraum für Programm und Daten (flat memory model) sorgt dafür, daß Sie sich endlich von den liebgewonnenen Seg:Off-Adreßberechnungen trennen können. Zugleich steht Ihnen mehr freier Speicher zur Verfügung, als Sie sich je bei einem 16Bit-Forth erträumt hätten. Dies schlägt sich in über 3500 Forth-Wörtern (ohne die fast 2000 Windows-Konstanten!) nieder, die gelernt und benutzt werden wollen.

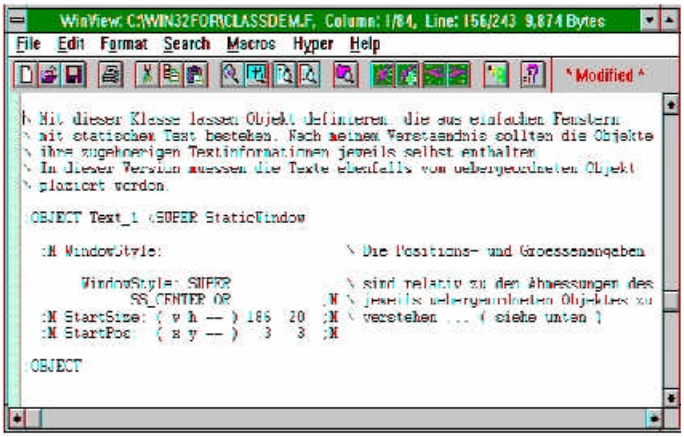
Komfortabel

Win32For kommt mit einer integrierten Entwicklungsumgebung (IDE) ins Haus: Diese umfaßt natürlich den Forth-Compiler selbst, den passenden Decompiler, einen 486er Assembler mit dem zugehörigen Disassembler, einen hypertextfähigen Editor und - weil wir alle Fehler machen - einen Debugger zur Fehlersuche.

Mit Forth arbeiten bedeutet, seine Lösung interaktiv im Dialog mit dem Forth-Interpreter zu entwickeln.
Damit ist eine gepufferte, editierbare Kommandozeile ein Muß, daß von Win32For mit Bravour erfüllt wird: Alle Tastatureingaben werden aufgezeichnet, stehen für die nächte Aufgabe zur Verfügung oder können vom System direkt als Quelltext verarbeitet werden. Damit nähert sich Win32For wieder einem Ideal aus den Anfangszeiten von Forth, als man davon träumte, eine Anwendung ausschließlich aus fehlerfreien, getesteten Modulen zusammenzusetzen. Dies ist mehr als Komfort, eher eine neue Dimension der interaktiven Programmentwicklung.

Tippfehler werden vermieden, indem man gerade die unzähligen Windows-Konstanten per Mausklick eingeben kann. Tastatur-Makros ermöglichen automatisierte Testläufe und vereinfachen die Eingabe.

Fehlt noch was?



Läuft Windows NT oder Windows 95 auf Ihrem Rechner, fehlt Ihnen zu einer vollständigen Win32For-Anwendung nur noch viel Fleiß, um die Dialog-Boxen für Ihr Programm von Hand zu machen. Oder alternativ eine Visual C++ 4.0 Entwicklungsumgebung für einen Ressourcen-Editor wie RESEDIT. Die FORTHDLG-Dateien in Win32Forth sind mit RESEDIT - einem Bestandteil des Microsoft Developers Toolset MS-SDK32 - entworfen und compiliert worden. Wenn man für Win32/NT/95 programmiert, ist umfangreiche Dokumentation darüber unerläßlich, da lohnt sich schon ein Abo der MS-Development-CD.

Möchten Sie Win32For auf der Kombination DOS/Windows 3.1 betreiben, ist dazu ein kleines Microsoft-Päckchen namens Win32s notwendig, das dem 16-bittigen Windows 3.1 zu einer 32Bit-Programmierschnittstelle verhilft.
Die OS/2-Anwender unter Ihnen können sich leider nicht auf ihr mitgeliefertes VWIN32S verlassen, auch Sie müssen Win32s installieren. Aber was tut man nicht alles für ein gutes Forth...

Fazit

Einer der wenigen Kritikpunkte wäre die noch fehlende Dokumentation, die - wenn Sie kommt - englischsprachig sein

wird. Dies wird zum großen Teil durch die Konformität mit dem gültigen ANSI-Standard und durch die Verfügbarkeit aller Systemquelltexte aufgewogen.

Sicher ist: Win32Forth ist ein umfangreiches leistungsfähiges Forth-System, dem kaum kommerzielle Konkurrenz gegenübersteht. Auch wenn man nicht direkt Windowsprogrammierung machen muß oder möchte, ist Tom Zimmers neueste Schöpfung ein rundherum gelungenes Forthsystem.

Literatur

Spricht man über Literatur zu Win32Forth, muß man drei Aspekte berücksichtigen:
Zum einen ist Win32Forth ein Programmiersystem für Windows(32), Sie benötigen also einen Leitfaden zur Windows32-Programmierung und eine Referenz der Win32-API. Ein mögliches Programmierhandbuch ist “teach yourself WINDOWS´95 PROGRAMMING in 21 days”, C.Calvert, SAMS Publ., ISBN 0-672-30531-3; eine mögliche API-Referenz ist “WIN32 API - programmer´s reference”, M&T, ISBN 1-55851-427-9. Beide Bücher sind in englischer Sprache verfaßt.

Zum anderen ist Win32Forth ein ANS-Forth-System - der ANS-Standard zur Forth-Programmierung darf also nicht fehlen. Der ANS-Standard ist englisch als DPANS6 elektronisch verfügbar.

Und zu guter Letzt hat Win32Forth eine Geschichte: Erwachsen aus den Ursprüngen des F83 von Laxen&Perry, gereift im Speicherkorsett des PC zum F-PC hat es eine Reihe von Eigenarten, Besonderheiten, Features, die auch beschrieben sein wollen. Eine Reihe solcher Beschreibungen - ebenfalls in englisch - findet sich im Lieferumfang des Win32Forth.

jst

Anmerkungen des Redakteurs:

Dieser Beitrag von Jörg Staben hat unverkennbar eine Weile bei ihm auf der Halde gelegen. Der Beitrag kommt spät – aber bei weitem nicht zu spät. Win32For lebt, wird ständig weiterentwickelt und erfreut sich einer wachsenden Gemeinde zufriedener Nutzer.
An dieser Stelle wäre es sehr interessant und sicher für viele Forther hilfreich, den aktuellen Leistungsstand von Win32For zu erfahren. ***Fühlen Sie sich aufgefordert, hierzu einige Zeilen (oder ganze Seiten) an die Redaktionsadresse zu senden.***
Ganz aktuell ist zum Beispiel die Frage, ob sich mittlerweile auch unter Windows95/98 DLLs mit dem Win32For erstellen lassen, oder ob man dazu nach wie vor WinNT als System benötigt.

fep

Fortsetzung von Seite 11

Also: Bis 2008 wird Forth mindestens (über)leben! Und das mit den phänomenalen Kräften kennen wir ja ;-)

Gruß Martin

Betreff: Re: Film und Forth
Von: Bernd Paysan <bernd.paysan@gmx.de>

...
Na ist doch klar, daß das in Forth ist, sonst könnte man das ja nicht Online machen. Und ehrlich: Wenn aus dem "Verdummungskasten" Forth-Zeilen flimmern, kann der doch gar nicht so dumm sein ;-).
Wahrscheinlich werden 2008 solche Chips von Point&Click-Programmierern aus Indien programmiert. Aus Indien, weil die fest dran glauben, daß man nach dem Tod im selben Level nochmal von vorne anfangen kann ;-).
...

Bernd Paysan

RCX –Ecke

Literatur

Ein einführender Artikel (9 Seiten, deutsch) über das "Robotics Invention System 1.5" mit zahlreichen Abbildungen findet sich in der Zeitschrift Elektor 4/2000 und 5/2000. Für den unbedarften Einsteiger zu viele spezielle Ausdrücke, die Vorkenntnisse erfordern, für den Programmiervorgebildeten, wie wir in der FG, zu sehr mit Nebensächlichkeiten beladen und zu oberflächlich. Aber 13 sehr viel versprechende kommentierte Internet-Adressen. Merke: Wissen ist nur ein Teil des Verstehens. Wirkliches Verstehen kommt erst mit der praktischen Erfahrung (Dr. Seymour Papert, Professor für Lernprozeßforschung am MIT).

beh.

Selbstbau

54 Druckseiten (HTML-Datei) darüber, wie man den "Programmable Brick" (den programmierbaren Lego-Baustein) auseinandernimmt und an seine Interna (einschließlich Opcodes und Unterprogramme des "Betriebssystems", die über "Reverse Engineering" zerpflückt wurden) herankommt, findet man unter http://graphics.stanford.edu/~kekoa/rcx/ . 4 Schrauben lösen und zwei Batterielaschen so heraushebeln, daß nichts zerbrochen wird ... In welcher Welt leben wir eigentlich, daß solche mit 450,- DM restlos überteuerten Dinger (meine Schätzung: DM 50,-) nur mit einem Trick auseinandergenommen werden können? Nur um der Informationsverheimlichung ein Schnippchen zu schlagen und mal schnell nachzusehen, welcher Baustein eigentlich drinsitzt. ... den Nippel durch die Lasche ziehen ... Für DM 450,- hätte ich doch eigentlich einen Schaltplan verlangen können, so wie er jedem besseren Fernsehapparat beiliegt, und ein Datenblatt. Und das ist doch bestimmt nicht der erste MC, der für Roboter eingesetzt wird. Aber ein ungeheurer Marketing-Aufwand! Lego hat eine eigene Abteilung dafür hier in München.

beh.

„Nach Hamburg“ haben sich eine ganze Reihe von Mitgliedern einen LEGO-RCX zugelegt. Haben Sie Interesse an einer regelmäßigen RCX-Ecke ? Dann schreiben Sie dafür...

fep

Auch Mathematiker können bissig sein - eine Buchvorschau -

Fred Behringer
behringe@mathematik.tu-muenchen.de
Planegger Str. 24, 81241 München

Soeren Tiedemann hat in der VD 1/00 eine Glosse veröffentlicht, in der Seitenhiebe auf verschiedene Berufssparten ausgeteilt werden und auch die Mathematiker nicht zu kurz kommen. Einen echten Forthler berührt das nicht. Ein echter Forthler kann alles, fast alles, und er läßt sich nicht gern in Kategorien einteilen. Individualisten sind wir, sagt der Fritz, unser Redakteur.

Unser Freund, M. Pilot aus Berlin, empfiehlt den "Wirtschaftlern", ein Wirtschaftsspiel zu entwickeln (VD 4/99). In Forth natürlich. Ein echter Forthler fragt nicht lange, ein echter Forthler geht hin und macht es. Und wenn er fünf Jahre Einarbeitungszeit braucht, dann arbeitet er sich eben fünf Jahre lang ein. In der Taschengeldserie von Top-Ware kann man sich für DM 9,95 unter dem Stichwort "Futuristische Wirtschaftssimulation" ein Wirtschaftsspiel auf CD unter dem Namen "Virtual Corporation" kaufen. Vielleicht könnte man das nach Forth übertragen? Das würde dann wahrscheinlich auch ohne Wirtschaftler gehen.

Wir konnten kürzlich in einer Übersetzung den sehr interessanten Artikel von Hugh Aguilar über das Knacken von Geheimcodes lesen (VD 1/00 und VD 2/00). Hugh Aguilar hat einen Bruder. Der ist Mathematiker und befaßt sich kritisch und herzerfrischend provozierend mit den Wirtschaftswissenschaftlern und ihren kaum drei Grundgleichungen, über die sie Kapitel um Kapitel diskutieren, ohne je zu einer Lösung zu gelangen. Er stellt eine eigene, axiomatisch begründete Wirtschaftstheorie auf. Irgendwie hatte ich mal die Idee, eine axiomatische Forth-Theorie zu entwickeln. So wie man in der Vorlesung einen Vektorraum axiomatisch einführen würde. Das braucht ja dann nicht gleich eine Theorie zu sein, die ernstgenommen werden will. Ich bin jedenfalls gespannt darauf, das Buch von Victor Aguilar mal in die Hände zu bekommen. Hier ist, was er selbst darüber sagt:

==========

AXIOMATISCHE WIRTSCHAFTSTHEORIE

von **Victor J. Aguilar**

Ich habe ein Buch geschrieben, das den Titel "Axiomatische Wirtschaftstheorie" trägt. Dieses Buch behandelt eine neue Theorie der Wirtschaftswissenschaften. Es stellt nicht einfach eine vereinfachte Version des heutzutage üblichen wirtschaftlichen Denkens dar. Es sagt nicht die Zukunft voraus, macht keine Angaben über Wohlstand oder Ruin in Amerika. Es ist ganz bestimmt nicht von der Art der "Wie-werde-ich-Kaufmann?"-Bücher, und es erhebt keinen Anspruch darauf, dem Leser zu erklären, wie er unter Ausnutzung und Inanspruchnahme der heute üblichen Wirtschaftsinstitutionen Geld verdienen kann. Es ist eine abstrakte Abhandlung. Das Buch verfolgt den Zweck, der Theorie der Wirtschaftswissenschaften eine axiomatische Grundlage zu verschaffen. Es ist wohlbekannt, wie erfolgreich die von Euklid (in der Geometrie), Kolmogoroff (in der Wahrscheinlichkeitsrechnung) und anderen (auf anderen Gebieten) angewandte axiomatische Methode ist, und ich möchte behaupten, daß sich derselbe Erfolg auch in den Wirtschaftswissenschaften erreichen läßt. Ich erkläre die Schaffung von Wohlstand zum Ziel der Wirtschaftswissenschaft, und nicht die Verteilung knapper Vorräte. Damit verschaffe ich ihr nicht nur eine solide Grundlage, sondern verschiebe auch die Paradigmen, auf denen sie beruht. Ich gehe aus vom Begriff "Preis und Vorrat". "Angebot und Nachfrage" funktioniert nicht. Damit weiche ich von den Hauptströmungen des heutigen Wirtschaftsgeschehens wesentlich ab, ein Schritt, der mit dem von Kopernikus in der Astronomie vergleichbar ist.

Ich behaupte, daß die Zukunft der westlichen Zivilisation nur dann gesichert sein wird, wenn man sich dazu bequemt, die Theorie der Wirtschaftswissenschaften von Grund auf neu zu gestalten. Die neue Wirtschaftstheorie muß ganz unten anfangen, sie muß das "Angebot-und-Nachfrage"-Denken zugunsten einer "Preis-und-Vorrat"-Haltung aufgeben, und sie muß von wohldurchdachten Axiomen ausgehen, von denen man Sätze im streng mathematischen Sinn herleiten kann. Es reicht nicht aus, Lippenbekenntnisse für eine axiomatische Behandlungsweise abzugeben und kompliziert aussehende Gleichungen in Fachzeitschrift-Artikeln einzustreuen; Gleichungen, die man lediglich aus Mathematik- oder Physikbüchern zusammengetragen hat. Will man, daß die westliche Zivilisation überlebt, dann muß man für eine Revolution der Theorie des Wirtschaftsgeschehens sorgen, eine Revolution, die wesentlich tiefer greift als die sogenannte Revolution von Keynes.

Die Wirtschaftswissenschaftler selbst sind unfähig, einen Wandel herbeizuführen. Mit dem Schreiben von zehn Seiten langen und von staatlichen Organisationen unterstützten Fachberichten, ihrer einzigen Beschäftigung, ist es nicht getan. Eine derartige Revolution kann nur aus der reinen Mathematik heraus kommen. Ich bin davon überzeugt, daß sie kommen wird, und das allein ist der Grund, weshalb ich nicht von einem Untergang der westlichen Zivilisation rede.

Die in den strengen Wissenschaften liegenden Möglichkeiten sind schwindelerregend. Den Ingenieuren und Technikern kann man nicht genug Anerkennung zollen für ihre Leistungen bei der Entwicklung der modernen Technik, und das trotz der MBAs, unter denen sie gewöhnlich arbeiten müssen. Ein Glück, daß es Ingenieure gibt. Wirtschaftler hätten uns längst

in den Ruin getrieben. Ohne eine neue Wirtschaftstheorie werden Rezessionen in Depressionen übergehen und die westliche Welt wird wieder ins Mittelalter zurückversetzt werden, auch wenn die Techniker und Ingenieure, die Biologen weiterhin in die innersten Geheimnisse des Lebens eindringen und die Widerstände erforschen, die sich ihm auf diesem Planeten entgegenstellen.

Was ich an mathematischem Grundwissen voraussetze, übersteigt die Kenntnisse der meisten Fachleute aus den Wirtschaftswissenschaften (drei Semester Differential- und Integralrechnung, ein Semester Wahrscheinlichkeitsrechnung und zur Vertiefung noch ein Semester Reelle Analysis), sollte aber sehr wohl in Reichweite eines Maschinenbauingenieurstudenten höheren Semesters liegen. Die zuletzt genannte Gruppe ist genau die, für die ich mein Buch geschrieben habe. Ich bin Mathematiker, und die meisten meiner Freunde an der Universität sind Physiker oder kommen aus den Ingenieurwissenschaften. So ziemlich alle von ihnen hatten im Verlaufe ihrer Ausbildung auch Betriebswirtschaft und Volkswirtschaft belegt. Mit Unlust. Sie konnten keinen Sinn darin erkennen, und die zur Sprache kommende Mathematik diente nicht dazu, Sätze herzuleiten, sondern lediglich dazu, Schlußfolgerungen zu erläutern, die längst allen klar waren. Funktionen wurden nie explizit eingeführt, so daß ein Mathematiker mit ihnen hätte etwas anfangen können. Ein f(x) nach dem anderen hüpfte in den Wirtschaftstexten herum, ohne je definiert worden zu sein, und es war zu vermuten, daß die Variablen und die Beziehungen zwischen ihnen überhaupt nicht definiert werden konnten. Meine Kollegen in den strengen Wissenschaften erkannten zwar die Mängel der "offiziellen" Wirtschaftswissenschaften, waren aber zu stark mit ihren eigenen Hauptfachrichtungen beschäftigt und nahmen sich nicht die Zeit, eine vollständige Theorie der Wirtschaftswissenschaften zu entwickeln. Ohne eine eigene Theorie war es ihnen zuwider, eine Wissenschaft zu kritisieren, die nicht im geringsten an eine Axiomatisierung dachte oder höchstens Lippenbekenntnisee für ein paar "Axiome" abgab, die sie dann nie dazu verwendete, irgendetwas zu beweisen. Es ist schwer, einer Wissenschaft logische Fehler nachzuweisen, wenn sie überhaupt keine Logik enthält. Beim Kritisieren der Wirtschaftswissenschaften konnte man leicht ausgleiten, und die wenigsten Kritiker überlebten. Trotzdem sollte man die Stimmen der Kritiker nicht überhören. Wenn Anfänger ein bestimmtes Gebiet einhellig ablehnen, dann sollten die Fachleute aufhorchen. Es ist viel wahrscheinlicher, daß das Gebiet unanschaulich dargestellt wurde, als daß alle Anfänger begriffsstutzig sind. Viele solche Anfänger sind führend auf ihrem eigenen Gebiet, nicht zuletzt auch in den strengen Wissenschaften.

Die Fachleute auf dem Gebiet der Wirtschaftswissenschaften vernebeln das, was sie schreiben, absichtlich. Einzig und allein ihre Anhänger können entziffern, was da eigentlich gefordert wird, meistens einfach nur mehr Staat im Wirtschaftsgeschehen. Den Gegnern des schreibenden Wirtschaftswissenschaftlers wiederum gelingt es nicht, ihn sofort und entschieden zu widerlegen, da sie in dessen Arbeit keinen einzige Spruch finden können, der klar genug geschrieben wäre, um herauszubekommen, über was er eigentlich spricht. Ich bin kein Fachmann auf dem Gebiet der Wirtschaftswissenschaft. Ich habe meine eigene Theorie entwickelt, und ich bin bereit, sie klar und deutlich jedem mitzuteilen, der hören will. Das größte Hindernis bei der Verbreitung dieses Buches sind die vielen Amerikaner, die bei der bloßen Nennung des Namens "Wirtschaftswissenschaft" in Buhrufe ausbrechen, bloß weil sie an die schrecklichen Erlebnisse mit diesem Fach während ihrer Ausbildung zurückdenken. Ich werde ganz bestimmt keine Vernebelungstaktik anwenden, um diese Leute zu erreichen.

Das einzige, was ein Student der Ingenieurwissenschaften einem Buch über Wirtschaftswissenschaften willig zugesteht, ist Kürze. Ich habe versucht, meine Darstellung so knapp wie möglich zu halten. Vom Leser setze ich voraus, daß er sich selbst motivieren kann. Wenn er das nicht kann, ist das Buch nicht für ihn bestimmt, egal, wieviele schöne Bilder es enthält, und wenn er es kann, wird er die Kürze begrüßen. Ich hege die Hoffnung, daß auch Studierende aus den strengen Wissenschaften Zeit finden werden, das Buch zu lesen. In der Einleitung findet der Leser eine vereinfachte Darstellung meiner Theorie und einen Hinweis auf den Grad an Mathematik, die verwendet wird. Die ersten drei Kapitel liefern eine Grundlage für die Wirtschaftswissenschaften und können von den weniger philosophisch orientierten Lesern übersprungen werden. Sie enthalten keine Mathematik. Das vierte Kapitel behandelt so ziemlich den gleichen Stoff wie die Einleitung, aber gründlicher und ohne das vereinfachende Axiom, das es der Einleitung ermöglicht, ohne die Relle Analysis auszukommen. Das Kapitel 5 kommt ohne Mathematik aus und konzentriert sich hauptsächlich auf Geldfragen. Im Anhang C werden makroökonomische Fragen der gängigen Art besprochen. Der Stoff in Anhang C hat mit dem eigentlichen Text des Buches nichts zu tun. Ich habe ihn hier nur deshalb eingefügt, weil ich es für wichtig halte, zunächst einmal meine Vertrautheit mit der üblichen Behandlungsweise aufzuzeigen, bevor ich meine eigene Theorie entwickle. Die heutzutage übliche Behandlung makroökonomischer Fragen ist im Grunde genommen recht einfach. Es sind einfach nur drei Gleichungen in drei Unbekannten zu lösen. Leider können die landläufigen Lehrbücher nur zweidimensionale Graphiken drucken. Und so wird Semester um Semester damit vergeudet, um die Probleme herumzureden, anstatt einfach nur zu sagen, wie die Lösung aussieht. Außerdem haben die Leute die Angewohnheit, ihre Maßeinheiten durcheinanderzuwürfeln. Ich werde diese Probleme ausbügeln und die gesamte bestehende Theorie in einem einzigen kompakten Abschnitt darstellen, so daß man sie verstehen und mit meiner eigenen Theorie vergleichen kann.

==========

Soweit die Vorschau auf das Buch von V.J. Aguilar, die man im Original unter http://www.axiomaticeconomics.com/ nachlesen kann.

Ein neues Codewort

Die E-Mail-Adresse des Autors lautet: *haguilar@dancris.com*

Und nun noch ein Zusatz in eigener Sache. Die obige Übersetzung ging über den großen Teich hin und her. Zu Victor Aguilar mit "ae" usw., zu Fritz mit "ä" usw.. Seit 40 Jahren beschäftige ich mich mit dem Umlautproblem, mal so, mal so. "Wo liegt das Problem?", wird vielleicht ein allzu Eilfertiger ganz schnell und hintergründig fragen. Was mache ich, darf ich entgegnen, wenn mein "Programm" den Ausdruck "neue Schreibweise" in "neü Schreibweise" rückwandelt, oder "bei den geringen Massen der Elektronen und Positronen ist die Auslenkung entsprechend stark" in "bei den geringen Maßen ... " usw ? In der deutsch-englischen und amerikanisch-deutschen Zusammenarbeit zur Gestaltung der VD habe ich es neuerdings wieder verschärft mit dem Umlautproblem zu tun (E-Mails mit Umlauten nach Amerika? Haben Sie das mal versucht?). Seit 15 Jahren lese ich die VD. Warum schreibt keiner über das Umlautproblem, ein echt europäisches Problem? Natürlich geht das über die reine Meisterschaft im Programmieren hinaus. Mehr Interdisziplinarität wird neuerdings von den Bildungsministern verlangt. Als ob das neu wäre! Ein echter Forthler ist von Hause aus interdisziplinär. Sonst hätte er sich nicht Forth verschrieben. Einen kurzen, schnellen und genialen Algorithmus brauche ich, keine Datenbank mit langwierig einzugebendem Kontext-Wörterbuch. In meinem Kopf hat sich natürlich längst ein Algorithmus festgenistet. Für den muß ich aber immer zwei Stunden Ausführungszeit opfern. Wahrscheinlich sucht der in meinem Gehirn auch in irgendeiner Form von Datenbank herum, wenn auch sicher assoziativ, aber immerhin.

Der Autor, von dem ich hier spreche, selbst Mathematiker, haut auf die Wirtschaftler (imma uff die Kleenhn, und imma uffn Kopp - Busch): "Die haben keine Ahnung von Mathematik". Andererseits ist aber auch die beste Mathematik nichts wert, wenn sie sich nicht an der Suche nach Lösungen für praktische Probleme orientiert - und die professionellste Programmierung nichts, wenn sie sich nicht mit der real existierenden Wirklichkeit auseinandersetzt.

Ich hätte in der VD gern ein brauchbares Prográmmchen zur Lösung des Umlautproblems gesehen!

beh

Anm. d. Red.

Was für ein Algorithmus sich bereits in Freds Hinterstübchen eingenistet hat, würde ich natürlich nur zu gerne wissen. Und eine brauchbare Lösung für das Umlauteproblem wäre für den Editor der VD eine große Hilfe. Bei manchen Beiträgen gehen viele Arbeitsstunden dafür drauf, Umlaute zu konvertieren. Und dann übersieht man meistens doch noch welche... Ein solches Programm sollte allerdings ‘allgemeinverständlich‘ geschrieben und relativ mühelos zum Beispiel in BASIC reproduzierbar sein. Dann könnte das Visual Basic ‚hinter‘ dem Word diese Arbeit übernehmen.

fep

Größter gemeinsamer Teiler, ohne Division, für ZF und Turbo-Forth, in 32-Bit-Breite

Fred Behringer
behringe@mathematik.tu-muenchen.de
Planegger Str. 24, 81241 München

Stichworte: Rationale Arithmetik, gemeine Brüche, divisionsloser GGT, ZF, Turbo-Forth, 32-Bit-Erweiterung für 16-Bit-Assembler

Der vorliegende Artikel behandelt eine mathematische Frage. Auf unserer diesjährigen Tagung in Hamburg war viel von Prozessoren die Rede. Entwirft man einen Prozessor, so braucht man Optimierungsverfahren. Am einfachsten nach geeigneter Linearisierung. Die lineare Optimierung kann man so formulieren, daß nur Additionen und Multiplikationen vorkommen. Bei den Koeffizienten der Praxis kann man sich ohne weiteres auf rationale Zahlen, also gemeine Brüche, beschränken - wenn man nicht gleich Fuzzy-Arithmetik einführen möchte. In den üblichen Forth-Implementationen kommen gemeine Brüche nicht vor. Es gibt aber Forth-Literatur, die sich mit rationalen Zahlen beschäftigt. In der Forthwrite 105 sind vier Arbeiten zu diesem Thema ("vulgar numbers") aufgeführt.

Wenn man nun aber schon mit rationalen Zahlen rechnet, so ist nicht einzusehen, warum man zwischendurch die ganze Rechnerei durch Einsatz von Gleitkomma-Technik und künstlich erzeugten Akkumulationsfehlern verfälschen soll. "Exaktes Rechnen" heißt das Stichwort in der Literatur.

Zähler und Nenner werden irgendwie im Speicher festgehalten. Schrittweise (z.B. beim Simplexverfahren) werden sie immer größer, unerträglich groß. "Kürzen" heißt das Stichwort. Kürzen durch was? Durch den größten gemeinsamen Teiler. Und damit bin ich beim Thema des vorliegenden Aufsatzes.

Der Aufsatz wurde in der Forthwrite 106 veröffentlicht. Die vorliegende Version ist eine um den Vorspann erweiterte deutsche Fassung.

Es wird viel Geschrei gemacht um Lehrmaterial für den Anfänger. Wie sieht aber dieser Anfänger eigentlich aus? Es laufen da viele "Anfänger" herum, die in Wirklichkeit gar keine Anfänger sind. Anfänger sind sie nur in bezug auf Forth. Ich meine, das beste "Lehrmaterial" ist ein kleines Programm, das der Anfänger ausprobieren und analysieren kann.

Das vorliegende Programm ist ein echtes "Spaßprogramm"

im Sinne von Hugh Aguilar (siehe Forthwrite 105). Natürlich verfolgt es einen bestimmten Zweck, aber es hat Spaß gemacht, es zu schreiben, und niemand hat mich beauftragt, das zu tun.

Das gleich folgende Programm ist eine Forth-Aufbereitung eines "Programms", das in algorithmischer Form in dem Buch "Prime Numbers and Computer Methods for Factorization" von Hans Riesel (Boston, Stuttgart, 1985) angegeben wurde. Es soll den größten gemeinsamen Teiler zweier 32-Bit-Ganzzahlen ermitteln. Das Besondere an diesem Programm liegt darin, daß es keine Division benötigt. Alles, was man braucht, wird von bloßen Subtraktionen und Bitverschiebungen geleistet. (Die Division benötigt 40 CPU-Takte, wohingegen die Subtraktion mit einem einzigen Takt auskommt. Selbst auf einem Pentium ist das Verhältnis immer noch 10 zu 1.)

Forth (ZF und Turbo-Forth) verwende ich eigentlich nur, um Vergleichsmöglichkeiten zu haben. In der Hauptsache möchte ich zeigen, wie leicht Maschinencode in ein Forth-Programm eingebettet werden kann, noch dazu 32-Bit-Maschinencode in ein 16-Bit-Forth. Portierbarkeit interessiert mich nicht. Ich arbeite mit einem PC-Kompatiblen. Der GGT ist ein grundlegender mathematischer Begriff, der kompliziert genug ist, seine Einbettung in eine CODE-Definition angezeigt erscheinen zu lassen. Ich schrecke nicht davor zurück, das Rad neu zu erfinden. Wenn mir jemand zeigen kann, wo die hier zu beschreibende Vorgehensweise in der Forth-Literatur bereits veröffentlicht wurde, stehe ich gern von Prioritätsansprüchen zurück.

Als zweites Ziel möchte ich zeigen, wie leicht 32-Bit-Code in ein 16-Bit-Forth eingebaut werden kann. Alles, was ich an 32-Bit-Erweiterungen für den ZF- oder Turbo-Forth-Assembler brauchte, waren zwei (nur zwei!) zusätzliche Colon-Definitionen.

Mir ist klar, daß mein ganzes Tun eigentlich darin besteht, Forth zum leichten und leicht anpaßbaren Assemblieren von Maschinensprache zu mißbrauchen.

In der bestehenden Forth-Literatur gibt es viele Arbeiten, die sich mit Code-Optimierung beschäftigen. Die beste Optimierung ist aber doch wohl die, die den zugrundeliegenden Algorithmus optimiert. Man betrachte also die vorliegende Arbeit als eine Aufforderung zu versuchen, den GGT in kürzerer Zeit oder mit einem kürzeren Programm zu finden.

Das Programm prüft zunächst, ob die Eingabewerte positiv sind. Negative Eingaben werden in ihre positiven Gegenstükke umgewandelt. Wenn eine oder beide Eingaben Null sind, springt das Programm ans Ende und liefert Null als den "größten gemeinsamen Teiler".

Was den Algorithmus selbst betrifft, erlaube ich mir, aus dem besagten Buch in Übersetzung zu zitieren

"Eine andere Art, Division und Multiplikation mehrfachgenauer Ganzzahlen zu vermeiden, besteht in der Verwendung der binären Version des Euklidischen Algorithmus. Diese benötigt lediglich Subtraktionen und macht von der Tatsache Gebrauch, daß die Teilung durch 2 in einem binären Computer ganz besonders einfach zu bewerkstelligen ist. Die Ganzzahl braucht nur s Stellen nach rechts verschoben zu werden, und schon hat man eine Teilung durch 2^s. Den Wert d=(a,b) findet man nach dem besagten Algorithmus wie folgt:

1. Angenommen, a endet mit u binären Nullen und b endet mit v binären Nullen.
   Sei d = 2^min(u,v).
2. Bilde a' = a/2^u und b' = b/2^v, d.h., schiebe a und b solange nach rechts, bis alle binären Nullen rechts herausgefallen sind.
3. Jetzt sind sowohl a' als auch b' ungerade Zahlen.
   Bilde c = a'-b' , eine gerade Zahl. Schiebe die Nullen von c heraus und wiederhole Schritt 3, dieses Mal mit dem reduzierten c und mit min(a',b'). Sobald das Ergebnis der Subtraktion 0 geworden ist, stellt der aktuelle Wert von a' den größten ungeraden Faktor d' von (a,b) dar.
4. Setze schließlich (a,b) = dd'.

Wie sich leicht zeigen läßt, benötigt die binäre Form des Euklidischen Algorithmus höchstens ungefähr ln max(a,b) = 3,32 * log max(a,b) Schritte. Die durchschnittliche Schrittzahl ist jedoch, wie beim normalen Euklidischen Algorithmus auch, kleiner, etwa 2,35 * log max(a,b). Da die in den einzelnen Schritten durchzuführenden Operationen wirklich einfach sind, kann also die binäre Form ganz gut mit der normalen Form des Algorithmus mithalten." - Ende des Zitats.

Das Programm wurde mit ZF und mit Turbo-Forth getestet. Zwischen den beiden Systemen bestand kein merklicher Unterschied. Das Programm wurde auch auf zwei verschiedenen Maschinen getestet, einem 80486/66 und einem K6-2/350. Das Verhältnis der Ausführungszeiten entsprach genau dem umgekehrten Verhältnis der Taktfrequenzen.

Ich habe nur das Worst-Case-Verhalten überprüft. Dazu habe ich als den einen Eingabewert 31 mal das 1-Bit (2147483647 dezimal) und als zweiten Eingabewert 30 mal das 1-Bit (119304647 dezimal) genommen. Diese beiden Zahlen sind relativ prim zueinander, d.h., ihr GGT beträgt 1, und die Ermittlung des GGT nimmt die maximale Anzahl von Schritten, also 31, in Anspruch. Außerdem habe ich die Worte

```
: XXX 30000 0 DO GGT LOOP ;
: YYY   100 0 DO XXX LOOP ;
```

zur Bestimmung derjenigen Zeit verwendet, die vergeht, wenn GGT 300000 mal ausgeführt wird, und

```
: XXX 30000 0 DO LOOP;
```

Ein neues Codewort

```
: YYY   100 0 DO LOOP;
```

um den Overhead abzuziehen, so daß man zur tatsächlichen Ausführungszeit von GGT gelangt.

Die Zahlen sind Dezimalzahlen. Zur Vermeidung von Stack-Überlauf und Stack-Overhead habe ich zu den POP-Operationen der Eingabewerte des Programms noch zwei sofort wirksame PUSH-Operationen hinzugefügt und darauf verzichtet, die GGT-Ergebnisse auf den Stack zu PUSHen. Die Ausführungszeiten von PUSH und POP können vernachlässigt werden. Es folgen die Testergebnisse für den schlimmsten Fall, den Worst-Case:

Die Ausführungszeit von GGT beträgt auf dem 80486/66er 157 Mikrosekunden und auf dem K6-2/350 beträgt sie 21,5 Mikrosekunden.

Es folgt das Programm-Listing (funktioniert sowohl in ZF als auch in Turbo-Forth):

```
\ Größter gemeinsamer Teiler (GGT).
\ Stackzellen = vorzeichenbehaftete
\ 32-Bit-Ganzzahlen. Negative Zahlen werden erst in
\ positive umgewandelt.
\ Ist eine Eingabezahl Null, wird als Ergebnis Null
\ auf den Stack gelegt.

ONLY FORTH ALSO
ASSEMBLER DEFINITIONS

HEX

\ Benötigt lediglich zwei Assembler-Erweiterungen.
\ Alles andere für den Zugriff auf die 32-Bit-
\ Register kann mit dem 16-Bit-Assembler erledigt
\ werden, den man in ZF (oder Turbo-Forth)
\ vorfindet.

: OPSIZE: 66 C, ;           \ 32-Bit-Datenzugriff;
                            \ EAX statt AX, usw.
: BSF ( xX yX -- )          \ Bit-Search-Forward;
                            \ yX zeigt das 1. 1-Bit
     0F C, MOV              \ in xX (von rechts her)
                            \ an; ersetze den
     0BC HERE 2- C! ;       \ Opcode MOV durch den
                            \ 2. Teil des Opcodes BSF

FORTH DEFINITIONS

CODE GGT ( ud1 ud2 -- ud3 )
     10 # CL MOV            \ Lade CL zur Verschie
                            \ bung um 2 Bytes
     OPSIZE: BX POP         \ ud2 in EBX
     OPSIZE: BX CL ROL      \ Vertausche nach
                            \ "little endian"
     OPSIZE: AX POP         \ ud1 in EAX
     OPSIZE: AX CL ROL      \ Vertausche nach
                            \ "little endian"
     OPSIZE: DX DX XOR      \ EDX = 0
     OPSIZE: DX INC         \ EDX = 1
             HERE           \ Sprungadresse für JS
     OPSIZE: AX NEG         \ Macht AX positiv
             JS             \ Wiederhole, falls
                            \ negativ
             HERE  5 +
             JNE            \ Wenn EAX = 0, dann
     OPSIZE: DX DX XOR      \ EDX = 0
             HERE           \ Sprungadresse für JS
     OPSIZE: BX NEG         \ Macht BX positiv
             JS             \ Wiederhole, falls
                            \ negativ
             HERE  5 +
             JNE            \ Wenn EBX = 0, dann
     OPSIZE: DX DX XOR      \ EDX = 0
     OPSIZE: DX BX XCHG
     OPSIZE: BX BX OR       \ Wenn mindestens ein
                            \ Eingabewert = 0,
     0<>                    \ dann Sprung zum Ende,
\ ---------------------------------------------
     IF                     \ mit dem Ergebnis DX = 0
       OPSIZE: BX DX XCHG
       OPSIZE: BX CX BSF    \ Z.B.: BX=8 -> CX=3
               CL DL MOV    \ Bewahre CL auf
       OPSIZE: BX CL SHR    \ BX = BX/2^CL
       OPSIZE: AX CX BSF    \ Z.B.: AX=4 -> CX=2
       OPSIZE: AX CL SHR    \ AX = AX/2^CL
               DL CL CMP    \ CL >= DL ?
               HERE  4 +
               JGE       \ Wenn nicht, dann CL -> DL
               CL DL MOV    \ DL = min(CL,DL)
                            \ auf jeden Fall
\ ---------------------------------------------
       BEGIN
         OPSIZE: AX CX BSF   \ Z.B.: AX=2 -> CX=1
         OPSIZE: AX CL SHR   \ AX = AX/2^CL
         OPSIZE: BX AX CMP   \ AX > BX ?
                 HERE  4 +
                 JG          \ Vertausche, falls
                             \ nicht
         OPSIZE: AX BX XCHG  \ Jetzt AX >= BX auf
                             \ jeden Fall
         OPSIZE: BX AX SUB   \ AX = AX-BX
         0=
       UNTIL
\ ---------------------------------------------
               DL CL MOV
       OPSIZE: BX CL SHL     \ BX = BX*2^DL
       10 # CL MOV           \ Lade CL zur Verschie-
                             \ bung um 2 Bytes
       OPSIZE: BX CL ROL     \ Vertausche nach
                             \ "little endian"
     THEN
\ ---------------------------------------------
     OPSIZE: BX PUSH         \ BX = ud3 = GGT.
     NEXT END-CODE
```

Nachwort: Auf dem Computer gibt es keine wirklich "reellen" Zahlen. Der Gleitkomma-Prozessor hat seine unbestrittenen Vorteile, die als Gleitkommazahlen darstellbaren Zahlen haben aber riesige Lücken, die noch dazu ungleichmäßig verteilt sind. Die Zahlen Pi und e lassen sich in keiner der üblichen Weisen darstellen. Auch und eben nicht in Form von endlichen Dezimalbrüchen. Letztere sind die Zahlen, in denen wir denken. Im Computer werden wir eigentlich mit Binärbrüchen konfrontiert. Bereits dabei gibt es schon Umwandlungsschwierigkeiten. Bei gemeinen Brüchen ist eine Darstellung in überraschend vielen Fällen ganz und gar nicht mehr möglich. Schon 1/3 macht Schwierigkeiten. Als periodischer Dezimalbruch oder als periodischer Binärbruch: ja ; als endlicher Bruch: nein. Man kann aber selbstverständlich mit 1/3, 7/5 usw. bei alleiniger Anwendung der arithmetischen Grundoperationen (+, -, *, /) "exakt"estens rechnen.

beh

Ein Web-Server in Forth

Bernd Paysan

Abstract

Ein HTTP-Server in Gforth dient als Anlaß, zu zeigen, daß man mit Forth durchaus auch String-orientierte Sachen machen kann. Die Entwicklungszeit (einige Stunden) zeigt, daß Forth auch hier das geeignete Werkzeug ist, um schnell zu einem Ergebnis zu kommen.

Vortrag anläßlich der Tagung in Hamburg

1. Einleitung

Nachdem ich in den vergangenen Jahren immer Vorträge zu bigFORTH/MINOS gehalten habe, soll dieses Mal wieder Gforth drankommen. Auch mit Gforth kann man tolle Sachen machen, und im Gegensatz zu manchen Unkenrufern geht eigentlich alles in Forth ziemlich einfach. Auch ein Web-Server.

Das Internet ist in Zeiten der „New Economy“ wichtig. Jeder ist „drin“. Außer Forth, das versteckt sich in der Embedded-Control-Nische. Ernsthaften Grund gibt's aber keinen. Der folgende Code ist in einigen Stunden Arbeit entstanden und verarbeitet hauptsächlich Strings. Das alte Vorurteil, Forth könne zwar gut Bits beißen, mache bei Strings aber Probleme, stimmt also nicht.

1.1 Motivation

Wozu braucht man einen Web-Server in Forth? Der eine hat Forth auf dem Meeresgrund oder im Krater eines Vulkans, und macht dort seine Messungen. Der andere hat Forth im Kühlschrank, und wenn der ausfällt, gibt's eine riesige Sauerei. Also hat man irgendeine Kommunikation eingebaut.

Ideal wäre es jetzt natürlich, wenn man nicht irgendeine Kommunikation eingebaut hat, sondern ein echtes Standard-Protokoll. HTTP kann auch der Browser im Web-Café auf Mallorca, oder mobiles Yuppie-Spielzeug wie PDAs und Handies. Vielleicht sollte man sowas in alle Herde und Badewannenmischbatterien einbauen, dann können die Leute im Urlaub über Handy jederzeit (alle drei Minuten) nachgukken, ob sie jetzt wirklich den Herd abgeschaltet haben.

Jedenfalls, der Kunde, Chef, oder wer auch immer das Produkt abnehmen soll, will sowieso, daß so ein Internet-Dingsbums da drin ist, wenn man schon nicht e-Bussiness macht. Und kosten darf das alles auch nichts.

Aber nun erst mal langsam, Schritt für Schritt.

2. Ein Web-Server, Schritt für Schritt

Eigentlich müßte man erstmal RFC([footnote] Request For Comments --- die Dokumente von Internet-Standards heißen so.) -Dokumente wälzen. Die in Frage kommenten RFCs sind RFC 1945 (HTTP/1.0) und RFC 2068 (HTTP/1.1), die verweisen natürlich ihrerseits auf andere RFCs. Da die Dokumente allein schon viel länger sind als der unten präsentierte Source-Code (und das Lesen würde länger dauern als das Schreiben dieser Sourcen), ersparen wir uns die für heute. Der Web-Server wird also nicht 100% RFC-konform sein (sprich: alle Features implementieren), sondern nur soweit, um mit typischen Clients wie Netscape zusammenzuarbeiten.
Ergänzungen können leicht durchgeführt werden.

Ein typischer HTTP-Request sieht etwa so aus:

```
GET /index.html HTTP/1.1

Host: www.paysan.nom

Connection: close
```

(Man beachte die Leerzeile am Schluß). Und die Antwort

```
HTTP/1.1 200 OK

Date: Tue, 11 Apr 2000 22:27:42 GMT

Server: Apache/1.3.12 (Unix) (SuSE/Linux)

Connection: close

Content-Type: text/html


<HTML>
...
```

Das sieht irgendwie recht trivial aus. Also, frisch ans Werk. Der Web-Server soll unter Unix/Linux laufen. Damit haben wir ein Problem schon mal vom Hals: Das Problem, wie wir an unseren Socket kommen. Das macht inetd, der Internet Deamon. Dem brauchen wir nur zu sagen, auf welchem Port unser Web-Server Daten erwartet, etwa wie folgt in der Datei/etc/inetd.conf:

```
# Gforth web server
gforth stream tcp nowait.10000
wwwrun   /usr/users/bernd/bin/httpd
```

Da wir erstmal nicht den vorhandenen Web-Server ersetzen wollen (es könnte ja noch etwas nicht funktionieren), brauchen wir einen eigenen Port, der kommt in die Datei /etc/services:

```
gforth  4444/tcp # Gforth web server
```

Beim nächsten Neustart (oder einem killall -HUP inetd) kriegt der Inetd das mit und startet dann für alle Requests auf Port 4444 unseren eigenen Web-Server. Aber nun brauchen wir erst mal ein ausführbares Programm. Gforth unterstützt Scripting mit #!, wie unter Unix üblich. Man muß nur auf das Leerzeichen achten:

```
#! /usr/local/bin/gforth

warnings off
```

Die Warnungen schalten wir lieber aus. Nun noch eine kleine String-Bibliothek laden (siehe Anhang).

```
include string.fs
```

Wir brauchen einige Variablen, für die URL, die der Server so verlangt, für Argumente, gepostete Argumente, das Protokoll und für Zustände.

```
Variable url          \ speichert die URL (string)
Variable posted       \ speichert Argumente von POST (string)
Variable url-args     \ speichert Argumente in der
                      \ URL (string)
Variable protocol     \ speichert das Protokoll (string)
Variable data         \ true, wenn Daten zurückgegeben
                      \ werden
Variable active       \ true für POST
Variable command?     \ true in der Request-Zeile
```

Ein Request besteht aus zwei Teilen, der Request-Line und dem Header. Trenner sind dabei Spaces. Das erste Wort einer Zeile ist dabei jeweils ein „Token“ des Protokolls, der Rest der Zeile, oder ein/zwei Wörter sind die Parameter.

Da wir einen Request erst abarbeiten können, wenn der ganze Header durchgelaufen ist, speichern wir alle Informationen zwischen. Dazu definieren wir uns zwei kleine Wörter, die ein Wort bzw. den Rest der Zeile in eine String-Variable speichern:

```
: get ( addr -- )  name rot $! ;
: get-rest ( addr -- )
  source >in @ /string dup >in +! rot $! ;
```

Wie gesagt, gibt's Header-Werte und Request-Kommandos. Damit wir die einfach interpretieren können, definieren wir zwei Wordlists:

```
wordlist constant values
wordlist constant commands
```

Doch bevor's richtig losgehen kann: Die URL könnte ja auch Spaces und andere Sonderzeichen enthalten, was macht man damit? HTTP schreibt vor, solche Sonderzeichen in der Form %xx zu übertragen, wobei xx für zwei Hex-Ziffern steht. Wir müssen also diese Zeichen in der fertigen URL ersetzen:

```
\ HTTP URL rework

: rework-% ( add -- ) { url }  base @ >r hex
   0 url $@len 0 ?DO
      url $@ drop I + c@ dup '% = IF
         drop 0. url $@ I 1+ /string
         2 min dup >r >number r> swap - >r 2drop
      ELSE  0 >r  THEN  over url $@ drop + c!  1+
   r> 1+ +LOOP  url $!len
   r> base ! ;
```

Uff, das ist geschafft. Doch halt! URLs bestehen aus zwei Teilen: Dem Pfad und optionalen Argumenten. Trenner ist das `?'. Also erst mal den String in zwei Teile zerlegen:

```
: rework-? ( addr -- )
   dup >r $@ '? $split url-args $! nip r> $!len ;
```

Na also, jetzt können wir ans Werk gehen. Requests besorgen sich also eine URL und ein Protokoll, zerlegen die URL in Pfad und Argumente und ersetzen die Sonderzeichen im Pfad durch darstellbare Zeichen (bei denen in den Argumenten warten wir vorerst noch mal ab, was damit überhaupt geschehen soll). Abschließend müssen wir auf ein anderes Vokabular umschalten, weil hinter dem Request ja der Header kommt.

```
: >values  values 1 set-order command? off ;

: get-url ( -- ) url get protocol get-rest
   url rework-? url rework-% >values ;
```

So, jetzt können wir unsere Kommandos definieren. Nach RFC brauchen wir nur GET und HEAD, POST gibt's also als Dreingabe

```
commands set-current

: GET   get-url data on  active off ;
: POST  get-url data on  active on  ;
: HEAD  get-url data off active off ;
```

Und nun zu den Header-Werten. Da wir für jeden Wert eine String-Variable brauchen und uns ansonsten einfach nur den String merken wollen, bauen wird das mit Create-DOES>. Also nochmal: Wir brauchen eine Variable und ein Wort, das den Rest der Zeile dort abspeichert. In zwei verschiedenen Vokabularen. Das letztere mit einem Doppelpunkt hintenran.

Immerhin liefert uns Gforth mit nextname dafür ein passendes Werkzeug: Wir basteln uns einfach genau den String zusammen, den wir brauchen und rufen Variable und Create danach auf.

```
: value:  ( -- )  name
  definitions 2dup 1- nextname Variable
  values set-current nextname here cell - Create ,
  definitions  DOES> @ get-rest ;
```

Und jetzt frisch ans Werk und alle nötigen Variablen definiert:

```
value: User-Agent:
value: Pragma:
value: Host:
value: Accept:
value: Accept-Encoding:
value: Accept-Language:
value: Accept-Charset:
value: Via:
value: X-Forwarded-For:
value: Cache-Control:
value: Connection:
value: Referer:
value: Content-Type:
value: Content-Length:
```

Es gibt zwar noch ein paar mehr (siehe RFC), das sind aber die, die wir im Moment brauchen.

3. Parsen eines Requests

Also, nun müssen wir nur noch den Request parsen. Eigentlich wäre das total trivial, wir lassen einfach den Forth-Interpreter werkeln. Hat aber zwei kleine Haken:

1. Endet jede Zeile mit CR LF, während Gforth unter Unix davon ausgeht, daß Zeilen nur mit einem LF enden. Wir müßen also das CR wegnehmen. Und

2. endet der Header mit einer Leerzeile und nicht irgendeinem ausführbaren Forth-Wort. Wir müssen also Zeile für Zeile mit refill einlesen, CRs am Zeilenende wegnehmen, und dann noch gucken, ob die Zeile leer war.

```
Variable maxnum

: ?cr ( -- )
  #tib @ 1 >= IF  source 1- + c@ #cr = #tib +!  THEN ;

: refill-loop ( -- flag )
  BEGIN  refill ?cr  WHILE  interpret  >in @ 0=
  UNTIL
    true  ELSE  maxnum off false  THEN ;
```

So, das Wichtigste ist schon geschafft. Da wir den Forth-Interpreter nicht einfach auf stdin loslassen können, müssen wir das selber machen. Wir initialisieren also ein paar Variablen, die wir auf alle Fälle auswerten und klauen Code aus “included“:

```
: get-input ( -- flag ior )
  s" /nosuchfile" url $!  s" HTTP/1.0" protocol $!
  s" close" connection $!
  infile-id push-file loadfile !  loadline off  blk off
  commands 1 set-order  command? on  ['] refill-loop
   catch
```

Moooment! Der Request ist noch nicht ganz fertig. Die Methode POST, die's als Dreingabe gibt, erwartet jetzt noch ihre Daten. Die Länge ist freundlicherweise als Base-10-Zahl im Feld „Content-Length:“ angegeben.

```
active @ IF  s" " posted $! Content-Length $@
     snumber? drop
     posted $!len  posted $@ infile-id read-file throw drop
THEN  only forth also  pop-file ;
```

4. Auf einen Request antworten

Ok, der Request wäre damit abgehandelt, wir müssen also die Antwort geben. Der Pfad der URL ist leider nicht gerade so, wie wir ihn uns wünschen: Wir wollen irgendwie Apache-kompatibel sein, d.h. wir haben ein „global document root“ und ein Verzeichnis im Home-Directory eines jeden Users, in dem der seine persönliche Homepage unterbringt. Also bleibt uns nichts anderes, als den Pfad erneut unter die Mangel zu nehmen und abschließend zu überprüfen, ob's die verlangte Datei überhaupt gibt:

```
Variable htmldir

: rework-htmldir ( addr u -- addr' u' / ior )
  htmldir $!
  htmldir $@ 1 min s" ~" compare 0=
  IF    s" /.html-data" htmldir dup $@ 2dup '/ scan
        nip - nip $ins
   ELSE  s" /usr/local/httpd/htdocs/" htmldir 0 $ins
   THEN
   htmldir $@ 1- 0 max + c@ '/ = htmldir $@len 0= or
   IF  s" index.html" htmldir dup $@len $ins  THEN
   htmldir $@ file-status nip ?dup ?EXIT
   htmldir $@ ;
```

Als nächstes müssen wir uns entscheiden, wie der Client die Datei darstellen soll, spricht, was für einen MIME-Typ sie hat. Entscheidungskriterium ist die Dateiendung. Die muß extrahiert werden.

```
: >mime ( addr u -- mime u' )  2dup tuck over + 1- ?DO
  I c@ '. = ?LEAVE  1-  -1 +LOOP  /string ;
```

Normalerweise werden wir die Datei so, wie sie ist, an den Client schicken (transparent). Dazu sagt man ihm am besten, wie lang die Datei ist (sonst müßte man die Connection für jeden Request schließen). Wir wandeln also einen Dateinamen in Größe und Filedeskriptor um und senden das dann an den Client.

```
: >file ( addr u -- size fd )
  r/o bin open-file throw >r
  r@ file-size throw drop
  ." Accept-Ranges: bytes" cr
  ." Content-Length: " dup 0 .r cr r> ;

: transparent ( size fd -- ) { fd }
  $4000 allocate throw swap dup 0 ?DO
     2dup over swap $4000 min fd read-file throw type
     $4000 - $4000 +LOOP  drop
  free fd close-file throw throw ;
```

Die Arbeit mit der Dateigröße machen wir uns für „Keep-Alive“-Connections, wie sie moderne Web-Browser gerne machen. Der Aufbau einer neuen Verbindung ist um einiges “teurer“, als einfach mit der bestehenden weiterzumachen. Auch auf unserer Seite kommt uns das zugute, denn Gforth nochmal starten ist auch nicht umsonst. Wenn die Connection Keep-Alive ist, geben wir das also zurück, erniedrigen maxnum um eins und teilen dem Empfänger mit, wie oft er noch darf. Wenn's der letzte Request ist, oder gar kein weiterer verlangt wurde, senden wir das auch zurück.

```
: .connection ( -- )
  ." Connection: "
  connection $@ s" Keep-Alive" compare 0= maxnum @
  0> And
  IF  connection $@ type cr
   ." Keep-Alive: timeout=15, max=" maxnum @ 0 .r cr
   -1 maxnum +!  ELSE  ." close" cr maxnum off
   THEN ;
```

Jetzt brauchen wir nur noch eine Möglichkeit, MIME-Dateienden in entsprechende Transmissions zu übersetzen. Auch bei der Rückantwort gilt: Zunächst einmal muß ein Header gesendet werden. Den bauen wir hier „von hinten her“ auf, weil die oberen Definitionen ihren Senf weiter vorne dazugeben.
Damit wir die Zuordnung Dateiende--MIME-Typ leicht haben, definieren wir für jede Datei einfach ein Wort. Das kriegt den MIME-Typ als String übergeben. transparent: macht uns das für all die Dateitypen, die transparent durchgereicht werden:

```
: transparent: ( addr u -- ) Create  here over 1+ allot place
  DOES>  >r  >file
  .connection
  ." Content-Type: "  r> count type cr cr
  data @ IF  transparent  ELSE  nip close-file throw
      THEN ;
```

So, nun gibt's ja hunderte MIME-Typen, wer will die alle eintippen? Nichts leichter als das, klauen wir einfach die MIME-Typen, die dem System schon bekannt sind, etwa aus /etc/mime.types. Die Datei listet links jeweils einen MIME-Typ, rechts einige Dateiendungen (auch mal keine).

```
: mime-read ( addr u -- )  r/o open-file throw
   push-file loadfile !  0 loadline ! blk off
   BEGIN  refill  WHILE  name
      BEGIN  >in @ >r name nip  WHILE
         r> >in ! 2dup transparent:  REPEAT
      2drop rdrop
   REPEAT  loadfile @ close-file pop-file throw ;
```

Einen brauchen wir noch: Für aktive Inhalte wollen wir Serverseitiges Skripting (natürlich in Forth) verwenden. Bei solchen Inhalten kennen wir die Länge nicht im Vorraus, also geben wir sie einfach nicht an, sondern machen die Verbindung zu. Damit entledigen wir uns auch des Problems, den Müll, den der User mit seinen aktiven Inhalten anrichtet (das ist schließlich Forth-Code!) wieder aufräumen zu müssen.

```
: lastrequest
  ." Connection: close" cr maxnum off
  ." Content-Type: text/html" cr cr ;
```

Also, los geht's mit der Definition der MIME-Typen. Her mit einer neuen Wordlist. Aktive Inhalte enden mit shtml und werden per included geladen. Ansonsten ein paar Beispieldefinitionen, und den Rest kriegen wir aus der oben genannten Systemdatei. Für unbekannte Dateitypen brauchen wir noch einen Standardtyp, text/plain.

```
wordlist constant mime
mime set-current

: shtml ( addr u -- )  lastrequest
   data @ IF  included  ELSE  2drop  THEN ;

s" application/pgp-signature" transparent: sig
s" application/x-bzip2" transparent: bz2
s" application/x-gzip" transparent: gz
s" /etc/mime.types" mime-read

definitions

s" text/plain" transparent: txt
```

5. Fehlermeldungen

Manchmal geht bei so einem Request auch was schief. Jedenfalls müssen wir darauf gefaßt sein und dem Client eine entsprechende Meldung machen. Er will wissen, welches Protokoll wir sprechen, was passiert ist (oder ob alles OK ist), wer wir sind, und im Fehlerfall ist eine Klartextmeldung (in HTML codiert) ganz nett:

```
: .server ( -- )  ." Server: Gforth httpd/0.1 ("
   s" os-class" environment? IF  type  THEN  ." )" cr ;

: .ok  ( -- ) ." HTTP/1.1 200 OK" cr .server ;
```

```
: html-error ( n addr u -- )
     ." HTTP/1.1 " 2 pick . 2dup type cr .server
     2 pick &405 = IF ." Allow: GET, HEAD, POST" cr
   THEN
    lastrequest
    ." <HTML><HEAD><TITLE>" 2 pick . 2dup type
    ." </TITLE></HEAD>" cr
    ." <BODY><H1>" type drop ." </H1>" cr ;

: .trailer ( -- )
   ." <HR><ADDRESS>Gforth httpd 0.1</ADDRESS>"
     cr
   ." </BODY></HTML>" cr ;

: .nok ( -- ) command? @ IF  &405 s" Method Not Allowed"
   ELSE  &400 s" Bad Request"  THEN  html-error
   ." <P>Your browser sent a request that this server "
   ." could not understand.</P>" cr
   ." <P>Invalid request in: <CODE>"
   error-stack cell+ 2@ swap type
   ." </CODE></P>" cr .trailer ;

: .nofile ( -- ) &404 s" Not Found" html-error
   ." <P>The requested URL <CODE>" url $@ type
   ." </CODE> was not found on this server</P>" cr .trailer ;
```

6. Top-Level-Definitionen

So, jetzt sind wir eigentlich fertig. Wir müssen die ganzen Stückchen noch zusammenkleben, damit ein Request richtig abgearbeitet wird: Den Input holen, die URL umformen, den MIME-Typ erkennen und das dann abarbeiten, jeweils mit Fehlerausgängen oder Default-Pfaden. Die Ausgabe müssen wir flushen, damit der nächste Request nicht hängenbleibt. Und das ganze evtl. n mal abarbeiten, bis wir den letzten Request erreicht haben.

```
: http ( -- )  get-input  IF  .nok  ELSE
   IF  url $@ 1 /string rework-htmldir
     dup 0< IF  drop .nofile
      ELSE  .ok  2dup >mime mime search-wordlist
        0= IF  ['] txt  THEN  catch IF  maxnum off THEN
      THEN  THEN  THEN  outfile-id flush-file throw ;

: httpd  ( n -- )  maxnum !
   BEGIN  ['] http catch  maxnum @ 0= or  UNTIL ;
```

Damit das dann beim Start von Gforth auch ausgeführt wird, patchen wir die Bootmessage damit. So können wir das auch als neues Systemimage abspeichern.

```
script?[IF]  :noname &100 httpd bye ; is bootmessage
        [THEN]
```

7. Scripting

Als besonderes Bonbon gibt's nun noch die aktiven Inhalte. Das ist wirklich ganz einfach: Wir schreiben unsere HTML-Datei so wie gewohnt, nur den Forth-Code in “<$“ und “$>“ eingerahmt (das Leerzeichen für die schließende Klammer ist natürlich Absicht!). Definieren müssen wir nur zwei Wörter, $> und damit das Ganze überhaupt losgeht, <HTML>:

```
: $> ( -- )
   BEGIN  source >in @ /string s" <$" search  0=
       WHILE
   type cr refill  0= UNTIL  EXIT  THEN
   nip source >in @ /string rot - dup 2 + >in +! type ;

: <HTML> ( -- )  ." <HTML>" $> ;
```

Das reicht in der Tat, mehr brauchen wir nicht, den Rest erledigt Forth, wie in folgendem Beispiel:

```
<HTML>
<HEAD>
<TITLE>GForth <$ version-string type $>
  presents</TITLE>
</HEAD>
<BODY>
<H1>Computing Primes</H1><$ 25 Constant #prim $>
<P>The first <$ #prim . $> primes are: <$

  : prim?  0 over 2 max 2 ?DO  over I mod 0= or  LOOP
    nip 0= ;

  : prims ( n -- )  0 swap  2
    swap 0 DO  dup prim? IF  swap  IF  ." , "  THEN
    true swap
    dup 0 .r 1+  1  ELSE  1+  0  THEN
    +LOOP  drop ;
    #prim prims $> .</P>
    </BODY>
    </HTML>
```

8. Ausblick

Das waren jetzt ein paarhundert Zeilen Code. Eigentlich viel zu viel. Ich habe auch einen fast vollwertigen Apache-Klon geliefert. Das wird man auf dem Meeresgrund oder im Kühlschrank nicht brauchen. Fehlerbehandlung ist auch nur Ballast. Und wenn man sich auf Einmalverbindungen beschränkt (Performance ist ja nicht das Ziel), kann man auch die ganzen Protokoll- Variablen einfach ignorieren. Ein MIME-Typ (text/html) reicht aus --- die Bilder speichert man auf einem anderen Server. Es besteht jedenfalls die Hoffnung, ein einigermaßen ordentliches HTTP-Protokoll mit server-side Scripting in einem Screen unterzubringen.

Anhang: String-Funktionen

Natürlich braucht man irgendwelche String-Funktionen, sonst geht's nicht. Die folgenden String-Bibliothek speichert Strings in normalen Variablen, die dann einen Pointer auf einen counted String enthalten. Statt einem Count-Byte gibt's gleich eine ganze Zelle, das sollte ausreichen. Die String-Bibliothek stammt ursprünglich aus bigFORTH, ich habe sie für Gforth (ANS Forth) angepaßt. Doch nun ie Funktionen im Detail. Zunächst brauchen wir einmal zwei Wörter, die bigFORTH schon so mitbringt:

```
: delete   ( addr u n -- )
  over min >r  r@ - ( left over )  dup 0>
  IF  2dup swap dup  r@ +  -rot swap move  THEN  + r> bl
fill ;
```

delete löscht die ersten n bytes aus einen Puffer und füllt den Rest hinten mit Blanks auf.

```
: insert   ( string length buffer size -- )
  rot over min >r  r@ - ( left over )
  over dup r@ +  rot move   r> move  ;
```

insert fügt einen String vorne in einen Buffer ein. Die restlichen Bytes werden nach hinten geschoben.

So, nun geht's richtig zur Sache:

```
: $padding ( n -- n' )
  [ 6 cells ] Literal + [ -4 cells ] Literal and ;
```

Damit wir die Speicherverwaltung nicht überfordern, gibt's nur bestimmte String-Größen; $padding sorgt für Vielfache von vier Zellen.

```
: $! ( addr1 u addr2 -- )
  dup @ IF  dup @ free throw  THEN
  over $padding allocate throw over ! @
  over >r  rot over cell+  r> move 2dup ! + cell+ bl swap c! ;
```

$! speichert einen String an einer Adresse ab. Falls da vorher schon ein String drin war, wird der freigegeben.

```
: $@ ( addr1 -- addr2 u )  @ dup cell+ swap @ ;
```

$@ gibt den gespeicherten String zurück.

```
: $@len ( addr -- u )  @ @ ;
```

$@len gibt die Länge eines Strings zurück

```
: $!len ( u addr -- )
  over $padding over @ swap resize throw over ! @ ! ;
```

$!len ändert die Länge eines Strings. Dazu muß sowohl der Speicherbereich vergrößert werden, als auch die Adresse und die Count-Cell geändert werden.

```
: $del ( addr off u -- )   >r >r dup $@ r> /string r@ delete
  dup $@len r> - swap $!len ;
```

$del löscht u bytes aus einem String mit Offset off.

```
: $ins ( addr1 u addr2 off -- ) >r
  2dup dup $@len rot + swap $!len  $@ 1+ r> /string insert ;
```

$ins fügt einen String am Offset off ein.

```
: $+! ( addr1 u addr2 -- ) dup $@len $ins ;
```

$+! hängt einen String an einen anderen an.

```
: $off ( addr -- )  dup @ free throw off ;
```

$off gibt einen String wieder frei.

Als Bonus gibt's noch Funktionen, um Strings zu zerlegen.

```
: $split ( addr u char -- addr1 u1 addr2 u2 )
  >r 2dup r> scan dup >r dup IF  1 /string  THEN
  2swap r> - 2swap ;
```

$split teilt einen String in zwei, als Trenner dient ein Zeichen (z.B. '? für Argumente)

```
: $iter ( .. $addr char xt -- .. ) { char xt }
  $@ BEGIN  dup  WHILE  char $split >r >r xt execute r> r>
     REPEAT  2drop ;
```

$iter zerlegt einen String Stück für Stück in seine Bestandteile, ebenfalls mit einem Zeichen als Trenner. Für jedes Teilstück wird ein übergebens Token aufgerufen. Damit kann man Argumente, die mit '& separiert sind, bequem trennen.

Bernd Paysan

Forthtagung 2001

Die nächste Forthtagung wird im April 2001 am Niederrhein stattfinden, dieses Mal am rechten (falsche Rheinseite), unteren Niederrhein, in Dingden (bei Wesel), unweit von Moers.

Martin Bitter wird diese Tagung mit Hilfe der **Moerser Gruppe** organisieren.

Nähere Informationen werden ab der Ausgabe 04/2000 in der VD bekannt gegeben.

Anmeldeformulare werden wieder in der Ausgabe 01/2001 enthalten sein, bzw. dieser beiliegen.

Merken Sie sich diese Tagung schon jetzt vor !

fep

Protokoll der Jahresversammlung 2000 der Forth Gesellschaft e.V.

Ort: Hamburg, Haus Rissen

Tag: 16.04.2000

Beginn: 09.00

Ende: 12.40

1. Begrüßung

Friederich Prinz begrüßte die anwesenden Mitglieder zur Jahresversammlung der Forth-Gesellschaft e.V.. Zunächst gab Friederich Prinz das Ergebnis der Abstimmung vom Samstag über das zukünftige Logo der Forth-Gesellschaft bekannt. Die anwesenden Mitglieder entschieden sich mit großer Mehrheit für das von Heinz Schnitter eingereichte Logo (siehe VD Ausg. 4/99 S.32). Als Belohnung wird Heinz Schnitter kostenfrei zur nächsten Forth-Tagung 2001 eingeladen. Fred Behringer (Drachenpreisträger der FG-Tagung 1999) überreichte im Namen des Drachenrats den Drachen an Egmont Woitzel und lobte ihn in seiner Laudatio für seine Leistungen beim Aufbau der Webseite unserer Gesellschaft.

2. Wahl des Protokollführers

Friederich Prinz schlägt Ewald Rieger als Protokollführer vor. Ewald Rieger wird per Akklamation zum Protokollführer ernannt.

3. Wahl des Versammlungsleiters

Ulrich Hoffmann wird per Akklamation mit der Leitung der Versammlung beauftragt.

4. Feststellung der Beschlussfähigkeit

Ute Woitzel berichtete, dass die FG zum April 2000 146 Mitglieder zählt, das sind 19 Mitglieder weniger als im Vorjahr. Die 17 anwesendenen Mitglieder ergaben somit eine beschlussfähige Versammlung.

5. Bericht des Direktoriums

5.1. Kassenbericht von Egmont Woitzel

Einnahmen 1999		
Mitgliedsbeiträge	DM	7470,46
(ohne DM 40,00 für die VD)		
Spenden	DM	1325,00
Überschuss Tagung 1999	DM	1032,42
Einnahmen VD	DM	6775,13
Porto Versand VD	DM	40,00
Erstattung MWST	DM	505,37
Summe	DM	17148,38
Ausgaben 1999		
Bürobedarf	DM	183,21
Bankgebühren	DM	125,46
Forth-Büro	DM	1826,07
(Kosten für 1 1/2 Jahre)		
Internetzugang	DM	2005,92
Druck VD	DM	1458,61
Frachtkosten VD	DM	294,50
Versand VD	DM	1826,07
Porto	DM	2456,19
MWST	DM	564,11
Summe	DM	10470,57
Überschuss	DM	6677,81
Kontostand 4/2000:		
Girokonto	DM	44342,47
Sparbuch	DM	373,85

A.Klingelnberg regte an, ein Teil des Geldes auf dem Girokonto zu besseren Konditionen anzulegen. Kassenprüfer Hans Reilhofer bestätigte eine ordnungsgemäße Führung der Kasse.

5.2 Friederich Prinz berichtet über die VD

Friederich Prinz apellierte an die Mitglieder mehr Artikel für die VD zuschreiben. Dies erhöhe das Interesse an der VD und hält Mitglieder in der FG. Fred Behringer lobte die hohe grammatische Qualität der VD und würdigte Birgit Prinz für Ihre hervorragende Arbeit beim Korrigieren der Artikel, um diesen hohen Standard zu erreichen. Als Dank überreichte er im Namen der FG einen Blumenstrauß. Weiteren Dank an Friederich Prinz für seine ehrenamtliche Arbeit als Editor der VD.

5.3. Thomas Beierlein berichtet über zukünftige Vorhaben des Direktoriums

Thomas Beierlein erläutert die Vorhaben mit den nachfolgend genannten Schwerpunkten.

- weiterer Ausbau der Internet-Aktivitäten
- Internationale Verbindungen zu anderen Forthgruppen ausbauen
- Inhalte der VD vermehren
- Mitgliederentwicklung

6. Entlastung des Direktoriums

Die Versammlung entlastete das Direktorium mit 14 Ja-Stimmen und 3 Enthaltungen.

7. Wahl des Direktoriums

Friederich Prinz teilte der Versammlung mit, dass er durch berufliche Mehrbelastungen nicht mehr für das Amt eines Direktors kandidieren möchte. Insbesondere müsse die zukünftige internationale Zusammenarbeit verbessert werden und er möchte deshalb diese Arbeit in die Hände einer dafür geeigneten Person legen. Er schlug Fred Behringer als seinen Nachfolger vor, der seit vielen Jahren internationale Kontakte knüpfte und pflegt. Die Direktoren Thomas Beierlein und Egmont Woitzel erklärten sich bereit, erneut zu kandidieren. Der Versammlungsleiter schlug vor, alle drei Kandidaten gemeinsam zu wählen. Die Versammlung folgte seinem Vorschlag und bestätigte per Akklamation die drei Kandidaten mit 14 Ja-Stimmen und 3 Enthaltungen.

8. Verschiedenes

8.1. Mitgliedsbeiträge und Buchhaltung

Egmont Woitzel schlug vor, die Buchhaltung der FG ab dem 1.1.2001 auf Euro umzustellen. Die versammelten Mitglieder einigten sich, die Mitgliedsbeiträge im Verhältnis 2:1 auf den Euro umzurechnen. Daraus ergeben sich zukünftig folgende Jahresmitgliedsbeiträge:

regulärer Beitrag	Euro 40.00
ermassigter Beitrag	Euro 16.00
fördernde Mitglieder	Euro 88.00

Diesem Antrag stimmten alle Mitglieder mit Ausnahme einer Enthaltung zu.

8.2. Internet

Die Versammlung schlug vor, das Webangebot weiter auszubauen. Die alten VD's werden ins Internet gestellt. Zusätzlich sollen die Beiträge im PDF-Format zum Download bereitstehen. Auch der Beitrag von Martin Bitter zur Forth-Tagung 2000, über den unter Forth programmierten Legoroboter RCX im Schulunterricht, möge ins Internet kommen. Zu diesem Thema soll eine eigene Rubrik für den RCX-Roboter im Web entstehen.

8.3. eMail-Zugang

Egmont Woitzel berichtet kurz über den Web-Server in Karlsruhe, über den unser Web-Angebot bereitgestellt wird. Er bestätigte eine zufriedenstellende Arbeit und hohe Verfügbarkeit des Servers. Um weitere Kosten einzusparen, schlug er vor, den inzwischen wenig genutzten Zugang über KBBS bei Holger Petersen abzuschaffen. Die betroffenen Mitglieder werden rechtzeitig über die Stilllegung informiert. Diesem Antrag stimmten alle Mitglieder bis auf eine Enthaltung zu.

8.4. Einsatz finanzieller Mittel

Egmont Woitzel schlug vor, die Ressourcen von Windows besser mit Forth zu nutzen. Dazu soll ein Interface zur DCOM-Schnittstelle entstehen. Forth ließe sich dann alternativ zum VBA als Scriptsprache unter Windows einsetzen. Zur Programmierung dieser Schnittstelle erklärte sich Hans Reilhofer bereit, einen Diplomanden für ein halbes Jahr zu finanzieren. Die Arbeiten erfolgen bei Egmont Woitzel in Rostock und werden nach Abschluss veröffentlicht.

Ein Vorschlag, von der FG bezahlte Redakteure zur Beschaffung von Artikeln für die VD einzusetzen, wurde schnell verworfen. Spontan erklärten sich die Mitglieder Hans Reilhofer und Arndt Klingelnberg bereit, im nächsten Halbjahr je einen Artikel für die VD zu schreiben. Egmont Woitzel sagte unsicheren Autoren beim Korrigieren ihrer Artikel seine Unterstützung zu.

Als Beilage zur VD soll ein aktueller Forth-Snapshot von Serverabzügen auf CD inclusive frei verfügbarer Forth-Systeme erscheinen.

Michael Kalus beantragte, der Schule von Martin Bitter einen RCX-Roboter-Ergänzungskasten im Wert von DM 500.00 zu spenden. Diesem Antrag stimmten alle anwesenden Mitglieder zu.

8.5. Kontakte zur holländischen Forth-Gruppe

Die Mitglieder begrüßten die Initiative, den Kontakt zur holländischen Forth-Gruppe zu verbessern mit dem Ziel, Beiträge von der dortigen Zeitschrift Feigenblatt in der VD abdrucken zu dürfen.

Das noch ausstehende Thema Forth-Buch konnte auf Grund der fortgeschrittenen Zeit nicht mehr behandelt werden (Hunger, Durst und das Mittagessen wartete bereits).

Ulrich Hoffmann beendete auf Wunsch der Beteiligten die Versammlung gegen 12.40

8.6 Nachtrag

Beim gemeinsamen Mittagessen bemerkten einige Mitglieder, dass noch kein Veranstalter für die nächste Tagung und Jahresversammlung 2001 gefunden wurde. Martin Bitter erklärte sich spontan bereit, die nächste Tagung von Freitag, dem 27. April bis zum Sonntag, dem 29. April 2001 auszurichten. Der Tagungsort ist die Akademie Klausenhof, Niederlassung Dingden bei Hamminkeln - Wesel - . Dafür herzlichen Dank an Martin Bitter.

gez. *Ewald Rieger*
Protokollführer

Forth-Gruppen regional

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

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

München
Jens Wilke
Tel.: 089-89 76 890
Treffen: jeden 4. Mittwoch im
Monat, **China Restaurant XIANG**
Morungerstraße 8
München-Pasing

µP-Controller Verleih

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

Gruppengründungen, Kontakte

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

Forth-Hilfe für Ratsuchende

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

Jörg Staben
Tel.: 02103-24 06 09 (p)

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

Spezielle Fachgebiete

Arbeitsgruppe MARC4	Rafael Deliano Tel./Fax: 089-841 83 17 (p)
FORTHchips (FRP 1600, RTX, Novix)	Klaus Schleisiek-Kern Tel.: 040-375 008 03 (g)
F-PC & TCOM, Asyst (Meßtechnik), embedded Controller (H8/5xx// TDS2020, TDS9092), Fuzzy	Arndt Klingelnberg, Consultants akg@aachen.kbbs.org Tel.: ++32 +87 -63 09 89 pgq (Fax -63 09 88)
KI, Object Oriented Forth, Sicherheitskritische Systeme	Ulrich Hoffmann Tel.: 04351 -712 217 (p) Fax: -712 216
Forth-Vertrieb	volksFORTH / ultraFORTH RTX / FG / Super8 / KK-FORTH Ingenieurbüro Klaus Kohl Tel.: 08233-3 05 24 (p) Fax : 08233-99 71 mailorder@forth-ev.de
Forth-Mailbox (KBBS)	0431-533 98 98 (8 N 1) Sysop Holger Petersen hp@kbbs.org Tel.: 0431-533 98 96 (p) bis 22:00 Fax : 0431-533 98 97 Helsinkistraße 52 24109 Kiel

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

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

Hinweise zu den Angaben nach den Telefonnummern:

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

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

Jahrestagung – Hamburg 2000

Interessante Vorträge,
gute Gespräche in angenehmer Runde,
Ausflüge mit viel Sehenswertem und
die Freude darüber, „alte“ Bekannte und „Freunde in Forth“ zu treffen, können Sie in 2001 am Niederrhein erneut erleben.

Seien Sie dabei !

(Nähere Informationen finden Sie in der Ausgabe 04/2000)