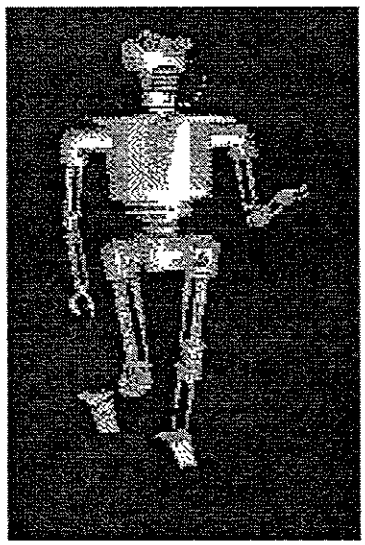
VIERTE DIMENSION

1/1997

13. Jahrgang 1997, 1. Quartal, DM 10.-



- Gforth auf MISC portiert
- Embedded Systems'97
- MISC
- Low-Level-Debug-Werkzeuge für fieldFORTH
- WIN32FOR - Nehmen wir unsere Umlaute mit?
- Forth Online
- Graphik `ohne Ende'
- Forth International - weiter geht's in Holland
- DN-1620 - Forth-Prozessor aus Weißrußland
- From the other side of the Big Teich
- Nucleus für Controller - Teil

7 FORTH MAGAZIN

Organ der Forth Gesellschaft e.V.

http://www.informatik.uni-kiel.de/~uho/VD/

Schon überwiesen?

Wer den Mitgliedsbeitrag für 1997 noch nicht bezahlt hat, sollte ihn jetzt auf den Weg bringen. Wer per Einzugsermächtigung zahlt, ist fein raus und braucht nichts zu tun...

Die Mitgliedsbeiträge der Forth-Gesellschaft e.V.:

Schüler, Studenten, Rentner und Arbeitslose (nur mit Nachweis)	64,00 DM
Ordentliche Mitglieder, Auslandsadresse	96,00 DM
Fördernde Mitglieder, Firmen und Institutionen	176,00 DM

Bankverbindung der FORTH-Gesellschaft:

Postbank Hamburg
Kontonummer: 5632 11-208
BLZ: 200 100 20

Rückfragen bitte an das FORTH-Büro:

PF 1110
D-85701 Unterschleißheim
Tel/Fax: 089 / 317 37 84
secretary@admin.FORTH-eV.de

Dienstleistungen und Produkte von Forthlern und/oder für Forthler (Anzeige)

Dienstleistungen und Produkte von Forthlern und/oder für Forthler (Anzeige)

IMPRESSUM

Name der Zeitschrift

FORTH MAGAZIN - VIERTE DIMENSION
Organ der Forth-Gesellschaft e.V.

Herausgeberin

FORTH-Gesellschaft e.V.
Postfach 1110
85701 Unterschleißheim
Tel./Fax: 089/3173784
secretary@admin.FORTH-eV.de

Redaktion & Layout

Claus Vogt

Katzbachstr. 23; D-10965 Berlin
Tel.: 030/786 84 60 (Fax nach Bedarf)
vd@FORTH-ev.de

Anzeigenverwaltung:

Ulrike Schnitter c/o Forth-Gesellschaft

Internationale Kontakte:

Fred Behringer

Planegger Str. 24; D-81241 München
behringe@statistik.tu-muenchen.de

Zeichnungen:

Rolf Kretzschmar;Hans-Georg Schmid

Redaktionsschluß

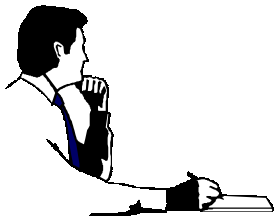
Erste Januar/April/Juli/Oktoberwoche.

Erscheinungsweise

Viermal im Jahr.

Preis

Einzelpreis: DM 10,-



Die Wilde Dreizehn ...

Manche Hotels sparen bei der Numerierung ihrer Zimmer die '13' aus. Abergläubige Gäste könnten sonst schlecht schlafen. Das Forth Magazin geht mit dieser Ausgabe mutig in den 13.Jahrgang. Welches Unglück sich dahinter auch verbergen mag - nächstes Jahr sind wir klüger oder haben es erfolgreich verschlafen.

Gleichzeitig ist 1997 der dritte Jahrgang 'Forth Magazin aus Berlin' unter meinem Editoriat. Ende dieses Jahres darf ich mich als dienstältester VD-Editor ins Guinness Buch der Rekorde eintragen lassen. Im dritten Jahr ihres Editoriats wurden meine geschätzten Vorgänger bislang immer von aufgebrachter LeserInnenschaft hinweggefegt oder von tödlicher Langeweile befallen zu neuen Ufern gerufen. Wobei mir persönlich die zweite Variante die unangenehmere zu sein scheint.

Die deutsche Forth-Gemeinde sollte sich durchaus bewußt überlegen, ob sie einen Daueredítor will. Marlin Ouverson leitet die amerikanische 'Forth Dimensions' schon einige zehn Jahre. Die wachsende Zustimmung, die unser (zumindest zu 80% deutschsprachiges) Forth Magazin auf hiesigem wie internationalen Parkett erfährt, hängt nicht zuletzt damit zusammen, daß die VD sich bei aller Zähigkeit und Kontinuität ihren Experimentalcharakter bewahrt hat und offen nach außen demonstriert. Das macht Spaß und spornt zur Mitarbeit an. Mich jedenfalls...

- Claus Vogt, Editor des Forth Magazins Vierte Dimension -

EuroForth'97 - Oxford, England -Call for Papers

Embedded Communications - September 26-28 1997

EuroForth, the annual European Forth Conference will focus this year on the increasing use of communications within applications, modems, Internet payment, process control, door access. EuroForth'97 is being held in the lovely setting of St Anne's College Oxford. This college is within easy walking distance to the city centre yet is free from the crowds of central Oxford.

Contact:

The Conference Organiser, EuroForth'97; c/o Microprocessor Engineering Limited;
133 Hill Lane; Southampton SO15 5AF; England;
Tel: +44 1703 631441; Fax: +44 1703 33691; net: mpe@mpeltd.demon.com.uk;

euroForth webpage of Peter Knaggs: http://cis.paisley.ac.uk/forth/euro/ef97.html

Die dritte Ausgabe des Newsletters wird ab März 1997 verfügbar. Sie enthält die Buanleitung für ein Programmiergerät, berichtet über die erste OTP-Variante und ist gespickt mit technischen Informationen.

Erhältlich sind die Marc4-Newsletters bei:

Forth Vertrieb Klaus Kohl
PF 1173
86406 Mehring

Zeppelinstr. 10
85415 Mering

Tel.: 08233-30524
Fax.: 08233-9971 f

❑


MARC4

Newsletter der FORTH e.V.

M48C260

Inhalt Ausgabe 3

1 Einleitung
2 Reset, Takt
3 Ports
4 Externe IRQs
5 EEPROM
6 Timer 1, Watchdog
7 Interrupts, Sleep
8 Timer 2
13 Programmiergerät
16 Impressum

Mitgliederwerbung

Die VD beabsichtigt in verschiedenen auflagenstarken Zeitschriften Kleinanzeigen unterzubringen, um neue Mitglieder zu gewinnen. Neue Mitglieder bringen neue Ideen und neues Geld in den Verein. Nebenstehend ein Layoutvorschlag von Rafael Deliano. Die Größe einer Anzeige sollte 62 x 43 mm (1/16 Seite) nicht überschreiten. Alle Mitglieder sind aufgerufen, bis zur Tagung weitere Entwürfe an die VD zu schicken. Auf der Tagung sollten wir uns dann entscheiden, welches Motiv dieses Jahr verwendet wird. Weiterhin möchten wir alle Mitglieder bitten, die Kontakt zu geeigneten Zeitschriften haben, hatten oder vermitteln könnten, sich mit uns in Verbindung zu setzen.

Claus Vogt, Editor, Tel.: 030-786 84 60

Die

Einladung

zur ordentlichen Mitgliederversammlung
der Forth-Gesellschaft e. V.

am
Sonntag, den 27. April 1997 um 9^{00} Uhr,

im
Europa-Hotel
Am Ludwigsplatz 5-6

finden Sie übrigens auf Seite 37 ...

1	14.133	22.03.97	1:23	mfil971.lst
2	3.052	22.03.97	1:32	inh971.lst
3	3.506	22.03.97	1:40	vdl971.lst
4	31.976	16.02.97	22:00	graphik\fastgraf.mod
5	8.232	16.02.97	22:00	graphik\fgbitmap.mod
6	5.104	16.02.97	22:00	graphik\fgflic.mod
7	2.488	16.02.97	22:00	graphik\fgkeyb.mod
8	15.256	16.02.97	22:00	graphik\fgmemory.mod
9	4.200	16.02.97	22:00	graphik\fgmouse.mod
10	3.176	16.02.97	22:00	graphik\fgpcx.mod
11	4.928	16.02.97	22:00	graphik\fgsound.mod
12	2.584	16.02.97	22:00	graphik\fgsprppr.mod
13	30.811	29.01.96	13:32	lowdeb\deco6811.cfw
14	6.315	16.01.96	11:00	lowdeb\dis6811.cfw
15	17.790	14.02.97	22:56	lowdeb\masklist.cfw
16	8.562	29.01.96	14:43	lowdeb\trc6811a.cfw
17	2.813	29.01.96	14:43	lowdeb\trc6811g.ffg
18	5.339	14.02.97	22:57	lowdeb\trc6811t.fft
19	2.305	2.03.97	14:30	lowdeb\trc6811t.txt

Anzahl Dateien: 19 Anzahl Bytes: 172.570

= Listing auf Diskette

Leserforum

Gefunden!

Beim Stöbern in Literaturverzeichnissen stieß ich auf den ‚Forth Schnellkurs" von C. Lorenz, F.Floegel, F. Ende und W. Hofacker. W.Hofacker 1989 ISBN 388963-264-5 (knappe 50 Seiten, hauptsächlich zu Laxen&Perrys F83). Man muss ihn nicht kaufen, aber in den Beispielsourcen gibt es eine Perle:

```
: eins ." Eins" ;       : zwei ." Zwei" ;
: drei ."Drei" ;        : vier ." Vier" ;
: Exec ( na -- ) swap 2* + perform ;

Create Liste ] eins zwei drei vier [

: Tst ( n -- ) Liste exec ;
```

Ich habe es bisher so gemacht:

```
Create Liste ' eins , ' zwei , ' drei , ' vier ,
```

Die Idee, den Compiler via] und [nochmal ein- und wieder auszuschalten hat mich überzeugt. Das ist es, was ich elegant nenne! P.S. Klappt in ZF, F83, Win32For!

M. Bitter, Feb'97

Nachwuchsspalte

Lieber Claus Vogt,

bei mir ist es so, daß ich ein recht unorganisierter Mensch bin. Deshalb kommt es häufig vor, daß ich Termine versäume oder es ganz verpasse auf etwas zu reagieren. Es tut mir leid, wenn ich Sie durch meine fehlende Antwort auf die Korrekturfahnen zur 64constant-Falle beunruhigt habe. Wenn auf eine Korrekturfahne von Ihnen keine Antwort meinerseits erfolgt, können Sie davon ausgehen, daß ich mit Ihren Vorschlägen ganz und gar einverstanden bin. So auch diesesmal! Nicht nur das, Ihre Ergänzungen (Kasten) stellen eine Bereicherung dar, ihre Korrekturen stützen und schützen mich.

Leider ist es so, daß ich zwar einige - wie ich glaube - schöne Codestücke für die VD habe, aber der letzte Schliff, die redaktionelle und darstellerische Aufbereitung mich viel Überwindung und Zeit kostet.

Um mich selbst zu verpflichten, sende ich Ihnen ein kompiliertes Programm mit (Salesman.exe), dessen Quellcode bestimmt auf mehrere Ausgaben verteilt werden muß. Einfach mal laufen lassen. Evtll. kann der ‚SALESMAN" ja auch der Aufhänger für eine Nachwuchspalte sein. Dazu bräuchte ich aber Mitstreiter. Persönlich kenne ich aus der Forth-Gemeinde Fritz Prinz und seine Moerser Gruppe. Aus der VD ist mir T. Beierlein als Mensch mit Nachwuchsinteresse bekannt.

Vielleicht ist es Ihnen möglich, mir noch zwei weitere potentielle Helfer zu benennen und einen ersten Kontakt zu vermitteln?

Über das wie und wann können wir uns bei Gelegenheit austau-schen. (Telefon?)

Es wäre für mich nützlich eine Aufstellung der Einsendetermine für die nächsten Vds zu haben.

Ein persönliches Wort zum Schluß: Ich bin sehr froh, daß Sie uns als Editor der VD erhalten bleiben. Ihre Briefe und Kommen-tare waren für mich stets angenehm.

mit frdl. Gruß Martin Bitter,
Möllenkampweg 1a, D-46499
Hamminkeln, 12.2.97

[Wer die Idee mit der Nachwuchsspalte unterstützen möchte, sollte sich mit der Redaktion in Verbindung setzen. Die gewünschten Termine des Redaktionsschlusses stehen im Impressum auf Seite 2. Es ist jeweils die erste Woche im Quartal. Kurzmeldungen dürfen auch etwas später kommen. /clv]

Generative Programmierung?

Das allesbeherrschende grundlegende Paradigma unserer Zeit scheint zu lauten: "Haltet alles in Fluss durch die laufende Erfindung neuer Paradigmen". Das Paradigma der Objektorientierung scheint jetzt durch das Paradigma der Generativen Programmierung abgelöst zu werden (?) Meine Frage: Was ist Generative Programmierung?

Weitere Frage: Wer schreibt uns, den Lesern der VD, einen forthbezogenen Artikel mit Beispielen über Generative Programmierung? Wer bringt Ordnung in die Begriffsvielfalt Objektorientierung, Komponentenorientierung, Generative Programmierung, Reverse Engineering und Wiederverwendbarkeit?

beh (Fred Behringer, München)
Feb'97

Zu "Die 64-CONSTANT-Falle in ZF" von Martin Bitter in der VD 4/96

Gute Idee, das zu vermelden! Was für ein Schreck: In meinem Transputer-Forth-System F-TP 1.00 (liegt in ftp://ftp.statistik.tu-muenchen.de/incoming/Forth) habe ich natürlich auch 0, 1, 2 und 3 vordefiniert. Der Schaden hält sich aber in Grenzen: Die von Martin Bitter aufgedeckte Schwierigkeit kann nur dann auftreten, wenn die vordefinierte Zahl mehrziffrig ist. Gehe ich beispielsweise auf Basis 2 und gebe 3 ein, dann liefert .S die Bildschirmanzeige 11. 3 ist vordefiniert und wird in diesem Fall einfach in Binärdarstellung ausgegeben.

Das ist akzeptabel, da es ja eigentlich so gemeint war. Wird 4 oder eine höhere Zahl eingegeben, so erscheint eine Fehlermeldung. Das hat auch seine Richtigkeit. 4 ist nicht vordefiniert und als Zahl, über NUMBER interpretiert, hat es in Basis 2 keine definierte Bedeutung. Gibt man umgekehrt die vordefinierte Zahl 2 in irgendeiner Basis höher als 2 ein, so bedeutet sie trotzdem immer dasselbe: 2 Einheiten. Erst wenn beispielsweise 10 (mehr als eine Ziffer) unter DECIMAL vordefiniert wird, dann hat die Eingabe 10 unter beispielsweise HEX eine falsche Wirkung.

beh (Fred Behringer, München)
Feb'97

Manöverkritik 4/96

Es war eine gute Idee, die Farbe der Briefmarke auf Seite 9 zu invertieren, da sie schwarz wesentlich besser zur Wirkung kommt. Jedoch sollten 2seitige Artikel immer auf einer geraden Seitenzahl anfangen. Man hat dann ohne Umzublättern den gesamten Text vor sich. Insbesondere wird dadurch die Zuordnung von Bild zu Text unkritisch.

jrd (Rafael Deliano) Jan'97

Schach

Am Flohmarkt "The BYTE Book of Pascal" von 1979 gefunden. Enthält Tiny Pascal für 8080 (threaded code natürlich). Und vor allem auch einen umfangreichen vierteiligen Artikel von Peter Frey und Larry Atkin über Schachprogrammierung samt Pascal-Source.

Leserbriefe:
Am liebsten kurz.Sonst trifft uns die Pflicht zur Kürzung.
Die Redaktionsadresse lautet:

Forth-Magazin 'Vierte Dimension'
Claus Vogt, Katzbachstr.23,
D-10965 Berlin, vd@FORTH-ev.de

Forth. Kurz und Knapp. **Das kommt**

Nützlich für jemand, der sowas programmieren will.

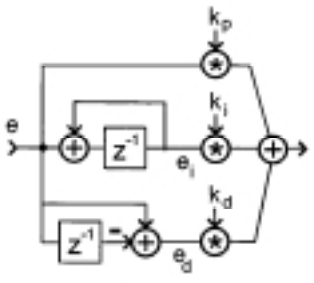
jrd (Rafael Deliano) Jan'97

zu 'N'bißchen wat Praxis' über PID von Wolfgang Führer in VD 4/96, Seite 31

Mein Artikel in VD 1/94 war nur eine Einführung, nicht der definitive Artikel über digitale PID-Regler, anhand dessen man implementieren sollte. Das angegebene PID-Modell ist an sich funktionsfähig. Es ist die übliche Parallelform wie sie z.B. im Motorregler-IC LM628 zur Anwendung kommt. Die I- und D-Anteile werden nach Rechteckregel berechnet. Die Einschätzung im damaligen Text, daß Trapezregel nicht mehr verwendet wird, ist wohl nicht richtig, da SIMATIC S5 nach Trapez rechnet. Bei realen Implementierungen sind für den D- und I-Anteil einige Tricks nötig, die in der vereinfachten Source des Artikels nicht berücksichtigt sind.

Das von Führer dargestellte Problem, daß bei Verzögerungen auf der Strecke, d.h. Totzeiten, Regler instabil werden können, ist wohlbekannt und gilt für alle linearen Regler.

Man kann zwar, wie es Führer macht, neue handgestrickte Reglerstrukturen erfinden. Aber unsystematische Lösungen passen dann eben



auch nicht für andere Regelstrecken und es gibt keine Einstellregeln für sie.

Aus dem Fundus für "Creeping FORTH", aus dem auch der Text von 1/94 stammt, liegt bei mir noch tief in der Schublade ein 5 seitiger Text zum Implementieren von PID. Aber da ich zu dem Thema immer noch in der Literatur stöbere, würde der auf 10-15 Seiten erweitert werden müssen, bis er wirklich den Stand der Technik widerspiegelt und als Grundlage für Implementierungen dienen könnte. Da das Thema aber doch etwas speziell ist, ist er dann zu lang für die VD.

Eine gute Literaturempfehlung zu machen ist auch schwierig. Es mangelt gewiß nicht an Literatur zum Thema Regelung, aber das meiste sind akademische Formelwüsten, die für Praktiker unverdaulich sind. Einen relativ guten Eindruck macht: Josef Giller "Vom Prozeß zur Regelung" Siemens AG 1990 ISBN 2-8009-1551-0

jrd (Rafael Deliano) Jan'97

Anfänger und Bücher

vgl. Stoyke: VD 3/96, Seite 8
und Freitag: VD4/96, Seite 6

Wie Klaus Seegebarth einmal richtig gesagt hat, gibt es in FORTH nur

zwei Benutzergruppen: die ewigen Anfänger und die Profis. Und kein Mittelfeld dazwischen. Bezüglich der FORTH e.V. ist es so, daß sich hier die Anfänger tummeln, weil ihr die Profis davonlaufen.

Artikelserie in VD

Stoyke hat in VD 3/96 den Wunsch geäußert, der Verein solle allgemeine Literatur publizieren. Das ist jedoch nicht nötig, solange man noch den deutschen Brodie für 50 DM in der Buchhandlung bestellen kann. Damit hat man eine allgemeine, gute Einführung, basierend auf FORTH 83. Ein mit der Fotokopiermaschine produzierter Text würde leicht ähnliche Kosten, nicht aber die Qualität erreichen. Eine Artikelserie in der VD ist nicht praktikabel. Zerlegt man ein Buch von 200 Seiten in Artikel zu 8 Seiten, hat man die Serie in 6 Jahren durch. Eine Kurzfassung z.B. als Sonderdoppelheft mit 50 Seiten, wird der Komplexität des Themas nicht gerecht. Es gibt zwar derzeit kein Einführungsbuch (weder Deutsch, noch Englisch) für ANS. Da der Anfänger aber nur den Grundbefehlsumfang benötigt, sind die Unterschiede zu FORTH 83 nicht zu gravierend.

Compilerhandbuch

Leider wird der Anfänger auch mit dem Brodie nicht weit kommen, wenn sein Compiler kein Handbuch hat. Damit wären wir bei Freitag (VD 4/9), der ein Handbuch für ein PD-Forth will. Für PD gibt es per definition kein Handbuch und keinen Support. Wenn PD für überhaupt jemanden geeignet ist (ich bezweifle es generell), dann bestimmt nicht für Anfänger.

Bezüglich F-PC hat er allerdings noch Glück, da hier mehrfach die Kommerzialisierung versucht wurde. Es gibt von Staben eine Loseblattsammlung von 270 Seiten Umfang, die auch optisch gut aufbereitet ist und nur mangels Verlag kein Buch wurde. Da Staben meist auch Zeitschriften mit seinem Talent beglückt, sind die Kapitel etwas kurzatmig geworden. Zudem schweift er gerne vom Kernthema FORTH ab. Dem selbstgesteckten Ziel, auch gleich eine Einführung in FORTH zu sein, wird der Text durch die unübersichtliche Gliederung nicht gerecht. Zusammen mit dem Brodie und Stabens Diskette(n) kann man jedoch damit preiswert FORTH programmieren lernen.

Generell anders geht Klingelnberg das Problem an. Der Stand von 1991 seiner Dokumentation der mir vorliegt, ist möglicherweise nicht mehr aktuell. Weil F-PC eben sehr umfangreich ist, hat er die Dokumentation in drei stabilen Heftern untergebracht. Leider verwendet er, um noch mehr hineinzupacken, ein etwas schwer lesbares Querformat. Der Inhalt ist überwiegend ein Ausdruck der englischen Dokumentation. Insofern weniger für Anfänger geeignet.

Neben diesen halbkommerziellen Lösungen zum Erlernen von FORTH, gibt es durchaus noch einige kommerzielle FORTHs aus deutschen Landen mit deutschem Handbuch, z.B. big-Forth oder comFORTH. Teilweise mit reduzierten Varianten zu reduziertem Preis, was besonders für Anfänger von Interesse sein sollte.

Warum wissen die Anfänger nicht, was verfügbar ist ?

Erstens weil in der VD recht selten in Form von Produktbesprechungen und Produktübersichten darüber berichtet wird. Das ist bedauerlich, aber ein Fehler der Anbieter selbst. In der Produktbesprechung des IX1 war ein Aufruf enthalten, Produkte für Tests zur Verfügung zu stellen. Die Redaktion ist daraufhin nicht eben damit überflutet worden.

[Leider nutzen Firmen im Forthumfeld unser Angebot praktisch nicht, Produkte oder Entwicklungen ihrer Firma in der VD selbst vorzustellen oder Produkte für unabhängige Tests zur Verfügung zu stellen. Fast alle diesbezüglichen Beiträge der letzten Jahre entstanden aufgrund von Initiativen unserer Autoren. Trotzdem versuchen wir weiter, die entsprechenden Rubriken zu erweitern. / clv]

Zweitens ist das Angebot kommerzieller oder auch nur halbkommerzieller Produkte spärlich geworden. Der Verein ist daran nicht unschuldig. Seit den Gründertagen von volksFORTH wird in der VD mit großem Eifer PD-Forth propagiert. Ohne daß der Pferdefuß von PD je zur Sprache kommt. Da ein hochgelobter kostenloser Compiler sich besser verbreitet, als ein hochgelobter kostenpflichtiger Compiler, sind die wenigen Anbieter aus dem Verein verschwunden. Die VD erscheint deshalb inzwischen praktisch anzeigenfrei. Zudem haben sich viele Anbieter auch aus dem unteren Preissegement zurückgezogen und unterstützen hauptsächlich industrielle Nischen für zahlungskräftige Kundschaft. Letztlich ist der Verein zwar auf die vielen Anfänger angewiesen, da man nur über hohe Mitgliedszahlen die VD in ihrer heutigen Form finanzieren kann. Der implizite Köder für die Mitgliedschaft, nämlich, daß man anhand der VD oder den Aktivitäten des Vereins aus den unbenutzbaren PD-Forths plötzlich vollwertige Werkzeuge machen kann, ist leider unerfüllbar. Die letzten 12 Jahre Vereins(un)tätigkeit beweisen das gnadenlos.

Mikrocontroller

Nur ein EPROM und eine Leiterplatte zu liefern, wie von Freitag vorgeschlagen, wird viele Anwender vor Probleme stellen, da die Definition, was ein "handelsübliches" Teil ist, doch etwas auseinandergeht. Durch zentralen Einkauf der Teile hat der Hersteller eines Boards den besseren Bauteilpreis und das Teil paßt auch sicher. Liefert man das Board als Bausatz, hat der Hersteller das lästige Problem eine Unzahl von Kleinteilen geeignet zu verpacken. Insbesondere kann er aber das Material vor Auslieferung immer noch nicht auf Funktion testen. Trotzdem bleiben Gewährleistung und Reparatur an ihm hängen. Letztlich sind also bestückte Boards die gangbarste Lösung.

Es kann jedoch keine speziellen VD-Projekte geben, da industrielle Boards teuer sind. Mit Layouter, Fotoplots und Rüstkosten liegt man >1k DM ohne Boards. Nur bei nennenswerten Stückzahlen wie sie Zeitschriften wie die ELRAD mit >20k Lesern bieten, ist damit Kostendeckung erreichbar.

Zudem ist auch kein wirklicher Bedarf erkennbar, solange der Controller-Verleih so selten benutzt wird.

jrd (Rafael Deliano) Jan'97

MARC4

Ähnlich wie bei den anderen Herstellern war der Stand der TEMIC auf der "Elektronica 96" dieses Jahr etwas nüchtern. Man konzentrierte sich auf wenige neue Produkte. Der MARC4 wurde im Bereich Automobilelektronik mit Anwendung in Transpondern und "Keyless Entry"-Systemen gezeigt. Die Entwicklung der Transponder ist noch von Eurosil begonnen worden. Keyless Entry wird alternativ als Infrarot- und Radio-

System angeboten. Die AnalogICs für diesen Teil kommen von Telefunken, die ja traditionell Hf-Chips für Konsumeranwendungen und Optoelektronik fertigt. Der MARC4 ist damit in bestimmte Systemlösungen der TEMIC eingebettet. Besser dieses Marketing als keines. Zwangsläufig wird sich der Newsletter auch mit diesen speziellen Themen befassen müssen, denn neue potentielle Anwender werden den Chip mit Blick auf diese Anwendungen aufgreifen.

Was ist ein Transponder?

Ein Transponder ist ein Datenträger, auf den auch aus einiger Entfernung zugegriffen werden kann. Eine übliche Bauform ist ein winziges Glasröhrchen, das den Chip und eine Ferritantenne enthält. Die Antenne im Lesegerät baut ein Hf-Feld (etwa 125 kHz) auf, durch das der Transponder mit Strom versorgt wird und Daten aus seinem Speicher gelesen werden können. Bei komplexeren Typen kann man das EEPROM auch schreiben. Der MARC4 kommt im Lesegerät zum Einsatz. Die Spezialität von TEMIC ist der Krypto-Transponder der eine chiffrierte Datenübertragung hat. Damit kann im Auto eine Wegfahrsperre realisiert werden, wenn der Transponder in den Zündschlüssel eingebettet wird. Dieser Schlüssel ist fälschungssicher und ohne seine chiffrierten Daten aktiviert die Autoelektronik den Motor nicht.

Was ist Keyless Entry?

Am besten wohl mit "elektronischer Schlüssel" übersetzt. Der Vorteil gegenüber dem herkömmlichen Schlüssel ist die Fernauslösung und die Fälschungssicherheit. Die Technik entspricht weitgehend einem Handsender bei einer Fernseh-Fernbedienung. Die Übetragung kann Infrarot oder über Funk erflogen. Der MARC4 kommt in den Handsender. Um eine kompakte Bauform zu erreichen, wie sie der Anwender von einem Schlüssel erwartet, verwendet man eine kleine Batterie und eine sparsame CPU. Andererseits muß genügend Flexibilität und Rechenleistung vorhanden sein, um den Chiffrieralgorithmus auszuführen.

jrd (Rafael Deliano) Jan'97

Forth-Gesellschaft: Mikrocontroller-Verleih Neu: CCD-Zeilenkamera

Die bisher verfügbaren Boards kommen dem Spieltrieb des Anwenders leider nicht sehr entgegen, weil keine Hardware mit typischen Anwendungen für Controller verfügbar ist. Für das nächste halbe Jahr steht nun eine Demoschaltung für einen Sony CCD-Zeilensensor zur Verfügung. Kern der Schaltung ist ein ProtoIII Einplatinencomputer auf dem F74 nanoFORTH läuft. Die 2048 Pixel des CCD-Bildsensors werden mit 8 Bit digitalisiert und im Speicher abgelegt. Mit der integrierten Software können sie dann am PC ausgegeben werden. Da FORTH online verfügbar ist, kann man die Software auch erweitern und z.B. Kantenerkennungsalgorithmen implementieren. Zum Betrieb braucht man irgendeinen Rechner mit einem VT52 Terminalprogramm (oder VT100, VT220) das 25 Zeilen à 80 Zeichen darstellen kann. Ferner wird ein V24-Kabel mit 9-Pin-Belegung auf Modemseite benötigt. Netzteil wird mitgeliefert. Derartige CCD-Sensoren kommen in Faxgeräten und Scannern zur Anwendung. Das Demoboard ist für Leute mit Interesse an Bildverarbeitung und Optoelektronik interessant.

jrd Jan'97

Firmen: Cinetix mit dibox

Wolfgang Schemmert, Ex-VD-Schreiber und Ex-Mitglied gibts immer noch. Sein Produkt "dibox" ist eine Familie von Steuerungskomponenten für den Bühneneinsatz im robusten Metallgehäuse. Die Controller ermöglichen die feldbusartige Vernetzung der Ausgabemodule, nicht nur über MIDI und DMX, sondern auch über CAN. Die Endstufenmodule sind z.B. Dimmer mit einprogrammierten Steuerkennlinien für Glühlampen oder Audio-Fader. FORTH versteckt sich hinter Formulierungen wie im "ASCII-Textformat programmierbare Controller" und wird dem Kunden vorzugsweise als "Java- ähnliche virtuelle Stackmaschine" empfohlen.

CINETIX GmbH
Gemündenerstr.27
60599 Frankfurt
Tel 069/825957
Fax 069/82375540

jrd (Rafael Deliano) Feb'97

Termine: Echtzeit'97

Die Echtzeit '97 findet zeitgleich mit der MessComp '97 vom 9. - 11. September in Wiesbaden statt.

Ein Stand für die Forth Gesellschaft ist bereits reserviert.

Über den Programmierwettbewerb habe ich noch nichts in Erfahrung bringen können.

Ich halte Euch auf dem laufenden.

Clemens@clforth.forth-ev.de (Winfried Clemens) in de.comp.lang.forth Jan'97

ANS-Forth: Nun auch ISO

Einer Nachricht von P. Knaggs zufolge, ist FORTH nun auch ISO standardisiert, und zwar unter dem Kürzel ISO/IEC DIS 1514. Nähere Informationen dazu findet man unter: http://www.iso.ch/cate/d26479.html

Über den Umfang ist nicht viel zu erfahren, aber da - wie in VD 3/96 Seite 37 erwähnt - ANS Forth im 'fast track' in das ISO Verfahren eingebracht wurde, wird sich der ISO Standard wohl mit ANS Forth decken. Hat jemand Lust auf DIN-Forth?

Ulrich Hoffmann (uho@informatik.uni-kiel.de) Jan'97

Forth inside: Forth and Javacode

I was reading a book published by Javasoft on the Java virtual machine and it doesn't sound all that different from the virtual machine used by FIG-like Forth's. Has anyone written a Forth which compiles to Java bytecode? (actually, you could use just about any language to create Java applets if you had the right compiler)

jmrubin@ix.netcom.com (Joel Rubin) jan'97

That said, it is possible to write something Forth-like using JVM bytecodes, but you can't use the JVM stack to implement the Forth stack. The JVM was designed really only for executing Java [exact reference not to hand] and other procedural, OO languages. However, a number of other languages have become available for it. I am currently implementing Modula-J, an in-house language derived from Modula-2, which is targetted to the JVM.

Paul Curtis <paul.curtis@ra-ltd.demon.co.uk>

Well, this probably isn't exactly what you meant, but I recently wrote a VM *in* Java which was based on the F21 chip whose instruction set is based on Forth's. (Which makes sense as the chip was designed by Charles Moore.) The lack of unsigned types was indeed a big pain but other than that it seems to be working. The VM is part of a system for breeding genetic algorithms to play Go. I may be cracked but I have this vision of someday actually building a machine out of a bunch of F21s especially for running this sort of system. (A massively parallel MISC machine???) But I don't know diddly about

actually putting together hardware so that's probably just a pipe dream.

seibel@sirius.com (Peter Seibel)

Personally, I'd like to see a Web Brouser Plugin which is a simple, sturdy machine independant Forth. I think that would be a much simpler approach, than to try and meld JAVA and FORTH.

I'd build it along the lines of PocketForth, or PigmyForth. Then I'd create the ability to stream code from a Website in the form of byte tokens.

This opens up the possibility of combining data and code into a single file which could easily be linked into a webpage, which the WebForth could then interprete.

Any ideas on this concept can be discussed here, or in email.

Jason (kodogr@typhoon.xnet-.com) Damisch

I know that in the RFC for Assigned Numbers there is an entry for FORTH. Perhaps somebody out here knows how to secure a MIME type for forth?

Reinout Heeck (reinz@Desk.nl)

In that case perhaps you might like to take a look at my web page, http://www.cl.cam.ac.uk/users/rrt1001/ where you can find Beetle, a VM designed to run Forth, which is:

- written in pure ANSI C [1]
- embeddable (can be used from within other C programs)
- robust and secure (can only access its own memory area unless you let it call the C program or provide library routines to do otherwise)
- fast (for an interpreted system)
- small (for a C program)

and pForth, an ANSI Forth compiler that runs on Beetle. A paper about the system will hopefully appear in ACM SIGPLAN Notices soon.

rrt1001@cus.cam.ac.uk (Reuben Thomas)

Have a look at the Misty Beach Forth Java Applet: http://www.misty-beach.com/

Not a plugin but a downloaded interpreter. Still in its early days but still worth having a look.

pjk@paisley.ac.uk (Peter Knaggs)

I did take a look, both on my Pentium running Netscape under WindowsNT, and on my PowerWave 132 (Mac Clone) running Netscape under MacOS 7.5.5.

It worked in both places. Limited functionality, but I like what it there.

When can I get the source code?

I had to ask!

ZForth@ix.netcom.com (Tom Zimmer)

[zusammengefaßt nach comp.lang.forth Jan/Feb'97 ('Forth and Javacode') von clv]

Forth inside: OpenFirmware

In article <32EFCE6B-.3633@spur.com>, "Clyde W. Phillips Jr" <cphillips@spur.com> wrote:

"OpenFirmware is FirmWorks (Mitch Bradleys) IEEE 1275 Firmware Forth system. I'd like to know which computers come with it these days but I'd *really* like to know if there are any PD projects where on could install as the booting firmware or better yet have a boot emulation environment that runs the firmware."

All PowerPC Macintosh's come with Open Firmware installed. Attach a serial cable from the modem port to a terminal set at 38400 N81, hold the command-option-O-F keys and you will pop into Open Firmware before the OS loads. The OF installed is an Apple version of the 1275 spec.

davec@apple.com in / comp/lang/forth, Feb'97

Forth inside: Open Boot Mac

Alle, denen die Benutzung ihres Apple Macintosh' schon immer zu kompliziert war, werden sich für die folgende Diskussion interessieren:

Now that I've got access to a new Power Mac with the PCI bus is there anyway I can do something during the boot sequence (like a I can with a sparcstation) so I can astonish my friends (and mangement:-) and show them Forth is still being used even in new, high volume products?

mef@aplcenmp.apl.jhu.edu (Marty Fraeman)

I doubt PCI Macs have anything remotely ressembling Forth boot code...

verec@micronet.fr (Jean-Francois Brouillet)

But they do, the OpenBoot is accessible over the serial port only.

regnirps@aol.com (Regnirps)

What about Power-PC systems from IBM running AIX? (We will be taking delivery of one in a few weeks.)

cjakeman@apvpeter.demon.co.uk (Chris Jakeman)

You will find the Open Boot Forth on any PCI bus computer. It is the mechanism for PCI "Plug and Play".

regnirps@aol.com (Regnirps)

The MACWORLD Expo in San Francisco in January was full of MAC's with Open Firmware on them.

RSL@forth.com (Randy Leberknight)

According to my sources at Apple, starting with 7200 all Macs have OF in the ROM. Currently that includes the 7200, 7500, 8500, and 9500.

erather@earthlink.net (Elizabeth Rather)

Is the 7200 a PCI Mac? I think it is 'option-o-b' all held down at powerup (or command-o-b or command-option-o-b?). But, the Open Boot communication is at 38Kbaud out the serial port, not to the screen! Hook up another computer with a terminal program.

regnirps@aol.com (Regnirps)

Hook up a terminal to the modem port, set it to 38400 8-0-1 and hit Command-Option-O-F. If you have on-board video you can switch the display to it by

"vci0/control/ output

and the keyboard by

"kbd" input

davec@apple.com (Dave Carlton)

nach comp.lang.forth Februar 1997 und Juni 1996: 'Finding open firmware on a PCI mac' und 'Open Boot on Apple Power PC' (clv)

Presseschau: DDJ 97/2 Carter robottet wieder

Von Klaus Kohl wurde der Artikel "Robots and Finite-State Machine" in DDJ 97/2 gefunden. Der Autor Everett Carter, Präsident der FIG, gibt einen hochinteressanten Einblick in die Probleme beim Bau gehender Roboter. Programmiert wurde in FORTH, als Controller dient ein 68332-Board von New Micros.

jrd (Rafael Deliano) Feb'97

Forth Online: electronic VD

Ich lasse mal den Original-Text intro.htm für sich sprechen...

Was lange währt, wird manchmal doch noch angefangen. So ähnlich ging doch wohl der Spruch.

Es geht irgendwann zum Ende des Jahres '95 zurück, als Claus Vogt und ich angeregt email unter dem Betreff 'Spinnerte Ideen...' austauschten. Es ging da um Möglichkeiten der Selbstdarstellung der Forth-Gesellschaft im Internet, eine geplante CDRom-Produktion, die Archivierung der gewesenen Ausgaben der "Vierten Dimension" und vieles andere...

Ich hatte damals eine Ausgabe der c't-Rom, ein Archiv des gleichnamigen Computermagazins im HTML-Format, und schlug Claus ein ähnliches Vorgehen für unsere Vereinszeitung "Vierte Dimension" vor. Wir waren der Meinung das HTML-Format - obwohl nicht eben forthtypisch - wäre geeignet für ein ziemlich

plattform-, hardware- und medienunabhängiges Projekt der Forth-Gesellschaft.

Es vergingen noch einige Monate - exakt bis zur Forth-Tagung '96 in Mittweida - um von neuen Diskussionen angeheizt, mit ersten praktischen Übungen zu beginnen. Claus Vogt hatte mir erfolgreich die Konvertierung seiner DTP-Dateien des Jahrgangs '95 der VD eingeredet. Ich arbeitete mich langsam in die Vielfalt der HTML-Tags ein, dann in die noch größere Vielfalt der HTML-Authoring-Software, und probierte eine Unmenge Stile aus (was man in dem einen oder anderen Artikel vielleicht immer noch sehen kann).

Mittlerweile ist der Jahrgang 1995 fertig geworden. Wie es meine Zeit erlaubt, werde ich auch das Jahr '96 in Angriff nehmen. Ein bißchen Feedback (auch mit Kritik und Verbesserungsvorschlägen) würde mich sehr freuen und meine Motivation steigern!

Meine Adresse ist:
Michael Schröder,
Magdeburger Str.26,
39387 Oschersleben,
email:
MSchr90828@aol.com oder
mis@kitchen.forth-ev.de

Zu finden ist das VD-Archiv des Jahrganges '95 unter http://www.informatik.uni-kiel.de/~uho/VD/

Also wie gesagt, vielleicht guckt sich der eine oder andere das Ganze mal an und schreibt hier in d.c.l.f. oder an mich darüber.

(Alle “links” zum folgenden Jahrgang ‘96 gehen übrigens noch ins Leere, das ist nicht schön aber trotzdem Absicht gewesen, um mir ein Update der schon fertigen Dateien zu ersparen...)

mis@kitchen.forth-ev.de
(Michael Schröder)
in de.comp.lang.forth 20.2.97

Forth-Gesellschaft: Neujahrsansprache der Mailbox

Nun ja, etwas verspätet; aber auch ich möchte mal eine kleineNeujahrsansprache verschicken...

Zuersteinmal: Das Jahr 1996 hat die KBBS eigentlich ganz gut überstanden. Ein paar mal allerdings mußte ich sie mit einem RESET dazu überreden, weiterzumachen. Im Sommer gab’s eine neue Harddisk. Die ständig zu volle 1 GByte wurde durch eine doppelt so große ersetzt. Leider ist mir mein EXABYTE-Bandlaufwerk kaputt gegangen; derzeit wird nun der Backup auf eine separate Harddisk gemacht. Nur könnten auf dem Platz natürlich schönere Sachen lagern...

Seit August ist auch ein direkter Zugang vom Internet aus möglich. Man kann zum Beispiel: http://www.kbbs.org in einem Browser eingeben; dann bekommt man eine kleine Seite mit Informationen zur KBBS und ihrem Kieler Umfeld. Ebenso natürlich ein Hinweis auf die Forth-Gesellschaft. Insbesondere habe ich das gemacht, weil ein Kawasaki-Dreizylinder-Motorradclub da mitmacht. Denn das RIESEN-Problem ist die Finanzierung: Jeder Anruf aus der ‘weiten Welt’ kostet mich (mindestens) eine Telefoneinheit plus die Gebühr für die übertragenen Byte’s. Wer allerdings eine eigene Homepage hier haben will, kann sich durchaus mal mit mir in Verbindung setzen...

Eventuell gibt es demnächst ja mal eine Standleitung...

Bis dahin werden die Newsgruppen und auch die E-Mails weiterhin per UUCP vom Provider geholt. Das Verfahren dazu möchte ich mal aus der Box-Sicht erläutern:

Wann immer ein Anruf (rein- oder raussgehend) zu neuen Nachrichten geführt hat, werden die bei E-Mail sofort in das betreffende Ausgangs-Verzeichnis gepackt; bei News erstmal in die betreffenden Unterverzeichnisse als Datei mit fortlaufender Nummer einsortiert. Dabei wird auch nachgeschaut, ob ein Downlink diese Nachricht bekommen soll. Wenn ja, wird in einer Liste (pro System) eingetragen, daß die Nachricht noch weiterverteilt werden muß.

Nach einstellbaren Zeiträumen werden dann die hier abgelegten Nachrichten gelöscht; ich habe eben keine unendliche Speicherkapazität...

In regelmäßigen Abständen wird nun diese Liste abgearbeitet. Zuerst überprüft das System, ob im Ausgangsfach noch genügend Platz ist. Dabei wird sowohl eine maximale Zahl an Dateien als auch ein maximaler Platzbedarf gemessen. Nur wenn beide Zahlen kleiner als das (einstellbare) Maximum sind, werden die vorgemerkten Artikel auch eingepackt. Dies passiert bei mir derzeit alle 20 Minuten.

- Wenn jemand selten pollt, kann es sein, daß beim Anruf die alten Nachrichten übertragen werden; und ein paar Minuten später (manchmal noch während des Anrufs!) dank freiwerdenden Platzes wieder neue Nachrichten eingepackt werden.

- Wer *zuverlässig* aber selten pollt, kann seine Limits höhersetzen (lassen).

- Wenn was nach dem Absenden beim Empfänger schiefgeht, kann ich nicht einfach “das Datum” oder einen sonstigen Merker zurücksetzen, weil es sowas nicht gibt. Bei zu alten Nachrichten kann es sogar sein, daß die eben schon gelöscht sind; dann kann man auch mit (mühevoller) Handarbeit nix mehr ‘nochmals’ bereitstellen.

MMF steht für ‘Make Money fast’, im übertragenem Sinn für jede Art von Missbrauch in Newsgruppen. Diese Nachrichten werden nun erstmal (weltweit) übertragen. Sehr bald findet sich aber jemand, der diese, wenn sie denn bestimmten formalen (nicht Inhaltlichen!) Kriterien genügen auch weltweit löscht. Das ist das sogenannte “Cancel’n”. Für *eigene* Nachrichten kann man das problemlos aus jedem News-Leser auch selbst machen. Das Problem dabei ist eben der zeitliche Versatz zwischen Original-Nachricht und Cancel. Zwar wird die Msg bei mir gelöscht; sie ist aber (meist) schon im Ausgabe-Bereich eingepackt.

UCE steht für ‘unsolicitated commercial Email’, also für die ‘unerwünschten Werbesendungen’. Die werden beim uucp-Verfahren eben sofort in den Ausgangspuffer geschrieben.

Es gibt Ansätze, solche Mails gar nicht erst zu empfangen. Das geht aber prinzipiell nur auf der Internet-Ebene. Und da ich keine Standleitung habe, muß mein Provider die Nachrichten erstmal an mich weiterleiten. Das große Problem an der Idee ist aber, eine Werbe-Mail automatisch als solche zu erkennen. Die Absender wechseln ja dauernd; und eine ‘künstliche Intelligenz’ zum Inhalte-Erkennen ist wohl noch weit weg. Die o.a. Versuche nehmen einfach eine ganze Absender-Domain als Anlaß, solche Nachrichten nicht zu empfangen.

BITTE BITTE bleibt ruhig, wenn Ihr so eine Mail bekommt oder in einer Newsgruppe lest. Oft ist der Absender gefälscht oder (in der Zwischenzeit) nicht mehr gültig. WENN einer dann auf die Idee kommt, viele oder lange Dateien zurückzusenden (mit dem Motto “Wie Du mir so ich Dir”) dann kommen die Nachrichten oft als ‘unzustellbar’ zurück - und ich muß beide Nachrichten transportieren (und bezahlen). Und wenn schon unbedingt ein ‘Follow-up’ in das Brett sein muß, bitte nicht die Nachricht im Stück ‘quoten’. Die anderen Leser kennen die zur Genüge...

Zur Finanzierung

Zu einem Teil betrachte ich den Betrieb ja als Hobby...

Ich bezahle neben Strom, Rechner-Verschleiß und Telefongebühren auch dem Internet-Provider etwas. Einerseits eine Grundgebühr, andererseits etwa 15 Mark pro Megabyte IP-Traffik. Dieser Betrag ist wegen ‘Großabnehmer-Rabatt’ ab Oktober billiger geworden! Damit sind E-Mail, www- Seitenabruf und ähnliches gemeint. Bei Schreibfehlern in der Adresse wird leider die unzustellbare Nachricht an den Absender zurückgeschickt, und ich bezahle dann zweimal...

Wenn ich richtig zusammengezählt habe, sind von Januar bis November hier übrigens 108 Megabyte Mail durchgeschleust...

Die Newsgruppen kosten ‘nur’ den Telefon-Anteil; der ist aber auch nicht unerheblich.

Dieser Zugang erlaubt es, die Newsgruppen frei weiterzuverteilen, und auch die kommerzielle Nutzung; man darf also z.B. die Mail-Adresse auf seine Visiten-Karte drucken.

Seit einiger Zeit verteile ich ja die Mail-Abrechnungen. Leider etwas unregelmäßig; so kommen September-November ‘heute’. Die Abrechnung für Dezember kann ich erst in etwa 2 Wochen machen, wenn ich die Daten vom Provider bekomme.

Bei einigen Leuten lohnt sich das ‘eigentlich’ nicht; andere haben schon etwas größere Mengen an Durchsatz. Für diejenigen, die meine Konto-Nummer noch nicht kennen ist sie hier noch einmal:

96 702 720 bei der Vereins- und Westbank Kiel BLZ 210 300 00

‘Ortskundige’ können natürlich auch bei Brinkmann mit einem Plausch Silke von der Arbeit abhalten und ihr was in die Hand drücken...

Ein gutes ” 1997 ” wünsche ich allen, Holger
(hp@kbbs.org) 7.1.97

Nachtrag

Einige nette Leute haben mich darauf aufmerksam gemacht, daß in den Mail-Abrechnungen ein Betrag von ” 0,00 DM ” ausgewiesen ist. Da habe ich einen kleinen Programmier-Fehler gemacht...

Aber die Anzahl an Kilo-Byte’s mit 1,5 Pfennigen zu multiplizieren sollte Euch allen möglich sein; zur Not kann es Euer Computer :-)

Gruss, Holger (hp@kbbs.org)
12.1.97

Forth-Systeme: Gforth 0.2.1 available

Gforth 0.2.1 is now available from ftp://prep.ai.mit.edu/pub/gnu/gforth-0.2.1.tar.gz and http://www.complang.tuwien.ac.at/forth/gforth/gforth-0.2.1.tar.gz

It will soon be available from other GNU mirror sites. Binary distributions for Linux-Intel and DOS/Windows/OS/2 are available at http://www.complang.tuwien.ac.at/forth/gforth/

Gforth is a fast and portable implementation of the ANS Forth language. It works nicely with the Emacs editor, offers some nice features such as input completion and history and a powerful locals facility, and it even has (the beginnings of) a manual. Gforth employs traditional implementation techniques: its inner innerpreter is indirect or direct threaded. Gforth is distributed under the GNU General Public license (see COPYING).

Gforth runs under Unix and DOS and should not be hard to port to other systems supported by GCC. This version has been tested successfully on the following platforms: i486-unknown-linuxaout, i586-unknown-linux, alpha-dec-osf3.2, mips-dec-ultrix4.3, sparc-sun-sunos4.1.4, sparc-unknown-netbsd1.2 (configured

with ac_cv_sizeof_long_long=0, see INSTALL), hppa1.1-hp-hpux9.05, hppa1.1-hp-hpux10.20, Windows 95 using DJGPP 2.0, and OS/2 3.0 using EMX 0.9c, the latter with manual help to finish compilation.

CHANGES:

- User-visible changes between 0.2.0 and 0.2.1
- Bug fixes
- User-visible changes between 0.1beta and 0.2.0
- Portability and Installation:
- Support architectures with buggy long longs (alpha-dec-osf). Better support for DOS and other non-Unix systems. Size changes through the command line are passed to the image (and saved with savesystem); the preamble specifies an interpreter and is propagated by save-system.

Tools:

- Improved etags support. more.fs allows output paging.
- Added compat/ directory containing ANS implementations of Gforth features.
- Added tiny multitasker (tasker.fs).
- Added two alternatives for object-oriented programming: oof.fs, objects.fs.
- Added ans-report.fs (reports which words are used from which wordset).
- New words:
- Changed POPEN and PCLOSE to OPEN-PIPE and CLOSE-PIPE.
- Added FORM, ROWS, and COLS.
- added primitives EMIT-FILE, STDOUT, STDERR.
- Added TABLEs (case-sensitive wordlists).
- added POSTPONE,.
- Added the ability to combine arbitrary interpretation and compilation
- semantics (INTERPRET/ COMPILE:); state-smart words were generally rewritten to use that mechanism.
- Changes to existing words:
- EMIT and TYPE now work through file words (and are redirectable). HEADER now stores the compilation wordlist in the header and REVEAL reveals into that wordlist.
- changed behaviour of SYSTEM (no longer returns wretval, but puts it in $?) added (SYSTEM) (c_addr u – wretval wior). ' and ['] now give an error for compile-only words.

Merry Christmas and a happy new year
anton@a0.complang.tuwien.ac.at (Anton Ertl) in comp.lang.forth
Jan'97

Forth-Systeme: Transputer-Forth F-TP 1.00

Mein Transputer-Forth-System F-TP 1.00 (für den T800) ist fertig. Besser gesagt, abgeschlossen, fertig wird sowas ja nie.

Es liegt auf: ftp://ftp.statisctik.tu-muenchen.de/incoming/Forth/f-tp-100.exe

- Forth 83, ANS-Forth (fast vollständig)
- Trigonometrische Funktionen
- Metacompiler (vom DOS-Host aus, in Turbo-Forth geschrieben)
- Metacompilation für Multisysteme vollautomatisch
- Parallelverarbeitung ähnlich wie bei OCCAM2, aber forthgerecht
- Multisystemmöglichkeit: Mehrere Forths in ein und demselben TRAM mit
- Interkommunikation
- DOSKEY-Nachbildung: FORTHKEY mit Hotkeys
- Bildschirmzeicheneinsammelmöglichkeit per Hotkeys
- Debugger DEBUG
- Disassembler DIS
- Decompiler SEE mit integriertem Disassembler DIS
- Assembler in UPN, forthgerecht, strukturiert (IF-THEN-ELSE usw.)
- Einfacher Aufruf beliebiger DOS-Befehle oder -Programme: DOS <name>.
- Alle Quelltexte
- Server, vom Host aus, in Turbo-Forth geschrieben
- Assemblieren, PEEKen, POKEn, Disassemblieren, DUMPen auch vom Server aus (für Notfälle)
- Leichtes Ändern der Linkadapter-Portadressen (interaktiv, menügesteuert): CONFIG.BAT
- Verändern der Endadresse des Systems (EOS) online (nachträgliche Vergrößerung oder Verkleinerung)
- Abspeichern eines veränderten Systems über SAVE-SYSTEM TFORTH.FYS
- Abspeichern eines veränderten Multisystems über SAVE-MULTI TFORTH.FYM
- Online-Hilfe in allen Vokabularen: HELP <wort>
- Ausführliche Erklärungen in diversen DOC-Dateien
- Abspeichern beliebiger TRAM-Bereiche als DOS-Dateien
- Einlesen von DOS-Datei-Inhalten in beliebige TRAM-Bereiche
- Programmabbruch an beliebiger Stelle mit [ESC], Sprung zum Server im Host
- Wiederaufnahme an derselben Stelle mit START
- Oder Aussprung auf die DOS-Ebene: BYE-FOR-DOS oder [ESC] BYE
- Wiederaufnahme von dort aus über TRESUME (Batch-Datei) unter Beibehaltung der Stackinhalte
- Neueinladen des Systems über LOAD-FTP
- Neueinladen des (eines) Multisystems über LOAD-FTP M
- DUMPen mit Blättern per Tastendruck
- Komfortables LEARN zur Einbeziehung der Online-Hilfe mit Statistik der dokumentierten und nicht dokumentierten Worte
- Erweiterung des Tastaturpuffers: KBUF128!
- Einladen eigener Programmteile per INCLUDE <name.FTH>
- 25% des Systems werden metacompiliert, der "Rest" wird per INCLUDE TCORE hinzugeladen
- Eingabe von KERNEL reduziert das System auf die eben erwähnten 25% an Grundsystem
- Leichtes Abändern der 75% an "Restsystem" durch Abändern der Quelldatei TCORE.FTH und nachträglicher Eingabe von KERNEL INCLUDE TCORE
- Einbeziehung von Umlauten in den Wortnamen: Umschalten von ENGLISH auf GERMAN
- TASSEMBLER als Crossassembler im hostresidenten Server
- Assemblieren und Disassemblieren (z.B. von OCCAM-Compilaten) auch vom Server im Host aus
- Einbeziehung eines Editors beliebiger Wahl über DOS <name> (z.B. DOS EDIT oder DOS EDLIN oder DOS CONTEXT ...)
- Aufruf eines zusätzlichen (Turbo-)Forth-Systems im Host von F-TP aus: DOS FORTH
- Aufruf von PCTOOLS von F-TP aus: DOS PCTOOLS
- Verbessertes LIST mit Vor- und Zurückblättern
- 32 Bit für Daten- und Returnstackbreite
- 32 Bit für Adressen
- 32 Bit für einfachgenaue Ganzzahlen
- 32 Bit für einfachgenaue Gleitkommazahlen
- 64 Bit für doppeltgenaue Ganzzahlen
- 64 Bit für doppeltgenaue Gleitkommazahlen

Zu einem "WIN32FORTH" wird F-TP 1.00, wenn man es per Mausklick von WINDOWS 3.11 oder WIN95 aus aufruft. Für Entwicklungsarbeiten erscheint ein solches Vorgehen wenig sinnvoll. Die per WORDS aufgerufene Wortliste kriecht, gemessen am textbasierten System, unglaublich langsam über den Bildschirm.

beh (Fred Behringer, München)
Feb'97

Gforth auf MISC portiert

von Bernd Paysan
paysan@informatik.tu-muenchen.de; Stockmannstr.14; D-81477 München

Die Möglichkeit, ein Forthsystem schnell auf andere Systeme und neue Prozessoren zu portieren, ist eines der beliebtesten Argumente für Forth. Aber wie lange dauert sowas? Drei Mannmonate? Drei Wochen? Drei Tage!

Den folgenden Erfahrungsbericht entnahmen wir der Newsgroup /de/comp/lang/forth vom 04.02.97.

Stichworte: Gforth Portierung MISC

Auch auf unseren monatlichen Münchner Forth-Treffen haben wir natürlich Leute (naja, meistens in der Einzahl ;-), die einfach nicht glauben wollen, daß man mit Forth alles viel schneller machen kann, vor allem das Portieren auf eine neue CPU. Und dabei ist der Mensch gar kein PC-Dödel, der da einfach sagt „Wozu neue CPU, mein Intel ist ja sowieso kompatibel", sondern arbeitet in einer ASIC-Schmiede („Mixed Mode"), die inzwischen nebenbei auch 'ne sehr kleine CPU gebastelt haben.

Nun ist das Design soweit fertig, die Boards sind bestellt und laufende Entwicklungssoftware (außer ein Assembler) nicht in Sicht. Nur das unglaubwürdige „Ein Wochenende" als Antwort auf die Frage „Wie lang braucht's, da ein Forth drauf zu portieren?". Und in drei Wochen oder so ist die 'Embedded', und da soll auf dem Stand von Mixed Mode 'was abgehen'! Naja, irgendwann wird man dann halt beim Wort genommen, und letzten Freitag um 17 Uhr ging's los (da war dann der Simulator fertig, das war Bedingung).

Die Kontrahenten

Auf der einen Seite ein Prozessor mit „einem" Befehl: dem Move. Naja, damit auch 'was gerechnet werden kann, gibt's eine in den Adreßraum gemappte ALU. Indirekte Adressierung erledigt man, indem man die Operanden für einen Befehl patcht (Opcode gibt's keinen, weil man für einen Befehl 0 Bit Opcode braucht). Zweck des Prozessors: Irgendwo auf einem FPGA zu hocken, und möglichst wenig Platz zu brauchen. Also so eine Art kundenspezifischer PIC, aber niedrige Stückzahlen.

Auf der anderen Seite Jens Wilke, Ex-Sysop, Ex-VD-Editor und jüngster Drachenträger der FG, ich, und gforth, eigentlich ein Desktop-Forth, mit C statt Assembler untendrunter. Nun ja, ein bißchen Vorarbeit war schon dabei: Die Anzahl notwendiger Primitives war etwa auf 20 reduziert. Außerdem konnte der Cross-Compiler von gforth schon immer 16-Bit-Images erzeugen, auch wenn das noch nie eingesetzt worden ist. Das ist beim Erzeugen von 64-Bit-Images so abgefallen.

Der Kampf

Nachdem uns die Leute von Mixed Mode den Simulator und den gerade aktuellen Befehlssatz in die Hand gedrückt haben, ging's Freitag 19 Uhr los. Assembler schreiben (ca. 20 Konstanten ;-), Intel-Hex-Dump erzeugen, denn der Simulator liest das, nicht einfach binär. Nebenher bringt Jens die Anzahl notwendiger Primitives noch weiter herunter. Irgendwann waren dann die Primitives geschrieben, dann ging's ans Testen (muß so um 0 Uhr gewesen sein). Natürlich erstmal Bauchlandung: Die Adressen sind ja Wort-Adressen, der Cross-Compiler erzeugt aber Byte-Adressen. Also Primitives umschreiben. Irgendwann ist es 3 Uhr und ich fahre heim. Jens arbeitet noch bis 6 Uhr weiter.

Nächstes Problem: Der Cross-Compiler verläßt sich auf den C-Loader, und spuckt nicht den richtigen Code aus. Das war dann am Samstag Abend korrigiert. Das Forth läuft auch schon einige Befehle weit und stockt dann. Jens hat unterdessen einen eigenen Simulator geschrieben, der schneller ist, und ist schon fast am Prompt. Allerdings stellt sich heraus, daß die Doku fehlerhaft ist, und nun zwar selbstgeschriebener Simulator und Primitives zueinander passen, aber nicht zum Chip ;-).

Am Sonntag also die letzten Fehler aus den Primitives entfernen, ein paar mehr Primitives bauen (sonst kommt man in endlicher Zeit nicht zu den Bugs ;-), und irgendwann um 1 Uhr als Intel-Hex-File an die Mixed-Mode-Leute schicken (wie versprochen).

Wie das System auf echter Hardware läuft, kann man wie angekündigt auf der Embedded sehen. Und mehr über den Prozessor und gforth als EC-Entwicklungssystem gibt's auf der Forth-Tagung.

Eines jedenfalls ist mir dabei klar geworden: Es reicht nicht aus, alles auf ein paar wenige Primitives zu reduzieren. Das läuft schon auf einem Pentium langsam. Es müssen die richtigen sein, und man muß die Primitives so aussuchen, daß jedes neue Primitive einen ordentlichen Performance-Schub gibt (das betrifft z.B. günstige Primitives zur Implementierung von um/mod, um* und (find)).

Derzeitiger Stand sind 47 Primitives (davon 7 Methoden zum Wörter ausführen):

```
:docol  :docon  :dovar  :douser :dodefer:dofield:dodoes
!  @  x! x@ execute ;s ?branch branch  (loop)
xor or and +  -  2/ 0= 0<> =  u< 1+ cell+ 8<<
8>> c@ 2*
>r r> sp@ sp! rp@ rp! drop   lit dup r@ over
swap   d+ d2*+   /modstep
```

❑

Embedded Systems'97

Die frühere „Echtzeit"-Messe mit dem Stand der FORTH e.V. findet nun im September in Wiesbaden statt und hat auch den Veranstalter gewechselt. Im Frühjahr gibt es unter anderem Namen, aber am gleichen Ort und mit gleichem Veranstalter die „Embedded". Der entscheidende Unterschied ist, daß die „Embedded" erfolgreich ist. In der Anfangszeit war die „Echtzeit" spärlich mit Ausstellern belegt und die Struktur der Aussteller war ziemlich einheitlich: mittlere deutsche Industriefirmen. Oft war mehr Standpersonal als Besucher im Saal.

Für die „Embedded" hat man die ungünstige Lage der Messehalle in Sindelfingen durch einen kostenlosen Shuttlebus zu S-Bahnstation und Parkplätzen entschärft. Und es ist vor allem gelungen, ein weites Spektrum von Ausstellern anzuziehen. Am oberen Ende sind die großen Halbleiterhersteller wie Motorola vertreten. Am untere Ende 1-Mann-Firmen und „EMUF-Hersteller". Parallel dazu alle wichtigen Verlage und Zeitschriften. Sowie Halbleiterdistributoren. Die Messe ist noch so klein, daß man sie ohne Fußschmerzen an einem Tag bewältigen kann. Und sie hat noch nicht die Hektik der „Electronica", man läßt sich bei Gespächen Zeit. Der Erfolg der „Embedded" zeigt sich in der Zahl von 4500 Besucher 1996, die dieses Jahr wohl noch übertroffen wird.

FORTH war dieses Jahr auch vertreten, wenn auch natürlich sparsam und meist verborgen.

Am offensichtlichsten war FORTH bei Forth Engineering, das ist Wolf Wejgaard, der HOLON auf dem 68HC11 vorführte. Er hatte einen kleinen Stand in guter Lage, der nicht zu übersehen war. Zudem hatte er seine Präsentation klar und optisch gut sichtbar aufgebaut. Er hat auch den Vortrag auf dem begleitenden Kongreß gehalten und nicht wie angekündigt Beierlein.

Ansonsten mußte man suchen, um FORTH zu finden. Gforth für den MISC gabs bei Mixed Mode. Dort wurden Flyer und Auszüge aus dem ANS-Standard sowie Disketten verteilt. Bei Glyn stand ein CCD-Demo von mir mit Flyern, die Forth erklärten. Lascar hatte die Boards von Triangle zwar nichtmehr ausgestellt. Aber man konnte dort auf Anfrage ein Demogerät mit dem neuesten System von Triangle zeigen. Die abgesetzten Stückzahlen seien in Deutschland zu gering, um aktiven Vertrieb zu rechtfertigen. Dort setzt man jetzt auf PC104. FS Flesch war natürlich auch da. LMI hat er noch im Katalog, aber FORTH ist dort Vergangenheit. Dito Diessner. Temic konzentrierte sich auf 8051/Sparclets. Dafür hatte der Stand der GEMAC nur ein Thema: MARC4. Und reichlich Infomaterial von Temic dazu.

Positiv sei auch vermerkt, was es nicht zu sehen gab: einen desorganisierten Stand der FORTH e.V., wo Hobbyisten in SWAP-Drachen-T-Shirts potentielle Anwender aus der Industrie erschrecken. Wenn die FORTH e.V. auf Messen auftreten will, soll man entsprechend Geld und Vorbereitung hineinstecken um adäquates Auftreten zu ermöglichen.

Für folgende Jahre wäre ein gemeinsamer Stand von mehreren FORTH-Anbietern/Anwendern wünschenswert. Ein einzelnes Auftreten ist nicht erfolgversprechend, weil die Kosten zu hoch sind.

Rafael Deliano

❑

MISC

von Rafael Deliano
Steinbergstr. 37; D-82110 Germering

In der VD 4/96 wurde das Thema CPU aus FPGA allgemein vorgestellt. Hier nun ein konkretes Beispiel. Das ASIC-Designhaus Mixed Mode hat auf der Messe „Embedded Systems" Ende Februar die erste Version des MISC-Prozessors vorgestellt. MISC steht für „Minimum Instruction Set Computer". Als Architektur wurde dafür die MOVE-Maschine gewählt.

Stichworte: MISC FPGA MOVE-Maschine

Das Konzept MOVE-Maschine ist neueren Ursprungs [1] [2] und wurde in Deutschland anscheinend erstmals 1990 im Rahmen einer Diplomarbeit an der FH Nürnberg realisiert [3]. Die damalige 16 Bit CPU wurde in Prototypenstückzahlen bei AMS in Graz gefertigt, belegt 2300 Gates und kommt nach den (reichlich optimistischen) Angaben der Nürnberger Entwickler hinsichtlich der Rechenleistung an die 68000 heran. Für die Implementierung von Mixed Mode waren die Randbedingungen erheblich härter, denn hier sollte die CPU in einem FPGA untergebracht werden. Der verwendete Baustein Lattice 6192 wird seit einem halben Jahr produziert und stellt damit den bei FPGAs bereits erreichten Stand der Technik dar. Er faßt die 2000 Gates der CPU und hat sogar noch etwas internes RAM, genug für Stacks jedenfalls. An seinem QFP-Gehäuse mit 208 Pins sind genügend Beinchen vorhanden, um den Daten- und Adreßbus mit jeweils 16 Bit Breite herauszuführen. Auf dem Demoboard sind extern 32k Word EPROM und 32k Word SRAM bestückt. Eine MOVE-Maschine hat nur einen Befehl:

```
MOVE source. destination
```

Der Befehl holt ein Datenwort aus „source" und speichert es in „destination" ab. Da es nur einen Befehl gibt, braucht man den Opcode nicht. Im Speicher befinden sich nur noch die beiden 16 Bit Adressen:

```
source . destination
```

Praktisch greift man hier auf Grundbefehle der CPU so zu, wie man bisher auf I/O zugegriffen hat. Beispiel: der „Befehl" für Addition ist eigentlich eine Zieladresse, nämlich 000Bh, und lautet: „ACCU+source->ACCU". Die Summe aus Akku und Quelladresse landet also wieder im Akku. Im Speicher stehen somit als Befehl diese beiden Adressen:

```
source . 000Bh
```

Nach diesem Schema wurde die minimal mögliche CPU definiert, deren Befehlssatz sich im Bild befindet. Dabei wurde gleich noch eine UART mit integriert, die fest auf 9600 Baud 8N1 eingestellt ist. Durch Kombinationen der Microcodebefehle konnte Mixed Mode einen Assembler definieren, der etwa 40 Befehle umfaßt und dem entspricht, was man bei einer herkömmlichen CPU findet.

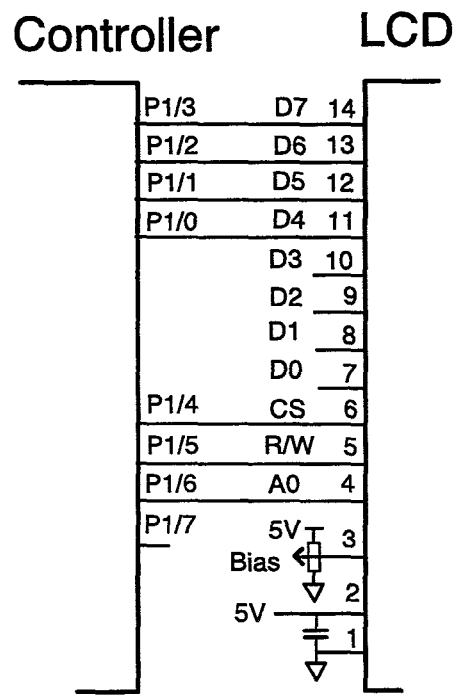
Allerdings ist der Anwender heute nicht mehr geneigt, seine CPUs in Assembler zu programmieren. Außerdem ist der MISC mit 16 Bit Datenwortbreite und 64k Adreßraum durchaus für Hochsprache geeignet. Während Mixed Mode selbst in C-Geschwindigkeit an einem C-Compiler dafür entwickelt, nahm man Kontakt zur FORTH Gesellschaft wegen eines FORTH-Compilers auf. Nachdem man Bernd Paysan und Jens Wilke am Freitag um 17.00 die Specs des MISC vorgelegt hatte, gelang es ihnen innerhalb von etwas mehr als 48 Stunden bis Sonntag 1.00 (nachts) Gforth zu portieren. Für die Messe, die zwei Wochen später stattfand, war vorgesehen, daß auf dem MISC über die UART auf einem ANSI-Terminal Tetris als Demo läuft. Das Layout der Leiterplatte, und insbesondere Probleme mit dem FPGA führten jedoch zu Verzögerungen. Am ersten Messetag war die Hardware zwar dann vorhanden, aber viel zu instabil um etwas live vorführen zu können. Zumindest konnte der Besucher aber reichlich Informationsmaterial und einen Simulator auf Diskette zu MISC und Gforth mitnehmen Während die meisten neuen RISCs nicht über ein VHDL-Simulationsmodell hinauskommen und nur wenige in Silizium gegossen werden, gibt es für den MISC also inzwischen auch Unterstützung durch Compiler.

[1] Jones „The Ultimate RISC" Computer Architecture News June 1988 S.48-55

[2] Griffin „The Ultimate Ultimate RISC?" Computer Architecture News Jan 1989 S. 26-32

[3] Eichele, Leuschner „Eine neue Risc-Architektur" Franzis-"Elektronik" 17/90 S. 49-54

❑



Low-Level-Debug-Werkzeuge für fieldFORTH

von Klaus Krieger, Rostock
E-Mail: klaus@kbbs.org

Bei der Entwicklung von Software für Microcontroller wird aus vielen Gründen oft mit Maschinensprache gearbeitet. Das Programmiersystem fieldFORTH kommt in diesem Aufgabengebiet zum Einsatz. Die vorhandenen Hochsprachen-Debugger von fieldFORTH antworteten bisher einsilbig, falls sie das Definitionswort eines zu untersuchenden Moduls nicht ermitteln konnten (das passiert bei low-level-Code-Passagen). Für diese Fälle galt es, die Gesprächigkeit der Tools zu verbessern.

Stichworte: fieldFORTH Debugger Controllerprogrammierung

```
TRC6811T TXT      2.305 02.03.97   14:30
MASKLIST CFW     17.790 14.02.97   22:56
TRC6811G FFG      2.813 29.01.96   14:43
TRC6811A CFW      8.562 29.01.96   14:43
DEC06811 CFW     30.811 29.01.96   13:32
DIS6811  CFW      6.315 16.01.96   11:00
TRC6811T FFT      5.339 14.02.97   22:57
```

FieldFORTH ist ein verteiltes System zur Programmierung von Microcontrollern (Target) von einem Entwicklungsrechner (Host) aus und läuft unter MS-Windows. Das Target kann selbst "körperlich" angeschlossen sein (online-Betrieb) oder durch einen Emulator nachgebildet werden (offline-Betrieb). Das System besteht aus mehreren, relativ eigenständigen Komponenten, die untereinander kommunizieren (siehe Bild).

Beim Start von ***fieldFORTH im online-Betrieb*** wird ein kleiner Forthsystemkern, der Nucleus, vom Entwicklungsrechner in den Speicher des Targets geladen und gestartet. Er beinhaltet keinen eigenen Interpreter, nur den Executer. Was der Nucleus selbst nicht hat, findet sich im Adapter auf dem Host. Nach dieser Prozedur "versteht" das Target Forth. Der Programmierer kann nun von der Shell den Controller so programmieren, als nutze er ein PC-Forth. Der Adapter, ein eigenständiges Forth-System auf dem Entwicklungsrechner, enthält alle targetspezifischen Teile von fieldFORTH auf der Seite des Entwicklungsrechners und verbirgt die speziellen Eigenschaften (z.B. Assembler, Kommunikationsinterface) eines Zielsystems vor der Shellkomponente. So ist es möglich, von einer Shell aus verschiedene Controller (über jeweils entsprechende Adapter) zu programmieren. Eine weitere Komponente bildet der Nucleus auf dem Target.

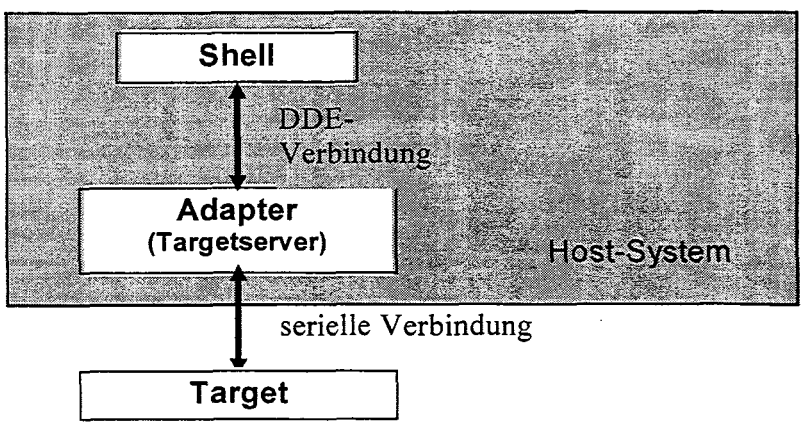


Bild 1: Komponenten des on-line-Betriebs von fieldFORTH (nach /WOIT95/)

Anforderungen

Die low-level-Debug-Werkzeuge bestehen aus einem Disassembler und einem Tracer. Die Bedienung soll der der Hochsprachenwerkzeuge weitgehend gleichen, um eine einheitliche Benutzerführung zu erreichen. Der Entwurf der Debugger sollte für verschiedene Controllerfamilien wiederverwendbare Lösungsansätze herausarbeiten. Auf dem eigentlichen Controller, auf dem die Ressourcen knapp sind, mußte der Umfang der für Testwerkzeuge implementierten Software möglichst gering bleiben. Die Kommunikationskanäle zwischen den Komponenten des verteilten Systems fieldFORTH, insbesondere das Interface zwischen Entwicklungsrechner und dem Controller, sollten durch die Debugwerkzeuge so gering wie möglich zusätzlich belastet werden. Um möglichst viel Code bei einem Controllerwechsel wiederverwenden zu können, sollten prozessorspezifische Programmpassagen der Tools gekapselt werden.

Entwurf

Bei der Untersuchung der Maschineninstruktionen gilt es herauszufinden, um welche Aktionen es sich

handelt, wo die Daten stehen, über denen sie auszuführen sind und wo es weiter geht. Die Maschineninstruktionen sind durch die Debugger exakt so zu deuten, wie durch die Decodierlogik des Prozessors. Da sowohl Disassembler als auch Tracer den zu untersuchenden Code zu entschlüsseln haben, benötigt man eine "Software-Decodierlogik" (***Decoder***). Er allein "kennt" die Sprache des Controllers und liefert als selbständiges Modul nach entsprechenden Vorgaben (wie zum Beispiel der Startadresse) alle benötigten Daten über sie. Die Schnittstelle zwischen Decoder und dem Rest der Welt bildet eine definierte Datenstruktur. In ihr sind nach jedem Aufruf des Decoders die aktuelle Adresse, der aktuelle Opcode, die aktuelle Codepage, die Adresse des nächsten auszuführenden Befehls und die Adresse des auf die aktuelle Instruktion folgenden Befehls (bei Verzweigungen sind die letzten beiden Werte verschieden, beide werden benötigt) zu finden. Um eine Instruktion zu identifizieren, vergleicht der Decoder ihr Bit-Muster nacheinander mit verschiedenen Bitmasken, die in einfach verketteten Listen hintereinander angeordnet sind (***Maskenlisten***). Beispielcode dazu findet sich im Abschnitt "Implementierung". Die Listen können verschachtelt werden (zum Beispiel, um verschiedene Codepages zu verarbeiten). Zu jeder Bitmaske steht in der Liste eine auszuführende Aktion. Bei der ersten Maske, zu der das Bitmuster des zu identifizierenden Opcodes paßt, wird die Suche abgebrochen und die zugeordnete Aktion gestartet. Dieses Konzept spart nicht nur Rechenzeit und Speicherplatz gegenüber dem Suchen in beispielsweise einer "starren" Tabelle. Es läßt auch viel Raum für geschickte Optimierungen bei der praktischen Implementierung. Das Vorgehen ähnelt dabei der Nutzung von Karnaugh-Tafeln bei digitalen Schaltnetzen. Die Opcodes werden zu möglichst großen Gruppen mit gleichen Eigenschaften zusammengefasst. Für jede dieser Gruppen wird nur ein Algorithmus zur Verarbeitung benötigt.

Der ***Disassembler*** wird durch ein Forth-Wort aufgerufen, das zur Einbindung in den Hochsprachen-Debugger geeignet ist. Der Hochsprachen-Rückübersetzer holt ihn automatisch zu Hilfe, wenn er das Definitionswort eines zu untersuchenden Moduls nicht ermitteln konnte (vgl. Bild 2).

Der Disassembler benötigt seine Startadresse auf dem TOS des Targets. Mit ihr initialisiert er den Decoder, ruft ihn in einer Schleife immer wieder auf und gibt dessen Erkenntnisse an den Benutzer weiter. Darüber hinaus muß er sich Sprünge merken, um später Entscheidungen treffen zu können, wann er aufhören darf. Da theoretisch jedes binäre Muster im Speicher eines Rechners ausgewertet werden kann, benötigt man sinnvolle Festlegungen, um zu ermitteln, wann der Disassembler genug geleistet hat:

- Unbedingte Sprünge stellen das Ende eines Programmpfades dar. Falls nicht zuvor eine Verzweigung auf die unmittelbar folgende Adresse zeigte, endet die Rückübersetzung. Das Forth-NEXT, das am Ende von Maschinensprach-Passagen folgen muß, endet mit einem solchen unbedingten Sprung.
- Der Disassembler arbeitet von niederen zu höheren Adressen. Zeigt ein Sprung auf eine niedrigere als die aktuelle Adresse, kann er im Sinne der Auswertung vernachlässigt werden.
- Sprünge auf höhere Adressen muß sich der Decoder merken, um sie später auswerten zu können.
- Der Disassembler bearbeitet nur durchgehende Code-Sequenzen. Wird die Folge, beispielsweise durch eine Datenstruktur, unterbrochen, endet hier auch die Rückübersetzung.

Funktion und Beispielcode zum Disassembler folgen unten.

Komplexer ist die Aufgabe, die Maschinensprache schrittweise auszuführen, dem Programmierer nach jedem Schritt die Umgebung des Controllers am Bildschirm zu präsentieren und nach seinen Kommandos den nächsten Schritt abzuarbeiten. Dazu muß der ***Tracer*** bei jedem Schritt die aktuelle Umgebung des Controllers exakt sichern, ihn mit den Nutzerwerten füttern, den Schrittbetrieb sicherstellen, die Steuerung an den zu testenden Code übergeben, die Ergebnisse aus den Nutzerwerten sichern und den Ausgangszustand wiederherstellen. Um die Funktion des Tracers zu realisieren, ist Code in allen online-Komponenten des Systems fieldFORTH erforderlich (siehe oben).

Während der Disassembler nach seinem Aufruf einfach durch den Code läuft, wartet der Tracer nach jedem Schritt (nach jeder Maschineninstruktion) auf die Wünsche seines Nutzers. Der kann zwischen den folgenden Befehlen wählen:

- nächsten Schritt ausführen (exakt und Schritt-für-Schritt den Code abarbeiten)
- nächsten Schritt ausführen (Subroutinen in der Codesequenz automatisch überspringen)

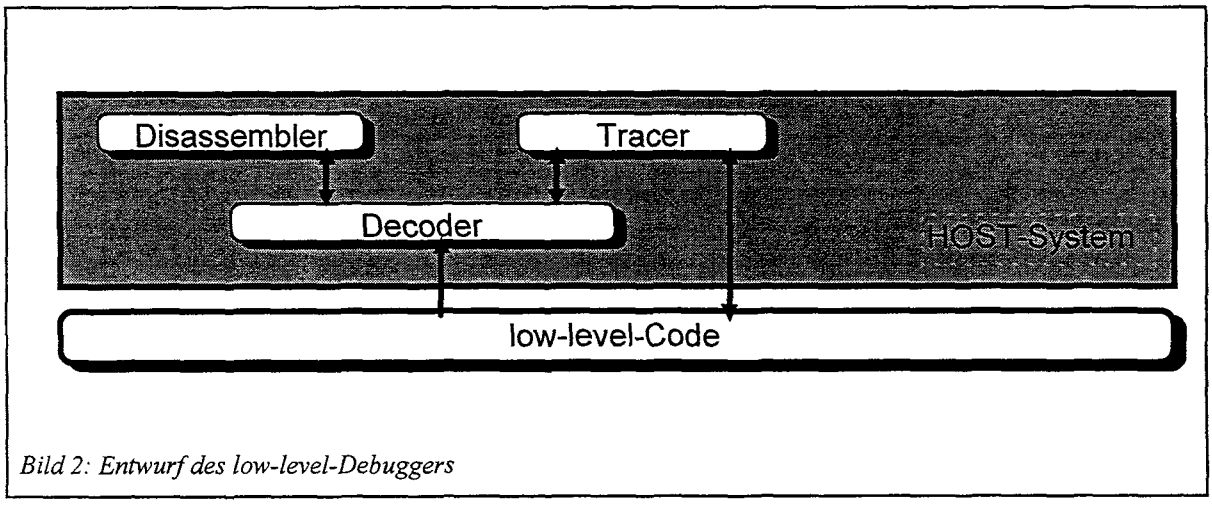


Bild 2: Entwurf des low-level-Debuggers

- Hilfetext ausgeben (ohne Unterbrechung des Tracelaufs)
- Schrittbetrieb beenden (im Echtzeitlauf den Code weiter verarbeiten)
- Schrittbetrieb beenden (zurück ins Forth ohne den Code weiter zu verarbeiten - die "Notbremse")

Nach dem Gebrauch der "Notbremse" kann der Nutzer interaktiv aus der fieldFORTH-Shell heraus die Umgebung des Tracers (Register, Stack) verändern und den Tracelauf an der Stelle fortsetzen, an der er unterbrochen wurde. So kann der Code auch nach dem Motto "Was wäre, wenn ..." getestet werden.

68HC11

Zielsystem bei der Implementierung der Werkzeuge war das "fieldFORTH Evaluationkit M68HC11" der Firma FORTecH Software. Auf diesem einfachen Controllerboard befindet sich ein 68HC11, der mit 9,832 MHz getaktet wird und mit 256 Byte internen RAM, 32 kByte externen RAM und 512 Byte EEPROM ausgestattet ist. Die Daten zwischen Entwicklungsrechner und Controllerboard werden über ein serielles Interface mit 19200 Baud transportiert /FORT95/.

Der MC 68HC11 ist ein Mikroprozessor mit 8 Bit Verarbeitungsbreite. Die daraus resultierende maximal mögliche Anzahl von 256 Maschineninstruktionen wird durch eine vierseitige Opcode-Karte ("opcode map") erweitert. Einzelne Opcodes aus dem Bereich der eigentlich verfügbaren 256 Möglichkeiten werden nicht als Instruktion ausgeführt, sondern schalten als sogenanntes "Prebyte" auf eine neue Karte mit 256 möglichen Opcodes (eine neue Seite der "opcode map") um. Diese Lösung vergrößert die Möglichkeiten der Maschinensprache der CPU. Sie führt aber bei der Ausführung der Prebyte-Befehle zu einer Verzögerung um einen CPU-Zyklus gegenüber gleichwertigen Instruktionen ohne Prebyte.

Der 68HC11 unterstützt sechs Adressierungsarten, um Daten aus dem Speicher zu lesen oder in den RAM zu schreiben. Ein Operand wird im Hauptspeicher adressiert. Falls ein zweiter benötigt wird, steht er in einem CPU-Register. Welches Register das ist und welche Operation zur Verknüpfung der Operanden durchzuführen ist, beschreibt der Opcode. Die Adresse, an der sich der Operand tatsächlich befindet, wird als "effektive Adresse" bezeichnet.

Der einfachste Adressierungsmodus ist der implizite ("inherent addressing"). Maschineninstruktionen bestehen nur aus dem Opcode. Falls Operanden benötigt werden, befinden sie sich in den CPU-Registern. Da der Hauptspeicher nicht angesprochen wird, werden auch keine Adressen benötigt.

Bei der unmittelbaren Adressierung ("immediate adressing") steht nicht die Adresse, sondern der Wert des Operanden hinter dem Opcode. Ob er 8 oder 16 Bit Länge hat, hängt vom genutzten Register ab, mit dessen Inhalt der Operand zu verknüpfen ist.

Bei erweiterter Adressierung ("extended adressing") steht die volle, effektive Adresse des einen Operanden im Maschinencode hinter dem Opcode. Mit diesen langen Adressen kann unkompliziert der gesamte Hauptspeicher adressiert werden. Ähnlich funktioniert auch der direkte Adressierungsmodus ("direct adressing"). Hinter dem Opcode steht aber nicht die volle Adresse des Operanden, sondern nur deren niederwertiges Byte. Das höherwertige Byte der Adresse steht immer auf 00h. Damit lassen sich nur die Adressen 0000h bis 00FFh erreichen. In diesem 256-Byte-Bereich des Speichers können oft benötigte Daten abgelegt werden, um mit den kürzeren, etwas schneller verarbeitbaren Adressen Zeit und Speicherplatz in den Programmen gegenüber langen Adressen zu sparen.

Die indizierte Adressierung ("indexed addressing") geht davon aus, daß sich in einem der Index-Register ein Zeiger auf den Anfang eines Speicherbereiches befindet (Index). Im Maschinencode steht hinter dem Opcode ein vorzeichenloser (immer nichtnegativer) 8-Bit-Offset. Die effektive Adresse ist die Summe aus der Adresse, auf die der Zeiger im Indexregister zeigt und dem Offset aus dem Maschinencode. Dieser Adressierungsmodus eignet sich beispielsweise für die Verarbeitung von Datenstrukturen.

Relative Adressierung ("relative adressing") nutzen Verzweigungsbefehle zur Berechnung des Sprungziels. Sie bestehen aus einem 8-Bit-Offset, der im Maschinencode auf der dem Opcode folgenden, nächsthöheren Adresse steht. Er wird zur Berechnung der effektiven Sprungadresse vorzeichenbehaftet zum Wert im Programmzähler PC addiert. Der Programmzähler zeigt auf die nächstfolgende Maschineninstruktion. Die maximale Sprungdistanz beträgt relativ zu diesem Wert 128 Byte in Richtung niedrigerer Adressen oder 127 Byte in Richtung höherer Adressen. Wird der Sprung ausgeführt (Verzweigungsbedingung erfüllt), wird die effektive Adresse in das Programmzählerregister PC geladen. Damit fährt die Ausführung an dieser Position fort /MOTO86/.

Zum Programmiermodell des MC 68HC11 gehören sieben Register. Die beiden 8-Bit-Akkumulatoren A und B lassen sich zusammen auch als ein 16-Bit-Register D ansprechen, wobei B das niederwertige Byte und A das höherwertige Byte darstellt. Der Prozessor verfügt über zwei 16-Bit-Index-Register IX und IY für die indizierte Adressierung des Hauptspeichers. Das 16 Bit breite Stackpointer-Register SP zeigt auf die Adresse des nächsten freien Platzes an der Spitze des Stapelspeichers. Der 16-Bit-Programmcounter PC enthält die Adresse des nächsten abzuarbeitenden Befehls. Die Prozessor-Status-Flaggen befinden sich im 8 Bit breiten Condition Code Register (CCR). Zum Programmiermodell zählt die "interrupt stacking order". Das ist eine Vorschrift, in welcher Reihenfolge die Registerinhalte bei einem Interrupt auf dem Stapel gesichert bzw. nach einem Interrupt auf dem Stapel zur Wiederherstellung des Ausgangszustandes erwartet werden. Bei jedem Interrupt oder der Rückkehr von einem Interrupt wird der Stapel um neun 8-Bit-Einträge verändert. Es werden alle oben beschriebenen Register des Programmiermodells, mit Ausnahme des Stapelzeigers SP, gesichert /MOTO88/.

Implementierung

Die Analyse einer Maschineninstruktion nimmt (wie im Entwurf beschrieben) vollständig der ***Decoder*** vor, der im fieldFORTH-Adapter angesiedelt ist. Bei der Implementierung auf dem Evaluationkit wurde im Decoder eine der oben beschriebenen Maskenlisten zur Zuordnung der mnemonischen Zeichenketten zu den Maschineninstruktionen angewendet.

```
RECORD: MATCHPOINT              \ Datenstruktur eines Listenelementes
        CELL FIELD: pMP         \ Zeiger auf nächstes Listenelement
        BYTE FIELD: bMask       \ Bitmaske
        BYTE FIELD: bPattern    \ geforderter Inhalt
        CELL FIELD: thcAction   \ Fadencode der Aktion bei Treffer
        :RECORD
```

```
: MATCHLIST: ( ps: ===> )( ib: name )( legt neue Liste an )
        CREATE 0 .
        ( ps: ===> 'ml )( liefert Adresse der Matchliste )       :

: MPC! ( ps: 'thc 'mp ===> )( schreibt Code für 'mp )
        MATCHPOINT thcAction ! :

: RUNMATCHLIST ( ps: xxx b 'ml ===> yyy ?t | xxx b ?f )
        ( sucht ersten Treffer in der Liste und ruft dann die )
        ( zugehörige Aktion auf. liefert dann ?t. sonst ?f )
        BEGIN   \ Listenende überprüfen
                @ DUP
        WHILE   \ Liste ist noch nicht leer
                R!                      \ ps: xxx b 'mp     rs: 'mp
                MATCHPOINT bMask C@     \ ps: xxx b mask    rs: 'mp
                OVER AND                \ ps: xxx b b'      rs: 'mp
                R@
                MATCHPOINT bPattern C@    \ ps: xxx b b' pat   rs: 'mp
                =                       \ ps: xxx b    rs: 'mp
                IF      \ Treffer
                        R>
                        MATCHPOINT thcAction @ CALL      \ ps: yyy
                        TRUE:           \ Abhauen
                THEN
                R>                      \ ps: xxx b 'mp
        REPEAT :


: +MP   ( ps: mask pattern 'ml ===> 'mp )( Matchpoint anbasteln)
        ( baut Matchpoint ohne Aktion am Listenende an )
        >R
        MATCHPOINT OBJECT R!        \ ps: mask pattern 'mp    rs: 'ml 'mp
        MATCHPOINT bPattern C!      \ ps: mask                rs: 'ml 'mp
        R@ MATCHPOINT bMask C!      \ ps:                     rs: 'ml 'mp
        [ ' NOOP >BODY @ ] LITERAL  \ leere Aktion zur Initialisierung
        R@ MATCHPOINT thcAction !   \ ps:                     rs: 'ml 'mp
        R> DUP 0 R> PRE LINK :

: :MP   ( ps: 'mp ===> )( ib: source : )
        ( weist 'mp den nachfolgend definierten Fadencode zu )
        :THC SWAP MPC! !CSP :
```

Ein Anwendungsbeispiel im Decoder zur Zuordnung der Mnemonics ist im Bild dargestellt.

Eine Anzahl weiterer solcher Listen dient der Berechnung der erforderlichen Parameter für die im Entwurf genannte Schnittstellen-Datenstruktur (da sie den TARGETCODE beschreibt, bekam sie auch diesen Namen).

Der ***Disassembler***, der mit der Definition T$SEE (im Adapter) aufgerufen wird, schreibt beim Start die Adresse vom Adapterstapel in die Struktur TARGETCODE und initialisiert die Verzweigungstabelle. Dann wird in einer unbestimmten Schleife der Decoder aufgerufen. Der Opcode wird bei jedem Schleifendurchlauf geprüft, ob es sich um eine Verzweigung handelt und gegebenenfalls deren Zieladresse gespeichert. Die Instruktion wird ausgegeben und die Adresse der folgenden Instruktion gesetzt. Die Rückübersetzung wird beendet, wenn die im Entwurf genannten Bedingungen erfüllt sind.

```
: .CODE ( ps: addr ===> )( Low-Level-Code-Stück anzeigen )
        !ADDRESS                \ aktuelle Adresse merken
        0!BTAB                  \ Branch-Tabelle putzen
        BEGIN
          DECODER               \ Analyse des OpCodes
          BRANCHES?             \ prüfen auf Verzweigungen
          .INSTRUCTION          \ Anzeige der Instruktion
          NEXTADDRESS           \ naechste Adesse bereitstellen
          VIEWEND?              \ noch weiter?
        UNTIL :

: T$SEE ( ps: ===>)( ts: addr ===> )( Modul zur Einbindung des Tracers in
den ff-Debugger)
        TGET .CODE :
```

Das Wort T$SEE (im Adapter) wird von Discompiler ***SEE*** von der fieldFORTH-Shell aus aufgerufen, wenn sich das Definitionswort des rückzuübersetzenden Codes nicht ermitteln läßt. Dann wurde dieser Code nicht in Forth-Hochsprache erzeugt. So antwortet der fieldFORTH-Discompiler mit integriertem Disassembler:

```
SEE DUP <Enter>
anonym: DUP                     unbekanntes Definitionswort von DUP
8425    PULX.                   lade Wert von Stapelspitze in Register IX
8426    PSHX.                   lege Wert aus Register IX auf den Stapel
8427    PSHX.                   lege Wert aus Register IX auf den Stapel
8428    Y LDX.                  .
842B    2 # LDAB.               .
842D    ABY.                    zurück ins Forth  (NEXT.)
842F    X LDD.                  .
8431    XGDX.                   .
8432    X JMP. ok               .
```

Für den ***Tracer*** wird vor jedem Schritt ein Datenblock namens USERREGISTER vom Adapter eingeflogen und unmittelbar nach dem Schritt mit den aktuellen Ergebnissen an den Adapter zurückgesendet.

```
Bitmaske (das Sieb, durch das der Opcode muß)

        dieser Wert muß bei der  VerANDung aus Opcode und Maske herauskommen,
        um die zugehörige Aktion auszulösen und die Suche hier zu beenden

          Name der Liste        Listenelement (an Liste MNEMONIC) anfügen
                                Treffer-Aktion für das neue Element definieren
                                hier: lege die Kette „ASL" auf den
                                Zeichenkettenstapel

CF 48   MNEMONIC +MP :MP [ "] ASL" ;
CF 49   MNEMONIC +MP :MP [ "] ROL" ;
CF 4A   MNEMONIC +MP :MP [ "] DEC" ;
CF 4C   MNEMONIC +MP :MP [ "] INC" ;
CF 4D   MNEMONIC +MP :MP [ "] TST" ;
EF 6E   MNEMONIC +MP :MP [ "] JMP" ;
CF 4F   MNEMONIC +MP :MP [ "] CLR" ;
8F 80   MNEMONIC +MP :MP [ "] SUB" ;
```

Bild 3: Anwendungsbeispiel im Decoder zur Zuordnung der Mnemonics:

```
CREATE  USERREGISTER  ( ps: ===> addr )( die „USER-CPU" )
        B/USERREGISTER ALLOT

: %A    ( ps: ===> addr )( user register A )     USERREGISTER :
: %B    ( ps: ===> addr )( user register B )     USERREGISTER 1+ :
: %D    ( ps: ===> addr )( user register D )     USERREGISTER :
: %X    ( ps: ===> addr )( user register X )     USERREGISTER 2+ :
: %Y    ( ps: ===> addr )( user register Y )     USERREGISTER 4 + :
: %SP   ( ps: ===> addr )( user register SP)     USERREGISTER 6 + :
: %PC   ( ps: ===> addr )( user program counter )USERREGISTER 8 + :
: %CCR  ( ps: ===> addr )( user register CCR )   USERREGISTER 10 + :
: %OP   ( ps: ===> addr )( breakpoint opcode )   USERREGISTER 11 + :
: %BP   ( ps: ===> addr )( breakpoint address )  USERREGISTER 12 + :
: %RSP  ( ps: ===> addr )( user return stack pointer )USERREGISTER 14 + :
```

Der Tracer sichert bei jedem Schritt zunächst die vom Target-Forth benötigten Werte, den "return stack pointer" RSP und den "instruction pointer" IP in einen Puffer FORTHREGISTER im Controller-RAM. Dieser Puffer wird nicht aus dem Target transportiert. Mit den gesicherten Werten wird die spätere Rückkehr zur Hochsprache ermöglicht. Dann wird der Opcode, der sich direkt hinter dem zu testenden Abschnitt befindet, gesichert und mit einer Breakpoint-Instruktion überschrieben. Als Breakpoint eignet sich ein nicht maskierbarer Software-Interrupt. Die Interrupt-Service-Routine, zu der die Ausführung beim Erreichen des Breakpoints verzweigt, muß die Sicherung aller für den Programmierer anzuzeigenden Werte, die Wiederherstellung der alten CPU-Umgebung und die Rückkehr ins Forth erledigen. Unmittelbar vor dem Rücksprung ins Forth wird mit der Instruktion "clear interrupt mask" CLI das I-Bit des "condition code register" CCR gelöscht, um maskierbare Interrupte wieder zuzulassen.

```
3F      Constant SWI                    \ Opcode SWI
FFF6    CONSTANT 'SWI                   \ SWI-Vektoradresse


LABEL SWICODE ( ps: ===> addr )( interrupt service routine für SWI )
   \ 1. alle Registerwerte (die der SWI auf den Stapel legte) in den
   \    Puffer USERREGISTER schreiben
   \ 2. Original-Opcode wiederherstellen; SWI entfernen
   \ 3. FORTH (RSP und IP) wiederherstellen
   \ 4. zurück ins FORTH
   USERREGISTER # LDX.              \ X zeigt auf USERREGISTER (Index)
   PULA.                            \ CCR vom Stack nach A laden
   %CCR USERREGISTER - .X STAA.     \ CCR aus A ins USERREGISTER schreiben
   PULB. PULA.                      \ D vom Stack nach D laden
   %D USERREGISTER - .X STD.        \ D aus D ins USERREGISTER schreiben
   PULA. PULB.                      \ X vom Stack nach D laden
   %X USERREGISTER - .X STD.        \ X aus D ins USERREGISTER schreiben
   PULA. PULB.                      \ Y vom Stack nach D laden
   %Y USERREGISTER - .X STD.        \ Y aus D ins USERREGISTER schreiben
   PULY. DEY.              \ PC vom Stack nach Y laden, Y dekrementieren
   %PC USERREGISTER - .X STY.       \ PC aus Y ins USERREGISTER schreiben
   \ User-Register sind gerettet

   \ Opcode restaurieren

   %OP USERREGISTER - .X LDAA. \ Original-Opcode aus USERREGISTER nach A
   Y STAA.                  \ über den SWI-Opcode schreiben (wieder original)
   TSY. DEY.                        \ SP nach Y kopieren und dekrementieren
   %SP USERREGISTER - .X STY.       \ und ins USERREGISTER schreiben
   RSP #) LDD.                      \ RSP nach D laden und
   %RSP USERREGISTER - .X STD.      \ ins USERREGISTER sichern

   \ FORTH-Register restaurieren

   FORTHREGISTER # LDX.             \ X zeigt auf FORTHREGISTER
   §RSP FORTHREGISTER - .X LDD.     \ RSP nach D laden
   RSP #) STD.              \ und von dort an die Adresse RSP schreiben
   §IP FORTHREGISTER - .X LDY.      \ IP nach Y laden (schon fertig)

   CLI.                             \ Interrupts freigeben
   NEXT.                            \ weiter in Forth
   END-CODE

: 0!SWI ( ps: ===> )( initialisiert SWI-Vektor ) SWICODE  'SWI  ! ;
```

Um die Steuerung an den zu tracenden Code im Targetsystem zu übergeben, werden alle erforderlichen Werte der USERREGISTER auf den Target-Stapel gelegt. Da diese Daten vor jedem Schritt aus dem Host übertragen werden, hat der Nutzer des Tracers die potentielle Möglichkeit des direkten Zugriffs auf die Register seiner "User-CPU". Die Ablagereihenfolge der Werte ist dabei genau so gewählt, wie sie für die Rückkehr von einem Interrupt vorgeschrieben ist, also nach der oben erwähnten "interrupt stacking order" des 68HC11. Dann wird die Instruktion "return from interrupt" RTI ausgeführt. Sie schreibt alle auf dem Stapel liegenden Werte in die Register der CPU. Danach fährt die CPU an der Stelle mit der Abarbeitung fort, auf die der (durch die Instruktion "return from interrupt" gesetzte) Programmzähler PC zeigt.

Die Steuerung ist an den zu testenden low-level-Code übergegangen.

```
CODE   (STEP) ( ps: ===> )( bis zum nächsten Breakpoint laufen )

   \ FORTH-Register retten

   FORTHREGISTER # LDX.             \ X zeigt auf den Puffer FORTHREGISTER
   §IP FORTHREGISTER - .X STY.      \ IP nach FORTHREGISTER sichern
   RSP #) LDD.                      \ Inhalt der Adresse RSP nach D laden
   §RSP FORTHREGISTER - .X STD.     \ RSP nach FORTHREGISTER sichern

   \ Breakpoint setzen

   USERREGISTER # LDX.              \ X zeigt auf den Puffer USERREGISTER
   %BP USERREGISTER - .X LDY.       \ Y zeigt auf Breakpoint
   Y LDAA.                          \ Original-Opcode nach A laden
   %OP USERREGISTER - .X STAA.      \ und von da ins USERREGISTER sichern
   SWI # LDAA.                      \ Opcode SWI nach A laden und
   Y STAA.                          \ über den Original-Opcode schreiben

   \ temporären Returnstack anlegen

   %RSP USERREGISTER - .X LDD.      \ User-RSP aus Puffer holen
   RSP #) STD.                      \ und eintragen

   \ Registerwerte aus USERREGISTER holen und auf den Stapel legen
   \ (in der Ordnung, als hätte sie zuvor ein Interrupt abgelegt)

   %PC  USERREGISTER - .X LDD.      \ PC aus Puffer USERREGISTER holen
   PSHB.  PSHA.                     \ und auf den Stapel legen
   %Y   USERREGISTER - .X LDD.      \ Y aus Puffer USERREGISTER holen
   PSHB.  PSHA.                     \ und auf den Stapel legen
   %X   USERREGISTER - .X LDD.      \ X aus Puffer USERREGISTER holen
   PSHB.  PSHA.                     \ und auf den Stapel legen
   2    %D   USERREGISTER - .X LDD.       \ D aus USERREGISTER holen
   PSHA.  PSHB.                     \ und auf den Stapel legen
   %CCR USERREGISTER - .X LDAA.     \ CCR aus USERREGISTER nach A laden
   PSHA.                            \ und auf den Stapel legen

   RTI.                  \ ab hier steuert der (zu testende) Maschinencode
   END-CODE
```

Nach der vorgesehenen Abarbeitung des low-level-Codes trifft die Ausführung auf den Breakpoint. Nun wird die Instruktion "software interrupt" SWI ausgeführt. Sie legt die Werte der Register in vorgegebener Ordnung (der "interrupt stacking order") auf den Target-Stapel. Das I-Bit im "condition code register" CCR wird gesetzt und damit maskierbare Interrupte gesperrt. Die Verarbeitung verzweigt zur Interrupt-Service-Routine des "software interrupt". Diese Routine wurde zuvor so initialisiert, daß sie auf den Label SWICODE zeigt.

```
: STEP  ( ps: ===> )( Initialisierung vorm STEPen )      0!SWI (STEP) ;
```

Fazit

- Debug-Werkzeuge auf Maschinensprachniveau lassen sich nicht nur zur Fehlerbehebung nutzen, sondern öffnen im Zusammenhang mit der Dokumentation nützliche Einblicke in die Funktion der Software insgesamt.
- Ein wichtiger Job der Low-Level-Debugger besteht darin, den Maschinencode exakt (so wie die CPU) zu deuten. Diese Funktionalität wird mehrfach benötigt und daher in einem eigenständigen Modul, dem Decoder, realisiert.
- Die knappen Ressourcen des Zielsystems sind zu schonen, um sie den eigentlichen Anwendungsprogrammen nicht zu entziehen. Notwendig sind vor allem Code zur Realisierung des Schrittbetriebs und Speicherzellen zur Pufferung der CPU-Umgebung.
- Schrittbetrieb mit Eintragung eines Breakpoints in den zu testenden Code kann nur im RAM realisiert werden, da Schreibzugriff erforderlich ist.
- Als Breakpoint-Instruktion für den Schrittbetrieb eignet sich ein nicht maskierbarer Software-Interrupt, der unabhängig vom aktuellen Zustand der Prozessorumgebung funktioniert. Mit ihm ist die Funktion des Tracers immer gewährleistet. Es müssen keine speziellen CPU-Zustände und andere Interrupte abgefangen bzw. verboten werden.
- Der Breakpoint-Interrupt des Tracers kann andere Interrupte stören. Beim Schrittbetrieb lassen sich jedoch ohnehin keine definierten Antwortzeiten (Echtzeitverhalten) der CPU erreichen. Deshalb kann dieser Nachteil in Kauf genommen werden.

Ein Beispiel für die Ausgaben des Tracers:

```
1 2 3  <Enter>  ok              drei Zahlen auf den Targetstapel legen
.S <Enter>                      Targetstapel ansehen
ts:   1 2 3  ok                 Ziffer 3 an der Stapelspitze soll dupliziert werden

'  DUP >BODY  $TRACE            Tracelauf starten
registers: D: 0                 Anzeige der Register der User-CPU
A:      0       B:      0
IX:     0       IY:     A949
SP:     0       PC:     8425
flags:
S=0 X=0 H=0 I=0 N=0 Z=0 V=0 C=0  Anzeige der Statusflaggen
stack:
ts:   1 2 3                     Anzeige des Targetstapels

8425  PULX.                     Anzeige Mnemonik (PULX: lade Wert von der Stapelspitze nach IX)
Kommando? (^J hilft): BL        Nutzerkommando Leertaste: Instruktion
        (hier: PULX. ) ausführen

registers:      D:      0       Anzeige der Register der User-CPU
A:      0       B:      0
IX:     3       IY:     A949
SP:     FA7B    PC:     8426
flags:
S=0 X=0 H=0 I=0 N=0 Z=0 V=0 C=0  Anzeige der Statusflaggen
stack:
ts:   1 2                       Anzeige des Targetstapels

8426  PSHX.                     Anzeige Mnemonik (PSHX: lege Wert aus IX 0auf den Stapel)
Kommando? (^J hilft): BL        Nutzerkommando Leertaste: Instruktion0 (hier: PSHX. ) ausführen

registers:      D:      0       Anzeige der Register der User-CPU
A:      0       B:      0
IX:     3       IY:     A949
SP:     FA79    PC:     8427
flags:
S=0 X=0 H=0 I=0 N=0 Z=0 V=0 C=0  Anzeige der Statusflaggen
stack:
ts:   1 2 3                     Anzeige des Targetstapels

8427  PSHX.                     Anzeige Mnemonik (PSHX: lege Wert aus IX auf den Stapel)
Kommando? (^J hilft): G  ok     Nutzerkommando G: Abbruch; weiter mit Echtzeitlauf

.S <Enter>                      Targetstapel ansehen
ts:   1 2 3 3  ok
```

Literatur:

FORT95

FORTecH Software GmbH:
fieldFORTH Evaluationkit M68HC11 EFF11.
Dokumentation, 1995

MOTO86

Motorola:
M68HC11 HCMOS Single-Chip Microcomputer
Programmer's Reference Manual. First Edition.
Printed in GB, 10/86

MOTO88

Motorola:
MC68HC11 HCMOS Single-Chip Microcomputer
Programming Reference Guide.
Motorola Ltd. Sales Office, Colvilles Road, Kelvin Estate
East Kilbride, Scotland G75 0TG, 5/88

WOIT95

Woitzel, Egmont; Lange, Stefan:
Modulare Cross-Entwicklungsumgebung unter Windows.
Beitrag zum Veranstaltungsprogramm der Jahrestagung der
Forth-Gesellschaft e.V., 1995.

❑

WIN32FOR: Von DOS zu Windows - Nehmen wir unsere Umlaute mit?

Autor: Michael Schröder
Magdeburger Str.26; 39387 Oschersleben; Tel.: 03949 / 40 86

Der Grund, warum ich das im folgenden Text vorgestellte Programm geschrieben habe, war nicht etwa der sehnliche Wunsch endlich wieder einmal ein Thema für die VD zu haben, sondern es war so, daß ich in der täglichen Arbeit ein (kleines) Problem antraf, daß ich mit W32F lösen konnte.

Im Nachhinein meine ich, daß Problemstellung und Lösung handlich genug wären, um einen Artikel für die VD abzugeben. Außerdem meine ich, daß Texte wie dieser zu einer Dokumentation für W32F auf der Basis einzelner Aufsätze zusammengetragen werden sollten (siehe auch "WIN32FOR: im Spiegel des USENET")

Ich habe kürzlich einige Firmenschriften über verschiedene IC-Familien angesehen und war vom Konzept der Dokumentation sehr angetan. Es wurden zuerst die (sehr trockenen) technischen Daten dargestellt, darauf folgten dann Aufsätze verschiedener Autoren zu praktischen Einsatzgebieten der IC's, Vergleiche mit anderen Chipsfamilien etc.

Stichworte: Zeichensatzkonvertierung Win32For

Wie fing es an?

Ich fand beim Stöbern in der neuen Umgebung W32F im Menü 'File' einen interessanten Eintrag: 'Pages Up Setup...'.

Dahinter verbirgt sich eine kleine Dialogbox, in der man wählen kann ob 1, 2 oder 4 logische Seiten auf eine physikalische Druckseite abgebildet werden sollen. Bei dem Umfang der Systemsourcen die ideale Funktion dachte ich mir und nutzte sie auch reichlich!

In meiner Arbeitsstelle muß ich einmal monatlich Datenlisten zur Archivierung ausdrucken. Dazu wollte ich nun auch den Minidruck von W32F benutzen, um so wenig Papier wie möglich für diese ziemlich sinnlose Sache zu "verheizen". Ich exportierte die Listen als (DOS)-ASCII File und siehe da, Umlaute und Grafikzeichen sahen auf dem Ausdruck etwas entstellt aus. Ich schaute mir daraufhin eine Windows-Zeichensatztabelle einmal genauer an und verglich sie mit entsprechenden DOS-ASCII Tabellen. Die Standard-ASCII Codes bis 127 haben sich erwartungsgemäß nicht geändert, wohl aber die darüberliegenden Codes, die auch unsere deutschen Umlaute enthalten. Mir ist dann auch wieder eingefallen, daß unter DOS die oberen Codes auch nicht immer gleich sind, sondern über einen Befehl in der CONFIG.SYS bestimmte 'Codepages' gewählt werden können. Ich stand also vor dem Problem Umlaute und einige Grafikzeichen in die entsprechenden Windowscodes zu konvertieren. Zuerst probierte ich es per Hand mit der Suchen/Ersetzen-Funktion des FPC-Editors. Es ging wohl, aber nach der ersten Datei hatte ich dann auch schon die Schnauze voll von dieser "Lösung". Die Idee ein Tastaturmakro für die Suchen/Ersetzen-Arbeit aufzuzeichen ging auch nicht, weil in FPC manchmal Funktionstasten mit Tastencodes über 128 kollidieren, außerdem wird der Status der Shift-Taste nicht mit aufgezeichnet. Also bis dahin: Dumm gelaufen!

Die nächste Idee war einen Filter zu schreiben, der die Quelldatei zeichenweise liest, die betreffenden Zeichen austauscht und in eine andere Datei schreibt. Machbar zwar, dachte ich mir schließlich, aber sehr umständlich. Schließlich erinnerte ich mich an einige Win3.1. SDK Bücher, die ich mir im Laufe der Zeit angeschafft hatte (ohne SDK) und blätterte im Band 3 "Funktions". In alphabetischer Reihenfolge fand ich ziemlich schnell eine meinen Wünschen umgekehrte Funktion "AnsiToOem" und vermutete richtig, es müßte dann auch eine Funktion "OemToAnsi" geben. Diese Funktion wandelt einen übergebenen String mit Zeichen der aktuellen DOS-Codepage (Oem) in einen anderen String mit Zeichen des Windowszeichensatzes (ANSI) um. Gut soweit, nun mußte ich noch herausfinden unter welchem Namen diese Funktion unter WIN32 angeboten wird. Gut, daß ich noch

eine alte Win32For Version 1.2xxx auf der Platte aufgehoben hatte, denn dort konnte man noch mit ”WORDS OEM <Enter>” die folgende Anzeige erhalten:

```
words_oem
----------- WINCON ----------------------------------------
VK_OEM_CLEAR              OEM_FIXED_FONT          OEM_CHARSET
ES_OEMCONVERT             CF_OEMTEXT              CBS_OEMCONVERT

----------- PROCEDURE -------------------------------------
OemToCharBuffA            OemToCharA              OemKeyScan
CharToOemBuffA            CharToOemA              SetFileApisToOEM

-------------------------------------------------------------
Displayed  12  of the  5380  words in the system. ok
```

Es konnte losgehen

Nun wußte ich endlich, daß ”meine” Funktion **OemToCharA** hieß und konnte experimentieren, sie mit den richtigen Parametern aufzurufen. Unter uns gesagt, einige Male bekam ich auch eine ”Exception-Error” Dialogbox zu sehen, mußte das System auch ein paar mal neu starten, aber dann konvertierte die Funktion doch noch eine Testzeile so wie sie es sollte!

```
: OemToAnsiString  ( src-addr dest-addr -- )
          2DUP  +NULL  +NULL
          1+ rel>abs  SWAP  1+ rel>abs
\ SDK 3.1: void OemToAnsi(hpszOemStr,  hpszWindowsStr)
\ Parameter erscheinen auf Forth Stack umgekehrt zur “C”-Notation!
          ( pszWindowStr  pszOemStr  -- ) CALL  OemToCharA
         DROP  ;
```

Der Rest - ein Wort zu schreiben, das eine Quelldatei zeilenweise liest, konvertiert und in eine zweite Datei schreibt - war dann eigentlich Routine.

Das TopLevel-Wort ConvertOemFileToAnsi liest die beiden folgenden Worte aus dem Inputstream und interpretiert sie als Dateinamen. Das erste Wort bezeichnet die Quelle, einen OemText, und das zweite die Zieldatei, einen AnsiText.. ConvertOemFileToAnsi übergibt die beiden Namen dann als Stackparameter Paare Adresse/Länge an _ConvertOemFile-ToAnsi, das die eigentliche Konvertierungsarbeit macht. _ConvertOemFileToAnsi wurde ausfaktoriert, um das Kommando unabhängig davon zu machen, ob die Argumente aus dem Inputstream kommen oder woanders her. Es wird später noch eine andere Anwendung für dieses Wort gezeigt.

_ConvertOemFileToAnsi erzeugt zuerst die Zieldatei neu und überschreibt damit eine evtl. vorhandene Datei mit gleichem Namen. Die Quelldatei wird zum Lesen geöffnet. Fehlerbedingungen werden lediglich mit einem simplen ABORT behandelt (sollte in einer größeren Applikation wohl gegen einen CATCH ... THROW Konstrukt ausgetauscht werden). Ist alles soweit gutgegangen, wird die Quelldatei in einer Schleife zeilenweise gelesen, bis das Dateiende erreicht ist. Eine gelesene Zeile wird mittels OemToAnsiString konvertiert und dann in die Zieldatei geschrieben. Abschließend werden beide Dateien wieder geschlossen.

```
0 VALUE  OemFile
0 VALUE  AnsiFile
CREATE   aline  max-path  allot
DEFER    show-aktivity    ( -- ) ' NOOP  IS  show-aktivity

: _ConvertOemFileToAnsi  ( addr-OemFname  slen  addr-AnsiFName  slen -
- )
     4DUP  COMPARE  0= ABORT” Arguments  must  be  different  Filenam
es!”
     W/O  CREATE-FILE
     ABORT” Ansi-TextFile  konnte  nicht  erzeugt  werden!”  TO  AnsiFile
     R/O  OPEN-FILE
     ABORT” OEM-TextFile  konnte  nicht  geoeffnet  werden!”  TO  OemFile
     BEGIN    aline  1+  max-path  1-
                 OemFile  READ-LINE  ABORT” line  read  error!”
          SWAP  aline  C!    ( -- NotEofFlag  )
\                ( adr  len  fileid  -- read-line  -- len  ?eof?  ior  )
     WHILE
          show-aktivity
          aline  DUP  OemToAnsiString   \ convert  string  onto  itself
          aline  COUNT  AnsiFile  WRITE-LINE  ABORT” line  write  error!”
     REPEAT
     OemFile  CLOSE-FILE  DROP   0  TO  OemFile
     AnsiFile  CLOSE-FILE  DROP  0  TO  AnsiFile
;

\ ----------  commandline  utility  -------------------------------

0 VALUE  loopcount
: _show-aktivity  ( -- )   \ sometimes  show  aktivity  on  the  screen
         1  +TO  loopcount
         loopcount  100  >  IF
              0  TO  loopcount  '.'  EMIT  1  ?CR
         THEN  ;

\ commandline  version,  which  get  filenames  from  input  stream
: ConvertOemFileToAnsi  ( -<OemFilename>-  -<AnsiFilename>-  -- )
     BL  WORD  COUNT               \ get  OemFilename
     new$  DUP>R  PLACE            \ save  it  in  dynamic  string  buffer
     BL  WORD  COUNT               \ get  AnsiFilename
     R>  COUNT  2SWAP
     DUP  0=
     ABORT” Usage:  ConvertOemFileToAnsi   <OemFilename>   <AnsiFilena
me>”
     CR  ['] _show-aktivity  IS  show-aktivity
     _ConvertOemFileToAnsi
     CR  ['] NOOP  IS  show-aktivity
;
```

Diese Version funktionierte tatsächlich und konvertierte meine Listen, so daß ich sie - für’s Archivgrab möglichst klein - drucken konnte.

Forth mit Windows ”look and feel”?

Ich hatte nun zwar ein funktionierendes kleines Forthprogramm, aber kann man es anwenderfreundlich nennen in den Zeiten von Gigabyte-Platten mit Hunderten von Directories und Tausenden von Files, zwei Dateipfade aus dem Gedächtnis einzugeben? Eigentlich nicht, und schließlich wollte ich damit ja auch noch vor den Augen einer elitären

Programmierer-Spezies, des sogenannten Forthers, bestehen (mit diesem Artikel... :-)

Ich versuchte deshalb noch etwas vom sogenannten Windows "look & feel" reinzutun, daß das kleine Prgrämmchen doch noch etwas bunter werde, jedoch ohne gleich drei Monate Arbeit zu kosten.

Das Richtige dafür fand ich in den "common dialogs", die Windows für den faulen Programmierer bereithält. W32F benutzt diese Dialoge auch selbst z.B. zum Öffnen, Sichern oder Drucken von Dateien und deshalb gibt es auch einen Satz von Objekt-Definitionsworten, um neue angepaßte Dialoge dieser Basisklassen zu erzeugen. Angepaßt werden können z.B. die Überschrift der Dialogbox und die vorgegebenen Dateinamenfilter. Im Listing 4 werden auf diese Art zwei Dialogobjekte OemFileToConvert und AnsiFileToConvert erzeugt und das erweiterte Kommando ConvertOemFileToAnsiEx benutzt sie anstelle des Inputstream zum Erfragen der Dateipfade. Alles weitere ist direkt aus der ersten Version des Wortes übernommen.

```
\ ---------- commandline utility with dialogs ----------

\ define two new common dialog objects
FileOpenDialog OemFileToConvert
"Waehle OEM-TextDatei zum Konvertieren" "Text Dateien
(*.txt)|*.txt|Forth Quellen |*.f;*.4th;*.seq|Alle Dateien (*.*)|*.*|"
FileSaveDialog AnsiFileToConvert
"Waehle Ansi-TextDatei zum Konvertieren" "Text Dateien
(*.txt)|*.txt|Forth Quellen |*.f;*.4th;*.seq|Alle Dateien (*.*)|*.*|"

\ improved command, which get the filenames using Windows
\ common dialogs FILEOPEN and FILESAVE
: ConvertOemFileToAnsiEx ( -- )
        conhndl start: OemFileToConvert COUNT    \ -- addr1 len1
        DUP 0= ABORT" Kein OEM-Text gewaehlt!"
        conhndl start: AnsiFileToConvert COUNT
                                                  \ --
 addr1 len1 addr2 len2
        DUP 0= ABORT" Kein ANSI-Text gewaehlt!"
        CR ['] _show-aktivity IS show-aktivity
        _ConvertOemFileToAnsi
        CR ['] NOOP IS show-aktivity
;
```

Auf die Schnelle testen!

Forth steht ja besonders bei uns Forth-Programmierern im Ruf schnelle Entwicklung, das sogenannte "rapid prototyping" zu fördern. Besonders der interaktive Aufruf kleiner und kleinster Programmteile soll das ermöglichen. Dazu gehören aber auch bis zur Unverständlichkeit abgekürzte Bezeichner, die (dem Eingeweihten) die schnelle Eingabe von Kommandozeilen ermöglicht. Wenn wir nun aber an das Windows-API stoßen, finden wir Mengen von langen Windowskonstanten und Funktionsnamen, die die interaktive Eingabe zur Geduldsprobe machen. Auch die von mir definierten Kommandos haben Namen, die fast länger sind, als ihr Code; eine Garantie für aufreibende Debugging-Sessions sollte man fürchten.

Win32For kennt hier jedoch noch einige hübsche Tricks, die jedoch mangels Dokumentation nicht offensichtlich und leicht zu finden sind. Hiermit dokumentiere ich also die von mir gefundenen verborgenen Funktionen, vielleicht habt ihr ja längst noch ganz andere und viel tollere gefunden.

Im Consolefenster von Win32For sieht man die vorausgegangenen Bildschirmausgaben bis sie nach oben wegscrollen, ähnlich wie bei einem zeilenorientierten DOS-Programm. Aus dem Bild gescrollte Zeilen werden jedoch in einem virtuellen Bildschirmspeicher noch eine Weile aufbewahrt und können durch Klicken auf den seitlichen Rollbalken wieder sichtbar gemacht werden. Interessante Sessions können jederzeit durch den Menupunkt 'Save Forth Console Buffer As...' im Menü 'File' gesichert werden. Ein CLS löscht jedoch das Consolefenster und auch den Bildschirmspeicher mit allen alten Ausgaben. Um lange interaktive Eingaben zu erleichtern, bietet das System verschiedene Möglichkeiten:

- Der schon vom F-PC her bekannte LINEEDITOR ersetzt das einfache Kernel-ACCEPT und bringt die gewohnten Fähigkeiten mit: Cursor zeichenweise und wortweise bewegen, zeichen- und wortweise löschen etc. Außerdem werden die 8 letzten Eingaben in einem Ringpuffer aufbewahrt und mit den Tasten 'Up' und 'Down' zurückgerufen. Diese einfache Historyfunktion ist gut, wenn man mehrmals die gleiche Eingabezeile (oder nur mit kleinen Änderungen) ausführen will.
- Neu und sehr vorteilhaft ist die Möglichkeit, auf ein Wort im Consolefenster doppelzuklicken, um es in die aktuelle Eingabezeile zu übernehmen (definiert in UTILS.F). Ich benutze es beim Testen von Code so:
  - Laden eines Quelltextes
  - Eingabe: WORDS <ENTER> <ESC> zur Anzeige der zuletzt definierten Worte aus dem eben geladenen Text.
  - erleichterte Eingabe der gewünschten Aktion, z.B.: DBG <Doppelklick auf ConvertOemFileToAnsiEx> <Enter> spart die Eingabe von 22 Zeichen
- Ebenfalls aus UTILS.F stammt die Möglichkeit, durch <Control-LinkeMaustaste> mehrere Worte vom linken Fensterrand bis zur Mausposition in die aktuelle Eingabezeile zu übernehmen und gleich auszuführen.

Besonders die Funktion des Doppelklicks im Consolefenster wird sicher niemand mehr missen mögen, der sie probiert hat. Auch das Risiko Compilerfehler durch simple Tippfehler zu erhalten, reduziert sich. Wenn ich eine der unzähligen Windowskonstanten brauche, deren Namen sich eh keiner merken kann, gebe ich z.B. WORDS WS_ <Enter> ein und erhalte eine Liste der WS_Konstanten aus der ich mir die richtige rausklicke.

Hoffe nun, den/die eine(n) oder andere(n) zum Ausprobieren von W32F angeregt zu haben. Vielleicht beschreibt der/die dann ebenfalls seine/ihre Erfahrungen in einem solchen Aufsatz, und es wird am Ende eine umfassende Dokumentation daraus.

verwendete Literatur

(1) Microsoft Windows 3.1 Programmers Reference, Volume 2, Functions, Microsoft Press - nicht die erste Wahl für die Programmierung von WIN32

Forth Online

Olaf Stoyke

os@cs.tu-berlin.de

In der heutigen Kolumne geht es im wesentlichen um das Suchen und damit natürlich auch um das Auffinden von Informationen. Da das Web sich mit ungebrochener Dynamik weiterhin ausbreitet, werden die richtigen Suchwerkzeuge und themenspezifischen Einstiegspunkte immer wichtiger…

Als Einführung ein Hinweis in - gewissermaßen - offizieller aber nichtsdestoweniger erfreulicher Sache: Die Adresse [1] (Siehe Seite 3, unten in VD 4/1996, geändert laut Mitteilung vom 17. Dezember 1996 in de.comp.lang.forth) steht für die elektronische Version der „VIERTE DIMENSION“ - kurz: „eVD“. Im Augenblick ist nur die Ausgabe 4/1995 als eVD erfaßt, aber dank Ulrich Hoffmann, der dies ermöglichte, und der Uni Kiel, die offensichtlich den Platz dafür zur Verfügung stellte, ist die VD nun im Web vertreten.

Eine wahre Fundgrube an Informationen über das „Digital Signal Processing“, über „Embedded Systems“ und über „Microcontrollers & Microprocessors“ sowie über eine unüberschaubare Fülle von weiteren Themenbereichen ist von EG3 Communications ins Web gestellt worden. Diese Firma hat es sich zur Aufgabe gemacht, alle im Internet verfügbaren Informationen zum Oberbegriff „Electronic Engineering“ zu verfolgen, zu ordnen und letztendlich zu präsentieren. Der Einstieg in das Angebot von EG3 ist über [2] möglich, wo mensch auf eine Liste von Schwerpunktthemen (Die oben genannten Themen enstammen dieser Liste!) und auf eine Zusammenstellung von weiteren Themenbereichen in Form einer Tabelle trifft. Einer der dort eingetragenen Bereiche trägt die Überschrift „FORTH“: Auf der dahinter verborgenen Seite (Für all jene, die den direkten Weg bevorzugen: die komplette Adresse ist [3]) findet mensch eine „Meta-Liste“ von Informationen rund um das Thema „FORTH“. Es handelt sich dabei um einen Einstiegspunkt zu zahlreichen weiteren Seiten zum Thema, die hier mit einer kurzen Beschreibung zusammengefaßt sind. Eine zusätzlich eingebaute Suchfunktion ermöglich das schnelle Auffinden von Suchbegriffen in eben jenen Beschreibungen. (Die Adresse [4] scheint man synonym verwenden zu können.)

Eine ganz spezielle Gattung von Problemen wird unter [5] zu kurieren versucht: Das (ewige) Problem der Dateiformate, das einem irgendwann immer wieder über den Weg läuft, da mensch ja nicht mehr in einem Vakuum sondern zwischen realen Software-Systemen mit realen Dateiformaten hin- und herprogrammiert. Auf diesem Rechner nun stehen Informationen über einige Dutzend Formate zum Abruf bereit, wobei inhaltlich keine Gewichtung auszumachen war: Spezifikationen zu Formaten von Grafik-, Sound-, Text-, Objektcodedateien sind genauso (und manchmal in mehreren Fassungen) zu finden wie die beinahe schon klassischen Formate von Tabellenkalkulations- und Datenbank-Softwares. Insbesondere für Entwickler im Bereich Compilerbau kann noch [6] empfohlen werden, eine zum Intel-Server gehörende Seite, auf der der „Tools Interface Standard“ (kurz: „TIS“) vorgestellt wird und auch kopiert werden kann. Dabei handelt es sich um eine Sammlung von Dateiformat-Spezifikationen für Dateien, die in einer Entwicklungsumgebung grundsätzlich anfallen: Dateien für den Objektcode (Formate OMF und ELF), die Debug-Informationen (CodeView, DWARF) und natürlich für das zusammengebundene Programm (PE). Federführend ist hier das „TIS Committee“, das sich aus Vertretern der Firmen Borland, IBM, Intel, Lotus, MetaWare, Microsoft, Santa Cruz und WATCOM. Mitgewirkt haben außerdem PharLap Software und Symantec, so daß durch die Marktverbreitung der genannten Unternehmen der TIS beinahe die Bedeutung eines ISO-Standards erlangt.

Bei der Suche nach Dateien im Internet kommt man unweigerlich in die Situation, daß die vielversprechendsten Dateien auf weit entfernten Rechnern liegen, die man nur mit erheblichem Aufwand an Zeit und Geduld (und nicht zu vergessen: Geld) auf den heimischen Rechner mit Programmen wie ftp herunterladen kann. Manchmal jedoch hat genau dies schon jemand getan und damit eine lokale und in der Konsequenz deutlich schneller verfügbare Kopie erstellt. Unter [7] steht ein sehr komfortabler und auch flinker Dienst (aus Norwegen) zur Verfügung, der basierend auf einem Dateinamen oder einem Fragment desselben die Archive der ftp-Welt durchsucht und damit Zugang zu den lokalen Kopien einer Datei ermöglicht.

Die Adressen:

[1] http://www.informatik.uni-kiel.de/~uho/VD/
Nachtrag: Die eVD umfaßt mittlerweile den kompletten Jahrgang 1995 der VD und einige wenige allgemeine Texte. Der Jahrgang 1996 ist bereits in Arbeit. Unser Dank gilt mis@kitchen.FORTH-eV.de (Michael Schröder). Vgl. auch seine Kurzmeldung in diesem Heft.

[2] http://www.cera2.com/ebox.htm

[3] http://www.cera2.com/forth.htm

[4] http://www.eg3.com/forth.htm

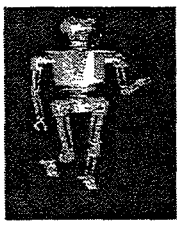
[5] http://www.wotsit.demon.co.uk/

[6] http://developer.intel.com/ial/tis/index.htm

[7] http://ftpsearch.unit.no/ftpsearch/

Mit der heutigen Kolumne ändern wir ein klein wenig das Format, mit dem die Adressen von Web-Seiten und ftp-Servern im Text erscheinen. Zuvor waren diese „inline“, was bei sehr langen Adressen, die ja oft anzutreffen sind, zu recht erheblichen Schwierigkeiten bei der Handhabung im Layout führen kann. Das neue Format benutzt eine Fußnoten-Matapher: Im Text wird mit Nummern (z.B. [3]) auf eine Adressenliste verwiesen, die am Ende der Seite zu finden sein wird.

❑



Graphik 'ohne Ende'

von Friederich Prinz
F.PRINZ@MHB.gun.de; Homberger Str. 335; 47443 Moers

Graphik mit FORTH 'zu machen', beschäftigt die Forther am Niederrhein schon ein wenig länger. Dabei schwebt uns durchaus das von Jörg Plewe auf der Forthtagung '96 angesprochene "Ballerspiel" vor, das gleichermaßen dazu angetan sein könnte, selbst ein wenig 'mehr Spaß am PC' zu entwickeln, als auch das Interesse an FORTH wieder wach zu rütteln...

Stichworte: Grafik turboForth ZF Fastgraf Anfänger

```
FASTGRAF MOD      31.976 16.02.97   22:00
FGBITMAP MOD       8.232 16.02.97   22:00
FGFLIC   MOD       5.104 16.02.97   22:00
FGKEYB   MOD       2.488 16.02.97   22:00
FGMEMORY MOD      15.256 16.02.97   22:00
FGMOUSE  MOD       4.200 16.02.97   22:00
FGPCX    MOD       3.176 16.02.97   22:00
FGSOUND  MOD       4.928 16.02.97   22:00
FGSPRPPR MOD       2.584 16.02.97   22:00
```

Auf der Suche nach einer möglichst unkompliziert zu bedienenden Graphik-Schnittstelle für das ZF, habe ich vor gut zwei Jahren einige wirklich sehr schnelle Routinen ' entwickelt', die alle Basisfunktionen wie das Zeichnen von Linien, Kästen und Kreisen mit der größtmöglichen Geschwindigkeit der jeweiligen Graphikkarte ausführen. Damit ließ sich durchaus etwas anfangen. Zum Beispiel konnte man sehr schön zeigen, daß Basic, Pascal, C und andere beim Zeichnen eines Kreises wesentlich langsamer sind als das ZF. Leider interessiert das kaum jemanden.

Das oben angesprochene Ballerspiel damit programmieren zu wollen, wäre nicht nur eine ausufernde Arbeit geworden - zu der sich zum Glück ohnehin niemand bereitgefunden hat - , sondern hätte über die 'reine' Graphik hinaus noch eine Vielzahl weiterer, noch nicht vorhandener Worte vorausgesetzt, mit deren Hilfe sich zum Beispiel die Maus oder ein Joystick steuern lassen.

Darum ist diese Schnittstelle relativ klanglos wieder in der Schublade verschwunden, abgeheftet im Ordner "Viel gelernt". Lediglich Martin Bitter hat ZF's Graphikpaket dazu genutzt, eine mehr als nur vorzeigbare Version des bekannten "Salesman" Problems zu definieren, bzw. dessen 'Lösung'.

ZF goes Pascal

Weil der an bunte Fenster gewöhnte User von einem ansprechenden Spiel wesentlich mehr erwartet als nur schnelle Linien, Kästen und Kreise, mußte 'etwas Anderes' her. Zu diesem Zeitpunkt habe ich mich wohl erstmals an Borland's BGI Treiber erinnert. Im Gegensatz zu ZF's Graphikpaket, das ausschließlich auf den Modus 640*480 Pixel bei 16 Farben ausgelegt war, können Borland's BGI Treiber sehr viel mehr Modi unterstützen. Damit käme man zumindest schon einige Schritte weiter, meinte ich - und machte mich daran, bei Borland die Informationen zu erbetteln, die ich brauchte, um dem ZF eine Schnittstelle zu diesen Treibern zu bauen. Via Internet sind die Kollegen in USA mittlerweile sogar an Sonntagen zu erreichen, on-line und live. Nur herausrücken wollten sie die erfragten Informationen nicht.

Zum Glück ließ sich auf einer CD mit einer großen Sammlung an Shareware Compilern auch Borland's "Construction Kit" für diejenigen Zeitgenossen finden, die BGI Treiber für eigene, spezialisierte Hardware programmieren wollen. Die darin enthaltenen Informationen reichten aus, um einen zweiten Anlauf für ZF's Graphik zu starten. Diesmal sollte HOLON sofort ebenfalls davon profitieren, weshalb ich damit begann, für beide Systeme parallel eine identische Schnittstelle zu 'bauen'. Neben zwar ein wenig langsameren, aber dafür wesentlich universeller einsatzbaren Basisroutinen als sie mir für das ZF gelungen waren, kamen Worte hinzu, mit deren Hilfe sich *.BMP Files auf den Bildschirm holen lassen. Damit, so dachte ich, könnte der geneigte Ballerspielprogrammierer die komplexen Hintergründe eines Bildes, oder 'Umgebungen' sehr viel einfacher erstellen. Malprogramme gibt es ja nun wirklich genug...

Ein dritter Weg

Während ich mir darüber Gedanken machte, wie ich wohl am einfachsten Schnittstellen zu Maus und Joystick definieren könnte, und ob so etwas vielleicht schon 'in der Szene' fix und fertig abzuholen wäre, ereilte mich eine Nachricht, die mich die gesamte, bis dahin bereits fertige Arbeit erneut in den schon erwähnten Ordner legen ließ.

Wolf Wejgaard schrieb mir, vermutlich mehrmals, ich solle mir doch unbedingt einmal Marc Petremanns TurboForth ansehen, bzw. dessen Graphikschnittstelle. Die wäre so leistungsfähig, daß ich mir wohl einen großen Teil meiner Arbeit sparen könne.

Tatsächlich 'kannte' ich das TurboForth bereits. Genauer gesagt: Ich hatte schon vor mehr als zwei Jahren die erste Version dieses FORTH nach einmaligem Hineinsehen ohne weitere Anstrengungen in meine Sammlung unterschiedlicher FORTH-Exoten getan. Marc Petremann hat nämlich alle Kommentare und Erklärungen in seinem System in seiner Muttersprache geschrieben. Die wenigen Vokabeln, die ich vor mehr als 25 Jahren aufzunehmen bereit war, hätten sicher nicht ausgereicht, die notwendige Übersetzungsarbeit zu leisten. Obendrein sind von diesen wenigen Vokabeln noch sehr viel weniger ebenso 'hängen geblieben', wie ich es meinerzeit eben wegen meiner Leistungen in Französich bin...

Aber Wolf hat es zumindest geschafft, mich diesmal etwas neugieriger zu machen. Deshalb begab ich mich an einige, kleinere Tests, mit dem Ziel nachzuweisen, daß Turbo-Forth's Schnittstelle gar nicht so berauschend sein kann, weil sie sicher-

lich um einiges langsamer arbeitet als ZF's bisherige Routinen, bzw. Borland's BGI Treiber. Das ging allerdings voll "in's Auge"; im wahrsten Sinne des Wortes.

Schon die allerersten, gemeinen Tests zeigten, daß TurboForth's Schnittstelle zum Beispiel Kreise um keinen CPU-Tick langsamer auf den Bildschirm bringt als meine eigenen Routinen - und damit tatsächlich deutlich flotter ist als Borland! Da mußte ich selbstverständlich noch genauer hinsehen.

FASTGRAF - FGDRIVER

Marc Petremann nutzt mit seinem TurboForth ein kommerzielles 'Graphikpaket' mit dem Namen FASTGRAF. FASTGRAF ist ein eingetragenes Warenzeichen von >Ted Gruber Software< in Las Vegas. In der kommerziellen Version besteht FASTGRAF neben einer ganzen Reihe unterschiedlichster Tools vor allem aus einer Sammlung von Libraries mit Graphikroutinen. Eine Liste der Softwarehäuser und Compiler die FASTGRAF unterstützt, liest sich wie das WHO IS WHO der nicht 'forthenden' Computerwelt. Leider ist kein einziges Forth darunter.

Fastgraf(Lite), eine Shareware Version des kommerziellen Pakets, besteht aus einem File mit dem Namen FGDRIVER.EXE. Dieser Treiber muß einer beliebigen Applikation, die seine Funktionen nutzen soll, vorgeladen werden. FGDRIVER.EXE legt sich als TSR in den Speicher, wo er sofort satte 120 Kbyte RAM für sich in Anspruch nimmt. Das sieht man der 69 Kbyte kleinen *.EXE natürlich 'von außen' nicht an. Die mir aktuell vorliegende "Evaluation Copy" des Treibers, in seiner Version 4.x, nutzt intern mehrere große Arrays, woraus sich unter anderem die hohe Geschwindigkeit erklärt, mit welcher FGDRIVER alle Basisroutinen der Graphik ausführt.

Mit rund 250 Funktionen deckt FGDRIVER beinahe alles das ab, was zur Erstellung eines 'professionellen' Spiels - oder einer beliebigen, anderen graphischen Applikation unter DOS notwendig ist:

- Initialisierung des Treibers für alle bekannten Graphikmodi, einschließlich CGA, Hercules und SVGA. 'Autodetect' Funktionen sind ebenso enthalten wie Testfunktionen, mit deren Hilfe sich der jeweils beste Modus für eine beliebige Aufgabe bestimmen läßt.
- Die 'Basisroutinen' ziehen Linien zu absolut und/oder relativ adressierten Koordinaten, führen diese Linien 'gefüllt' oder nach einem frei vorgebbaren 'Strichmuster' aus, malen ungeheuer schnell gefüllte und ungefüllte Kreise, Ellipsen und Rechtecke und zeichnen Polygone aus zuvor gefüllten Arrays heraus. Das Füllen von Flächen kann wahlweise mit 'vollen' Farben geschehen, oder 'gedithert'.
- Zugriffe auf die Farbregister und auf die Register der DAC Tabellen, bzw. der darin enthaltenen RGB Tripel sind vollständig implementiert.
- Horizontales und vertikales 'Rollen' sowohl des ganzen Bildschirms als auch frei zu definierender Bildschirmbereiche sind problemlos möglich.
- Textausgaben werden in den Pixelhöhen der jeweiligen Hardware-fonts unterstützt. Links- und rechtsbündige Textausgaben, sowie Zentrierungen stehen auf beiden Achsen zur Verfügung.
- Die Steuerung des Graphikcursors beinhaltet Konvertierungsroutinen zwischen Koordinaten des Textbildschirms und der graphischen Ausgabe.
- Auf der Hardwareebene stehen Funktionen für ein sehr viel exakteres Timing zur Verfügung, als dies üblicherweise durch den Timer möglich ist.
- Eine Low-Level Steuerung der Tastatur ermöglicht es u.a., die Tasten NUM, SCR und CAPS abzufragen, bzw. deren Stati in eine beliebige Applikation einzubinden.
- Eigene BitMap Funktionen und ein eigenes Datenformat ermöglichen schnelle Manipulationen und Transaktionen von Bildschirminhalten in Speichern, bzw. schnelles Kopieren zwischen Speichern.
- Memory-Funktionen helfen bei der Nutzung von EMS, XMS und virtuellem Speicher.
- FLI/C Animationen können 'direkt' aus einem File heraus 'abgespielt' werden. Ein kleiner Satz spezieller Worte ermöglicht zusätzliche Manipulationen mit FLI/C Animationen.
- PCX Files steht ein eigener Satz von Worten zur Verfügung.
- Zwei- und Dreitastenmäuse können beliebig abgefragt werden. Das Aussehen des Mauszeigers ist frei definierbar.
- Der Joystick kann sowohl in seiner Positionierung abgefragt werden werden, wie auch zu den Stati seiner Feuerknöpfe.
- Sound bietet FGDRIVER in einer relativ einfach zu handhabenden Implementierung der "Basic-Music-Strings".
- Ein eigenes, spezialisiertes Dateiformat für Bildschirminhalte ermöglicht den schnellen Zugriff auf entsprechende Files und/oder Speicherinhalte.

Diese Kurzbeschreibung der Möglichkeiten von FGDRIVER hat mich, wie bereits gesagt, dazu bewogen, alle meine bisherigen "Arbeiten" an eigenen Graphiktreibern einzustellen. Im TurboForth wird gezeigt, wie man 'seinem' Forth graphisch 'Beine machen' kann. An dieser Stelle: meinen herzlichsten Dank den fleißigen Franzosen...

Marc Petremann und Philippe Guillaumaud haben dem TurboForth eine Schnittstelle zu FGDRIVER 3.x spendiert. Weil es stets relativ einfach ist, etwas Gutes zu verbessern, habe ich mich entschieden, ZF und HOLON eine Schnittstelle zu FGDRIVER 4.x zu bauen. Die dazu notwendigen Informationen ließen sich aus verschiedenenen Quellen via Internet besorgen (siehe Anhang). Das machte es allerdings notwendig, die gesamte Schnittstelle neu zu schreiben.

Im Prinzip ist es, wie immer wenn man es ordentlich gezeigt bekommt, recht einfach: FGDRIVER legt sich als TSR in den Speicher. Sobald der Treiber sich in das RAM des DOS einnisten konnte, überwacht er den USER-Interrupt 62h. Wann immer ab diesem Augenblick der Interrupt 62h aufgerufen wird, verteilt der Treiber die angeforderte Arbeit über einen internen Dispatcher

entsprechend der Funktionsnummer die er im Register AL genannt bekommt. Das Register AH bleibt immer 'leer'. Alle anderen Register können (und werden) zur Übergabe der Parameter genutzt, die notwendig sind, um zum Beispiel einen Kreis in einer wählbaren Farbe mit einem wählbaren Radius um einen wählbaren Mittelpunkt zu zeichnen...

Das ist eine altbewährte Technik in der DOS-Welt, sozusagen 'vom Einfachsten'. Und es tut mir ausgesprochen gut, nach wenig befriedigenden Ausflügen in die bunte Fensterwelt wieder einmal ein Beispiel echter Programmierkunst zu sehen - auch wenn für mich selbst kaum mehr zu tun übrig blieb, als Worte der folgenden Art zu schreiben:

```
CODE fgLoadPCX
( $addr flags – status )
   CX POP
   BX POP
   CX PUSH
   ES POP
   6A >FGDRIVER
   1PUSH
END-CODE
```

Das größte Problem bei dieser Arbeit war, die Informationen darüber zu bekommen, welches Register mit welchem Inhalt welcher Funktionsnummer zugeordnet werden soll. Selbstverständlich hat Ted Gruber genau diese Informationen nicht in die Manuals aufgenommen, die man ansonsten problemlos aus dem Netz 'fischen' kann. Dafür gibt es aber die nicht umsonst weithin berühmte (berüchtigte ?) Interruptliste von Ralph Brown! Und die läßt sich in ihrer jeweils aktuellsten Form ebenfalls aus dem Internet 'saugen'.

So sind rund 250 Worte entstanden, die sich einfach und bequem nutzen lassen, und die dem Forther die ganze Vielfalt von FGDRIVER eröffnen.

HighLevel Makro

Interessant sind an den recht simplen Assembler-Definitionen eigentlich nur zwei Dinge. Das ganze für das ZF geschriebene Paket läßt sich völlig problemlos entweder von anderen FORTHen sofort übernehmen (sofern der Assembler sich an die UPN hält), oder ist zumindest ohne großen 'Denkaufwand' in andere FORTHe portierbar. Zum Zweiten möchte ich an dieser Stelle das Augenmerk auf Marc Petremann's "Makro" lenken:

```
6A >FGDRIVER
```

Die Funktionsnummer 6Ah wird einem Wort übergeben, das es in den einschlägigen Assemblern gar nicht gibt. Tatsächlich hat Marc Petremann das Wort wie folgt definiert:

```
  HEX

: >FGDRIVER ( nfunc – )
   # AX MOV
   62 INT ;

  DECIMAL
```

'Gewöhnlich' werden Code Definitionen in Colon Definitionen aufgerufen, bzw. dazu benutzt, eine komplexere Anweisung zu beschreiben. Hier wird eine Colon Definition in einer Code Definition 'aufgerufen'.

Selbstverständlich ist der Begriff des Aufrufens an dieser Stelle falsch gewählt. Das 'Nennen' von >FGDRIVER bewirkt keinen Sprung in den Code dieses Wortes, sondern arbeitet statt dessen als 'Makro', oder besser: "PreCompiler", der bei Nennung seines Namens den nachfolgenden Code in die im Aufbau befindliche Definition einfügt. Hand aufs Herz: "Hätten Sie's gewußt ?" Mir war diese Vorgehensweise bisher jedenfalls unbekannt. Und ich finde das Wort >FGDRIVER so interessant, daß ich hoffe, mir 'demnächst' einige Tage für das TurboForth freimachen zu können!

Graphik ohne Ende...

ist mit dem FGDRIVER und der bereits fertigen Schnittstelle für das ZF auch mit FORTH zukünftig kein Problem mehr. Die Portierung auf andere FORTHe habe ich bereits angesprochen. Wenn die VD mit diesem Bericht erscheint, wird die in der Syntax identische Schnittstelle zum HOLON ebenfalls längst fertig sein. Folglich bleibt mir nur noch zu wünschen, daß ich recht bald die ersten überzeugenden, ultimativen Action-Comic-Strategie-Ballerspiele mit möglichst bunter und schneller, animierter Graphik zu sehen bekomme. Selbstverständlich gehe ich dabei davon aus, daß neben dem Namen der jeweiligen Autoren immer auch der Name FORTH auftaucht ;-)

Zum Schluß...

will ich noch verraten, wo und wie sich weitere Informationen zum FGDRIVER und/oder die bereits fertigen Schnittstellen für ZF und HOLON beschaffen lassen:

FGDRIVER. User Guide und Manuals für Implementierer (leider nicht für Forther und nicht für Assembler) gibt es bei:

```
http://WWW.MIRINAE.CO.KR
==> FTP (auswählen)
==> PROGRAMMING
==> GAME
==> FG
```

dort liegt dann (fast) alles herum, was man sich zum FGDRIVER so wünscht, einschließlich der Evaluations Version 4.x. RALPH BROWN's Interruptliste in ihrer aktuellsten Fassung läßt sich finden bei:

http://WWW.DELORIE.COM/DJGPP/DOC/RBINTER

Die postalische Adresse des Herstellers von FGDRIVER, den man selbstverständlich nach angenehm verlaufenen Test der Prüfversion bezahlen sollte ($ 49,-), lautet:

TED GRUBER SOFTWARE
PO BOX 13408
LAS VEGAS, NV 89112

Und ZF's Schnittstelle, ebenso wie die für das HOLON, bekommt man wie immer am besten direkt bei mir. Anruf oder E-Mail genügen. Zusätzlich zu den Schnittstellen selbst habe ich mir 'zu eigenem Nutzen' eine Art 'Implementierer Handbuch' aus den verschiedenen Quellen zusammengestellt. Das umfaßt gut 100 DIN A4 Seiten. Selbstverständlich gebe ich auch das gerne her...

❑

Und weiter geht's in Holland

Von der Niederländischen Forth-Benutzergruppe ist wieder ein Feigenblättchen herausgekommen: Vijgeblaadje Nr. 2 vom Oktober 1996. Das Redakteurproblem scheint noch nicht ganz gelöst, die sonstigen Aktivitäten gehen aber ungebrochen weiter. Ich zitiere aus dem Impressum:

Forth International

von Fred Behringer

Planegger Str. 24; D-81241 München

behringe@statistik.tu-muenchen.de

Das Vijgeblaadje erscheint ein- bis zweimal pro Jahr.

Ziele der HCC-Forth-Benutzergruppe

* Gemeinsam ein Hobby ausüben, in diesem Fall Beschäftigung mit der Programmiersprache Forth.

* Anderen Mitgliedern bei Forth-Problemen helfen.

* Für die Verbreitung von Forth-Implementationen und Anwendungen sorgen.

* Diverse Aktivitäten und Veranstaltungen für Mitglieder organisieren.

* Analyse und Erweiterung der Sprache und Programmierumgebung Forth, auch in internationaler Zusammmenarbeit.

Aktivitäten

* Zusammenkünfte für Forth-Interessierte organisieren.

* Herausgabe des Vijgebla(a)d(je) (Feigenbla(e)tt(chen)).

* Betrieb eines eigenen Forth-Fido (BBS), das von der D.F.W. gesponsort wird (Tel.: 026-4422164, rund um die Uhr erreichbar).

Vorsitzender

Coos Haak , Catharijnesteeg 5 , 3512 NZ Utrecht , 030 - 2328 726 .

Sekretär

Willem Ouwerkerk . Boulevard Heuvelink 126 , 6828 KW Arnhem , 026 - 4431 305 .

Zusammenkünfte

Jeden zweiten Samstag an jedem zweiten Monat.

Forth-Produkte

CHForth 16-Bit-Multisegment-ANS-Forth für den PC/AT, mit Handbuch. F 60,-

JIN-Forth 32-Bit-ANS-Forth für den ATARI-ST auf 3.5-Floppy. F 25,-

ANS Standard-Dokument (ANSI X3.215-1994) auf Floppy. F 7,50

Was mir, dem Berichterstatter, besonders auffällt, ist die "gemeinsame Pflege eines Hobbys" und die Ausrichtung aufs "Internationale"

❑

Gehaltvolles

zusammengestellt und übertragen von Fred Behringer

Het Vijgeblaadje 2 der hcc FORTH gebruikersgroep, Niederlande

4. Quartal 1996

2 AT89Cx051 ontwikkel omgeving
Willem Ouwerkerk

Der Autor beschäftigt sich seit geraumer Zeit mit dem 8051-ANS-Forth, einem 16-Bit-Forth, das für F 90,- von der holländischen Forth-Gruppe auf EPROM bezogen werden kann. (Ein weiteres Exemplar dieses EPROMs kostet dann nur noch F 25,-.) Es ist für die B+-Karte eingerichtet, die für F 130,- bezogen werden kann. Der Autor beschäftigt sich im nichtprofessionellen Sinn mit Robotern, die er gern mit "Intelligenz" ausrüsten möchte. Kürzlich ist er auf den Mikrocontroller AT89C2051 der Firma ATMEL gestoßen. Damit hat er eine Entwicklungsumgebung aufgebaut, die u.a. ByteForth, einen Forth-Makrocompiler, enthält und binnen kurzem über die holländische Forth-Gruppe vertrieben wird.

3 Implementatie van TO met behulp van 'private wordlists'
Albert Nijhof

Hier geht es um Implementationsschwierigkeiten mit dem ANS-Forth-Wort TO und deren Ausräumung. Mit kommentiertem Listing.

4 Een pincode deuropener
Willem Ouwerkerk

Beispiel für eine Anwendung in 8051-ANS-Forth: Ein mit Hilfe eines Relais und einer 3x4-Matrixtastatur verwirklichtes elektronisches Türschloss. Kommentiertes Forth-Programm. 11 Worte.

❑

Ich möchte in dieser Kolumne in unregelmäßiger Folge über Forth-Aktivitäten in anderen Ländern berichten. Ich bin fleißig am Sammeln, würde mich aber natürlich auch über Zusendung von Material von anderer Seite freuen.

Verständlich sind für mich die Sprachen Englisch, Holländisch, Italienisch und Französisch.

Fred Behringer

Gehaltvolles

zusammengestellt und übertragen von Fred Behringer

Forth Dimensions der Forth Interest Group, USA

Juli/August 1996

9 Safety Critical Systems, Teil 2
Paul E. Bennett

Der Autor stellt fest, dass sich Forth ganz besonders für solche Aufgaben eignet, die formalen Kriterien, wie Zuverlässigkeit und Sicherheit, genügen müssen. Er beleuchtet die Voraussetzungen für solche Systeme, bespricht diverse Modelle samt Umwelteinflüssen und zeigt auf, in welcher Weise er Forth verwendet, um Risiken und erforderliche Garantieleistungen zu minimieren.

13 More Than a Simple State Machine
Devin Wilson

Kann ein 15jähriger einen Verkehrskreuzungs-Controller programmieren? Er kann das, wenn er sich selbst Forth beigebracht hat. John Ribles "State-Machine" und C.H. Tings Hardware liefern eine passende Umgebung, um mehr über Forth und Echtzeitprobleme zu lernen und darüber, wie scharfsinnige Entwicklungsideen unser aller Leben beeinflussen können.

18 A CGI Shell for the Apple Macintosh
Ronald T. Kneusel

Das World Wide Web besteht nicht nur aus passiven, miteinander verbundenen Dokumenten. Es stellt eine interaktive Verbindung zwischen Dienstanbieter und Nachfrager her. Der Grad der Interaktivität lässt sich durch CGI-Anwendungen steigern, was für den Benutzer eine Wertsteigerung bedeutet und für den Netzanbieter einen Prestigegewinn. Die hier zu besprechende Shell für den Mac wird Sie anregen, Ihre Vorstellungen ebenso wie die Netzverbindungen auszubauen. CGI-Experten: Was halten Sie von einem Online-ANS-Forth-Lehrprogramm mit selbstüberprüfenden Programmierbeispielen?

22 PC Floppies for non-DOS Hardware
Dwight Elvey

Für auf den Anwender zugeschnittene eingebettete Steuerungssysteme wäre es oft gut, wenn sie in der Entwicklungsphase und zur Datenprotokollierung im Betrieb ein Diskettenlaufwerk zur Verfügung hätten. Die Arbeit und die Ausgaben für ein auf den Kunden zugeschnittenes Interface lassen sich leicht vermeiden, wenn man allgemein zugängliche Hardware für den PC verwendet. Der Autor führt uns Schritt für Schritt durch diverse Formatfragen, Portadressen und andere Gesichtspunkte, die sich bei einem solchen Projekt ergeben.

30 hForth: a Small, Portable ANS Forth
Wonyong Koh

hForth ist ein auf minimal getrimmtes, auf eForth aufbauendes Public-Domain-ANS-Forth für eingebettete Steuerungssysteme. Die mit ROM oder/und RAM ausgestatteten Grundmodelle wurden mit Blick auf Portierbarkeit entwickelt, können aber auch für spzielle CPUs optimiert werden, wie am 8086-EXE-Modell gezeigt werden soll. Der Autor griff dabei Ideen und Methoden auf, die von der Forth-Gemeinde beigesteuert wurden. Darunter befindet sich auch ein eleganter neuer Multitasker.

❑

FORTHWRITE der FIGUK, Großbritannien

Nr. 90, November 1996

Ein Leserbrief von Jack Brien beschäftigt sich mit Leuten, die sich keinen komfortablen Internet-Zugang leisten können. Er schlägt vor, eine "Email-Box" zu entwerfen, die im wesentlichen nur aus Floppy-Laufwerk und Modem besteht, ein Gerät, das ähnlich wirkt wie ein Faxgerät. Die einzuschiebende Diskette bestimmt, welche Verbindung aufzubauen ist und welche Daten gesendet werden sollen.

4 Building an Outer Interpreter
Jack Brien

"... Ich habe einen neuen äußeren Interpreter konstruiert, der F83 übergestülpt wird, um das ANSI-Core-Wordset zu implementieren. Damit bin ich in der Lage, mit unterschiedlichen Versuchssystemen zu experimentieren, ohne Forth jedesmal neu schreiben zu müssen. ..."

8 Dear Gil, ...
Ray Allwright

Ein Brief an den Editor, in dem es um die Jahresbeiträge geht: Eine Erhöhung bringt die Gefahr mit sich, dass Mitglieder abspringen. Eine Beibehaltung des jetzigen Betrages lässt den Club auf lange Sicht eingehen. Für uns (VDler) ist es interessant, beiläufig zu erfahren, dass der Jahresbeitrag 10 englische Pfund beträgt. Es kann sein, dass ich (der Rezensent) die Zuschrift falsch verstanden habe und sich darin Ironie versteckt. 10 Pfund kann doch gar nicht sein (?)

10 Forth and Java
Chris Jakeman

"... Java, ähnlich wie Forth, läuft auf einer virtuellen Maschine. Das verringert die Wahrscheinlichkeit, sich (z.B. bei der Ausführung eines über WWW eingeladenen Hintergrundprogramms) ein Virus einzufangen ..." Es folgen Auszüge aus comp.lang.forth. Ich zähle 24, einer davon von unserem Mitglied Bernd Paysan. Elizabeth Rather ist der Meinung, dass Java ein Zwischending zwischen "tokenized Forth" und "tokenized C++" ist. In diesen Beiträgen wird außerdem der ShBoom-Mikroprozessor von Patriot Scientific Corporation besprochen.

19 Frequently Asked Questions
Chris Jakeman

... F.: Wo kann ich ein kostenloses Forth für meinen PC bekommen? A.: in der FAQ in ftp://forth.org/pub/Forth wird eine ganze Reihe davon aufgezählt.

19 Newcomers Wanted
Chris Jakeman

Aufruf an alle Neueinsteiger, einen Artikel über den Umgang mit Forth aus ihrer Sicht zu schreiben.

20 Sell-By Date ?

... Wozu ist Forth gut? ... Die Philosophie von Moore: Mach es möglichst einfach, spekuliere nicht herum, mach es selbst. ... Forth ist anders ... Pre-Processing, Pattern-Matching ... Forth erreicht seine einzigartige Flexibilität dadurch, dass es den Zugriff auf seinen einfachen Compiler gestattet.

22 Library Assets
Chris Jakeman

Bibliotheksmitteilungen. Verwalterin: Sylvia Hainsworth, die Gattin von Chris Hainsworth, dem Vorsitzenden.

24 Triangle Digital Services
Peter Rush

7 Seiten Reklame mit Preislisten.

❑

Gehaltvolles

zusammengestellt und übertragen von Fred Behringer

Forth Dimensions der Forth Interest Group, USA

Mai/Juni 1996

7 More Than a Simple State Machine
John Rible

Vertraut mit "Verkehrskreuzungs-Karten", die für einen Programmierkurs angefertigt wurden, fiel dem Autor auf, dass man diese gut zur Veranschaulichung von "State-Machines" heranziehen könnte. So schrieb er eine einfache "Minisprache" für "State-Machines" in Forth. Aufgrund der Hardware-Orientierung des Vorhabens wurde die normale Anordnung des Lexikons auf den Kopf gestellt. Das Ganze läuft darauf hinaus, dass die Logik-Gleichungen, Zeitgeber und Flip-Flops in Form von Software-Gleichungen nachgebildet werden, die sich unmittelbar als Schaltpläne darstellen lassen.

12 DOS-Compatible Disk Access for Targets
Dwight Elvey

Als Entwicklungsumgebung für Target-Systeme leistet der PC gute Arbeit. Wenn sich aber die Anwendung ändert, muss neuer Code in das Target-System gebracht werden. PROMs brennen wäre eine Lösung. Praktikabler wäre es, wenn PC und Target-System über serielle Schnittstellen miteinander verbunden werden könnten. Es wäre besser, dem Target-System eine Disketten-Ein- und -Ausgabe zu spendieren. Den Codezugang hat man dann vom Target-System aus über Disketten, die man einfach auswechseln kann. Damit hat man dann auch die Möglichkeit, vom Target-System aus Datenprotokolle in einer Form anzulegen, die im PC gespeichert und dort weiterverwendet werden können.

22 Safety Critical Systems
Paul E. Bennett

Die weite Verbreitung von mikroprozessor-gesteuerten Systemen in praktisch allen Bereichen der Lebensgemeinschaften der "ersten Welt" bedeutet, dass immer mehr von uns mit Arbeiten betraut werden, die große Auswirkungen haben - gute wie schlechte. Das bringt automatisch Verantwortlichkeiten und möglicherweise Verpflichtungen mit sich. Konstrukteure, Hersteller und Anlagenbetreuer haben nicht mehr nur die Aufgabe, ein System zu liefern, das funktioniert (und das unter den verschiedensten Bedingungen zuverlässig funktioniert), sondern sie müssen sich auch noch gegen mögliche Regressansprüche und gegen Vorwürfe der Unlauterkeit absichern, die ihnen die Karriere zerstören können.

27 Taming Variables and Pointers
Chris Jakeman

Forth-Programmierer erfreuen sich eines unbeschränkten Zugangs zu Compilat und Computer. Es gibt aber auch Momente, wo zuviel Freiheit eher hinderlich ist. Zum Glück stehen stets Werkzeuge bereit, um neue Ideen auszuprobieren und diese dann vielleicht mit Hilfe einiger Forth-Kollegen "an der Strippe" noch zu verfeinern. Als der Autor bei der Fehlersuche in einem Programmstück mit vielen Zeigern Schwierigkeiten hatte, hängte er an sein Forth ein neues Wort, eine Alternative zu VARIABLE, um herauszufinden, wo was falsch ist.

31 Does Late Binding Have to Be Slow?
András Zsótér

Die objektorientierte Programmierung (OOP) lockt viele Programmierer an, die durch sie verursachten Leistungseinbußen schrecken aber wahrscheinlich ebenso viele andere ab. Viele Forth-Dialekte haben schon OOP in irgendeiner Form eingebaut. Es sind aber auch Stimmen aus der Forthgemeinde zu hören, die da sagen, der mit OOP verbundene Zusatzaufwand sei für zeitkritische Anwendungen inakzeptabel. An sie ist der vorliegende Artikel gerichtet. Er beschäftigt sich hauptsächlich mit dem Ziel, objektorientierte Methoden effizienter zu gestalten, um sie jenen attraktiver erscheinen zu lassen, die gern maschinennah operieren.

❑

Het Vijgeblaadje 3 der hcc FORTH gebruikersgroep, Niederlande

Februar 1997

2 FLYER
Albert Nijhof

Compiler-Worte, Immediate-Worte, die etwas compilieren, wie TO und S", aber auch SLITERAL oder C" oder ." oder ABORT", können mit FLYER in einem einheitlichen Sinn state-smart gemacht werden. FLYER gleicht deren interaktives Verhalten dem Runtime-Verhalten derjenigen Worte an, die durch sie definiert werden. ...

4 FOP
Paul Wiegmans

Ein preiswertes und vielseitiges Eprom-Programmiergerät, das von der Eprom-Programmier-Arbeitsgruppe Paul Wiegmans und Willem Ouwerkerk entwickelt wurde. Zurückgegriffen wurde dabei auf eine Entwicklungsvorstufe (preFOP) von Willem Ouwerkerk.

❑

DN-1620 - ein Forth-Prozessor aus Weißrußland

von Thomas Beierlein
Thomas-Mann-Str. 9; D-09648 Mittweida
tb@tb.forth-ev.de

Die Entwicklung von Forth-Prozessoren in der ehemaligen Sowjetunion begann Ende der achtziger Jahre. Der erste Typ war der DOFIN DN-1610 (DOFIN steht dabei als Synonym für DO Forth INstructions*). Bei diesem Prozessor handelte es sich um einen direkten Nachbau von Cuck Moore's Novix NC4016. Die Fertigung erfolgte in Minsk, Hauptzentrum der Computerentwicklung und -fertigung der SU. Nach 1990, mit dem Zerfall der Sowjetunion, wurde das Projekt von einer kleinen Firma übernommen, die aus einer der Halbleiterfertigungsstätten ausgegliedert und privatisiert worden war.*

In dieser Firma wurde das Design des Prozessors weiterentwickelt, was schließlich zum aktuellen DN-1620 führte. Inzwischen ist der Chip in die Sereinproduktion überführt worden und wird nach Herstellerangaben sowohl für in- als auch ausländische Kunden eingesetzt.

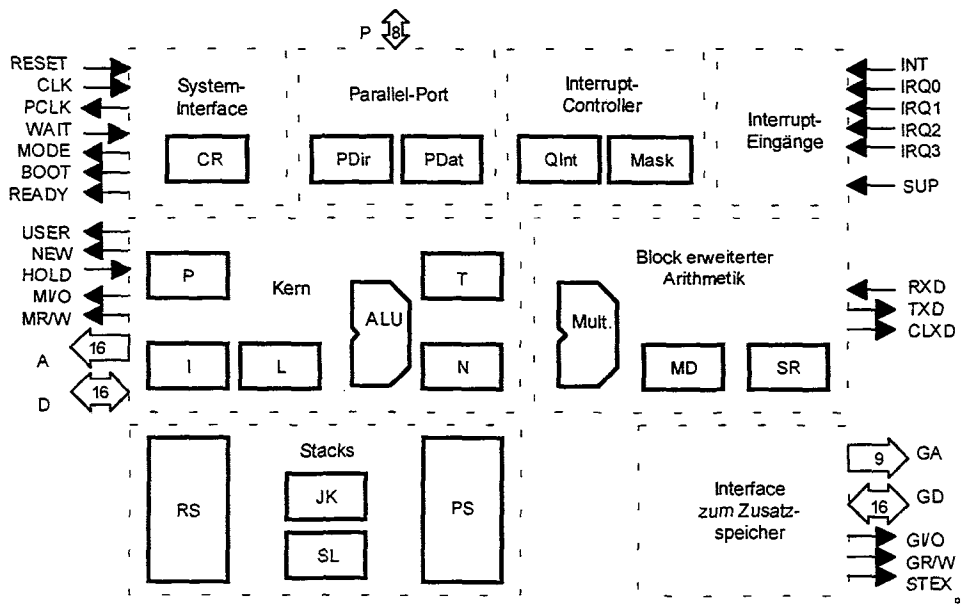
Stichworte: DN-1620 Forthprozessoren Minsk

Der Prozessor im Überblick

Der DN-1620 ist ein 16-Bit RISC-Prozessor, der auf die Abarbeitung von Forth optimiert ist. Das Design lehnt sich sowohl an den Novix NC4016 als auch an den Harris RTX2000 an. Sowohl Hardware als auch der Befehlssatz wurden erweitert und stellen eine eigenständige Lösung dar.

Die wichtigsten technischen Eigenschaften des Prozessors sind:

- vollstatisches CMOS-Design (max. 6..10 MHz Takt)
- Zweistack-Architektur mit wahlweise internen oder externen Stacks
- Zwei Speicher-Interface für Anschluß von Hauptspeicher bzw. für externe Stacks und I/O-Einheiten
- 64 kWorte Hauptspeicher, von denen nur die unteren 32 kWorte als Programmspeicher nutzbar sind
- erweiterte ALU mit 16 * 16 Bit Hardwaremultiplizierer
- serielles Interface
- bidirektionaler 8-Bit Port
- 5 externe und 3 interne Interrupts
- Befehlsausführung in 1..3 Takten
- Prozeduraufruf und -rückkehr typisch in einem Takt



- Das Hauptspeicherinterface ist eng verwandt mit dem des RTX2000. Es kann mit langsamen Speicher arbeiten und ist busmasterfähig. Weitere Steuersignale erlauben die Trennung von Programm- und Datenspeicher, die Ansteuerung eines Boot-ROM's und den separaten Zugriff auf USER-Daten.
- Coprozessor-Interface

Blockschaltbild

Die Abbildung zeigt das Blockschaltbild des DN-1620. Der Kern beinhaltet das Hauptspeicherinterface, die ALU und einige Register: das Befehlsregister L, den Programcounter P, die obersten beiden Einträge des Datenstacks T und N sowie den Top of Returnstack I.

Die erweiterte Arithmetik umfaßt die Register MD und SR sowie den Hardwaremultiplizierer. Neben der Multiplikation unterstützt sie die Berechnung von Division und Quadratwurzel, sowie interessanterweise das serielle Interface.

Das Systeminterface beinhaltet ein Konfigurationsregister zur Einstellung von Arbeitsmodii sowie Anschlüsse für Taktversorgung, Reset und anderes.

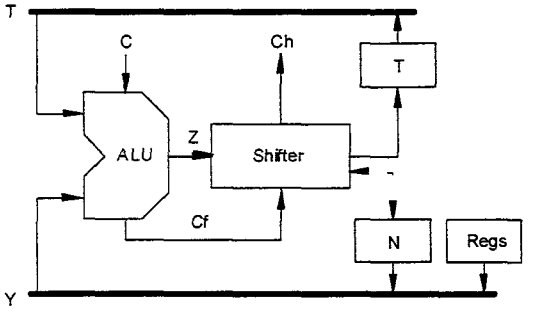
Stackmodul und Zusatzspeicherinterface beinhalten jeweils die internen Stacks mit dem Stackcontroller bzw. die Ansteuerung der externen Stacks und der maximal 512 externen I/O-Einheiten

Der Interruptblock umfaßt die entsprechenden Eingänge sowie die Logik zur individuellen Freigabe bzw zur Quittierung anliegender Interrupt-Anforderungen.

Abgerundet wird der Prozessor durch einen frei verwendbaren 8-Bit Parallelport, dessen Pins individuell für Ein- bzw. Ausgabe konfiguriert werden können.

Datenverarbeitung

Die Recheneinheit des DN-1620 besteht aus der eigentlichen ALU und einem nachgeschaltenen Shifter. Sie verarbeitet maximal zwei Operanden. Auf dem T-Bus wird dabei stets der Inhalt des Registers T (Top of Stack) bereitgestellt. Der zweite Operand auf dem Y-Bus stammt entweder aus dem Register N (Next of Stack) oder einem anderen prozessorinternen Register. Das Resultat wird im Register T abgelegt.



Zu den ALU-Operationen gehören die logischen Verknüpfungen AND, OR und XOR inklusive ihrer negierten Formen, Addition und Subtraktion mit bzw. ohne Carry sowie die Übertragung von T oder Y an den Ausgang - sowohl direkt als auch invertiert.

Der Shifter stellt die üblichen Schiebe- und Rotationsoperationen bereit, teilweise sogar für 32-Bit Operanden in den Registern T und N. Weiterhin kann das Vorzeichen des Operanden direkt in ein Flag konvertiert werden ("0<"-Operation).

Die erweiterte Arithmetik unterstützt die schrittweise Berechnung von Multiplikation, Division und Quadratwurzel. Der Hardwaremultiplizierer erlaubt die Berechnung eines 16 * 16 Bit Produkts in einem Takt. Das Ergebnis der Multiplikation kann wahlweise in den Registern MD und SR als 32-Bit Summe akkumuliert werden, wodurch die effektive Bearbeitung von Signalverarbeitungsalgorithmen möglich wird.

Stacks

Der Prozessor enthält zwei interne Stacks, die jeweils 16 Einträge groß sind. Alternativ kann jeder der Stacks extern realisiert werden (max. 256 Elemente). Beide Stacks werden über einen Stackcontroller verwaltet, der bei Stacküber- bzw. -unterlauf einen Interrupt auslösen kann.

Für den externen Anschluß der Stacks steht nur ein Speicherinterface zur Verfügung. Sind beide Stacks extern angeordnet, kann ein Konflikt entstehen, wenn auf beide Stacks gleichzeitig zugegriffen wird. Dies wird durch den Stackcontroller automatisch gelöst - in diesem Fall wird ein zusätzlicher Takt eingeschoben.

Befehlsausführung

Der Befehlscode enthält eine Reihe Bitfelder, die sowohl die Arbeit der ALU, des Shifters als auch der Stacks parallel ansteuern können. So ist oft die gleichzeitige Ausführung mehrerer elementarer Forth-Operationen in einem Befehl möglich.

Die Ausführung der meisten Befehle erfolgt in einem Takt. Bei Zugriffen auf den Hauptspeicher werden zwei Takte benötigt. Bei Stackkonflikten (s.o.) kann ein weiterer Takt hinzukommen.

Aufruf und Rückkehr aus einem Forth-Wort erfolgen typisch in einem Takt - dies ermöglicht die Bearbeitung modular faktorisierter Programme mit sehr geringem Overhead.

Der Befehlssatz weist einige interessante Neuerungen auf. Bei den bedingten Sprüngen wird ein Branch-on-Carry unterstützt, was insbesondere die Verarbeitung größerer Wortlängen vereinfacht. Ein spezieller Befehl ermöglicht die Übergabe von Befehlen an einen, am Hauptspeicher angeschlossenen, Coprozessors.

Ausführlichere Informationen zum Prozessor kann man dem User-Manual entnehmen, welches inzwischen in Deutsch vorliegt. Für die Softwareentwicklung stehen ein Assembler und ein Forth-Crosscompiler zur Verfügung. Auch ein C-Compiler soll verfügbar sein.

Ausblick

Inzwischen wird in Minsk an einem Nachfolgetyp des Prozessors gearbeitet. Als wesentliche Erweiterungen soll der DN-1630 einen interner Programm-ROM, ein I^2C-Interface sowie größere interne Stacks (je 32 Einträge) erhalten.

❑

From the other side of The Big Teich

Henry Vinerts
v.vinerts@genie.com

11/24/96, SVFIG at Cogswell College

Hello, Friederich,

The Nr.1 refers to you. It has been almost a year and a half since you and I started corresponding, but you have connected me with others, and now I am trying to figure out how to talk to everybody without making anyone wait too long. So, this is one way; I hope that Claus and Wolf do not mind that this time they are on the "carbon copy list". You have waited the longest.

...

Secondly, I told Claus that I would be pleased to contribute to his VD section of letters from the other side of the Big Teich, but that you may have to chop them down to size. Therefore, what follows is an attempt to describe last weekend's Forth Day meeting at Cogswell College in Silicon Valley, next to Lockheed and not too far from the Golden Gate Bridge.

There were about 20 people to start with, at 10 a.m., but only one of them wore the Swap-Drachen sweatshirt. Leonard Morgenstern, the leader of the North Bay Forth Interest Group, was the first one to address the audience, and he described his Forth Tutorial which is available at http://members.aol.com/nbfig. It seems that there are some people who call themselves novices at other than my Forth group, but I still call myself the "oldest Forth novice". Anyway, all novices who have comments about Len's Tutorial may send them to nleonard@aol.com.

Next came Dwight Elvey, as usual with lots of knowledge about lots of things, this time in particular describing a call-threaded Forth that he had developed for use on one of his DSP projects.

Then Jeff Fox with a report on iTV's Internet access box. This will be an inexpensive way for many people to venture into the Internet without having to buy computers. It is scheduled to appear in the market next spring. It uses Chuck Moore's 400 MIPS 20-bit Forth microprocessor, and its trade name is Pegasus.

John Rible had time to tell about his experiences in teaching Forth to Open Firmware programmers and users. Extensions in Open Firmware are handy for teaching, he says. Some of the students were doing Fibonacci series in Forth in no time at all. (I'll have to work on that. Didn't I ask you how to do it in ZF Forth?) By now I counted about 30 people in the room, but then the barbecue lunch--with hamburgers and hotdogs--was about to start, under the leadership, cooking expertise, and joviality of none other than the man with the best attendance record at Forth gatherings: Dr. C.H. Ting. (At this point I want to mention though, that in my count at the SVFIG meeting, the man who has missed the fewest is George Perry, who presently chairs the Chapter.)

Since the rain this Saturday was sort of intermittent, the outdoor barbecue drew most of the members. This was the time to exchange stories, world views, and reports on the scarcity of jobs that allow one to make a living with Forth.

Thus, the next speaker after lunch, Jim Schneider, gave a description of his method of doing garbage collection in Forth, but told us not to mention to his boss that he is using Forth on the job. Jim's wife stopped by, with the baby (did he say she is 6 months old already?), so George (Perry) had to change his call for the next session by saying "Ladies and gentlemen", which generally does not happen very often at the SVFIG meetings.

Well, the next session really did not need a call for everyone to gather at all, because Charles Moore was on. By now I counted about 35 people in the room. Chuck is always so interesting to listen to. I read somewhere that G.B. Shaw said "Few people think more than two or three times a year; I have made an international reputation for myself by thinking once or twice a week." Well, Chuck certainly has a reputation in the Forth Community, not only because he invented Forth, but also because he is _always_ thinking. Most of his talk this time was about the iTV Pegasus, the "Customizable Internet Access Box." His chip has 16000 transistors (wow!) and works with 27 instructions, 2 nanoseconds per instruction. 0.8 micron CMOS generates 650 picosecond pulses. Impossible? Some interesting thoughts: "Sometimes an improved chip is not possible because there is no time to update the schematic"; "The map is not the territory, however." It is unbelievable how much the man can accomplish with the least amount of complexity.

iTV's Internet access box with Chuck Moore's 400 MIPS Forth processor

At 4 p.m. the meeting room had to be vacated, and a good number of the attendees went to continue the day at a local restaurant, but since I had dinner waiting at home, my report ends here.

The next meeting will be a week before Christmas. Chances are that I'll get to "talk" to you all before Christmas.

Actually, I have to get another note off to Wolf on the subject of Holon. It seems that the HOLON4TH.ZIP file that

he sent me on my wife's AOL line (my GEnie is too slow) did not come in fully due to an interruption in transmission. ...

It is too late tonight to do so. Will try tomorrow, or whenever. Been working overtime too much. So forgive me for slacking off with my responses.

Guten Abend! --Henry

28XII96 Happy New Year! SVFIG Dec 21st

Hello from California!

Before the Ferienzeit runs out, I better keep my promise and send you a note about the latest in Silicon Valley FIG-land.

The December meeting, on the 21st almost didn't happen, since Cogswell College was closed for the holidays, and we had to find someone with a key to let us in. We got lucky. About 16 people were anxious to start at 10 a.m.; by noon I counted 24, and at quitting time, 4 p.m., there were as many as, I understand, had attended this year's EuroFORTH in St. Petersburg.

Dr. Ting started with a long talk about FPGAs of the Atmel AT600 family. When he works on something, he goes all out. Must be one of those guys who hardly sleep. The rest of us at least try to take short naps during the Forth meetings.

Bob Smith has been studying True Type for Windows 95, in a project of his, for display and printing of musical notes. By now he is quite an expert on True Type, as his lecture showed. True Type looks like Fig Forth, Bob says, but it appears to have been designed by a "fairly large committee." Someone suggested that Bob look also at Postscript--another forthlike language--but it seems that he has had enough with one herculean effort.

After lunch, 5 or 6 of the attendees of FORML gave their impressions of the conference. The theme had been "Experimenting with the Standard." About a dozen papers were presented, supplemented by impromptu talks. Klaus Schleisiek brought the most elegant way to crash Forth from the EF96. 1 dup base ! . (Was that it?) Fifty percent or better of the discussions were on ANSForth. PowerMacForth is only ANSForth that is in the public domain. Win32Forth is "fairly compliant." Wil Baden is working on a new issue of "Starting Forth," converted for use with ANSForth, but apparently it will involve a considerable rewrite (of the book--not ANSForth, alas!--my comment). Let's hope that the Swap-Dragon stays!

That is about it, until next year, and I hope that it is a good one for all of you. I'll be writing to Friederich again, I hope soon, since we have a bit of a gymnastic exercise going on with Wolf Wejgaard's Holon. I am still working on my "forward roll", but I have great hopes to eventually learn to do the "salto mortale" without hurting myself or my computer.

-- Henry

January 26, 1997. Lost VD, etc

Dear Friederich and Claus,

So far I have not received the 4/1996 issue, but I did manage to stay awake all through yesterday's meeting, and will give you a short report. (Actually, I have never been caught snoozing in the SVFIG meetings yet, but I have to confess that sometimes I try to catch up on my reading while the lectures go on.)

All in all probably more than 30 people came by over the 6-hour period. We even had a visitor from Taiwan.

Jim Schneider started off with a short tip on anonymous objects in Forth. You can read all about it via ftp:/ /ftp.netcom.com/pub/ ja/japs. Also Jim's article on garbage collection is "forthcoming" in the Jan/Feb Forth Dimensions. For a fellow who claims that he is in the lower ranks of the "employables" because of a lack of a college degree, he sure packs a lot of knowledge about many subjects, not only Forth, Unix, Linux, etc. And he is only a "youngster."

Dr. Ting's well-prepared and well-executed lecture on fitting P16 on 4K gate QuickLogic's FPGA filled the rest of the morning.

Skip Carter, our new FIG president, brought another "baby" robot that is in the process of developing and will be appearing on Taygeta on the Web. The previous 6-legged robot of Skip's has already claimed some fame, by appearing on the cover of February 1997 issue of Dr.Dobbs. Skip's article on page 50 tells you more about the fickle rascal.

So far I have not received the 4/1996 issue, but I did manage to stay awake

After Paul Snyder's tips on getting PostScript to do things through GoScript, etc, we came to a discussion of what to do in the next 6 months or, perhaps, the rest of 1997. There is a consensus that Win32Forth needs a good manual, but this leaves me out, since I have no intentions of finding a computer that will run Windows 95, which is necessary to get started. (This letter is written with VDE on a Toshiba T1000SE with no hard disk.)

So much for now. I am not sure what kind of news from "across the Big Pond" may be of interest to you, and then again, not being one of the pundits of Forth, what I can amuse you with, but give me some feedback, and keep the VD coming to the above address.

Tschuess, Henry

Nucleus für Controller

- Teil 1 -

von Rafael Deliano
Steinbergstr. 37; D-82110 Germering

Da in den letzten Jahren vermehrt neue Controller auf den Markt gekommen sind, muß man FORTH wieder öfters portieren, um den Anschluß nicht zu verlieren. Ein guter Anlaß unter diesem Gesichtspunkt Grundlagen zu behandeln und dabei das oft beklagte Defizit an Literatur für Anfänger abzubauen. Die hier gegebene Darstellung konzentriert sich auf 8 Bit Controller.

Stichworte: Nucleus Controller Threading Portierung

FORTH ist bekanntlich wie eine umgekippte Pyramide aufgebaut (Bild 1): viele in Hochsprache definierte Befehle greifen auf den Nucleus zu, der aus wenigen in Assembler definierten Befehlen besteht. Man hat dadurch eine überschaubare Schnittstellenzur Hardware, was Portabilität und Debugging fördert. Neuerdings werden in der Industrie Mikro- und Nano-Kernels wegen dieser Eigenschaften als Neuheiten angepriesen. Konzepte also, die immer schon Teil von FORTH waren. Neuimplementierung beschränkt sich in FORTH somit meist auf Anpassung des Nucleus.

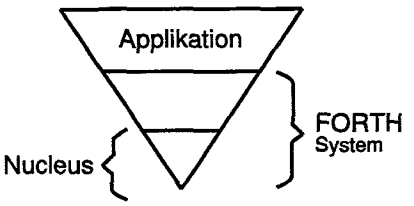


Bild 1

Stack

Bei kleinen Controllern wird meist ein 16 Bit Paramterstack verwendet. Er muß über ein Indexregister, das bei Motorola-CPUs z.B. X heißt, in Software simuliert werden. Es wird dabei die ‚X-Adressierungsart verwendet:

```
02 #. LDX,   \ X-Register mit dem Wert 02 laden
01 ,X LDA,   \ Akku mit Offset aus X laden
```

Die effektive Adresse, mit der der Akku geladen wird, ist die Basisadresse 01 plus der Inhalt des X-Registers. Also wird der Inhalt der Adresse 03 geladen. Man kann den 16 Bit Stack entweder mit zwei getrennten 8 Bit Blöcken realisieren (Bild 2). Dann hat man eine etwas löcherige Speicherzuteilung. Aber es genügt ein Inkrement-X-Befehl, um den Stackpointer um eine Zelle weiterzubewegen. Oder man legt einen kontinuierlichen Block an, bei dem die beiden Bytes aufeinanderfolgen (Bild 3). Das ist die übliche Implementierung, die hier weiterverfolgt werden soll. Einige CPUs haben für die Adressierung auch einen festeingestellten Basisoffset. Z.B. der 68HC05 mit einer 0,X-Adressierung. Dann ist ohnehin nur letztere Variante möglich.

Bei den beiden aufeinanderfolgenden Bytes gibt es unterschiedliche Festlegungen des Halbleiterherstellers, welches Byte zuerst kommt. Obwohl es auf einem simulierten Stack technisch nicht zwingend ist, sollte man den Vorgaben des Herstellers folgen, um mit dem Assembler der CPU übereinzustimmen. Bei Intel-CPUs (und dem 6502) kommt erst das untere Byte. Sie sind damit „little-endian". Bei Motorola-CPUs kommt erst das obere Byte, also „big-endian".

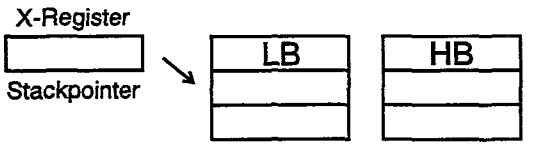


Bild 2

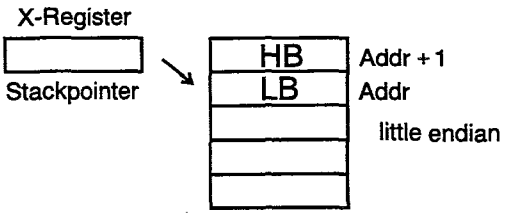


Bild 3

Das Indexregister der CPU ist meist 8 Bit breit. Das genügt völlig. 16 Bit sind eher nachteilig, besonders wenn man das Register ins RAM retten will. Möglichst viele Befehle der CPU sollten als Adressierungsart über den Offset mit dem Indexregister verfügen. Nur so können die Stackbefehle von FORTH effizient ausgeführt werden.

Returnstack

Als Returnstack kann normalerweise immer der Stack der CPU verwendet werden. Dieses ist ein sequentieller Stack, bei dem der Stackpointer ein spezielles Register der CPU ist. Solange nur PUSH- und POP-Befehle verwendet werden, braucht man sich um den Stackpointer nicht kümmern, weil er automatisch nach-

geführt wird. Man kann ihn aber auch genauso wie den Parameterstack bearbeiten. Dazu wird der Stackpointer der CPU erst in ein Indexregister kopiert. Nun ist man nicht mehr auf PUSH und POP beschränkt, sondern kann Daten direkt auf dem Stack modifizieren, z.B. inkrementieren. Sinnvoll ist das z.B. für Laufvariablen von DO...LOOP. Nach Ende des Befehls muß der Stackpointer wieder in sein Register zurückkopiert werden.

Leider hat es dabei gerade bei neuen Controllern eine ungünstige Rückentwicklung des CPU-Stacks gegeben. Der 68HC05 verarbeitet zwar Unterprogramme und Interrupts klaglos, hat aber keine PUSH- und POP-Befehle. Entweder man verzichtet generell darauf, Daten auf dem Returnstack zu speichern, oder man simuliert ihn langsam in Software. Andere CPUs, besonders PICs, haben eine arg begrenzte Stacktiefe, die oft nur für zwei Unterprogramme reicht. Hier kann man JSR-threading nicht implementieren, weil dafür die Schachtelungstiefe nicht mehr ausreicht.

Threaded Language

Praktisch alle historischen FORTHs sind gefädelt implementiert worden. Insbesondere das sehr einflußreiche fig-FORTH, das anhand eines Implemtierungsmodells definiert war. Dadurch hat sich der Irrglaube eingebürgert, daß nur ein gefädeltes FORTH ein echtes FORTH ist. Fädelung ist nur ein anderer Name für Interpreter. Fädelung ist eine Implemtierungstechnik und kein Teil der Sprache.

Man kann grob drei Implementierungsarten unterscheiden die in FORTH-Neusprache den sinn-entleerten Zusatz „threaded" erhalten haben: (a) token threaded; (b) direct threaded, indirect threaded ;(c) subroutine threaded, native code. Zugrunde liegt immer der Gegensatz von Geschwindigkeit gegen Speicherverbrauch. Den geringsten Speicherverbrauch hat der Token-Interpreter, er ist aber dafür am langsamsten. Am schnellsten ist native code, echter Assembler. Dafür ist hier der Speicherverbrauch sehr hoch. Die eigentlichen Threaded-Varianten liegen zwischen beiden Extremen.

Bei den FORTH-Befehlen gibt es Unterschiede zwischen den „primitives" und den „secondaries". Die Primitives sind die Befehle, die direkt in Assembler definiert wurden und den Nucleus ausmachen. Secondaries sind Befehle, die aus Primitives zusammengesetzt wurden. Sie wurden bereits aus FORTH-Source compiliert.

Ferner gibt es wenigstens drei unterschiedliche Grundtypen von Befehlen: Erstens die simplen, interpretierbaren Befehle. Zweitens Daten, die auf den Stack gelegt werden sollen. Sie werden „literals" genannt. Und schließlich Sprungbefehle. In folgenden Beispielen werden die letzte beiden Sonderfälle nicht behandelt, weil sonst die Übersichtlichkeit leiden würde.

token threaded

Byte-Tokens werden gerne in BASIC-Interpretern verwendet. Genauso Smalltalk und Java. Es wurde jedoch anscheinend noch nie ein FORTH tatsächlich so implementiert. Jedes Byte wird als Offset in eine Tabelle verwendet, die 16 Bit breite Sprungziele enthält (Bild 4). Das Sprungziel ist die Adresse des eigentlichen Unterprogramms. Man ist dabei nicht auf 256 Befehle beschränkt. Vielmehr definiert man die wichtigsten 255 Befehle in der Haupttabelle und benutzt einen Offsetwert, meist 00, als Umschalter in die nächste Untertabelle die wieder 255 Einträge hat. Ein Token für die Untertabelle ist damit zwei Byte lang, das erste Byte ist 00. Der Mechanismus hat damit die Charakteristik des Huffman-Datenkompressionsverfahren. Die gängigen Befehle, werden mit der Haupttabelle codiert und verbrauchen damit nur ein Byte.

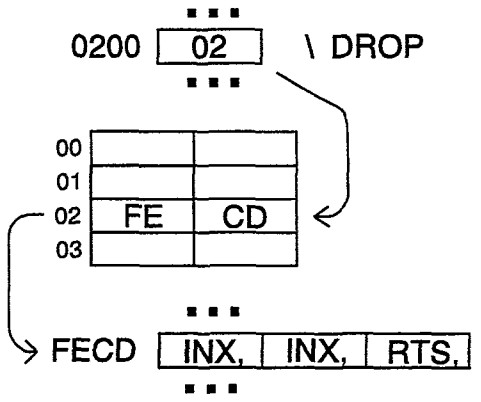


Bild 4: Token Threading

direct threaded

Das ist das klassische Verfahren für FORTH. Im Programmspeicher liegt die Adresse des Unterprogramms, das ausgeführt werden muß (Bild 5). Jeder Befehlsaufruf belegt damit 16 Bit im Speicher, statt 8 Bit wie beim Token-Interpreter. Genau wie bei diesem ist ein kurzes Interpreterprogamm nötig, das den Befehl tatsächlich zur Ausführung bringt. Abhängig von den Gegebenheiten der CPU gibt es eine Unzahl von Varianten, bei denen die umständlicheren meist „indirect threaded" genannt werden.

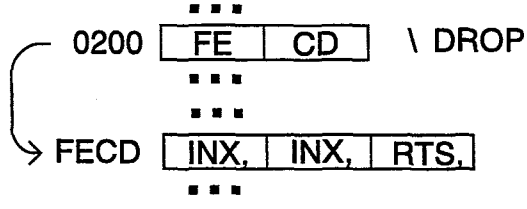


Bild 5: Direct Threading

subroutine threaded

Man braucht das Interpreterprogramm nicht, wenn man das Unterprogramm durch die CPU mittels eines Unterprogrammaufrufs zur Ausführung bringt (Bild 6). Man gewinnt an Geschwindigkeit, braucht aber nun gleich drei Bytes.

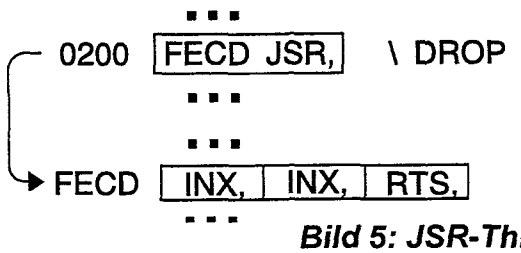


Bild 5: JSR-Threading

native code

Man muß auch kein Unterprogamm aufrufen, sondern man kann die benötigten Opcodes direkt ins Programm einfügen (Bild 7). Leider täuscht das im Bild gewählte Beispiel bezüglich des typischen Speicherverbrauchs. Die meisten Befehle belegen so auf einer 8 Bit CPU mehr als 10 Bytes im Speicher. Auf komplexen CPUs wie dem 68000 oder Stackprozessoren sind die Opcodes jedoch so leistungsfähig, daß Nativecode kompakt und damit anwendbar wird.

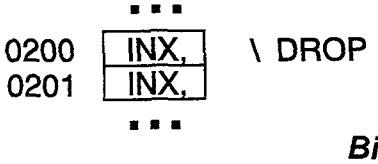


Bild 7: Native Code

Die Entscheidung, welche der drei Techniken man verwendet, hängt von den Anforderungen der Anwendung ab. Mit den Interpretern verbraucht man weniger Speicher, was bei OTPs, also Controllern mit integriertem EPROM sinnvoll sein kann. Verwendet man hingegen externe EPROMs, ist Speicherverbrauch nicht mehr so kritisch. Bei historischen FORTHs fiel die Wahl auf direct threading, weil das zu Zeiten des 2716 EPROMs der beste Kompromiß zwischen Geschwindigkeit und Speicherverbrauch war. Heute verschiebt die Entwicklung der Speicher den Kompromiß zu Subroutine-(JSR)-threading. Bei CPUs mit 16 Bit Befehlen, wie dem 68HC11 und optimierenden Compilern ist teilweise auch native code möglich. Bei CPUs mit sehr umfangreichem Befehlssatz, also mikrocodierten Typen, aber auch dem Z80, ist eventuell die Erzeugung von Nativecode sinnvoll, um möglicht viele der Opcodes verwenden zu können. JSR-threading greift immer nur auf die wenigen Opcodes zu, mit denen der Nucleus codiert wurde. Dieses Verfahren ist damit für CPUs mit wenigen Opcodes, wie den 6502 und 68HC05 günstig.

Aufgrund der geringen kommerziellen Verbreitung von FORTH, wurde hier wenig technische Weiterentwicklung betrieben. Insbesondere PD-Implementierungen beruhen meist immer noch direkt auf fig-FORTH und damit auf Steinzeittechnologie.

Standards wie FORTH-83 schreiben die Implementierungsmethode zwar nicht explizit vor. Aber FORTH-83 wurde meist direct-threaded implementiert und den Programmierern ist selten klar, welche Teile der Sprache tatsächlich durch den Standard abgedeckt werden. Viele Implementierer sind deshalb hier bei direct-threaded geblieben, um Portabilitätsprobleme generell zu vermeiden.

❑

Einladung

zur ordentlichen
Mitgliederversammlung
der Forth-Gesellschaft e. V.

am
Sonntag, den 27. April 1997
um $9^{\underline{00}}$ Uhr,

im
Europa-Hotel
Am Ludwigsplatz 5-6
67059 Ludwigshafen

Tagesordnung

1. Rechenschaftsbericht des Direktoriums
2. Kassenbericht und Bilanz 1996
3. Aussprache und Entlastung des Direktoriums
4. Wahl des Direktoriums
5. Vorhaben und Ziele der Gesellschaft
6. Planung des Finanzhaushalts 1997
7. Verschiedenes

Ergänzende Tagesordnungspunkte sind bis zum 12. April über das Forth-Büro beim Direktorium einzureichen.

Die Teilnahme an der Mitgliederversammlung ist kostenlos. Sie findet traditionsgemäß direkt nach Ende der Jahrestagung der Forth-Gesellschaft an deren Veranstaltungsort statt. Wir bitten um zahlreiches Erscheinen, ausreichend Zeit und viele gute Ideen für das nächste Jahr der Forth-Gesellschaft. Die Mitgliederversammlung wird voraussichtlich etwa 3 Stunden dauern.

Das Direktorium

Bücher

Matthias Kalle Dalheimer: Java Virtual Machine

Sprache, Konzept, Architektur

O'Reilly, Köln 1997
209 Seiten (deutsch)
Paperback DM 29,--

Das Buch wendet sich an den potentiellen Java-Bytecode-Interpreter-Konstrukteur. Mit Java als Sprache beschäftigt es sich nur am Rande. Auf die heutzutage erhältlichen Java-Compiler geht es aber ein. Diese erzeugen aus Java den Maschinencode der Java-Maschine, den "Bytecode". Wesentlicher Bestandteil der Sprache Java ist die "Klasse". Wesentlicher Bestandteil der an den Interpreter weiterzugebenden Bytecode-Information ist die Klassen-Datei. Hier setzt das Buch an.

Neben dem Aufbau der Klassendatei werden die Maschinenbefehle ausführlich besprochen. Es ist nicht zu erkennen, ob hundertprozentig vollständig, da von den 255 vorgesehenen "Byte"-Codes (noch) nicht alle verwendet werden. Die Befehle der ("virtuellen") Java-Maschine bestehen aus dem Opcode von einem Byte Länge und dem oder den Operanden der (die) weitere 0 bis 4 Byte beansprucht (beanspruchen).

Viele Operationen verwenden zum Einholen der Eingabedaten und zum Ablegen der Ergebnisse den Stack, andere lokale Variablen. Gotos sind in bedingter wie auch in unbedingter Form, wie auf Maschinenebene nicht anders zu erwarten, reichlich vorgesehen, direkte Manipulation des (virtuellen) RAMs in dem Sinne, wie es beispielsweise für selbstmodifizierenden Code nötig wäre, scheint aber nicht möglich zu sein.

Das Buch enthält auch ein kurzes Kapitel über Sicherheitsfragen: Kann man mit Java-Code über das Internet andere Maschinen lahmlegen? Schließlich sind noch ein paar Seiten über die Entstehungsgeschichte und über die Zukunftsaussichten von Java zu lesen.

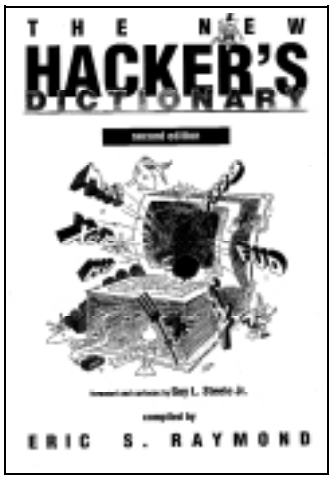
Fazit: Das Buch ist gut. Nach dem Durchlesen ist mir klar geworden, dass man statt Java auch jede andere Hochsprache in den Bytecode der Java-Maschine übersetzen könnte, beispielsweise auch Forth ("Forth-Online"), und dass man als Bytecode auch jede andere Maschinensprache nehmen könnte. Vielleicht nicht gerade die des 6502, aber auf jeden Fall doch wohl die des Transputers T800, die ja ebenfalls den IEEE-754-Spezifikationen genügt. Was zur Erklärung der derzeitigen Sogwirkung von Java angeführt werden kann, ist wohl doch nur die geballte Werbekraft der irgendwie am Projekt beteiligten Firmen.

beh

Bücher

Eric S. Raymond: The New Hackers Dictionary

MIT Press, 1993
ca. 500 Seiten, Englisch
ISBN 0-262-68079-3
ca. 36.-DM



In der Buchhandlung habe ich dieses Buch beim Stöbern im Fach "Kryptologie" gefunden. Wo soll man auch sonst ein Buch reinstellen das die Bedeutung von "foo" und "bar" erklärt ? Auf den ersten Blick eine 450 Seiten lange Sammlung von Insiderjokes. Hier werden jedoch viele Randbereiche und praktischen Probleme der Computertechnik angesprochen, die sonst nur durch mündliche Überlieferung zugänglich sind. Dadurch ist das Buch auch von praktischem Nutzen. FORTH und andere tote Sprachen finden keine Erwähnung, da der Hintergrund des Autors mehr auf LISP, C und UNIX basiert.

"Bohr bug /bohr buhg/ [from quantum physics] n. A repeatable bug; one that manifests reliably under a possibly unknown, but well-defined set of conditions. Antonym of heisenbug; see also mandelbug, schroedinbug."

Rafael Deliano

Bücher

Peter Gerdsen, Peter Kröger: Digitale Signalverarbeitung in der Nachrichtentechnik

Springer Verlag, 1993
340 S., mit 3,5''-Diskette mit Source für TurboPascal 5.5
ISBN 3-540-55520-X
48,00 DM

Es gibt eine Unmenge DSP-Bücher für Anfänger wo einem FFT, IIR- und FIR-Filter erklärt werden. Dieses Buch geht ein Stück weiter und beschreibt nützliche Dinge wie CORDIC, Sinuserzeugung, Hilbert-Transformation, PLLs, AM- und FM-Modulatoren und Demodulatoren. Dabei wird hier nicht akademisch theoretisiert, sondern es gibt dazu Programme. Diese sind in HLL geschrieben und nicht in einem unportierbaren DSP-Assembler. Verwendet wurde Pascal und das ist als Pseudocode sicherlich lesbarer als C oder BASIC (oder FORTH). Weiterhin ist erfreulich, daß die Source dort ist, wo sie hingehört, nämlich auf der Diskette. Auf den 340 Seiten des Buches hingegen werden die Algorithmen mit Text und vielen Illustrationen durchgesprochen.

Rafael Deliano

Forth-Gruppen regional

Kiel
Ulrich Hoffmann
Tel.: +4351 - 712 217 pf

Rhein-Ruhr
Jörg Plewe
Tel.: +208 -49 70 68 p
Treffen: jeden 1. Samstag im Monat im S-Bahnhof Derendorf
Münstererstr. 199,

Moers
Friederich Prinz
Tel.: +2841 -5 83 98 p
Treffen: jeden Samstag 14:00
Arbeitslosenzentrum,
Donaustr. 1, Moers

Darmstadt
Andreas Soeder
Tel.: +6257 -27 44

Mannheim
Thomas Prinz
Tel.: +6271 -28 30 p
Ewald Rieger
Tel.: +6239 -86 32 p
Treffen: jeden 1. Mittwoch im Monat, Vereinslokal Segelverein Mannheim e. V.,
Flugplatz Mannheim-Neuostheim

µP-Controller Verleih

Thomas Prinz
Tel.: +6271 -28 30 p

Gruppengründungen, Kontakte

Regional
Stuttgart
Wolf-Helge Neumann
Tel.: +711 -8 87 26 38 p

Fachbezogen
8051 ... (Forth statt Basic, e-FORTH)
Thomas Prinz
Tel.: +6271 -28 30 p

Forth-Hilfe für Ratsuchende

Forth allgemein
Jörg Plewe
Tel.: +208 -49 70 68 p

Karl Schroer
Tel.: +2845 -2 89 51 p

Jörg Staben
Tel.: +2173 -75 708 p

Spezielle Fachgebiete

Arbeitsgruppe Marc4
Rafael Deliano
Tel./Fax.: +89 -841 83 17 pf

Anfänger und Wiedereinsteiger
Gerd Limbach
Tel.: +2051 -25 51 12 p

FORTHchips (FRP1600, RTX, Novix...)
Klaus Schleisiek-Kern
Tel.: +40 375 008-03 g

F-PC & TCOM, ASYST (Meßtechnik), embedded controller(H8/5xx// TDS2020, 8051 ... eFORTH...), FUZZY
Arndt Klingelnberg
Tel.: ++32 +87-63 09 89 a

HS/Forth (Harvard Softworks)
Wigand Gawenda
Tel.: +30 -44 69 41 p

Standardisierung, ANS-Forth KI, OOF
Ulrich Hoffmann
Tel.: +4351 - 712 217 pf

KI (Künstliche Intelligenz), OOF (Object Oriented Forth)
Birgit Steffenhagen
Tel.: +38204 - 129 33 pa
+381 - 498 35 52 g

Forth-Vertrieb
volksFORTH/ultraFORTH, RTX/FG/Super8/KK-FORTH
Ingenieurbüro Klaus Kohl
Tel.: +8233 -3 05 24 p
Fax: +8233 -99 71 f

Forth-Mailbox (KBBS)
+431-533 98 98 *:8N1*
Sysop: Holger Petersen
Fax: +431-533 98 97 f
Tel: +431-533 98 96 p *bis 22 Uhr*
Mail: hp@kbbs.org

Hinweise

Zu den Telefonnummern

f == FAX
a == *Anrufbeantworter, hier können Sie Ihren Ansprechpartner eventuell vorinformieren,* erwarten Sie bitte keinen (kostspieligen) Rückruf
g == geschäftlich, zu erreichen innerhalb typischer Arbeitszeiten
p == privat, zu erreichen außerhalb typischer Arbeitszeiten

Die Adressen des Forth e. V. (Forth Büro) und der Redaktion/ finden Sie im Impressum

FORTecH Software

- Forth-Entwicklungsumgebung comFORTH unter DOS oder Windows
- interaktive Crossentwicklungssysteme für Mikroprozessoren von Intel, Motorola, Zilog, TI ...
- Softwareentwicklung für PC und Mikrocontroller
- System- und Anwendungsprogrammierung unter Windows

comFORTH

- Forth-Entwicklungsumgebung für Windows
- interaktive Benutzbarkeit aller Windows-API-Funktionen und -Strukturen
- kombinierbar mit anderen Programmiersprachen
- Unterstützung von DDE, DLL, VBX, ...

fieldFORTH

- Forth-Entwicklungssystem für eingebettete Systeme
- interaktive Programmierung off-line und on-line
- verfügbar für diverse 8-, 16- und 32-Bit Mikrocontroller und -Prozessoren (TMS320C40, M68332, M68HC11,...)
- **NEU!!!** Evaluation-Kit M68HC11 inclusive Board MINI-HC11 296,70 incl. MwSt.

FORTecH Software GmbH
J.-Jungius-Str. 9 • D-18059 Rostock • Tel: (03 81) 4 05 94 72 • Fax: (03 81) 4 05 94 71