VIERTE DIMENSION

2/1996

12. Jahrgang 1996, 2. Quartal, DM

FORTH MAGAZIN

Organ der Forth Gesellschaft e.V.

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

Ingenieurbüro Dipl.-Ing. Wolfgang Allinger

Tel. (+Fax.) 0+212-66811
Brander Weg 6
D-42699 Solingen

Entwicklung von µC, HW+SW, Embedded Controller, Echtzeitsysteme 1 bis 60 Computer, Forth+Assembler PC / 8031 / 80C166 / RTX2000 / Z80 ... für extreme Einsatzbedingungen in Walzwerken, KKW, Medizin, Verkehr / >20 Jahre Erfahrung.

FORTecH Software GmbH

Tel.: 0+381 -405 94 71 (Fax: -4059.471)
Joachim-Jungius-Str. 9
D-18059 Rostock

PC-basierende Forth-Entwicklungswerkzeuge, System comFORTH für DOS und Windows, Cross- und DownCompiler für diverse Microcontroller, Controllerboards mit 80C196, 80C537 und H8, Softwareentwicklung für Microcontroller und PC's, auch unter Windows (und fremdsprachig)

ETA Elektrotechnische Apparate GmbH

Tel.: 0+9187 -10.0 (Fax: -10.397)
Industriestr. 2-8
D-90518 Altdorf (b. Nürnberg)

Produkte für Echtzeitanwendungen
FRP1600: Echtzeitprocessor optimiert für Forth
RP-PB1: FRP1600 Prototyping Board.

Ing.Büro Klaus Kohl

Tel.: 0+8233-30 524 (Fax: --9971)
Postfach 11 73
D-86406 Mering

FORTH-Software (volksFORTH, KKFORTH und viele PD-Versionen). FORTH-Hardware (z.B. Super8) und -Literaturservice. Professionelle Entwicklung für Steuerungs- und Meßtechnik.

Dipl.-Ing. Arndt Klingelnberg

Tel.: 0+2404 -61648 (Fax: -63039)
Strassburgerstr. 12
D-52477 Alsdorf (b. Aachen)

Computergestützte Meßtechnik und Qualitätskontrolle, Fuzzy, Datalogger, Elektroakustik (HiFi), MusiCassette High-SpeedDuplicating, Tonband, (engl.) Dokumentationen u. Bed.-anl.

Möchten auch Sie oder Ihre Firma hier aufgeführt werden? Bitte wenden Sie sich an die Anzeigenverwaltung (s. Impressum).

Ihre Anzeige plus 3 Zeilen je 45 Zeichen Text kosten 90.-DM (incl. 20.-DM Einrichtung/ Änderung, je Zusatzzeile 10.-DM) und das komplett für ein ganzes Jahr.

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
Mail: secretary@admin.FORTH-eV.de

Redaktion & Layout

Claus Vogt
Ebersstr. 10
D-10827 Berlin
Tel.: 030 / 782 81 79 (Fax & BBS nach Bedarf)
Mail: vd@FORTH-ev.de

Anzeigenverwaltung: Ulrike Schnitter c/o Forth-Ges.; PF 1110; 85701 Unterschleißheim.
ANS-Forth: Ulrich Hoffmann; uho@informatik.uni-kiel.de; Sehestädter Str. 26; 24340 Eckernförde.
Forth international: Fred Behringer; Planegger Str. 24; 8124 1 München.
Zeichnungen: Rolf Kretzschmar;

Titelbild: Hans-Georg Schmid

Redaktionsschluß '96

Erste Januar-, April-, Juli-, und Oktoberwoche.

Erscheinungsweise

Viermal im Jahr.

Preis

Einzelpreis: DM 10,-

Forth-Tagung 96 - ein Erfolg

Die diesjährige Forth-Tagung in Mittweida hat mir gut gefallen. Dank gebührt Thomas Beierlein von der Fachhochschule Mittweida für die ausgezeichnete Organisation von Tagung, Unterkunft und Rahmenprogramm.

In den Vorträgen kam die spielerische Seite von Forth nicht zu kurz (Jörg Plewe), aber auch an fundiertem Wissen wurde überraschend viel geboten.

Interessant der Vortrag von Ulrich Hoffmann (Uni Kiel) über Fragen der Softwarezuverlässigkeit (immer aus der Sicht von Forth). Dieser Begriff scheint sich tatsächlich von dem Begriff der mathematischen Zuverlässigkeit (Intaktwahrscheinlichkeit), dessen Theorie von v.Neumann vorbereitet und von Moore und Shannon begründet wurde (1956), zu unterscheiden. Oder vielleicht doch nicht? Man spricht ja sowieso schon von Forth-"Maschinen", versteht darunter aber bestimmte Software-System(kern)e.

Interessant auch ein Vortrag von Birgit Steffenhagen (Uni Rostock), in dem Fragen der unscharfen Regelung (fuzzy control) angeschnitten wurden. "Fuzzy" scheint heute, 30 Jahre nach Zadeh, tatsächlich allgemeiner Bestandteil unseres Sprachgebrauchs geworden zu sein. Jedenfalls hatte keiner gegen Begriffe wie Zugehörigkeitsfunktion, Aggregationsoperation und linguistische Variable ernsthafte Einwände. Ich sehe da aber wieder Parallelentwicklungen heraufziehen: Die Arbeiten über unscharfe Entscheidungen (bei mehrfacher Zielsetzung) aus der Entscheidungstheorie und dem Operations Research schlechthin - sie liegen mir näher als Fragen der unscharfen Steuerung, überschneiden sich aber natürlich mit diesen - werden in der Literatur über Steuer- und Regelungstechnik kaum berücksichtigt.

Interessant schließlich Friederich Prinzens virtueller Kampf gegen WIND(OWS)mühlen: WINDOWS - nein, danke; WIN32FORTH aber eigentlich doch. Auch ich habe mich jahrelang gegen WINDOWS gesperrt: Es rattert los und man weiß nicht, was geschieht. Seitdem es mir aber gelungen ist, dem TrueType-Zeichensatz ARIAL über CORELDRAW 3.0 selbstgemachte abgeänderte mathematische Zeichen hinzuzufügen, beuge ich mich der unerforschlichen Allmacht WINDOWS'.

Viel Lob wurde Claus Vogt für die hervorragende Arbeit bei der Herausgabe der VD gezollt. Sehr zu Recht, meine ich.

Fred Behringer, München, Ressort "Intenationales" des Forth-Magazins

EchtZeit '96 | iNet '96

18. - 20. Juni 1996
Karlsruher Kongreß- und
Ausstellungszentrum, Stadthalle

Aufruf zur Teilnahme am EchtZeit-Programmierwettbewerb

Während der diesjährigen Kongreßmesse EchtZeit/ iNet '96, die vom 18. bis 20. Juni in Karlsruhe stattfindet, wird zum siebten Mal der EchtZeit-Programmierwettbewerb qusgetragen, bei dem unter den Augen des Fachpublikums Programmierer und Tools ihre Eignung für die hardwarenahe Programmierung beweisen müssen.

Wettbewerbsziel ist es, eine bis zum Beginn des Wettbewerbs unbekannte Echtzeit-Programmieraufgabe zu lösen, etwa die Steuerung eines mechanischen Modells. Dazu stehen maximal dreieinhalb Stunden zur Verfügung. Zur Lösung der Aufgabe dürfen beliebige Rechensysteme - vom C64 bis zum Pentium - herangezogen werden. Auch die Wahl der Software-Werkzeuge ist freigestellt. Teilnehmen können einzelne Personen oder Teams mit bis zu drei Mitgliedern.

Es gewinnt dasjenige Team, das die zu Beginn des Wettbewerbs demonstrierte Funktion des Modells mit seiner Hard- und Softwareausrüstung als erstes nachbilden und vorführen kann. Wenn kein Team die gestellte Aufgabe innerhalb der vorgegebenen Zeit lösen kann, werden die Plätze nach dem Umfang der vorhandenen Teillösungen vergeben. Es können maximal zehn Teams am Wettbewerb teilnehmen.

Die Aufgabe wird in diesem Jahr vom Sieger des Vorjahres gestellt, Herrn Dieter Peter. Er ist selbständiger Elektronik-Entwickler und arbeitet als Consultant für die Firma Microchip.

Der Wettbewerb findet am Donnerstag, dem 20. Juni 1996, ab 12.00 Uhr im Ausstellungsbereich der EchtZeit/ iNet statt. An melden kann man sich bis zum 30. Mai 1996 bei der Redaktion der Zeitschrift Elektronik, Telefax (0 89) 99115-520. Anfang Juni erhalten die Teilnehmer nähere technische Informationen über die erforderliche Mindestausrüstung sowie die Anschlußbedingungen des Modells, wobei ein handelsüblicher PC für die Lösung der Aufgabe ausreichend ist.

Die Siegerpreise setzen sich vollständig aus Spenden von Sponsoren zusammen, die ausschließlich Aussteller der EchtZeit/iNet '96 sind:

- Delta t GmbH, Hamburg
- Hitex Systementwicklung, Karlsruhe
- HSP GmbH, Münster
- Dr. Rudolf Keil GmbH, Dossenheim
- QNX Software Systems, Ontario, Kanada
- SYSGO Real-Time Solutions GmbH, Mainz
- Thomson Software Products/Alsys GmbH, Karlsruhe
- Wind River Systems, Ismaning

Nähere Einzelheiten über:

Redaktion Elektronik, Franzis-Verlag
Dornacher Str. 3d
D-85622 Feldkirchen

Tel. (0 89) 99115-511, Fax (0 89) 99115-20

Ihr Ansprechpartner: Joachim Kroll

oder von :

NETWORK GmbH
Wilhelm-Suhr-Str. 14
D-31558 Hagenburg

Tel. (0 50 33) 70 57, Fax (0 50 33) 79 44

Ihre Ansprechpartnerin:
Christiane Kiel

Einem Teil der Auflage liegt eine

Freikarte

zum Besuch der

Echtzeit'96 bei

Vielen Dank an die Ausrichterin

1	3.161	3.03.96	20:43	vdl962.lst
2	1.989	22.04.96	22:26	disco\discotin.lst
3	14.296	12.05.96	12:00	pcmess\terminal.lst
4	16.384	5.05.96	23:35	pcmess\terminal.scr
5	637	20.03.96	13:29	sps\readme.txt
6	57.344	20.03.96	13:00	sps\sps7.scr
7	21.587	20.03.96	11:36	sps\sps7.txt
8	40.399	20.03.96	12:51	sps\vfsps.com
9	31.778	10.02.96	18:27	sps\volks4th.com
10	10.013	4.05.96	11:45	tropfen\tropfen.f

Anzahl Dateien: 10 Anzahl Bytes: 197.588

= *Listing auf Diskette*

zu ‘kleinstes Forth’ im Leserforum VD 1/96, Seite 7

Was Deine Suche nach Last-Minute-Fax-Deutschkenntnisse-Korrektoren angeht: Ich tät' mich da schon anbieten, wenn ich wüßte, ob es nur um Deutsch geht oder auch um Inhaltliches: Wäre ich z.B. verpflichtet, bei allfälliger Korrektur der Notiz ‘nochmal kleinstes Forth’ von Fred Behringer denselben darauf anzusprechen, daß nach meinen Unterlagen der Herr Pierce sich PEIRCE schreibt, sein Operator die NOR-Verknüpfung ist, während NAND von Herrn Sheffer stammt und sich die Aussagenlogik sowohl allein mit NAND als auch mit NOR aufbauen läßt?! Ich hoffe doch: NEIN!

Denn obiges zu schreiben, ist mir nur möglich, weil ich mich eben mal SPS-mäßig mit zweistelligen Operatoren beschäftigt habe ...

Mit freundlichen Grüßen,
Gerhard Raisig (rai)

[Die Korrekturen sollten rein sprachlicher Art sein. Hinweise auf inhaltliche Probleme sollten an Editor und Autor gehen, die dann über Änderungen oder Diskussion im Leserforum nachdenken. Du hast es also genau richig gemacht, lieber Gerhard. /clv]

zum Titelthema ‘Profi oder Hobby’ in VD 1/96

Hier eine spontane Ovation für den VD-Artikel ”Forth braucht keinen ... Fachverband”. Besser kann man nicht ausdrücken, wie auch ich empfinde.

mfg (f=forthig)
100522.135@compuserve.com
(Johannes Teich)

Heute - 25/03. - ist die VD angekommen. Ich habe sie gleich verschlungen. Super “outfit” !!!

Ebenfalls SUPER - der Aufsatz vom Fred Behringer. Den will ich “unterschreiben”. Ich bin auch kein “Professioneller”. Das will ich auch gar nicht. Dazu bin ich viel zu gerne Bergmann! Eine Zertifizierung von Forth-Programmierern würde ich konsequent ignorieren - und mich damit brüsten, NICHT ZERTIFIZIERT zu sein !

F.PRINZ@MHB.gun.de (Friedrich Prinz)

Prozessorgeflüster:
Free Forth FPGA

Auf dem letzten Treffen der Münchner Forth-Gruppe wurde der Vorschlag gebracht (recht massiv), aus den Teilen meines 4stack-Prozessors (sobald oder während er fertig wird) einen richtigen Forth-Prozessor zu basteln (durch Auskommentieren aller überflüssigen Teile). Natürlich wird der Prozessor dadurch nicht schneller, aber er wird klein genug, daß man ihn auf ein FPGA (ein “Field Programmable Gate Array”, also ein Gate Array, das man aus einem EPROM laden kann) quetschen kann.

Die größten FPGAs haben inzwischen 50K Gates, das ist deutlich mehr als ein 68000, und sie haben Gatterlaufzeiten unter einer Nanosekunde, das heißt, ein Prozessor kann mit etwa 33 MHz laufen. Damit wäre man (zumindest solange man Forth verwendet) im Bereich von üblichen teureren Controllern, also 486ern oder R3000-Dingern.

Der Nachteil: Das ist alles noch eine Idee. Die Teile zum Ausschlachten gibt es zwar schon. Manche Sachen gehn aber auf einem FPGA nicht so wie mit einem Gate-Array (die wertvollen Gatter als Cache verbraten is nicht - und 33MHz sind für DRAM schon zu schnell). Und der Befehlssatz ist natürlich auch nur teilweise vorgegeben - andere Teile müssen noch erfunden werden.

Und bei alledem ist die Anzahl “Freiwilliger” in der FG, die auch die fachliche Kompetenz haben, äußerst klein, will sagen, die Arbeit bleibt an mir hängen.

paysan@informatik.tu-muenchen.de (Bernd Paysan),
März’96

Direktorium fördert neue Mitglieder

Auf der Jahrestagung der Forthgesellschaft hat das neue Direktorium von der Mitgliederversammlung den Auftrag bekommen, sich zukünftig mehr um neu in den Verein kommende Mitglieder zu kümmern. In einem ersten Schritt wird das Direktorium diesem Auftrag dadurch entsprechen, daß neue Mitglieder ein Anschreiben der Direktoren erhalten, in welchem Sie persönlich begrüßt und in der Forthgesellschaft willkommen geheißen werden. Der Inhalt dieses Schreibens interessiert sicher nicht nur die Teilnehmer der Mitgliederversammlung, sondern alle Mitglieder der Gesellschaft. Darum wird dessen Inhalt an dieser Stelle veröffentlicht.

Für die Direktoren der
Forthgesellschaft
Friederich Prinz

Leserforum

Lieber Forthfreund,

im Namen der Mitglieder unserer Gesellschaft, im Namen des Forthbüros und im Namen des Direktoriums der Gesellschaft heißen wir Sie in unserem Verein herzlich willkommen.

Die FORTH-Gesellschaft e.V. ist ein Verein, in dem sich Forther aus den Bereichen Wissenschaft, Kommerz und Hobby zusammengefunden haben, um die ”Sprache” Forth und ihre nach wie vor überlegenen Konzepte zu fördern und ihre Verbreitung zu betreiben. In diesem Kontext ist auch das Direktorium bemüht, ganz besonders solche Forth-Systeme zu unterstützen, die für den Bereich ”Lehren und Lernen” geeignet erscheinen. Hierzu zählen wir sowohl kleine, überschaubare und hinreichend dokumentierte Forth-Systeme aus dem PD- und Sharewarebereich, als auch solche Systeme, die von kommerziellen Anbietern in ”Lite”- oder Testversionen als Einstieg in professionelle Entwicklungsumgebungen zur Verfügung gestellt werden. Sowohl die nationale als auch die internationale ”Forthszene” hält hierzu ein reiches Angebot unterschiedlicher Systeme bereit, die Ihnen zum gesamten Spektrum der Programmierung mit Forth wertvolle Hilfen, Unterstützung bei Ihren Aufgaben, Tips und Hinweise oder einfach nur Anregung bieten.

In diesem Zusammenhang bitten wir Sie, uns mitzuteilen, für welche Aspekte von Forth Sie sich ganz persönlich und besonders interessieren. Wir möchten Ihnen den - vielleicht schon erfolgten - Einstieg in Forth erleichtern, und Ihnen gleichzeitig zeigen, daß die Forth-Gesellschaft weit mehr ist als ein über die gesamte Bundesrepublik ”verstreuter” Verein. Vielleicht können wir Ihnen ein Forth-System vermitteln, daß Ihren besonderen Interessen entgegenkommt, oder Material zum Studium eines besonderen Systems, oder einfach nur einen Ansprech-

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

Forth-Magazin 'Vierte Dimension'
Claus Vogt, Ebersstr. 10,
D-10827 Berlin, vd@FORTH-ev.de

Forth. Kurz und Knapp. Das

partner, der sich womöglich ganz in Ihrer Nähe befindet.

"Ganz in Ihrer Nähe" finden Sie relativ viele Forther - wenn Sie über einen Modem verfügen und das Angebot der FORTH-Gesellschaft und der Kieler Mailbox KBBS nutzen wollen. Dort stehen Ihnen über Ihre eigene Internetadresse Forther aus ganz Deutschland und der ganzen Welt ständig in Diskussionen zur Verfügung, und meist auch mit Rat und Tat.

Die "Vierte Dimension", unsere Quartalsschrift, werden Sie als Mitglied der Gesellschaft zukünftig regelmäßig erhalten. Sie finden darin viele interessante Aufsätze, Notizen und Hinweise. Auf zwei Hinweise möchten wir, die Direktoren der Gesellschaft, Sie besonders aufmerksam machen.:

- Falls Sie die Datenfernübertragung bereits nutzen, oder zukünftig nutzen wollen, ist sicher die Liste mit den Mailadressen der dfü-treibenden Vereinsmitglieder interessant für Sie.

- Auf der vorletzten Seite der "VD" finden Sie darüber hinaus eine Liste mit Ansprechpartnern in lokalen Gruppen der Gesellschaft, sowie Ansprechpartner zu besonderen Themen- und Fachgebieten.

Bitte machen Sie von beiden Listen Gebrauch ! Ihre neuen "Vereinskameraden" werden sich freuen, von Ihnen zu hören.

Und noch eine Bitte zum Schluß: Schreiben Sie den Direktoren (oder rufen Sie uns einfach an), und teilen Sie uns bitte mit, was Sie sich von der Forth-Gesellschaft "erhoffen", was wir für Sie tun können. Und vielleicht haben Sie auch etwas für den Editor der VD ? Ein Aufsatz, Hinweise auf "Material zu Forth", Hinweise auf Literatur-...wir freuen uns über alles, was Sie zum Gelingen unserer Gesellschaft beitragen wollen.

In diesem Sinne begrüßen wir Sie noch einmal ganz herzlich in Ihrer Forth-Gesellschaft und verbleiben

(für das Direktorium)

Literatur:
Tagungsbände von den Forth-Tagungen

Das Direktorium erwägt, die Tagungsbände auch öffentlich herauszugeben und möchte hierzu zunächst prüfen, ob seitens der Mitglieder Interesse daran besteht. Anfragen nach dem Tagungsband bitte an das Forth-Büro oder direkt an direktorium@FORTH-ev.de

Forthsysteme:
WinFORTH von LMI

Fritz Prinz erkundigte sich bei Ray Duncan nach den Unterschieden der verschiedenen Versionen und den Preisen. Im folgenden Drucken wir Rays Antwort ab:

Yes, the Explorer version is $100. This is basically the same as the "Shareware" version except that the SAVE and TURNKEY commands are operational. The manual, demo programs, etc. are pretty much the same (maybe there have been some minor updates & additions since the Shareware version was posted).

The differences between the Explorer and the Professional are mainly:

1) Professional package includes source code for assembler, editor, floating point, source code, etc.

2) Professional package includes object modules and libraries that allow you to rebuild the kernel (if you have Borland C++ 4.5 also).

3) Professional package supports the creation of DLLs written in WinForth (there are some significant limitations to what these DLLs can do, see the on-line documentation for more details).

In general we recommend that people get the Explorer package to start with and not spend the money for the Professional package until they really need it. Most people never need it. You can always upgrade later for the difference in price.

The shipping to Germany would be $1 or so if you just get the $100 package without printed manual; about $25 if you get the $150 package with printed manual.

Regards, Ray Duncan (rduncan-@mailgate.csmc.edu)

Forthsysteme:
Win32Forth 3.1 Beta II released

Der Tom hat offenbar eine wahnsinnig produktive Phase...

Jedenfalls ist Win32For 3.1 Beta II Release fertig. Die Version läuft wieder unter OS/2, bzw. WINOS/2. Die Probleme beim Drucken unter WINDOWS95 sind laut Toms Beschreibung ebenfalls behoben. Und das Teil liegt wieder in der MHB zum Download bereit. W32F31B2.EXE ist 1.221 KByte groß.

Viel Spaß
F.PRINZ@MHB.gun.de (Friedrich Prinz), März'96, Forthgruppe Moers

Forthquellen:
PID-Regler für ZF-Forth

Ich habe eine Reglersimulation in ZF-Forth geschrieben. Es kann eine Meßstrecke angepaßt werden (z.B. Mengen- oder Standregelung) und ein Regler mit P-,I- und D-Parametern eingestellt werden. Eignet sich also zum Üben dieser Einstellungen.

Ein paar Ausführungen dazu werden in der VD erscheinen. Wer mehr Interesse daran haben sollte, soll schreiben - ich mach' dann vielleicht noch eine genaue Anleitung für alle Funktionen. Der Upload wird bald in der KBBS in FORTH/MEORS/ als PID.SEQ sein.

wfuehrer@uni-upn.forth-ev.de (Wolfgang Fuehrer), feb'96

Prozessorgeflüster
DOFIN-1620 Ein Forth-Prozessor aus Weißrußland

In Minsk, der Hauptstadt Weißrußlands, wird mit dem DOFIN-1620 ein 16-Bit Forth-Prozessor gefertigt. Bei einer kürzlichen Dienstreise erhielt ich erste Muster des Prozessors und die zugehörige Dokumentation.

Während sich der Befehlssatz stark an den Novix NC4000 anlehnt, ist man im Hardwaredesign eigene Wege gegangen. Der Chip kann jeweils mit internem oder externem Daten- und Returnstack arbeiten. Intern stehen jeweils 16 Stackeinträge zur Verfügung, extern 256. Im Unterschied zum Novix teilen sich die externen Stacks einen gemeinsamen Bus. Eventuelle Konflikte bei gleichzeitigem Zugriff werden durch eine eingebaute Logik gelöst und verlängern den Befehl um einen Takt. Ansonst benötigen alle Befehle einen oder zwei Takte bei einer Taktfrequenz von 6 Mhz.

An Entwicklungswerkzeugen existiert ein konventioneller Assembler sowie ein Forth-Crosscompiler. In nächster Zeit sollen weitere Chips beschafft und ein Musterboard aufgebaut werden.

tb@tb.forth-ev.de (Thomas Beierlein)

Forthgesellschaft
Gemeinsame Teilnahme an der Echtzeit'96

Auf der Forthtagung in Mittweida hat sich Winfried Clemens freundlicherweise bereiterklärt, die Teilnahme der Forth Gesellschaft e.V. an der Echtzeit'96 vom 18. bis 20.6.1996 in Karlsruhe und den Informationsstand der Gesellschaft zu koordinieren. Wer Interesse an der gemeinsamen Teilnahme hat oder den Stand unterstützen möchte, wende sich bitte direkt an ihn. Erste Erfolge haben die Bemühungen von Herrn Clemens bereits ergeben. Wie Sie sicher bereits bemerkt haben, liegt dieser Ausgabe des Forth-Magazins 'Vierte Dimension' eine Freikarte zum Besuch der Echtzeit und iNet'96 bei. Wir bedanken uns für die freundliche Zusammenarbeit bei der Organisatorin, der Network GmbH.

Kontakt: Winfried Clemens, c/o EchtzeitSysteme, Tel: 02461-690-380, Fax: -100, clemens@clforth.FORTH-ev.de

Firmen
FORTecH umbenannt

Die FORTecH Software GmbH (Rostock) hat einen neuen Namen. Sie heißt jetzt "FORTecH Software - Entwicklungsbüro Dr.-Ing. Egmont Woitzel". Die Gründe für den Wechsel der Geschäftsform sind formaler Natur. Adresse und Telefonnummer bleiben davon unberührt. Auch die Geschäftspolitik wird unbeirrt weitergeführt und selbstverständlich wird auch der bewährt gute Support für die beliebten Forthsysteme - am bekanntesten ist wohl comForth - in gleicher Qualität weiterlaufen. (*clv)*

Firmen
DELTAt gliedert SEND aus

Auch bei der Firma DELTAt (Hamburg) gibt es Änderungen, mit denen die sich erweiternden Tätigkeitsbereiche der Mitarbeiter jetzt noch besser den steigenden Bedürfnissen des Marktes angepaßt werden sollen. Der Mitinhaber Klaus Schleisiek übernimmt die Leitung der neu gegründeten Firma SEND GmbH (Signal-Elektronik und Netz-Dienste). Die neue Firma hat ihren neuen Sitz im Vorsetzen 42 in D-20459 Hamburg, wo sie unter Tel: 040-375008-03 (Fax:-93) für ihre Kunden erreichbar sein wird. Sie übernimmt, wie der Name schon so treffend sagt, jene Firmenbereiche, die sich um die Signalelektronik und die Netzdienste drehen. Vermarktung und Support des von DELTAt entwickelten Forth-Prozessors iX1 (vgl. VD 4/95 Seite 13) verbleiben wegen der guten Markteinführung bei DELTAt. *(clv)*

Firmenpleite
Holger Dyja Geräteentwicklung

Die Firma Holger Dyja Geräteentwicklung (Berlin) hat, wie wir erst jetzt erfuhren, im November 1995 ihren Geschäftsbetrieb eingestellt. Angesichts der wachsenden Konkurrenz auf dem Markt der Geräteentwicklung mit der Programmiersprache Forth konnte die Rentabilität langfristig nicht mehr gesichert werden. Die laufenden geschäftlichen Verbindlichkeiten werden noch abgewickelt. Für Kunden, die Interesse an weitergehendem Support haben, sind entsprechende Möglichkeiten in der Vorbereitung. Sie werden gebeten, sich zwecks Erhebung des Bedarfs schriftlich mit Herrn Dyja in Verbindung zu setzen. *(clv)*

Preese(vor)schau Elrad
Forthtagung 1996

Den folgenden Text hat das Direktorium im Mai'96 der Elrad mit der Bitte um Veröffentlichung übersandt. Ob die Elrad dieser Bitte nachkommt, oder ob die Leserinnen und Leser auch weiterhin ohne Information über die Tagung weiterleben müssen, ist ungewiss. Hinweise über erfolgte Veröffentlichungen zu forthigen Themen nimmt die Redaktion (nicht nur in diesem speziellen Fall) immer gerne entgegen.

Tagung der Forth Gesellschaft e.V. 1996

Am 19.-21. April fand bei Mittweida die jährliche Tagung und Mitgliederversammlung der Forth Gesellschaft e.V. statt.

Neben den für Forth typischen Themen aus dem Bereich Echtzeit- und Mikrocontrollerentwicklung gab es Beiträge zum Thema Qualitätssicherung und deutliche Grenzübertritte zu Lisp und Java, die beide jeweils deutliche Parallelen zu Forth zeigen. Die etwa 35 Teilnehmer aus Industrie, Forschung und dem Hobbybereich konnten sich weiterhin über echtzeitfähige Garbagecollectoren oder die Programmierung von Actionspielen informieren. Neuartige, datenbankgestützte Quelltextsysteme sorgten für z.T. kontroverse Diskussionen der Fachleute.

Insgesamt zeigte die Tagung, daß sich Forth zum einen stetig weiterentwickelt und zum anderen immer wieder mit Erfolg eingesetzt wird.

Am Sonntag, den 21.4. fand die Mitgliederversammlung des Vereins statt, die Ihrerseits der Gesellschaft neue Impulse für das kommende Jahr geben konnte. Das Vereinsmagazin sowie der kostenlose Usenet-Zugang für die Mitglieder werden aber *[wieso "aber"? / rai]* weiter aufrechterhalten.

Forth Gesellschaft e.V.
Postfach 1110
85701 Unterschleißheim
Tel. 089/3173784
secretary@admin.Forth-eV.de
direktorium@Forth-eV.de

Be up to date!

Kann ein Leserbrief schon vor Erscheinen des zugehörigen Artikels bei der Redaktion eingehen? Hier ist der Beweis:

zu 'Lieber Forthfreund' in VD 2/96, Seite 7

.. Noch eine Anregung für VD und den Direktoriumsbrief an Newcomer: Eine Auflistung/Anzeige des Forth-Software-Vertriebs wäre schön und nützlich für Leute, die noch nicht am Internet hängen (wie ich z.B.)

Mfg Gerhard Raisig (rai), Erster Last-Minute-Korrekturleser des Forth-Magazins

.... haben Sie ein Faxgerät und Schulkenntnisse der deutschen Sprache? Dann werden auch Sie Last-Minute-Korrekturleser bei Ihrem Forth-Magazin

Wanted

Das Forth-Magazin 'Vierte Dimension' bittet um Hinweise!

Aufgrund einer technischen Panne sind eine Reihe von Kurzmeldungen in den Untiefen des Redaktionscomputers verschwunden. Dies betrifft jene Kurzmeldungen und Leserbriefe, die zwischen dem 22.4. und dem 10.5.1996 einsortiert wurden. Es dürfte sich fast ausschließlich um zwischen dem 15.4. und 1.5.1996 eingegangene Mails handeln.

Falls Sie eine wichtige Information vermissen, eine neue Information aufschnappen oder wohl gar eine selbst geschriebene Nachricht nicht wiedergefunden haben, wenden Sie sich bitte an die Redaktion.

Vielen Dank für Ihr Verständnis,

- Claus Vogt, Editor -

Bücherecke

Bücher zu Forth in München wenig gefragt

Arndt Klingelnberg hat auf der diesjährigen Forth-Tagung angeregt, man solle sich doch in den größeren Städten bemühen, Forth-Literaturquellen aufzuzeigen. Hier sind meine Bemühungen aus München:

Buchhandlung Lachner, Theresienstraße:

- Leo Brodie, "In Forth denken", Hanser-Verlag 1986. DM 54,-
- Leo Brodie, "Programmieren in Forth", Hanser-Verlag. DM 52,-
- Konrad Koller, "Forth und Forth-Prozessoren", Expert-Verlag 1993. DM 34,- .

In fünf weiteren größeren Buchhandlungen, auch in den beiden auf Computerliteratur spezialisierten, war nichts über Forth zu finden. In der Buchhandlung Kanzler erhielt ich die Auskunft, Forth sei schon seit längerem nicht mehr gefragt. In der Liste der lieferbaren Forth-Bücher seien aber 8 Eintragungen zu finden und die könne man mir selbstverständlich bestellen.

In dem Computer-Discountgeschäft DosCom, in der Schillerstraße, gibt es die

- CD "Dr. Dobb's Journal, Source Code & Index" 1982-1995. DM 59,-
- Dr. Dobb's Journal enthielt in früheren Jahren sehr viele Artikel über Forth. Auf der CD scheinen auch die dazugehörigen Listings vollständig enthalten zu sein.

In der Bibliothek der Technischen Universität München, die man natürlich auch von jeder anderen deutschen Hochschule per Fernleihe erreichen kann, sind folgende Forth-Bücher:

- Dr. Dobb's Toolbook of Forth /1
- Dr. Dobb's Toolbook of Forth /2
- Aumiller, Rainer: ATARI ST 32FORTH-Compiler (1988)
- Aumiller, Rainer: Programmieren mit FORTH ATARI ST (1987)
- Beilstein, Hans-Walter: Wie man in FORTH programmiert (1987)
- Brockmüller, Gerd H.: Beschleunigung von Mikroprozessoren durch einen ausgelagerten Forth-Koprozessor für Anwendungen der Fertigungstechnik (1994). Doktorarbeit TH Zürich.
- Brodie, Leo: In FORTH denken (1986)
- Brodie, Leo: Programmieren in FORTH (1984)
- Brodie, Leo: Starting Forth (1981)
- Chirlian, Paul M.: Der Einstieg in FORTH (1985)
- DeGrandis-Harrison, Richard: Forth on the BBC Microcomputer
- Feierbach, Gary: Forth Tools and Applications (1985)
- Floegel, Ekkehard: FORTH (1984)
- Floegel, Ekkehard: Forth on the ATARI (1983)
- Geere, Ron: Forth, the next step (1986)
- Geisse, Hellwig: FORTH als Brücke zwischen Roboterhardware und Expertensystem (1987). Doktorarbeit TH Darmstadt.
- Hogan, Thom: Discover Forth (1982)
- Kail, Paul: Forth (1989)
- Katzan, Harry: Invitation to Forth (1981)
- Knecht, Ken: Einführung in FORTH (1984)
- Koller, Konrad: FORTH und FORTH-Prozessoren (1993)
- MacCabe, C.K.: Programmieren in FORTH (1988)
- Mader, Manfred: Einführung in FORTH 83 (1989)
- Matthews, John: FORTH (1989)
- Oakey, Steve: Forth for micros
- Pountain, Dick: Object-oriented Forth (1987)
- Reymann, Joseph: FORTH (1985)
- Roberts, Sam D: Forth Applications (1985)
- Salman, W.P: FORTH (1984)
- Vack, Gert-Ulrich: Programmieren mit FORTH (1990)
- Winfield, Alan F.: The Complete Forth (1983)
- Woehr, Jack: Forth: The New Model (1992)
- Zech, Ronald: Forth83 (1987)
- Zech, Ronald: Die Programmiersprache FORTH (1984)
- Vierte Dimension: ab 1988
- The Newsletter of the Association for Computing Machinery's Special Interest Group on Forth (ACM-SIGFORTH) Heft 1 (1989). --- Da scheint es nur dieses ein und einzige Heft zu geben ???

In meinem eigenen Besitz befinden sich noch folgende weitere Bücher:

Goppold, Andreas und Roger Bouteiller: Forth, ein Programmiersystem ohne Grenzen. Edition Aragon 1985

Monadjemi: Das Trainingsbuch zu FORTH. Ein DATA-Becker-Buch 1984.

Petremann, Marc et al.: Forth 83-Standard, Manuel de référence (1987)

Petremann, Marc: Turbo-Forth, Guide de référence. Editions REM CORP 1990.

Petremann, Marc: Turbo-Forth, Manuel d'apprentissage. Editions REM CORP 1990.

Tracy, Martin, and Anita Anderson: Mastering FORTH, New York 1989.

Ich würde es gern unternehmen, eine möglichst vollständige Bibliographie aller forthbezogenen Literaturerscheinungen zusammenzustellen und in der VD zu veröffentlichen, und ich sehe Literaturhinweisen von jedem und jeder gern entgegen. Vielleicht könnte man auch Zeitschriftenartikel-Hinweise (FORTHWRITE, Vijgeblad, VD, FD) sammeln und in einer stichwortabrufbaren Datenbank zusammenfassen (?) So, wie es seit neuestem ganz allgemein in den Katalogsälen der Bibliotheken geschieht. Das wäre doch etwas für die nächste Weihnachts-CD (?)

Fred Behringer, München

Anstatt einer Buchbesprechung:

Ravi Sethi:

Programming Languages
Concepts and Constructs

Addison Wesley, 1996
478 Seiten (englisch)

Und wieder ist ein Buch über Programmiersprachen erschienen, in welchem Forth nicht erwähnt wird, jedenfalls weder im Inhalts- noch im Stichwortverzeichnis. **Ist Forth (k)eine Sprache?** Es werden alle möglichen Programmierparadigmen besprochen, es wird aber nicht gesagt, daß Forth diese seit eh und je alle schon enthält. Es wird überhaupt nicht über Forth gesprochen. Ist das Versehen oder Absicht? Man könnte vermuten, der Autor erkenne Forth als Sprache gar nicht an. Ich darf jedoch aus dem Vorwort zitieren: "A language can be described by writing a definitional interpreter for it, so called because its purpose is to define the interpreted language, efficiency is not a concern." Also doch nur ein Versehen!

beh

Forth Online

Olaf Stoyke

os@cs.tu-berlin.de

Auf der Online-Speisekarte für die FORTH-Gemeinde stehen heute: Französisches, Amerikanisches, der Macintosh, Compiler, Magazine und Journale und als Nachspeise ein paar Ergänzungen und Anmerkungen zum ANSI-Standard für die Programmiersprache FORTH...

Als Einleitung zur diesmaligen Online-Seite präsentiere ich ein eklatantes Negativbeispiel einer Web-Seite, d.h. schlechte Gestaltung, schlechte Dokumenthierarchie, fehlende Angaben über Autoren, etc. Wer also sehen will, wie eine FORTH-Web-Seite auf gar keinen Fall aussehen sollte, werfe einen Blick auf die finnische Seite mit der Adresse http://www.sci.fi/~at/hc11forth.html. Der Inhalt dieser Seite? Ein Assembler-Listing eines fig-FORTH fuer den MC68HC11A1. Nicht mehr und nicht weniger.

Wer Französisch zu lesen in der Lage ist, findet auf der Seite http://ourworld.compuserve.com/homepages/mp7die Firma MP7, die hier das Turbo-FORTH-Paket vorstellt und auch eine FORTH-CD-ROM anbietet. Auf der Seite (und den von dort aus erreichbaren Seiten) werden Handbücher und Programmpakete (z.B. FASTGRAF, siehe VD 3/95 auf Seite 31) zu Turbo-FORTH beschrieben. Verantwortlich für diese Seite ist Marc Petremann (100647.3306@compuserve.com, siehe VD 1/96).

Eine alphabetische Liste von FORTH-Worten mit Kurzbeschreibungen ist auf http://sherman.pas.rochester.edu/Forth/forth-words.html zu finden. Diese Seite gehört zu "An Introduction to Forth" von J. Kevin McFadden, wo man landet, wenn man obige Adresse ohne "forth-words.html" benutzt. Das ganze scheint von einem Entwicklerteam eingerichtet zu sein (und hier ist ein Verweis auf oben genannte Qualitätskriterien angebracht: Die Seite enthält keinen Hinweis auf die Firma DSPSYS, die wohl irgendwie da mit drinhängt...) und beleuchtet einzelne Aspekte eines Forth-Systems; damit gehört dieses Angebot auf die Speisekarte eines FORTH-Novizen.

Auf ftp://ftp.uu.net/vendor/minerva/uathena.htm findet sich allerhand Interessantes zum Thema ANSI-Standard, insbesondere jedoch sieben Texte, die bestimmte Definitionen im Standard selbst erläutern - hier werden sie "Clarifications" genannt und sind von A0001 bis A0007 durchnumeriert. Am Ende der Seite findet sich eine kurze Liste der wohl bekanntesten Web-Seiten zum Thema FORTH (FORTH, Inc., Taygeta, SIMTEL...).

Etwas farblos und trist präsentiert sich die FIG-Gruppe Maryland, die sich auf Seite http://www.bcpl.lib.md.us/~nbuck/fig.html der netzreisenden Bevölkerung vorstellt. Auch dort zu finden, sind ältere Berichte und Einführungen zu einem FORTH-Betriebssystem zusammen mit einer Aufforderung, sich mit Kommentaren und Dokumenten zu beteiligen. Vielleicht sollte man mal nachfragen, wie weit dieses Projekt gekommen ist?

Das letzte Mal habe ich eine Seite aus der Schweiz vorgestellt, die die Artikel aus der Newsgroup comp.compilers ansprechend und vor allem flink beinahe lokal zur Verfügung stellt. Nun, um fair zu sein, werde ich heute das Original vorstellen, das vom Moderator jener Newsgroup eingerichtet wurde, von John R. Levine. Unter http://www.iecc.com/compilers/ findet sich das Archiv von 1986 bis 1996 und zwar einschließlich der aktuellen Diskussionsbeiträge, was eigentlich ja nur recht und billig ist, wo dieses Archiv doch schon an der Quelle sitzt... Einen kleinen Test konnte ich mir natürlich nicht verkneifen: Waehle die Seite mit der Suchmaske an, die es erlaubt, einzelne Jahrgänge aus der Suchmenge ein- und auszublenden, und lasse nach "FORTH" suchen, Resultat: 13 Treffer. Ein umfangreiches Software-Archiv und Kataloge freier Compiler und zugehoeriger Tools und die FAQ von comp.compilers runden das gelungene Programm ab.

Ein freies FORTH-System fuer den Macintosh mit "round the clock support via this page" [O-Ton] kann mensch unter http://chemlab.pc.maricopa.edu/pocket.html finden. Pocket Forth ist dort als kleines, auf figFORTH und Leo Brodie's Buch basierendes System beschrieben und ist im Quellcode mit umfangreicher Dokumentation erhältlich, d.h. aus dieser Seite kopierbar.

Zum Abschluß noch die Adressen einiger amerikanischer Zeitschriften, die momentan noch in traditioneller Weise auf Papier gedruckt vertrieben werden. Auf diesen Seiten findet sich meist eine Vorschau auf die kommenden Ausgaben, die Möglichkeit die dort diskutierten Software-Schnippsel herunterzuladen und mit den Leuten selbst in den Kontakt zu treten:

- www.byte.com (Stichwortsuche möglich, besonders interessant: auf der Seite http://www.byte.com/www/www.htm findet sich eine Inserentenliste der Februar-Ausgabe des Magazins BYTE mit rund 100 Firmen samt anklickbaren Web-Adressen. Nie war es so einfach, Werbematerial zu ordern... ;-)
- www.ddj.com (Die im Dr. Dobbs Journal dargestellten Algorithmen und Lösungen sind genauso abrufbar wie ausgewählte Artikel, der Themenindex weist zu "FORTH" rund 60 Einträge aus...)
- http://www2.psyber.com/~tcj/ (Web-Seite von "The Computer Journal", das seit 1983 über neue und alte Hard- und Software berichtet; die Seiten selbst sind weniger bunt und komfortabel als die der ersten beiden Magazine)

❑

DisCo-light

von Klaus Schleisiek, SEND GmbH
040-37 50 08-03; Fax: -93 ; Vorsetzen 42; D-20495 Hamburg

Zum Geburtstag hatte ich Pieris Berreiter, einem 16-jährigen in Los Gatos, das DISCO-light von Klaus-Peter Schleisiek geschenkt, in der Hoffnung, er könne sich selber einen Eindruck von den Möglichkeiten von 4th machen.

Stichworte: DISCO FPC Ausbildung

Schon nach dem Auspacken und der ersten Inbetriebnahme (telefonische Rücksprache mit KPS war erforderlich) dämmerte mir, daß das für Pieris eine echte Herausforderung war. Naja.

Im März habe ich ihn dann zu Hause besucht und, wie vermutet, das DISCO-light war noch nicht zum Leben erweckt. Zusammen haben wir uns dann vor seinen PC gesetzt, ich an der Tastatur, er mit Bleistift und Papier bewaffnet und den folgenden Code zusammengehackt, mit dem man das DISCO-light ganz gut für Spielereien unter Kontrolle hat. Nach einigem Suchen haben wir dann auch die low-level Ansteuerung DISPLAY im mitgelieferten Code und den fiesen Trick mit $F0 PEMIT als “Sesam-öffne-dich”

File: e:discotin.lst vom 22.4.1996

```
 1 \ tiny DISCO-light package                                    14/04/96
 2
 3 \ Written by Klaus Schleisiek and Pieris Berreitter
 4 \ March 1996, Los Gatos, USA
 5 \ using FPC by Tom Zimmer.
 6
 7 \ ------------------------------------------------------------------------
 8 \ low level disco light driver                                14/04/96
 9 decimal
10
11 Create discobuffer   8 allot
12 \ 1st byte is top row, 8th byte is bottom row
13 \ “1”-bit is left side, “80”-bit is right side
14
15 : display  ( -- )
16    $F0 pemit    discobuffer dup 7 +
17    DO  I c@ $80 xor pemit  -1 +LOOP ;
18
19 \ : dark  ( -- )    discobuffer 8 erase display ;   \ PICTURE: DARK later on
20
21 : 2**n  ( n-1 -- 2**n )
22    1 swap ?dup IF  0 DO  2*  LOOP  THEN ;
23
24 : >addr ( col -- addr )  $7 and discobuffer + ;
25
26 : set  ( col row -- )
27    2**n swap >addr >r r@ c@ or r> c! display ;
28
29 : clear ( col row -- )
30    2**n not  swap >addr >r r@ c@ and r> c! display ;
31
32 \ ------------------------------------------------------------------------
33 \ defining pictures                                             14/04/96
34
35 : twist ( n1 -- n2 )            \ reverse bit order of a byte
36    0  7 0 DO  over $80 and or
37               swap 2*  swap 2/
38          LOOP  swap $80 and or ;
39
40 : Picture: ( b0 .. b7 -- )
41    Create   7 0 DO  twist c,  LOOP
42   DOES> discobuffer 8 cmove display ;
43
44 : binary  ( -- )  2 base ! ;
45
46
47 \ delay between pictures to make a movie
48
49 Variable delay  500 delay !
50
51 : wait  ( -- )  delay @ ms ;
52
53 \ ------------------------------------------------------------------------
54 \ some basic pictures                                           14/04/96
55 binary
56
57 00000000
58 00000000
59 00000000
60 00000000
61 00000000
62 00000000
63 00000000
64 00000000 Picture: dark
65
66 10000000
67 10000000
68 10000000
69 10000000
70 10000000
71 10000000
72 10000000
73 10000000 Picture: leftcolumn
74
75 10000000
76 01000000
77 00100000
78 00010000
79 00001000
80 00000100
81 00000010
82 00000001 Picture: backslash
83
84 decimal
längste Zeile hat 76 Zeichen
```

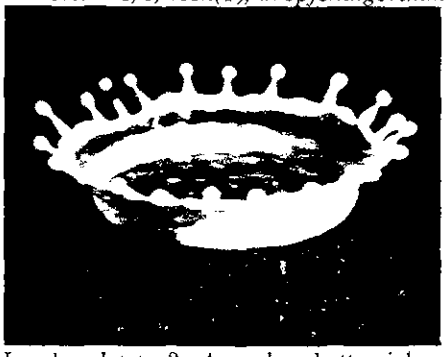
Steter Tropfen hölt den Stein

Lösung des Rätsels aus der VD 4/95

von Martin Bitter
Möllenkampweg 1a, 46499 Hamminkeln,

Der Tropfenalgorithmus zur Umrechnung von beliebig langen Ziffenketten in verschiedene Zahlensysteme.
Darstellung transzendenter Zahlen in "exotischen" Zahlensystemen.

Stichworte: PI, e, cosh(1), Tropfenalgorithmus, ZF, WIN32For



In der letzten? Ausgabe hatte ich versprochen, die Lösung des Rätsels und seine mathematische Begründung zu liefern, soweit mein Verständnis dazu reicht.

Das Wort Krypt2 aus dem Listing (VD 4/96) gibt nacheinander die ersten 32 Stellen der Zahl PI aus.

Der Wirkung beruht auf zwei Gegebenheiten. Zum einen läßt sich PI auf viele verschieden Weisen darstellen und ggfs. berechnen. Zum anderen erlaubt es der Tropfen-Algorithmus, eine dieser Darstellungen zur Berechnung von PI auszunutzen.

Vorspiel

In der Literaturwissenschaft gibt es den Begriff des implizieten Lesers, das ist der Leser, an den ein Autor denkt, wenn er schreibt. Die folgenden Erläuterungen sind recht lang geraten. Dies liegt zum einen an der (für mich) komplexen Materie, zum anderen an meiner Unsicherheit in der Ausgiebigkeit der Darstellungen.

All die jenigen, die den Text zu lang, zu redundant und zu trivial (Zech) finden, sowie diejenigen, die den Ausführungen wegen stilisctischer und sachlicher Mängel nicht folgen können bitte ich im Voraus um Verzeihung. Diejenigen oder demjenigen, die/der einwenig Spaß an der Lektüre hat, sei er herzlich gegönnt, ich jedenfalls habe mir mit Vergnügen und einigen guten Tropfen die Nächte um die "Ohren geschlagen".

Frage an alle: Ist in der VD Platz für Algorithmen, die ja nicht unbedingt forthspezifisch sind?

Darstellung von PI

PI läßt sich neben anderen durch folgende Gleichung beliebig genau darstellen.

$$PI=\left(2+\frac{1}{3}\left(2+\frac{2}{5}\left(2+\frac{3}{7}\left(2+\frac{4}{9}\left(2+\frac{n}{2n+1}\right)\ldots\right)\right)\right)\right)$$

(Diese Darstellungsform ist übernommen; s.h. Literatur, leider kann ich sie nicht herleiten.)

Stellenwertsysteme

Wir sind es gewohnt, im täglichen Umgang mit dem Dezimalsystem zu rechnen, das auf der Basis 10 aufbaut. Mit dem Aufkommen der Informatik wurden andere Systeme, die auf den Basen 2, 8 oder 16 beruhen, in der Praxis benötigt und verbreitet. (Mathematiker kannten diese schon länger (Leibniz))

Diese Systeme sind alle Stellenwertsystem, d.h. der Wert, den eine Ziffer repräsentiert ist abhängig von dem eigentlichen Wert der Ziffer und von der Stelle, den diese Ziffer in Bezug auf eine fest definierte Position innehat. In Europa ist dies meist das Komma, in anderen Ländern ein Punkt.

Für die Ziffernreihe 123,456 bedeutet dies, das sich ihr Wert im Dezimalsystem aus der Summe von sechs Zahlen zusammensetzt:

$1*100+2*10+3*1$ für die Vorkommastellen, zuzüglich

$4*\frac{1}{10}+5*\frac{1}{100}+6*\frac{1}{1000}$ für die Nachkommastellen.

Bei der Basis 16 repräsentiert die gleiche Ziffernfolge eine gänzlich andere Zahl, die sich ebenfalls aus der Summe von sechs Zahlen

$1*256+2*16+3*1$ für die Vorkommastellen, zuzüglich

$4*\frac{1}{16}+5*\frac{1}{256}+6*\frac{1}{4096}$

zusammensetzt.

In der Dartellung des Stellenwertes durch Exponenten, wird die Abhängigkeit des Wertes der einzelnen Summanden von der jeweiligen Stellung noch deutlicher: $1=10^0$ $10=10^1$ $100=10^2$ bzw. $1=16^0$ $16=16^1$ $256=16^2$ $4096=16^3$. Für das weitere ist wesentlich, das unabhängig von der verwendeten Basis der Wert der ersten Vorkommastelle immer EINS ist, d.h. die Ziffer der ersten Stelle wird mit eins multipliziert.

Da für die Darstellung von PI (fast) nur Nachkommastellen relevant sind, beziehe ich mich im folgenden nur auf Nachkommastellen.

Die dezimale Ziffernfolge ,456 bedeutet also:

$$4*\frac{1}{10}+5*\frac{1}{100}+6*\frac{1}{1000}$$

oder in anderer Schreibweise:

$$\frac{1}{10}\left(4+\frac{1}{10}\left(5+\frac{1}{10}\left(6\right)\right)\right)$$

Die hexadezimale Ziffernfolge ,456 steht für

$$4*\frac{1}{16}+5*\frac{1}{256}+6\frac{1}{4096}$$

oder

$$\frac{1}{16}\left(4+\frac{1}{16}\left(5+\frac{1}{16}\left(6\right)\right)\right).$$

Umwandlung in andere Systeme

Umrechnungsmöglichkeiten von einem Zahlensystem ins andere sind allgemein bekannt.

Am Beispiel des hexadezimalen Ziffernstranges ,5B6 soll nun (hoffentlich) die Vorgehensweise des Tropfenalgorithmus verdeutlicht werden.

Diese Ziffernfolge steht für die Summe: 5*1/16+11*1/256+6*1/4096 oder 1/16(5+1/16(11+1/16(6))).

Der Tropfenalgorithmus rechnet nun nicht diese Summe aus und gibt sie dezimal wieder - sondern er liefert der Reihe nach isoliert die Ziffern des entsprechenden dezimalen Ziffernstranges.

Zur Erinnerung: unabhängig von der verwendeten Basis hat die erste Vorkommastelle immer den Wert eins - die dort stehende Ziffer muß mit eins multipliziert werden, ihr Wert ändert sich dadurch nicht.

Das Ganze kommt einer Reihe von Verschiebungen der Ziffernfolge nach links gleich, wobei mit jeder Verschiebung die jeweils gültige Einerstelle erscheint.

... du hast das Schieben raus

Eine Verschiebung nach links bedeutet nichts anderes, als die ganze Zahl mit ihrer Basis zu multiplizieren. Da zur Basis 10 umgewandelt werden soll, muß der Ziffernstrang - Ziffer für Ziffer - mit der Zielbasis 10 multipliziert werden.

Beispielrechnung:

1/16(50+1/16(110+1/16(60)))

An der dritten Nachkommastelle ist aus dem Ausdruck 1/16(6) nun 1/16(60) geworden. Ausgerechnet hieße dies 60:16=3 Rest 12.

Dieses (Teil-)Ergebniss ist so zu interpretieren: 6/4096tel mit 10 multipliziert ergeben 12/4096tel plus 3/256tel.

Spätestens jetzt sollten Sie zu Bleistift und Papier greifen, eine Stellenwerttabelle zeichnen und die Rechnungen Schritt für Schritt nachvollziehen.

Die 12/4096tel bleiben an ihrer Stelle stehen, die 3/256tel werden zur davorliegenden Ziffer addiert: 110+3.

Die neue Ziffernfolge heißt:

1/16(50+1/16(113+1/16(12)))

Ein gleiches für die zweite und erste Nachkommastelle:

113:16=7 Rest 1 -->

1/16(57+1/16(1+1/16(12)))

57:16=3 Rest 9 -->

3+1/16(9+1/16(1+1/16(12))) (I)

Die erste Ziffer im Dezimalsystem lautet also 3.

Nun das Selbe in grün:

1/16(90+1/16(10+1/16(120)))

mit der Quellbasis 16 verrechnen -->

5+1/16(11+1/16(1+1/16(8))) (II)

Orientiert man sich an der üblichen Schreibweise in Stellenwertsystemen, so werden die letzten beiden Ziffernfolgen (I und II) folgendermaßen notiert: **3** 9 1 12

5 11 1 8 .

Wenn sie die einzelnen Rechenschritte wiederholen so ergeben sich der Reihe nach folgende Ziffernfolgen:

1	1/16	1/256	1/4096
6	14	15	0
9	5	6	0
3	5	12	0
3	9	8	0
5	15	0	0
9	6	0	0
3	12	0	0
7	8	0	0
5	0	0	0

Die fett gedruckten Ziffern bilden die dezimale Nachkommaziffernfolge der hexadezimalen Zahl, das heißt, 0,4B6(hex) = 0,35693359375!

Mit Hilfe des Tropfenalgorithmus können (fast) beliebig lange Ziffernfolgen eines Zahlensystems in ein anderes umgerechnet werden.

Frankensteins Nichte oder zusammengesetzte Basen

Das Spiel mit unterschiedlichen Basen kann noch weiter getrieben werden. Aus der Darstellung eines Stellenwertsystems in Exponentialform:

(b=Basis) $Zahl=aB^0+bB^1+cB^2+...xB^n$ läßt sich ein Bildungsgesetz erkennen. Bei den "gewöhnlichen" Systemen (Basis 2, 8,10,16 ...) kann es umgangssprachlich wie folgt formuliert werden: Die Ziffer an einer Position ist mit der potenzierten Basis, dieser Position zu multiplizieren. Dabei entspricht der Exponent der Laufzahl der Position.

Bei Nachkommaziffern wird durch die potenzierte Basis dividiert.

Das Bildungsgesetz des wiederholten Multiplizierens bzw. Dividierens wird in der folgenden Schreibweise für Nachkommastellen deutlich, besonders gut ist zu sehen, daß für jede Position (Stelle) die gleiche Basis potenziert wird.

$$\frac{1}{10}\left(a+\frac{1}{10}\left(b+\frac{1}{10}\left(c+\frac{1}{10}(d)\right)\right)\right)$$

Nun ist es durchaus möglich, bei entsprechender Vereinbarung, ein anderes Bildungsgesetz für die Werte der einzelnen Positionen festzulegen. Machmal ist das sogar sinnvoll!

Änderte man abhängig von der Position einer Ziffer die Basis, sähe das z.B so aus: (Nachkommastellen)

$$\left(a+\frac{1}{3}\left(b+\frac{2}{5}\left(c+\frac{3}{7}\left(d+\frac{4}{9}\left(x+\frac{n}{2n+1}\right)...\right)...\right)\right)\right)$$

Für die Anwendung des Tropfenalgorithmus auf eine solche Zahlendarstellung ergeben sich kaum Probleme.

Rechenschema zur Ermittlung von PI

	1/10	1/3	2/5	3/7	4/9	5/11	6/13	7/15	8/17	9/19	10/21
0.	2	2	2	2	2	2	2	2	2	2	2
1.	20	20	20	20	20	20	20	20	20	20	20
2b.	30	32	32	32	30	32	27	28	29	20	
2a.	**3** R0	10 R2	6 R2	4 R4	3 R3	2 R10	2 R1	1 R13	1 R12	1 R1	0 R20

3.

	1/10	1/3	2/5	3/7	4/9	5/11	6/13	7/15	8/17	9/21
0.	0	2	2	4	3	10	1	13	12	1
1.	0	20	20	40	30	100	10	130	120	10
2b.	13	40	53	80	95	148	104	186	120	
2a.	**1** R3	13 R1	10 R3	11 R3	10 R5	13 R5	8 R0	12 R6	7 R1	0 R10

3.

	1/10	1/3	2/5	3/7	4/9	5/11	6/13	7/15	8/17
0.	3	1	3	3	5	13	0	6	1
1.	30	10	30	30	50	130	0	60	10
2b.	41	34	63	78	110	142	28	60	
2a.	4 R1	11 R1	12 R3	11 R1	12 R2	12 R10	2 R2	4 R0	0 R17

3.

	1/10	1/3	2/5	3/7	4/9
0.	1	1	3	1	2
1.	10	10	30	10	20
2b.	18	24	36	18	
2a.	**1** R8	8 R0	7 R1	2 R4	2 R2

gebrochenes Herz - gebrochene Basis - geliebtes PI

Ein kurzer Blick auf den Anfang dieses Artikels zeigt, daß die dort zitierte Darstellung der Zahl PI nichts anderes ist, als eine Zahl unendlicher Periode mit einer gebrochenen Basis nach dem gerade gezeigten Bildungsgesetz.

$$PI = \left(2+\frac{1}{3}\left(2+\frac{2}{5}\left(2+\frac{3}{7}\left(2+\frac{4}{9}\left(2+\frac{5}{11}\left(2+\frac{6}{13}\left(2\right)\right)\right)\right)\right)\right)\right)$$

Insbesondere ist es nun möglich, den Tropfenalgorithmus auf diese Zahl anzuwenden:

1. alle Ziffern mal 10 (Zielbasis, links Schieben)

2a rechts beginnen, durch den Nenner der Quellbasis modular dividieren

2b Quotient mit dem Zähler der Quellbasis multiplizieren und zur nächsten führenden Spalte addieren.

3. Die Reste aus der ersten Division als Ziffernwerte für den nächsten Durchgang verwenden.

Eine Beispielrechung für die ersten drei Nachkommastellen finden Sie unten.

Hier noch einmal explizit für die Spalte 8/17 der zweiten Rechnung:

Von rechts nach links jede Ziffer mit der Zielbasis multipliziieren. Für die Position (8/17) ist dies (12*10=120).

Danach sukzessive für jede einzelne Ziffer: Modular durch positionsabhängigen Nenner (17) der Quellbasis dividieren (120 mod 17=7 R1).

Quotient (7) mit dem positionsabhängigen Zähler (8) der Quellbasis multiplizieren (56) und zur nächsten führenden Ziffer (130) addieren (186).

Für die neue Position (7/15) die eingerückten Schritte wiederholen.

(Ganz links angekommen befinden wir uns Dezimalsystem, deshalb ist die positionsabhängige Basis dort 1/10.)

Mit jedem Durchgang erscheint dort in dezimaler Darstellung die jeweils folgende Ziffer von PI.

Aus Gründen der Darstellung habe ich die Rechnungen in den ersten drei Durchgängen für jede ursprüngliche Ziffer durchgeführt. Das ist jedoch nicht nötig, da für jede Ziffer von PI im Dezimalsystem ca. drei Ziffern mit gebrochener Basis benötigt werden. Das heißt, mit jedem Durchgang verwandeln sich die jeweils letzten drei Stellen in Rundungsschrott. Die zu beachten spart Rechenaufwand und Zeit!

Die gezeigten Rechengänge nachzuvollziehen und evtl. zu verstehen mag schwer fallen. Mir hat es sehr geholfen, sie mehrmals in bekannteren Zahlensystemen, die ja Spezialfälle eines allgemeinen Stellenwertsystems sind, durchzuführen.

Bspl.:

0,456

	*1/10	*1/10	*1/10
0.	4	5	6
1.	40	50	60
2b	45	56	
2c	4 R5	5 R6	6 R0
3.			

	*1/10	*1/10	*1/10
0.	5	6	0
1.	50	60	0
2b	56		
2c	5 R6	6 R0	0
3.			

	*1/10	*1/10	*1/10
0.	6	0	0
1.	60	0	0
2b			
2c	6 R0		

Praktischer Nutzen?

Neben der Befriedigung die es (mir) verschafft, solche Algorithmen und zugrundeliegenden Mechanismen zu verstehen hat der Tropfenalgorithmus vielleicht einen praktischen Nutzen:

Ich bin nun in der Lage einen beliebig langen Ziffernstrang in einerm beliebigen Zahlsystem in ein zweites beliebiges Zahlsystem umzurechnen, und dies in den meisten Fällen unter Verwendung einfachgenauer Integeroperationen.

Der Algorithmus ist auch auf Vorkommazahlen anwendbar (hier nicht erörtert).

Für Zahlenmystiker ist es hübsch? zu sehen, daß bei entsprechend ausgewählter gebrochener Basis PI regelmäßig aussieht. PI, eine Zahl die vielen Zufallsgeneratoren zugrunde liegt, weil es bisher noch nicht gelungen ist, in ihrer Ziffernfolge eine Regelmäßigkeit zu entdecken!

Darüberhinaus zeigen einige andere Zahlen und Konstanten ein gleiches Phänomen:

e = 2,11111.... bei der Basis $\frac{1}{n+1}$,

cosh(1) = 1,101010... Basis $\frac{1}{n+1}$,

PI = 2,22222222... Basis $\frac{n}{2n+1}$!

Zur Implementation

Der gezeigte Algorithmus wurde sowohl unter ZF-forth als auch unter WIN32for implementiert und getestet. Alles Nötige zum Verständnis sollte sich im obigen Text befinden. Drei Gegebenheiten müssen berücksichtigt werden:

1. Pro gültiger Ziffer im Dezimalsystem werden 10/3 + 1 Stellen im gebrochenen Zahlensystem benötigt.
2. Je nach der Zahlenbreite in Quell- und Zielbasis kann es vorkommen, daß Überläufe auftreten.
   Zum Beispiel liefert der algorithmus bei der Zahl PI an der 31. Stelle eine 4, an der 32. Stelle eine 10. Dies bedeutet, das der Darstellungsbereich des Dezimalsystems übergelaufen ist. Aus den 10 Einern müssen 1 Zehner und 0 Einer gebildet werden. Folge: an Stelle 31 erscheint eine 5, an Stelle 32 eine 0.
   Bei einer Reihe von Neunen, müssen diese und die davorliegende Ziffer korrigiert werden(wird abgefangen).
3. Durch das wiederholte Multiplizieren der einzelnen Ziffern können Zahlen entstehen, die den Darstellungsbereich des Forthsystems überschreiten (ZF 16bit, WIN32for 32bit). Weiterrechnen führt ebenfalls zu ungültigen Ergebenissen (wird mit Fehlermeldung abgefangen).
   Eine Lösung ist möglich, indem ab der Stelle der Bereichsüberschreitung die einzelnen Ziffern invers behandelt werden (dividieren statt multiplizieren, von links nach rechs usw., Array wieder nach rechts verlängern).
   In ZF-Forth kann PI bis auf 535 Stellen berechnet werden. die letzten zwei Ziffern sind ungenau.
   In Win32For konnte ich die Grenzen nicht ermitteln, mir lag ein Referrenz-PI nicht in erschöpfender Länge vor (midestens 6840 Stellen).

Weitere Einzelheiten sind (hoffentlich) aus den Kommentaren im Quelltext ersichtlich.

Ein weiters ZF-File zum Spielen mit dem Tropfenalgorithmus liegt bei der VD vor (wie sieht PI in hexadezimal aus? wie in binär? usw.) Dort wird für Vorkommazahlen eine Lösung für das Problem Nr. 3 gezeigt.

❑

Eine SPS-Simulation in PC-VolksFORTH

von Gerhard Raisig
Oppauer Str.6; D-68305 Mannheim
0621-7 88 84 25; 74 76 12 fax

Einige Artikel in VD 3&4(1995) (2,3,4), den FORTH-Dimensions (5) und ein antiquarischer Buchfund (1) regten mich als Hobby-Forther an, aus dem PC mittels FORTH als Metasprache den Prototyp einer speicherprogrammierbaren Steuerung (SPS) zu machen. Steuerungen dienen der Automatisierung von Maschinen und Anlagen. Bewegungs- und Arbeitsabläufe werden durch Signalgaben ausgelöst und beeinflußt. Abfolge und Verknüpfung dieser Signale sind in einem Programm festgelegt, das bei 'verbindungsprogrammierten Steuerungen' als feste Verdrahtung der Bauelemente vorliegt, während es bei einer SPS in elektronischen Speichern abgelegt ist und somit leicht geändert werden kann. Bei der FORTH-SPS sollten die LEDs auf dem Monitor blinken und je nach SPS-Programm sollte sich der PC in die abstrakte Steuerung einer Waschmaschine, eines Getränkeautomaten oder einer Parkhausanlage verwandeln. Das SPS-Steuerwerk sollte ohne Hardware-Ergänzungen emuliert werden, für Ein- und Ausgabe von Zustandsvariablen sollten nur Tastatur und Monitor eingesetzt werden. Die SPS-Sprache sollte unter weitgehender Beibehaltung von Syntax und Semantik in FORTH abgebildet werden. Mit der hier vorgestellten FORTH-SPS ist es möglich, Steuerungen mit bis zu 32 Zustandsvariablen in SPS-Sprache zu entwerfen und zu testen.

Stichworte: SPS volksForth Steuerung

Aufbau und Sprache einer SPS

Vorbild dieser FORTH-SPS ist die MELSEC FX von Mitsubishi. Die Steuerung arbeitet seriell-zyklisch: Die Programm-Anweisungen für die Verknüpfung der binären Gebersignale (Schalter, Taster, Sensoren) zu den Gerätesignalen (Motoren, Lampen, Ventile etc.) werden nacheinander ohne Sprünge oder Schleifen abgearbeitet bis zu einer END-Anweisung; von dort aus wird zum Anfang der Anweisungssequenz zurückgekehrt. Der Ablauf wiederholt sich, bis die SPS angehalten wird. Das SPS-Programm besteht aus einer Folge von Anweisungsfolgen, die mit einer Start-Anweisungsfolge beginnt und von einer Ende-Anweisung abgeschlossen wird. Die Anweisungsfolge besteht aus:

```
Anweisung ... Anweisung ... Anweisung ...
```

Die SPS-Anweisung hat den Aufbau:

```
Adresse Operator Operand ,
```

bzw. für Zähler und Zeitstufen:

```
Adresse Operator Operand
    Konstante .
```

Die symbolische Adresse bezieht sich auf den Programmspeicher, in dem die Anweisungen nacheinander abgelegt sind. Die Start-Anweisungsfolge beginnt daher mit

```
0 Operator Operand .
```

Die Ende-Anweisung lautet:

```
Adresse END .
```

Die Operanden identifizieren

1 Speicherstellen eines Zwischenspeichers, in den vor Zyklusbeginn die anstehenden binären Gebersignale eingelesen werden;

2 die SPS-Funktionselemente Zähler, Zeitstufen, Merker und Konstanten;

3 Speicherstellen des Resultatspeichers, in den die Ergebnisse der Anweisungsfolgen übertragen werden.

Dies ist der Ausgang der SPS zur Geräteseite.

Semantisch betrachtet ist in einer Anweisungsfolge die schaltalgebraische Verknüpfung binärer Eingangszustände zu einem Ausgangszustand festgelegt. Die Verknüpfung der über die Operanden ausgewählten Zustände erfolgt über ein Ver-knüpfungsregister. Es stehen folgende Operatoren zur Verfügung:

```
1) einstellig:Zuweisung   OpZustand ->Verkn            : LD  Operand
                  "       OpZustand  inv. ->Verkn      : LDI Operand
                  "       Verkn ->OpZustand            : OUT Operand
                          Setzen speichernder Ausgang  : SET Operand
                          Rücksetzen   "               : RST Operand
2) zweistellig:Verknüpfung OpZustand ,Verkn            : AND Operand
                  "              "     ,  "            : OR  Operand
                  "    OpZustand inv.  ,Verkn          : ANI Operand
                  "              "     ,  "            : ORI Operand.
```

Die Anweisungsfolge muß daher die Form haben:

```
    Zuweisung    Op->Verkn
Verknüpfung      Op,Verkn
      ....       ....
Verknüpfung      Op,Verkn
    Zuweisung    Verkn->Op
      ....       ....
    Zuweisung    Verkn->Op   .
```

Während des SPS-Zyklus' ist der Inhalt des Zwischen-speichers, aus dem die Zuweisungen erfolgen, konstant; neue Geberzustände werden erst im Folgezyklus erkannt. Die Programmlänge bestimmt die Zykluszeit und damit die Reaktionszeit einer SPS auf Zustandsänderungen.

Die SPS-Zuordnungsliste

Vor dem Schreiben von SPS-Anweisungsfolgen muß definiert sein, wie bei Aktionen auf der Geberseite die Eingangszustände lauten müssen, damit nach ihrer Verknüpfung in der Anweisungsfolge der Ausgangszustand die geforderte Aktion auf der Geräteseite bewirkt. Ein Hilfsmittel dazu ist die Zuordnungsliste.

Beispiel: Ein Motor darf nur laufen, wenn der Motorschalter S1 betätigt ist und der NOT-AUS-Taster S2 nicht betätigt ist.

Zuordnungsliste:

```
X00 S1   (Schließer)      Motor ein/aus
X01 S2   (Öffner)         NOT-AUS
Y01 Motor
```

S1 betätigt liefert Signal 1,
S2 nicht betätigt liefert Signal 1.

Die Anweisungsfolge lautet daher:

```
... LD    X00
... AND   X01
... OUT   Y00 .
```

Wäre dagegen S2 ein Schließer (was sicherheitstechnisch nicht zulässig ist!), müßte die Anweisungsfolge lauten:

```
... LD    X00
... ANI   X01
... OUT   Y00 .
```

Die Operanden werden durch Buchstabe+Ziffern charakterisiert. Für die MELSEC FX ist festgelegt:

```
Geber:  Eingänge.....X..      Merker.....M..
Geräte: Ausgänge.....Y..      Zähler.....C..
        Timer.....T..         Konstanten.....K..
```

Der Entwurf einer SPS-Parkhaussteuerung

Ein Parkhaus mit je einer Ein- und Ausfahrt, die mit Lichtschranken versehen sind, soll eine Einfahrts-Ampel erhalten. Die Anzahl noch freier Plätze soll angezeigt werden.
Die Elemente der Zuordnungsliste sind schnell definiert:

```
X01    S1 (S).....Lichtschranke Einfahrt
X02    S2 (S).....Lichtschranke Ausfahrt
CR2       .....Zähler freie Plätze K05 = 5

Y01    Melder.....(simuliert Ampel)
```

Um die SPS-Anweisungsliste schreiben zu können, müssen wir die Spezifikation des SPS-Zählers kennen:

❷ Die Zählrichtung des Auf/Ab-Zählers wird vom Zustand eines zugeordneten Richtungsmerkers MCXX gesteuert:

```
MCXX = 0 ->zähle aufwärts
MCXX = 1 ->zähle abwärts ;
```

❷ Der Zähler hat einen Zähleingang und einen Ausgang, der abhängig von Zählrichtung und Zählstand folgende Zustände annimmt:

Richtung	**Zählstand**	**Zustand**
aufwärts	cnt < K	0
	cnt = K	1
abwärts	cnt > 0	1
	cnt = 0	0 .

Ferner muß eine SPS-Grundregel beachtet werden: Im Programm darf einem Ausgang nur einmal ein Verknüpfungsergebnis zugewiesen werden, um Fehlverhalten zu vermeiden.

Die Anweisungsliste zur Zählersteuerung (Anzeige f r e i e Plätze) lautet daher:

```
LDI  X02   OUT  MCXX   (Ausfahrt not -> aufwärts)
LD   X01   OR   X02    OUT  CR2  K05   (Zählsignal).
```

Der von Zählstand und -Richtung abhängige Zählerzustand muß nun den Ampelzustand Y01 steuern: Bei freie Plätze = 0 -> Ampel rot, sonst grün. Dazu wird Y01 als speichernder Ausgang programmiert:

```
LD   X02    OR   CR2    SET  Y01  -  Ausfahrt oder noch fr. Plätze: grün
LDI  X02    ANI  CR2    RST  Y01  -  'Einfahrt auf letzten Platz': rot .
```

Vordergründig ist die Aufgabe nun erfüllt. Was geschieht aber, wenn - so unwahrscheinlich es sein mag - im gleichen SPS-Zyklus der Einfahrt- und der Ausfahrtgeber betätigt sind? In praxi bleibt die Zahl der freien Plätze gleich; das SPS-Programm zählt aber wegen LD X01 OR X02 OUT CR2 (Grundregel!) falsch auf. Um diesen Grenzfall zu erfassen, wird die bisher reine 'Verknüpfungssteuerung' mittels eines Merkers in eine 'Ablaufsteuerung' umgeformt:

```
Ein- u n d Ausfahrt: IF zähle nicht, ELSE zähle THEN
```

Die SPS-Sequenz für das Zählsignal lautet nunmehr :

```
LD  X01    AND  X02    OUT  M10
LD  X01    OR   X02    ANI  M10   OUT  CR2   K05  ;
```

In SPS-Nomenklatur sind Ablaufsteuerungen sog. Schrittketten, für deren graphische Darstellung es eine Norm gibt. In unserem einfachen Beispiel entscheidet der Zustand des Merkers M10 darüber, ob der Schritt 'Zählen' ausgeführt wird oder nicht. Schrittmerker können zu 'Weiterschaltbedingungen' verknüpft werden, so daß Sprung- und Schleifenstrukturen programmiert werden können.

Die SPS-Simulation in FORTH: Aufbau und Arbeitsweise

Im Bild ist die linearzyklische Arbeitsweise einer SPS und ihre Simulation dargestellt.

Das Programmdesign ist bestimmt durch ein Feld, das als Elemente die Operanden-Namen mit einem Adress-Verweis in ein zweites Feld enthält. Dort sind alle Informationen, die zu einem Operanden gehören, in fester Abfolge angeordnet. Im einzelnen sind dies: Zustand als FORTH-Flag, Farbattribut für Zustandsanzeige als 'LED' auf dem Monitor, row/col-Werte für diese Anzeige (noch nicht benutzt), Zählerstände für Zähler und Timer und ein 'countflag', das zum Schalten der Zähler benötigt wird. Die gleichartige Anlage dieses Feldes für alle Operanden vereinfacht die Op-Auswahl und ermöglicht bequemes Umfunktionieren von Operanden durch Namensänderung. Auf diese Felder sind Zwischen- und Resultatspeicher der SPS abgebildet. Durch Ändern der Syntax auf postfix-Notation vereinfacht sich die Formulierung der FORTH-SPS-Operatoren wesentlich; es gilt also

```
: Anweisung := Operand Operator
```

```
--SPS--      -Simulation auf PC in FORTH-


 GEBER     - Tastatur: 'X01 on cr'
  I
ZWISCHENSPEICHER  - Feld 'Zustandsvar':Element ni,
  I       Teilelem. 'xflag' = Operand
  I
PROGR-> OPERANDENAUSWAHL < - Element X01, M10, Y01 etc.
 OPERATION   ^ - 'LD' 'OUT' als FORTH-Worte
  I    ^
RESULTATSPEICHER > - Variable 'Verknüpfung'
  I
AUSGANGSSPEICHER  - Feld 'Zustandsvar':Element yi,
  I      Teilelem. 'xflag'
 GERÄTE     - LED-Simulation auf Monitor
```

Die linear-zyklische Arbeitsweise einer SPS und ihre Simulation in Forth

'Adresse' entfällt, die Anweisungsfolgen werden in einer oder mehreren Colon-Definitionen zusammengefaßt, deren Aufruf den seriellen Ablauf bestimmt. Semantisch mußte die OUT - Anweisung aufgefächert werden. Da die FORTH-SPS nur mit Software-Zählern und -Zeitstufen arbeitet, ist ein Unterschied zu machen zwischen der Zuweisung eines Verknüpfungsergebnisses an einen Operanden vom Typ Zustandsvariable (X.., Y.., M..) und derjenigen an eine Funktion vom Typ Zähler oder Timer durch das gleichlautende Wort OUT. In FORTH-SPS gibt es daher die Anweisungen X..(Y..,M..) OUT , C.. OUT/C bzw. OUT/C+ und T.. OUT/T . OUT/C bzw. OUT/T enthalten den linearen Teil der Zähler-/Timerfunktion, der durch die wiederholte Abarbeitung im SPS-Zyklus zur Schleife ergänzt wird.

Der 'Zyklus-Motor' der FORTH-SPS ist eine BEGIN/UNTIL-Schleife im Wort 'run':

```
: run geber? BEGIN platzh xshow? key? UNTIL ; ,
```

Die SPS-Anweisungsfolge wird mittels Defer/Is in 'platzh' eingehängt. In 'run' erfolgt auch die Monitor-Ausgabe der Operanden-Zustände, und eine Stoppuhr-Funktion kann zugeschaltet werden. Der 'Motor' wiederum ist Teil einer äußeren BEGIN UNTIL-Schleife, in der der Rest der Maschinerie untergebracht ist: Start/Stop, Programm-Wahl, Geber-Eingabe über Tastatur, Monitor-Ausgabe der Menues und Zuordnungslisten:

```
: go   initcntr  allxfloff  progr
       BEGIN .namerow xshow .eingabe query interpret
           run ( das SPS-Programm) .geber? stop?
       UNTIL .sps_aus ;  .
```

Operandenauswahl: ‘Feld:’**, ‘**(elem’ **und** ‘(part’

Das SPS-Steuerwerk, dem die Operandenauswahl obliegt, ‘steckt’ in den Create/does> - Worten Feld: , (elem und (part. Dazu gehören die Pointer *zustandsvar, *element und das Wort frst. Mit Feld: werden immer 32 * 13 Bytes im Speicher mit 00 vorbelegt:

```
: Feld:
    Create  32 0 DO  13 0 DO    0 c, LOOP LOOP
    Does> *zustandsvar ! ;
```

Dies ist der Platz für die Operanden-Informationen. Das Wort (elem baut die Operanden-Liste auf, die aus den OpNamen und absoluten Start-Adressen der zugehörigen Elemente im Feld besteht. Außerdem bringt (elem gleich die nfa ins Feld:

```
: (elem ( n ad -- n' ad')
    dup here 4 + swap ! 13 + swap
    Create dup *zustandsvar @ + , 13 + swap
    Does> @ *element ! ;
```

Lediglich für diese Arbeiten wird *zustandsvar benötigt, im laufenden Betrieb wird nur noch der Pointer *element auf das Op-Element gesetzt. Die mittels (part definierten Offsets zu den Op-Informationen liefern zuaddiert dann die benötigte Teilelement-Adresse für den Operator-Zugriff:

```
: (part
    Create dup c, 1+
    Does> c@ *element @ + ; ( -- teilelemadr )

    0 (part xflag 1+ (part nfa (part xcntr  3 +
    ... (part cflag drop
```

Das Wort frst setzt ‘implizit’ den Pointer *element auf das 1. **E**lement des ‘explizit’ über *zustandsvar eingeschalteten Feldes:

```
: frst *zustandsvar @ *element ! ;
```

Nur über diesem Feld arbeiten die Routinen für die Monitor-Ausgabe der Zustandsvariablen und Zählerstände: .namerow, .ledrow, xshow, .cnt , und zwar konstant für alle 32 Elemente, die jedoch nicht alle belegt sein müssen. In diesem ‘Hauptfeld’ sollten alle Operanden definiert sein, deren Anzeige für das SPS-Programm wesentlich ist, also Ein- und Ausgangszustände, aber auch evtl. Zähler und Merker. Alle übrigen Operanden, vor allem die SPS-internen, die zur Simulation benötigt werden, können dann in weitere Felder ohne Anzeige gelegt werden. Auch können so Op-Felder einzelnen SPS-Programmen zugeordnet werden. Über die Op-Namen muß natürlich Buch geführt werden, sonst erscheint ’... exists’ und die SPS-Operatoren greifen falsch zu. Eine SPS-Anweisung ‘X01 LD’ läuft also wie folgt ab: ‘X01’ setzt *element auf die Adresse des Elements X01 im Feld;

```
: LD xflag @ verkn ! ;
```

legt die Adresse von xflag X01 auf den Stack (= Operand) und bringt den Wert in das Verknüpfungs’register’ verkn .

Anzeige der Zustandsänderungen: ‘xshow?’

Das Wort xshow? in der SPS-Zyklusschleife ‘run’ steuert den Sprung aus dem Zyklus zur Monitor-Ausgabe. Sehr bald zeigte sich, daß aus Geschwindigkeits- und Richtigkeitsgründen nur eine ‘ereignisgesteuerte’ Anzeige der Zustandsänderungen in Frage kam: “Immer wenn, aber nur wenn!”. Xshow? arbeitet wie folgt: Am Ende jedes Zyklus’ werden alle 32 Op-Zustände (xflag) abgefragt. Bei jedem ‘true’ wird eine -1- stellenwertrichtig mit or in 2 * 16b auf dem Stack gebracht. Dann wird der Inhalt von change? mit den zwei Stackwerten verglichen. ‘False’ sorgt dann für Anzeige, change? wird angepaßt. Die Stack-Akrobatik in xshow? entsteht also aus 2 * 16 xflag-Abfragen, den 2* -Operationen, Bit-1-Speicherungen und dem Vergleich:

```
2Variable change?
: xshow? ( -- ) frst xflag
    2 0 DO 0 1 rot 16 0 DO dup @ ( xflag)
        IF -rot dup rot or swap 2*
        ELSE -rot 2*
        THEN rot 13 + LOOP
        swap drop
    LOOP
    drop 2dup change? 2@ d=

 IF 2drop    ELSE change? 2! xshow delay THEN ;
```

Das Prinzip der Inline-Zähler: ‘OUT/C’

Nun soll die Arbeitsweise des im Parkhaus-Beispiel verwendeten Auf/Abzählers gezeigt werden. Während die als SPS-’Timer’ getarnten Zähler bei jedem Zyklus-Takt hochzählen dürfen, bis die entsprechend große Zeitkonstante erreicht ist oder das Freigabesignal zurückgesetzt wird, darf ein SPS-Zähler nur auf einzelne Geber-Signale reagieren, und dies auch nur im 1.Takt der viele Takte anstehenden Zustandsänderung. Zum Erkennen dieser 0/1 -Taktflanke wird ein Hilfsflag eingeführt, mit dem das Eingangssignal xor-verknüpft wird. Erst wenn das Zählsignal zurückgesetzt worden ist, geht auch das cflag wieder auf ‘false’ und gibt den Weg frei für das nächste Zählsignal:

```
: OUT/C ( -- ) verkn @ cflag @ xor
    IF verkn @ dup cflag !
        IF  xcntr 2@ richtung 2dup
            xcntr 2! .cnt
        THEN
    THEN xcntr 2@ ende? xflag ! ;
```

Der Zähler arbeitet also in folgender Zustandstabelle:

Anfang Geber	OUT/C cflag	in xor	xcntr	Ende cflag	Bemerkung
0	0	0	-	0	kein Zählimp.
1	0	1	+/- 1	1	Zählflanke
1	1	0	-	1	Imp. dauert an
0	1	1	-	0	Imp. zu Ende .

Der Endzustand von cflag ist jeweils der Anfangszustand für den Folgezyklus. Die Funktion 'Richtungsmerker MCXX' hängt in jedem Zyklus in die Defer-Worte 'richtung' bzw. 'ende?' Worte zum Auf- oder Abzählen bzw. Abfragen auf den Zählerstand = K oder 0 ein.

Anmerkungen zu Stoppuhr und SPS-Zeit: '(1RCHTIMP',(TIMER'

Um eine Vorstellung von SPS-Zykluszeiten zu bekommen, die ja von der Programmlänge abhängen, wurde die Stoppuhr entworfen. Ihr Aufbau ist insofern interessant, als die SPS-Funktionen 'Richtimpuls' und 'Timer' die Parameter für IF -Entscheidungen auf FORTH-Ebene liefern, die den weiteren Ablauf auf SPS-Ebene steuern. Der Stack wird zur Drehscheibe. Die Funktion 'Richtimpuls' gibt nach dem Einschalten einer SPS einen einmaligen Impuls aus, der zum Initialisieren von Zählern und Merkern verwendet werden kann:

```
: (1RCHTIMP   HM4 LD HM5 OR HM5 OUT
              HM5 LDI HM4 OUT ;
```

Nach dem Durchlaufen dieser Sequenz im 1.Zyklus ist HM4 = 1, wird aber ab 2.Zyklus auf 0 gehalten, während HM5 auf 1 bleibt. Mit dem HM4-Takt wird nun die Startzeit der Stoppuhr ausgegeben und ein Zähler gestartet, der bis 10.000 zählt. Jetzt wird die Stoppzeit ausgegeben. Die Zeiten werden durch Auslesen der PC-Zeit mittels DOS-Interrupt $21,Funkt. $2C00 gewonnen ('time1.,time2.').

```
: (TIMER   (1RCHTIMP HM4  LD verkn @   ( -- f )
   IF                                 : nur 1.Zyklus
      HM4 LDI  HT3 OUT/T              : Zähler-Reset
      time1. BELL   THEN              : Startzeit
    HM5 LD HT3 ANI verkn @ ( -- f)
 IF                                   : ab 2.Zyklus
  K100 HT3 OUT/T  : Timer als Zähler
  HT3  LD verkn @ ( -- f)
  IF time2. Tdiff. BELL THEN
         : Timer abgelaufen
   THEN ;
```

Die Auswertung der Messergebnisse ist gar nicht so einfach. Die ausgegebene Zeit steht für den Durchlauf aller Befehle in der run - BEGIN UNTIL - Schleife und zwar auf allen Sprach- und Systemebenen. So steckt die Zeit für das Uhrprogramm selbst mit darin; eine Erhöhung der Blinkfrequenz eines Blinkgeberprogramms 'BLINK' erhöht die Anzahl der Monitor-Ausgaben, FORTH-Wortbildungen schleppen wechselnden Overhead mit sich. Heisenberg läßt grüßen. Aber trotzdem erhält man nützliche Informationen: So konnte durch Optimieren von xshow? mittels Stackakrobatik die Zykluszeit des BLINK-Programms halbiert werden. Richtwerte für Modul-oder Anweisungszeiten erhält man aus geeigneten Differenzmessungen z.B. .. BLINK BLINK .. zu .. BLINK .. ergibt die Zeit für BLINK. Auf meinem System vergehen 7,19 sec, bis Parkhaus- plus Stoppuhrprogramm 10.000mal durchlaufen sind; für sich selbst hat die Stoppuhr 4,67 sec gemessen. Der SPS-Lauf darf natürlich nicht durch Tastatur-Eingaben unterbrochen werden! Diese Geber-Simulationen zerstören zwar den Bezug der SPS- Zeit zur Realzeit; der so gestückelte SPS-Zeitlauf bleibt aber in sich logisch richtig, da ja die Inline-Timer Zähler sind, die alle zugleich unterbrochen werden bzw. miteinander weiterlaufen.

Abschließende Bewertung der Arbeit an der FORTH-SPS

Nachdem die Definition der wenigen SPS-Sprachelemente und ihrer rudimentären Syntax in FORTH abgeschlossen war und die SPS-Anweisungen sich bereits in einer Definition der SPS-Programmierumgebung breitmachten (siehe die Stoppuhr-Funktion), befiel mich ein Bedauern, das ich durch Fortsetzen eines Zitats von Symonds *(2)* beschreiben möchte: *"In dem Moment, wo die Sprache ideal an das Problem angepaßt ist, ist aus Forth eine überaus lesbare und damit wartbare Programmiersprache geworden ..."*, aber, füge ich hinzu, Forth ist auch verschwunden! Ich kann also jetzt in SPS meinen Kaffeeautomaten auf den Monitor bringen, aber wo bleibt der Stack-Thrill von xshow? !? Andererseits ist mir auch klar geworden, daß mit der Definition der SPS-Anweisungen noch nicht viel geleistet ist: Das Spielen mit dieser Demo-SPS setzt stillschweigend Forth-Kenntnisse voraus: Variablenfelder anlegen, Module definieren und einhängen etc. erfolgen nach wie vor in Forth. Es fehlt also noch die SPS-Programmier- und Run-Umgebung für 'Nur-SPSler'. Gelernt habe ich jedenfalls nicht nur viel über SPS, sondern auch über *"die metasprachlichen Fähigkeiten und Funktionen von Forth, (...) , die als elementare Bestandteile des Forth-Interpreters und -Compilers offenliegen" (2) .*

Literatur:

(1) K.H.Borelbach u.a.: Speicherprogrammierbare Steuerungen mit der MELSEC FX , Verlag Europa-Lehrmittel 1993

(2) M.Symonds: Forth als Metasprache, VD 3&4,p.14 1994

(3) M.Symonds: Vorwärts- und dann..., VD 3&4,p.21 1994

(4) R.Kern: Computer, die man nicht sieht, VD 3&4,p.31 1994

(5) Ken Merk: Forth In Control, Forth Dim. 2/XVII,p.18 '95.

❑

```
Seite  1             volksFORTH83 der FORTH-Gesellschaft eV      SPS7.SCR
```

0

```
 0 Simulation einer SPS in Volks4th PC V3.81:  Ver.1.7  rai 12mr96
 1  Dieses Programm verwandelt mittels FORTH als Metasprache und
 2  ohne Hardware-Ergaenzungen den PC in den Prototyp einer
 3  'speicherprogrammierbaren Steuerung -SPS-' (Lit. siehe SCR 53).
 4  Gliederung des Programms:
 5  SCR  2- 3       Feld der SPS-Zustandsvariablen
 6  SCR  4- 7       Bildschirmanzeige und Eingabe der SPS-Zustaende
 7  SCR  8          Die SPS-Maschine
 8  SCR  9-10       Der SPS-Befehlssatz
 9  SCR 11-14       SPS-Elemente: Zaehler, Timer, Zubehoer
10  SCR 15-18,22-25 SPS-Programm-Module
11  SCR 19-21       Stoppuhr zum Messen der Zykluszeit
12  SCR 43-52       Programm-Menue und Zuordnungslisten.
13           -- Die FORTH-SPS wird mit  go  gestartet. --
14  Autor:   Gerhard Raisig  Oppauer Str.6  68305 Mannheim
15           Tel. 0621 - 7 88 84 25    Fax 0621 - 74 76 12
```

1

```
 0 \ Simulation einer SPS: Loadscreen                  rai 20mär96
 1
 2 \  forget 2Constant
 3    onlyforth
 4  : 2Constant  Create , , Does> 2@  ;
 5  : 2Variable  Create 0 , 0 ,  ;
 6    19 load
 7    vocabulary sps  sps also definitions
 8
 9     2 18  thru    20 25  thru
10    43 52  thru
11 \ : runalone sps also pkhs also go ;
12 \   ' runalone Is 'restart
13 \    savesystem vfsps.com
14
15
```

2

```
 0 \ Simulation einer SPS: Feld fuer Zustandsvariablen  rai 28feb96
 1
 2   Variable  *zustandsvar    *zustandsvar off
 3
 4
 5  : Feld: Create 32 0 DO   13 0 DO  0 c,  LOOP  LOOP
 6        Does> *zustandsvar ! ;
 7
 8    Feld:  Zustandsvar                    \   Zustandsvar
 9
10    Variable   *element      *element off
11
12  : (elem  ( n ad -- n' ad') dup  here 4 +  swap ! 13 +  swap
13             Create  dup  *zustandsvar @ +  , 13 +  swap
14             Does>   @  *element ! ;
15  : frst   *zustandsvar @  *element  !  ;
```

28

```
Die linear-zyklische Arbeitsweise einer SPS        rai 06feb96

                    --SPS--                 -Simulation auf PC in Forth-
                ----------------            ----------------------------
                     GEBER            --    Tastatur: 'x03  on'
                       |
                ZWISCHENSPEICHER      --    'Zustandsvar': Elemente ni,
                       |                    Teilelement xflag (Operand)
PROGR-> OPERANDENAUSWAHL <-   --    Elemente X01, M02, Y00  etc.
          OPERATIONEN      |  --    LD,OUT etc. als FORTH-Worte
                       |   |
                RESULTATSPEICHER ->   --    Variable 'Verknuepfung'
                       |
                AUSGANGSSPEICHER      --    'Zustandsvar': Elemente yi,
                       |                    Teilelemente xflag
                   GERAETE            --    LED-Simulation auf Monitor
```

29

```
SPS-Zuordnungsliste und -Programmnotation          rai 19jan96
Beispiel: Nach Betaetigen des Gebers S1 soll der Melder leuch-
ten, auch wenn S1 nicht mehr betaetigt wird. Nach Betaetigen
des Gebers S0 soll der Melder nicht mehr leuchten.
Zuordnungsliste:
SPS-Element  Anlageteil
Eingang X00  Geber S0 (Oeffner)     S0 betaetigt -> X00 = 0  !
Eingang X01  Geber S1 (Schliesser)  S1 betaetigt -> X01 = 1
Ausgang Y00  Melder H1              Y00 = 1 -> Melder leuchtet
Arbeitstabelle: ( + = betaetigt )   SPS-Programm:
 S1  S0   H1                        0  LD   X01  Geber S1 (S)
 -   -    0                        1  OR   Y00  (Selbsthaltg)
 +   -    1                        2  AND  X00  Geber S0 (Oe)
 -   -    1                        3  OUT  Y00  Melder H1
 -   +    0                        4  END
 +   +    0  !: AUS hat Vorrang vor EIN (Sicherheitsbestg.!)
```

30

```
                                                       rai 10feb96
Pointer auf Feld 'Zustandsvar' mit den Elementen n0-n31
(Laenge je 13 Bytes)

||Zustandsvar||00|00|0|00|..|00|nfa||00|00|0|00|..|00|nfa||...
               n0                  n1                  n..

- Feld: legt das Feld 'Zustandsvar' an,  Zustandsvar initiali-
  siert bei Aufruf den Pointer *zustandsvar (siehe SCR#3)
- Pointer auf Element ni im Feld
- (elem speichert die nfa im Feld , erzeugt |n1,fadr|n2,fadr..
  = Anfangsadr von ni im Feld  und
  setzt bei Aufruf eines Elements *element auf dessen Feld-
  adresse  ||*element|ni|| .
- setzt *element aufs 1.Element des Feldes.
```

Der Rest des Listings zur SPS (insgesamt 9 Seiten mit 54 Screens) findet sich zusammen mit dem ausführbaren File und dem volksFORTH83 zum Experimentieren auf der Listingdiskette

Jahrestagung der Forthgesellschaft 1996

... aus der Sicht von Jörg Plewe

Haarzopfer Str. 32; D-45472 Mühlheim

Wem der inzwischen gleichmäßige Puls der Vierten Dimension vertraut ist, der wird es auch sicher im Gefühl haben, daß bei jedem vierten Schlag etwas Besonderes passiert: Die Jahrestagung der Forthgesellschaft e.V.

In diesem Jahr hat es die Schar der Teilnehmer am 19./20. April nach Höfchen bei Mittweida bei Chemnitz verschlagen; dort, wo die Oststraßen noch authentisch sind. Thomas Beierlein hatte uns dorthin eingeladen. Das Tagungshotel selbst erinnerte die ostdeutschen Teilnehmer stark an die Ferienlager ihrer Jugend und war mit einer eigenwilligen Idylle überzogen, die entsteht, wenn bei schönem Wetter zwischen vielen Gebäuden nur wenig Menschen zu treffen sind. Der Tagungsraum selbst war mit allem ausgestattet, was sich ein Vortragender nur wünschen kann, von der Tafel bis zum Beamer, und so konnte es dann auch relativ pünktlich losgehen. Als kleine Besonderheit habe ich einen Preis, die Forth-Weihnachts-CD, für die beste Performance der Tagung gestiftet. Heinz Schnitter, Jörg Staben und Ralf Neuthe stellten sich dafür als Juroren zur Verfügung.

PD, VD, Spiele ...

Friedrich Prinz, der 95er Drachenhüter, eröffnete den Reigen mit seinen Erfahrungen bei der Windowsprogrammierung mit Win32Forth, dem letzten PD-Schrei aus USA. Seine Ausführungen brachten ihn schließlich zu der Forderung nach einem VisualForth, das den Entwickler von der Kenntnis der Windowsprogrammierung befreien sollte und seine Konzentration auf das Inhaltliche ermöglichen kann. Diese Forderung fand breite Unterstützung. Jedoch wurde in einer lebhaften Diskussion klar, daß zum einen Windowsprogrammierung ohne Windowskenntnisse eine Utopie bleiben wird und daß zum anderen ein VisualForth mit den vorhandenen Entwicklungskapazitäten kaum zu machen sein wird.

Der inzwischen wohlbekannte und bewährte Editor unserer VD Claus Vogt zog anschließend ein Resümee seiner Arbeit im letzten Jahr und machte uns auch mit den Problemen seiner Arbeit vertraut. Daraus entwickelte sich eine lebhafte Diskussion darüber, ob z.B. orthographische Korrekturen an Artikeln vorgenommen werden dürfen oder nicht. Zu einem klaren Ergebnis kam man dabei nicht, so daß diese Frage weiterhin (sinnvollerweise) in der Verantwortung des Editors bleibt.

Der Gastgeber Thomas Beierlein weihte uns in seinem ersten Vortrag in seine ungewöhnliche Nutzung des kooperativen Forth-Schedulers ein. Sein System ermöglicht ein Scheduling über mehrere virtuelle Maschinen, so daß zum Beispiel sowohl sein Holon-Entwicklungsystem als auch sein auf dem gleichen PC befindliches Target im Multitaskingbetrieb arbeiten können. Und das auch dann noch, wenn die beiden virtuellen Maschinen unterschiedlich Threadingmechanismen (sprich: ein anderes NEXT) einsetzen. Möglich wird dies alles durch einen dezentralen Scheduler, bei dem das Deaktivieren und das Aktivieren von Tasks von unterschiedlichen Programmteilen, die auf der jeweils passenden Maschine residieren, durchgeführt werden.

Nach eine kurzen Pause durfte ich die Zuhörer kurz in die Welt des Entertainment entführen, indem ich von meinen Erfahrungen beim Versuch, ein Actionspiel in Forth zu entwickeln, berichtete. Faszinierend ist dabei die Tatsache, daß man bei einem solchem Unternehmen jede Ecke seines PC wirklich aus der Nähe kennenlernt. Ein interessanter Punkt war auch, daß der verwendete 3D-Renderer Real3D als Makrosprache Forth nutzt (allerdings ohne DOES>). Ein einfaches Videospiel kommt auf einer mannshohen Projektionsfläche übrigens gut rüber.

Den Abschlußvortrag des ersten Tages hielt Alexander Burger über sein System LIFO, ein Forthsystem mit Lisp-Eigenschaften oder umgekehrt. Alexander konnte an

einem offensichtlich weit fortgeschrittenen System in Forth-Syntax die typischen Vorteile von symbolischer Listenverarbeitung demonstrieren. LIFO ist kein akademisches Experiment sondern steht im Bereich von GUI-Entwicklung mitten in der Praxis.

Nach dem Abendessen ging es natürlich noch lange mit Diskussionen und Experimenten weiter, was ich als Augenzeuge aber nur bis etwa 3.00 Uhr belegen kann....

Microcontroller

... denn kurze Zeit später ging es mit den Samstagsvorträgen weiter, und zwar mit den eher Forth-typischen Themen der Microcontrollerentwicklung.

Wolf Wejgaard stellte uns sein System Holon vor, das bereits seit einiger Zeit auch prominenter Diskussionspunkt in den Netzen ist. Holon zeichnet sich besonders durch sein System zur Quelltextverwaltung aus, aus der die klassischen Sourcefiles oder Blöcke komplett verschwunden sind und durch eine hierarchische Datenbank ersetzt wurden. Die Source wird so in Module (wie Basis, Forth, Tasking, ...), Gruppen (wie Datenstack, Returnstack, ...) und Worte (wie DUP, ROT, SWAP...) eingeteilt. Mit dem Holon-Editor kann man durch diese Hierarchie navigieren und erhält so einen übersichtliches Gesamtbild der Anwendung. Zur Kompilation kann Holon die notwendigen Sourceteile extrahieren und den Code auf das Target herunterladen. Holon bleibt dabei interaktiv mit dem Target verbunden, und es ist sogar möglich, Veränderungen am laufenden System vorzunehmen, was Wolf mit Hilfe von Thomas Beierlein, einem überzeugten Holon-Anwender, und einem Tetris in Holon demonstrierte. Thomas hatte es nicht leicht gegen ein Spiel, bei dem der Programmierer zur Laufzeit die Regeln ändert!

Als Target stehen neben verschiedenen µC auch der PC selbst zur Verfügung, was besonders zum Experimentieren mit Holon einlädt. Zu diesem Zwecke ist auch Holon-lt kostenlos zu bekommen. Solcherart Neues führt natürlich zu Diskussionen, die sich nun auch nach der Tagung im Netz fortsetzen.

Ebenfalls mit datenbankgestützter Quelltextverwaltung hat sich Stefan Lange beschäftigt. Er setzt im Gegensatz zu Wolf für comFORTH auf kommerziell erhältliche, relationale Datenbanktechnologie, die unter Windows via ODBC angesprochen wird. Dabei gibt der Entwickler selbst die Einteilung seiner Source in Module vor, so daß er nicht auf das starre Holon-Schema festgelegt ist. Der zugehörige Browser, natürlich mit Windows-Controls, war leider zur Tagung noch nicht ganz fertig geworden.

OOF, Prüfung, Module ...

Birgit Steffenhagen brachte feine Kost für Spezialisten, indem sie über ihre Arbeiten zu objektorientierten Programmierwerkzeugen für echtzeitfähige embedded-control Anwendungen (eooS) referierte. Besonderes Augenmerk richtete sie dabei auf eine echtzeitfähige Speicherverwaltung mit Garbagecollector, wie sie für OO-Systeme notwendig ist. EooS ist streng modular aufgebaut, damit die Module Method-Object-Binder (MOB) und object-oriented MMU (ooMMU) als Koprozessoren auch in Hardware ausgeführt werden können. Implementiert ist das System aber vollständig in Software mit einem M68332-Target. Die weitere Entwicklung wird jetzt Klassenbibliotheken für die Meßwertverarbeitung basierend auf eooS bringen.

In einen ganz anderen und für Forthler gänzlich unbekannten Bereich verschlug uns der Vortrag von Ulrich Hoffmann, der sich mit den Möglichkeiten zur Erstellung zuverlässiger Programme beschäftigt. Er hat klargemacht, daß bei Programmen, die einmal vom TÜV abgenommen werden sollen, ein enormer Dokumenationsaufwand entsteht (lückenlose Dokumentation), wobei jedes Softwarestück exakt in seiner Funktionsweise zu beschreiben ist. Die Eigenschaft von Forthprogrammen, aus kurzen Worten zu bestehen, macht diese Forderung erfüllbar. Durch die Testbarkeit einzelner Forthworte ist es zudem möglich, direkt bei der Kompilation automatische Whitebox-Tests durchzuführen, d.h. das Wort mit bestimmten, möglicherweise kritischen Parametern auf korrekte Ergebnisse hin abzuklopfen.

Malte Köller brachte uns anschließend wieder in etwas bekanntere Gewässer, indem er seine Ideen für eine Modulladetechnik vorstellte. Vorher führte er in die grundlegenden Probleme dabei ein, wie Relokation oder das Auflösen offener Referenzen. Sowohl Quelltextmodule als auch binäre Overlays führen zu spezifischen Problemen. Als Ausweg bot Malte eine Mischung beider Techniken an, in der ein

Foto: kk & fep mit Drachen vor Palme (akg 4/4-5/96)

Der Swap-Drache wurde dieses Jahr Klaus Kohl für seinen aufopferungsvollen Einsatz in der Forth-Gemeinde verliehen. Am bekanntesten wurde er durch das langjährige erfolgreiche Vertreiben von Forth-Systemem und seine Verantwortung für die nächste Forth-CD.

tokenisierter Quelltext als Modul gespeichert wird. Das kann man sich als ein binäres Quelltextäquivalent vorstellen, das beim Laden von einem speziellen Compiler übersetzt wird. Am Ende seines Vortrages konnte Malte an Benchmarks zum Laden von Modulen die gute Performance dieses Ansatzes nachweisen. Da sich unter den Zuhörern auch der ein oder andere Systemimplementierer befand, ergab sich die Gelegenheit, auch noch von anderen Alternativen zu erfahren.

Komponentensoftware ist ein vielgelesenes Stichwort! Was man sich aus der Sicht eines Forth-Entwicklers darunter vorzustellen hat, hat uns Egmont Woitzel erklärt. Komponenten sind einzeln übersetzbare Module, deren Dienste andere Module oder übergeordnete Applikationsprogramme über definierte Schnittstellen anfordern können. Also gibt es auch eine Forth-Komponente, deren Funktionalität z.B. aus C-Programmen heraus genutzt werden kann. Problematisch ist hierbei eine effiziente Schnittstelle, wofür Egmont dynamischen Token-Code vorschlägt, bei dem sich die Anwendung zunächst ein Token für eine bestimmte Forth-Funktionalität (=Wort) besorgt (mittels eines Tokenbeschaffungstokens) und mittels dieses Tokens jederzeit die Funktionalität anfordern kann. Implementiert ist das System mit comFORTH und einer DLL als Interface für den Klienten. Diese DLL enthält die notwendigen Funktionen, um via DDE einen Forth-DDE-Server anzusprechen.

Russischer Forth-Prozessor

Den letzten Vortrag hielt wiederum Thomas Beierlein über einen Chip, den er von einem Besuch in Weißrußland mitgebracht hat. Dabei handelt es sich um einen 16bit-RISC-Prozessor mit 2 Stacks, die jeweils mit 16 Worten intern oder mit 256 Worten im externen Speicher ausgelegt sein können. Ein Forth-Chip also, natürlich nicht öffentlich als solcher bezeichnet. Weitere Details zu diesem Chip ließen den Verdacht aufkommen, daß es sich im wesentlich um einen Nachbau des bekannten RTX2000 handelt. Ulrich Hoffmann sah dabei auf jeden Fall ein Patentproblem, da ein Zwei-Stack-Prozessor von Charles Moore patentiert wurde.

Ziemlich geschlaucht wurden wir nun allesamt in Richtung Grillplatz entlassen, wo in rustikaler Atmosphäre weiter diskutiert wurde, denn es was noch nicht Schluß...

....da am späten Abend noch Workshops stattgefunden haben!

Ein offener Forth-Prozessor aus München scharte eine handvoll Leute um Bernd Paysan, um sich mit dem Chip-Design eines Forth-Prozessors, den Bernd entworfen hat, zu beschäftigen. Stichworte wie Verilog und VHDL (?) ließen nicht vielen Tagungsteilnehmern eine Chance, dem Thema näherzukommen. Ich kann nur hoffen, daß Bernd in diesem Heft vielleicht einmal beschreiben wird, um was es da geht.

Ja, wa?!

Reges Interesse fand letzlich der Workshop über Gemeinsamkeiten und Unterschiede von Forth und Java, bei dem ich selbst bereits etwas angetrunken in das Thema einführen durfte. Ganz offensichtlich leben Forthler nicht hinter dem Mond, denn einige hatten bereits tiefgehende Detailkenntnisse zu Suns jüngstem Kind. Dadurch glitt die Diskussion oft viel zu stark in die Tiefe, wo doch das Hauptinteresse der meisten auf Fragen wie Was ist das eigentlich? oder Wie sieht das praktisch aus? lag. Ergebnis: Obwohl Forth und Java virtuelle Stackmaschinen implementieren, sind sie sich bei genauerem Hinsehen doch nicht allzu ähnlich.

Nachdem sich auch dieser letzte Workshop aufgelöst hatte, ging es in diversen Grüppchen und Einzelgesprächen weiter. An den aufgebauten Maschinen wurde noch heftig mit Java-Entwicklungspaketen, Actionspielen in Forth, Forth-Lisp oder Lisp-Forth, oder schlicht dem Flipper aus dem Windows95-Plus-Paket auf der Leinwand experimentiert. Und das, da für alle Beteiligten ebenso interessant wie amüsant, erneut bis in die frühen Morgenstunden.

Ach ja, der Preis für die beste Performance. Die Jury hat ihm am nächsten Tag auf der Mitgliederversammlung vergeben. Für sie war Thomas Beierleins verwegener Kampf auf verlorenem Posten (”Warum geht das jetzt nicht?”) gegen Wolf Wejgaards Holon-Tetris zusammen mit seiner rundherum gelungenen Tagungsorganisation DIE herausragende Leistung. Und Recht hatten sie! Jetzt hat Thomas eben 2 Forth-Weihnachts-CDs.

Natürlich stellt so ein Tagungsbericht im wesentlichen eine Vortragszusammenfassung dar. Das wird der Tagung eigentlich nicht gerecht, da sie immer auch außerhalb der Vorträge einen hohen Erlebniswert hat. Ich kann daher nur jedem, der bisher noch nicht dabei war, nur empfehlen, im nächsten Jahr auch mal hinzufahren.

Foto: Zeitschriften auf rotem Grund (akg1/4-5/96)

Die Ausstellung "Technik - Kunst - Gestaltung am Rande der Tagung..

.. erwies sich als voller Erfolg. Leider fiel die angestrebte Diskussion über gestalterische Aspekte des Forth-Magazins wegen eines Ungleichgewichts zwischen Cocktail-Produktionsleistung der Redaktion und Trinkfähigkeit der Tagenden aus. Dafür wurde Frank "Genius" Rothkamm (vgl. VD 3/95, S.24) fünfmal verkauft und sogar zweimal vorgespielt, diverse Zeichnungen *(rechts)* der (leider viel zu wenigen) für die VD Kulturschaffenden wurden betrachtet und die beeeindruckende Zahl und Schönheit aller bisher erschienenen VDs *(links)* ließ einige Lobeshymnen (insb. für den Künstler-Editor Rolf Kretzschmar) erschallen und alte Erinnerungen wieder aufleben. *(clv)*

Foto: Bildchen auf Pinnwand (akg2/4-5/96)

Der Drachenrat und die Forthgesellschaft.

M.Kalus; Plöner Str.; D-23714 Malente

Man bat mich, einmal aufzuschreiben, was es denn mit dem Drachenrat in der Forthgesellschaft auf sich habe, woher er gekommen sei, was er mache und vor allem wie es darin denn nun tatsächlich zu gehe. Schließlich sei ich schon lange dabei und hätte doch gewisse Einblicke. Nun, das ist schon wahr, aber auch mir ist beileibe nicht alles bekannt.

Zugetragen hat sich die erste Zusammenkunft so: In einer noch kühlen Frühlingsnacht ganz im Norden der Republik, da wo vor langer langer Zeit die Wickinger lebten, unweit von Haitabu, da traf sich 1994 eine Schar merkwürdiger Gestalten in einem abgelegenen Dorfkrug, dem Neukirchner Hof, zu einem noch viel merkwürdigerem Tun. Sie nannten sich Forthler und machten ihre Jahrestagung. Weder zuvor noch danach wurden sie dort je wieder gesehen, einzeln nicht und schon gar nicht gemeinsam. Und doch gingen sie zur festgesetzten Stunde alle daran, am verabredeten Ort geheimnisvoll zu tun. Und obschon sie von weit her aus ganz unterschiedlichen Gegenden, den Bergen im Süden, den Meeren im Norden, dem Hügelland im Osten, den Flußtälern im Westen und den Ebenen in der Mitte des Landes gekommen waren unter mancherlei Gefahr, schienen sie genau zu wissen, was nun hier an diesem abgelegenen Ort geschehen sollte. Etliche von ihnen trafen sich schon Jahr für Jahr in dieser Weise, doch immer an anderem Ort.

Und einige von ihnen, äußerlich nicht von den anderen zu unterscheiden, traten zur späten Stunde etwas Abseits zusammen, um etwas wichtiges zu beraten. Und man nannte sie von da an “Drachenrat”.

Und dann erwischt es am anderen Tage einen Tagungsteilnehmer hinterücks: Er wurde vom jüngsten Mitglied des Drachenrates, ob er wollte oder nicht, zum neuen Drachenhüter für ein weiteres Jahr bestimmt, und wurde danach selber Ratsmitglied. Und so ist es nun in jedem Jahr.

Und wohl weil das keine leichte Aufgabe ist, jeweils den einen Menschen aus der Forthtagung heraus zu finden, der neuer Hüter des SWAP wird - so nennen sie ihren Drachen - weil viele Teilnehmer sich hervorgetan und in der einen oder anderen Weise gezeigt haben, daß sie des SWAP-Drachens würdig sind, bleibt geheim, was beraten wurde und noch geheimer, wie dies geschah. Noch nie drang auch nur ein Sterbenswörtchen nach außen, und so kann ich hier auch darüber nichts weiter berichten, als was man so im Volke darüber spricht.

Es soll gesehen worden sein, wenn auch nur von weitem, das sich der Rat um einen alten Tisch versammelt, und der Älteste von ihnen, der den Drachen zuerst hütete, beginnt. Man will dann gesehen haben, daß alle gleichzeitig die Köpfe zusammenstecken, seltsames murmelnd, und Gerüchte gehen, es würden Zauberformeln gesprochen, die klingen wie ein dumpfes DUP, auch zischeln sei zu vernehmen wie SWAP FETCH STORE und gelegentlich ein lautes DROP. Doch hat noch kein Außenstehender den Sinn erfassen können.

Es heißt ferner, immer nur in jener Nacht spucke der Drache dann Feuer aus seinen zwei Köpfen und genau dann müßten, ebenfalls mit Feuer im Rachen, alle versammelten Drachenhüter in einer gewaltigen gemeinsamen Anstrengung den Swap-Drachen zu seinem neuen Hüter treiben. Weißer Rauch steige dabei gewöhnlich auf, sagt man, wenn mit ernsten Minen an Tischen so alt, daß Bäume hindurchwuchsen, erleuchtet nur von hunderten Jahren versammelter Weisheit, schließlich im Zeitsprung die Entscheidung fällt.

Zuletzt geschah dies im April des Jahres 1996 in Höfchen bei Transsylvanien in einem Gasthof mit dem bezeichnenden Namen “Drachenquelle”. Es wurde der neunte Drachenhüter bestimmt. Wie groß der Rat denn werden kann? Das vermag auch ich nicht zu schätzen, einige noch oder hundert gar? Aber beim Zehnten jeweils soll sich der Swap-Drache verändern, so lautet eine Weissagung des Rates. Im nächsten Jahr werden wir es sehen.

Die Hüter des Swap-Drachens - Jahr und Ort der Verleihung:

- 1. Michael Kalus, 1986, Heide.
- 2. Heinz Schnitter, 1989, Aachen.
- 3. Jörg Staben, 1990, Frankfurt.
- 4. Klaus Schleisiek, 1991, München.
- 5. Ulrike Schnitter, 1992, Rostock.
- 6. Jens Wilke, 1993, Nürnberg.
- 7. Jörg Plewe, 1994, Malente-Neukirchen.
- 8. Friederich Prinz, 1995, Berlin.
- 9. Klaus Kohl, 1996, Mittweida-Höfchen.

PS: Kreiert wurde der Wanderpreis von Klaus Schleisiek 1986. Den zweiköpfigen Gummidrachen, der immer lecker nach Erdbeeren duftet, handelte er in USA einem kleinen Jungen Namens Pieris Maida ab. Die Figur stammt aber aus einer Fabrik in Taiwan. Bei der ersten Verleihung versprach der so geehrte feierlich, er werde den Preis in Bronze gießen lassen, wenn die Forthgesellschaft zehn Jahre alt wird. Das war 1994 in Malente, auch hatte sich Rolf Kretschmar bereit gefunden, die Bronze zu fertigen. Aber dann kam doch etwas dazwischen, und sie verloren sich aus den Augen. Doch in 1996 trafen sich die beiden wieder und verabredten die Arbeit an der Plastik. So hoffen wir nun alle, daß zur 10. Verleihung des Preises der Swap-Drache in Bronze fertig wird.

❑

Serie: PC-Meßtechnik, Teil IV

Die serielle Schnittstelle

von Klaus Kohl / BMC Systeme GmbH
Zeppelinstr. 10; 86406 Mering; Tel.: 08233 / 30524

Da bei der nächsten Folge mit dem Joystick-Port schon unmittelbar ein A/D-Wandler angesprochen wird, soll hier zum letzten Mal ein nicht so streng auf Meßtechnik ausgerichteter Artikel den Aufbau und die Programmierung der RS232-Schnittstelle des PC´s erklären. Da diese Schnittstelle ebenfalls Interruptfähig ist, werden wir sie in späteren Folgen der Serie für die Ansteuerung externer Meßsysteme nutzen. Als Beispiel werden wir hier ein einfaches, interruptgesteuertes Terminalprogramm entwerfen und damit ein Multimeter zur Messung von Spannungen abfragen.

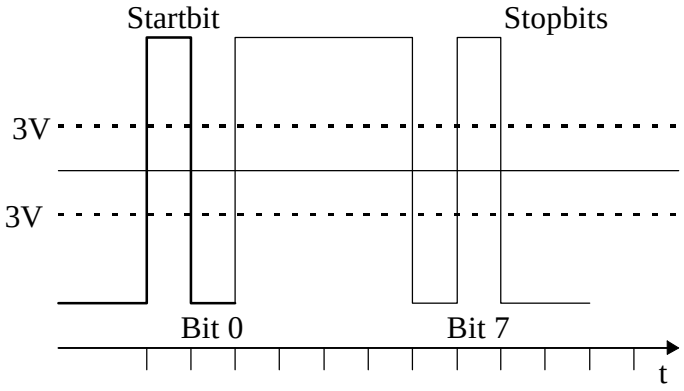
Stichworte: Meßtechnik, PC, RS232, V24

1. Einführung

Inzwischen steht jedem PC über einen 9- oder 25poligen Stecker eine RS232-Schnittstelle zur Verfügung. Neben den beiden für die bidirektionale Datenübertragung notwendigen Pins sind weitere Ein- und Ausgänge vorhanden, die für den Handshake genutzt werden können. Wie wir später noch sehen werden, ist dabei die Ansteuerung dieser Leitungen reine Programmsache und kann nicht automatisch vom verwendeten SIO-Baustein 8250 abgenommen werden.

1.1 Begriffserklärungen

Zuerst muß hier der prinzipielle Ablauf der Übertragung gezeigt werden. Dabei ist vor allem zwischen den 1 (+5V) und 0 (0V) der Datenbits und der tatsächlich an den Datenleitungen TD und RD der seriellen Schnittstelle anstehenden Spannungen zu unterscheiden. Nach Vorschrift sind Spannungen über +3V bis +15V als Startbit oder als 0-Datenbits (SPACE) verwendet. Stopbits oder 1-Datenbits (MARK) werden durch Spannungen von -3V bis -15V gekennzeichnet. Früher wurde durch geeignete Treiber meist ±12V erzeugt. Durch Verwendung integrierter Bausteine wie MAX232 beträgt die Spannung meist nur noch ±10V oder bei 3.3V-Hardware sogar nur noch ±6V. Meist wird dann einfach eine Spannung über +3V als Startbit bzw. 0-Datenbit und Spannungen unter +3V als 1-Datenbit erkannt. Bei der Übertragung ist die Datenleitung außerhalb des PC´s im Ruhezustand auf -12V. Ein +12V-Impuls mit angegebenen Bitlänge (= 1/Baudrate) markiert den Start der Übertragung. Anschließend folgen ohne Pause ebenfalls mit der gleichen Dauer die Datenbits, wobei mit dem niederwertigsten Bit begonnen wird. Im Anschluß an die 7 bzw. 8 Datenbits folgt evtl. noch ein Paritybit und 1, 1.5 oder 2 Stopbits.



Datenübertragung bei RS 232

Baudrate / Baud

Die Baudrate gibt an, wieviel Bits pro Sekunde übertragen werden. Üblich sind hier Werte von 300 (alte Modems) bis 115200 Baud (höchster Wert für PC), wobei zur Zeit folgende Werte am häufigsten verwendet werden:

2400 Baud PC-Mäuse
9600 Baud Modems, externe Peripherie
38400 Baud Verbindung zu Mikrocontroller

Eine Baudrate von 9600 Baud besagt, daß ein Bit für eine Dauer von 1/9600 Sekunden (entspricht 0.104ms) ausgegeben wird. Da wir für die Ausgabe eines Bytes (8 Bit) einschließlich Start- und zweier Stopbits mindestens 11 Bits benötigen, können maximal 872 Zeichen pro Sekunde übertragen werden.

Da ab der steigenden Flanke des Startbits nur noch die Dauer

eines Bits über die richtige Erkennung der Daten entscheidet, müssen bestimmte Toleranzen eingehalten werden. Falls nur in der Mitte eines Bits der Wert beachtet wird, muß der Frequenzfehler bei einem Startbit und 8 Datenbits kleiner als 0.5/9 oder 5,6 % sein. Meist arbeiten die SIO-Bausteine mit einer bis zu 16fachen Überabtastung und einer 9aus16-Erkennung und erlauben deshalb nur Fehler von 7/16/10 oder 4,8%.

Datenformate

Bei vielen (Modem-)Anschlüssen ließt man die Angabe 1200-7-N-2. Diese bedeutet, daß bei Verbindungsaufnahme mit diesem Empfänger bei einer Geschwindigkeit von 1200 Baud nur 7 Datenbit gesendet werden. Das N steht für “None Parity” und zeigt an, daß kein Prüfbit erwartet oder gesendet wird. Weitere Möglichkeiten sind hier E für “Even Parity” und O für “Odd Parity”. Mit der 2 am Schluß wird noch angegeben, daß zwei Stopbits erwartet werden. Dies gibt langsamen Empfängern die Zeit zur Auswertung des Zeichens.

Halbduplex / Fullduplex

Einige Controller (z.B. der RTX-2000) lösen die Ansteuerung der RS232-Schnittstelle in reiner Software. Diese sind nicht in der Lage, gleichzeitig zu Senden und zu Empfangen und arbeiten deshalb in “Halbduplex”. Über Handshake-Leitungen wird dann dem Kommunikationsparter angezeigt, ob der Empfänger bereit ist. Serielle Bausteine können dagegen gleichzeitig Senden und Empfangen und Arbeiten deshalb “Fullduplex”. Bei intelligenten Bausteinen oder geeigneter Software wird über die Handshake-Leitungen angezeigt, daß der Empfangspuffer voll ist.

Mark

Der Ruhezustand der Leitung (-12V = 1-Datenbit) wird als MARK bezeichnet. In seltenen Fällen wird auch als Parity-Bit M angegeben und dann ein 1-Datenbit ausgegeben.

Parity (Odd/Even)

Zur Kontrolle, ob das Zeichen richtig übertragen wird, dient ein zusätzliches Bit. Bei “Odd Parity” wird dieses Bit so gewählt, daß die Anzahl der 1-Bits einschließlich des Paritybits ungerade ist. Bei “Even Parity” muß dagegen die Zahl der gesetzten Bits gerade sein. In dem PC übernimmt sogar der Baustein die Prüfung dieses Bits, wobei auch MARK- oder SPACE-Parity angebbar ist.

Protokolle

Nicht unmittelbar von den einzelnen Bits abhängig sind die vereinbarten Protokolle. Diese dienen bei der Übertragung größerer Datenmengen durch zusätzliche Prüfsummen zur besseren Fehlerkontrolle und können zum Teil automatisch diese Fehler durch erneute Anforderung der Daten korrigieren. Bekannte Protokolle sind z.B. Kermit , X- oder Z-Modem.

RS-232 / V.24 / DIN66020

Historisch ist das hier angesprochene Übertragungsverfahren erstmals 1969 von der EIA (Electronic Industries Association) als RS-232 formuliert worden. Sie diente zur Definition des Datenaustauschverfahrens zwischen Computer oder zwischen Computer und Modem. Sie wurde später nochmals überarbeitet und existiert noch heute in der entgültigen Form als RS-232-C. Von der internationalen Standardisierungsorganisation CCITT wurde das Gegenstück V.24 herausgegeben. Sie enthält jedoch nur funktionelle Eigenschaften. Die ergänzenden elektrischen Kennwerte zur RS-232-C sind in einer eigenen Empfehlung V.28 zusammengefaßt. Die deutsche Version des RS232-Standards hat die Bezeichnung DIN66020.

Space

Das Gegenstück zum Mark-Bit ist das Space-Bit, welches identisch mit den Stopbits oder den 0-Datenbits ist. In seltenen Fällen wird es auch als Paritybit genutzt.

Startbit

Zur Synchronisierung der Datenübertragung wird zuerst mit dem +12V-Signal ein Startbit übertragen. Es wird sonst nicht weiter berücksichtigt, muß aber die Dauer von genau einem Bit haben.

Stopbit

Um genügend Raum zur Synchronisation mit dem nächsten Datenbit zu lassen oder bei reinen Software-Lösungen Zeit zur Auswertung des erfaßten Wertes zu lassen, wird mindestens 1 Stopbit nach den Daten gesendet. Es hat den Pegel der in Ruhe befindlichen Leitung.

Synchron / Asynchron

Bei der RS232-Schnittstelle spricht man immer von einer asynchronen Schnittstelle. Im Gegensatz zu der synchronen Schnittstelle hat man hier keine Taktleitung, auf die sich die Übertragung “synchronisiert”. Statt dessen wird das Startbit und ab dann nur noch eine Zeitsteuerung verwendet.

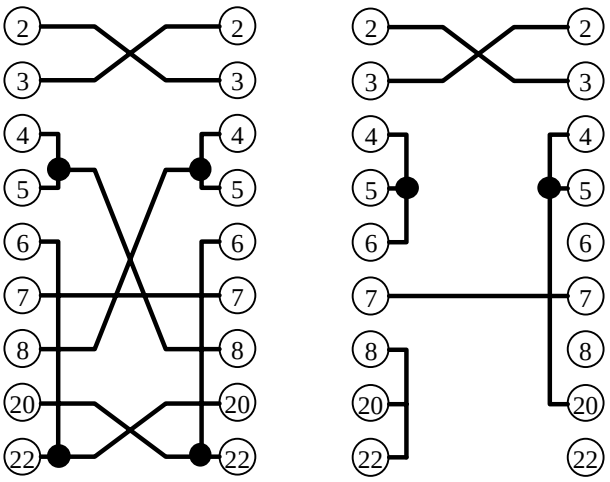
BREAK

Als Break wird eine Folge von Startbits bzw. 0-Bits länger als eine Zeichenlänge erkannt. Die Ursache dazu kann z.B. eine Leitungsunterbrechung sein. Die meisten Bausteine erkennen dies automatisch und erlauben einen Interrupt bei diesen Zustand. Ansonsten führt dieser Zustand zur ständigen Erkennung von 0-Bytes.

1.2 Pinbelegung

Folgende Liste zeigt die Pinbelegung des 9- und 25poligen Steckers. Bei manchen älteren Schnittstellen ist auch noch Pin 23 des 25poligen Steckers für die Erkennung der Baudrate reserviert, wird aber bei den PC´s nicht genutzt.

Pin (25pol)	Pin (9pol)	Richtung	Verwendung
1		.....	Masse für Abschirmung
2	3	Ausgang	TD = Transmitted Data
		.....	Vom PC ausgehende Datenleitung
3	2	Eingang	RD = Received Data
		.....	Zum PC gehende Datenleitung
4	7	Ausgang	RTS = Request to Send
		.....	Ein +12V-Signal zeigt an, daß Daten bereit stehen
5	8	Eingang	CTS = Clear to Send
		.....	Ein +12V-Signal zeigt an,
		.....	daß der Empfänger bereit ist
6	6	Eingang	DSR = Data Set Ready
		.....	Früher hat ein Modem (=”Data Set”) über diese
		.....	Leitung angezeigt, daß es vorhanden und bereit ist.
7	5	.....	Signalmasse....
8	1	Eingang	



*Nullmodem und Smart*modem

DCD = Data Carrier Detect
20 4 Ausgang..... DTR = Data Terminal Ready
22 9 Eingang ..... RI = Ring Indicator
..... Modems zeigen über diese Leitung
..... das Klingeln des Telefons an.

1.3 Anschlußart

Da die RS232-Leitungen von den üblichen Empfängern wie Drucker oder Modem meist nicht vollständig unterstützt werden, hat sich eine große Vielfalt von Anschlußbelegungen gebildet. Bestimmte Verknüpfungen sind oft notwendig, weil ein Gerät einen bestimmten Pegel an seinen Eingängen oder das sendende PC-Programm einen Handshake erwartet. In der Praxis werden dabei die Versionen Nullmodem und Smart Modem am häufigsten eingesetzt.

1.4 Speicheradressen

Beim Bootvorgang sucht der PC automatisch auf vier Adressen ($3F8, $2F8, $3E8 und $2E8) nach einem seriellen Baustein. Damit aber intelligente Karten weitere RS232-Ports installieren und Anwenderprogramme diese Adressen ermitteln können, gibt es in den BIOS-Variablen vier Speicherstellen für die Basisadresse der vorhandenen Schnittstellen. Nicht verfügbare Ports werden mit dem Wert 0 gekennzeichnet.

Adresse der BIOS-Variable	**Verwendung**
$0040:$0000/1	COM1-Basisadresse (meist $03F8)
$0040:$0002/3	COM2-Basisadresse (meist $02F8)
$0040:$0004/5	COM3-Basisadresse
$0040:$0006/7	COM4-Basisadresse

Bei jedem angegebenen RS232-Port werden bis zu 8 Adressen für die Programmierung belegt. Zusätzlich werden üblicherweise die Interrupts 3 (für COM2 und COM4) und 4 (für COM1 und COM3) reserviert.

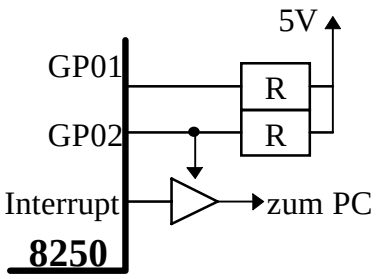
Die 8 Adressen verwalten sogar 11 Register, wobei Zugriffe auf Offset 0 beim Schreiben die Ausgabe eines Zeichens starten und beim Lesen des gleichen Ports das zuletzt erkannte Zeichen auslesen. Über ein Bit im Line-Control-Register (Offset 3, Bit 7) kann der Teiler für den Baudratengenerator auf Offset 0/1 eingeblendet werden (s.Tabelle).

Der Baudratenteiler für die Übertragungsfrequenz wird durch einen 1.8432MHz-Quarz gespeist, der früher vom 4,77MHz-Bustakt geteilt durch 3 kam. Nach einer Teilung durch 16 (für die oben angesprochene Überabtastung) können folgende sinnvollen Werte eingestellt werden:

Teiler:	1	3	6	9	12	48	96	384
Baudrate:	115200	38400	19200	12800	9600	2400	1200	300

Der 12800Baud-Wert wurde z. B. für eine Mikrocontroller-Entwicklung benötigt, wo von einem 8MHz-Quarz die eingebaute SIO gespeist wurde. Die dort einstellbare Frequenz von 12500 Baud liegt noch innerhalb der Fehlertoleranz.

Offset	**Bit**	**Bedeutung**
0	0-7	Schreiben: Zeichenausgabe an TD starten Lesen: Abfrage des zuletzt erfaßten Zeichens
1		Interupt Enable Register
	0	1: Interrupt, wenn Daten verfügbar (RxRDY)
	1	1: Interrupt, wenn Sendepuffer leer ist (TBE)
	2	1: Interrupt bei Übertragungsfehler
	3	1: Interrupt bei Änderung der Eingangsleitungen
	4-7	immer auf 0
2		Interupt Identification Register Bit 0=0: Interrupt steht an Wert 0: RS232-Eingangsleitungen geändert Wert 2: TBE..... Wert 4: RxRDY ..... Wert 6: BREAK oder Serialize Error
3		Line Control Register (Übertragungsprotokoll)
	0/1	Anzahl Datenbits (0=5, 1=6, 2=7 und 3=8)
	2	Anzahl Stopbits (0=1 und 1=2 Stopbits)
	3..5	Parity (0=None, 1=Odd, 3=Even, 5=Mark, 7=Space)
	6	Break Controll (0=aus, 1=an)
	7	DLAB: bei 1 liegt Baudratenteiler auf Register 0/1
4		Modem Control Register (steuert Ausgänge)
	0	DTR (Pin 20)
	1	RTS (Pin 4)
	2	GP01
	3	GP02 (Interrupt Enable; siehe unten)
	4	Loopback (für Tests)
	5-7	immer auf 0
5		Line Status (Bit 0-5 können Interrupt auslösen)
	0	RxRDY (Empfangspuffer voll)
	1	Overrun Error
	2	Parity Error
	3	Framing Error
	4	Break Detect
	5	TBE (Sendepuffer leer)
	6	TXE (Ausgabepuffer leer, Übertragung beendet)
	7	immer 0
6		Modem Status (Bit 0..3: Änderung seit letztem Lesen)
	0	Flag für Änderung an CTS
	1	Flag für Änderung an DSR
	2	Flag für Änderung an RI
	3	Flag für Änderung an DCD
	4	Aktueller Status an CTS.....(RTS bei Loopback)
	5	Aktueller Status an DSR.....(DTR bei Loopback)
	6	Aktueller Status an RI.....(GP02 bei Loopback)
	7	Aktueller Status an DCD.....(GP01 bei Loopback)
7		freies RAM (wird nicht genutzt)
8		Auf Register 0 gelegt, wenn Bit DLAB gesetzt ist enthält niederwertige Byte des Baudratenteilers
9		Auf Register 1 gelegt, wenn Bit DLAB gesetzt ist enthält höherwertige Byte des Baudratenteilers



Interrupt

Ähnlich können natürlich auch andere Teilerwerte genutzt werden.

1.5 RS232-Interrupt beim PC

Wie schon im ersten Artikel dieser Serie erwähnt, muß man bei einem PC außer den Register des entsprechenden Bausteins noch die Interrupttabelle (zur Festlegung der Adresse des Interruptprogrammes) und den Interruptcontroller (zur Freigabe der Leitung) programmieren. Im PC ist es üblich, für die erste serielle Schnittstelle die Interruptleitung 4 und für die zweite RS232 die Interruptleitung 3 zu nutzen. Falls noch mehr Schnittstellen zur Verfügung stehen, muß man sich diese Leitungen teilen. Damit man einen RS232-Controller von der Interruptleitung abschalten kann, wurde schon bei den IBM-Entwicklern einer der beiden nicht genutzten 8250-Portleitungen (GP02) zur Ansteuerung eines Treiberbausteins herangezogen. Nur wenn diese Leitung auf 1 ist, kann der 8250 auch einen Interrupt auslösen.

Neben der üblichen Bestätigung am Interruptcontroller muß auch auf dem 8250 eine geeignete Aktion zur Beendigung des Interruptzustandes erfolgen. Folgende Möglichkeiten sind vorgesehen:

Interruptart	**Notwendige Aktion**
Receive Error oder Break	Line Status Register auslesen
Receive Data	Datenregister lesen

File: terminal.lst vom 12.5.1996

```
 1
 2 TERMINAL.SCR
 3
 4                    Screen 0                                                    Screen 8
 5
 6  0 \\ Verwaltung der COM-Schnittstelle bei PC      ( 05.05.96/KK )  \ -DTR   +DTR  = Steuerleitung ein/aus              ( 16.08.93/KK )
 7  1                                                                  ( DTR-Leitung auf +12 V = logisch Null )
 8  2 System:            KKF_PC V1.2/2                                  Code -dtr      ( -- )
 9  3 Änderungen:        16.08.93  KK: aus KKFT2 entnommen               (portaddr #) dx mov,   4 # dx add,    \ Modem-Kontrollregister
10  4                                                                     dx byte in,                \ Inhalt zum AL-Register
11  5                         Hinweise:                                   $1c # al and,              \ DTR und RTS auf 0 setzen = +12 V
12  6  * Verwendet kleinen 1K-Buffer im FORTH-Speicher                  dx byte out,               \ AL wieder in Port schreiben
13  7  * Keine Handshake's verwendet                                     next,                      \ nächster FORTH-Befehl
14  8                                                                     End-Code
15  9  Quelle für SIO-Programmierung: PC-volksFORTH 3.81.41 (KS)
16 10                                                                  ( Analog dazu DTR und RTS wieder auf 1 setzen = -12 V )
17 11  ACHTUNG:  Register BX wird im KKF_PC als TOS verwendet            Code +dtr      ( -- )
18 12              Spoolergröße: 128 Byte                              (portaddr #) dx mov,   4 # dx add,
19 13                                                                    dx byte in,    $fc # al and,    1 # al or,    dx byte out,
20 14                                                                    next,
21 15                                                                  End-Code
22
23
24            Screen 1                                                    Screen 9
25
26  0 \ Loadscreen                                    ( 15.04.94/ki )  \ s-int = Interruptroutine; rx?                ( 05.05.96/KK )
27  1                                                                  | Proc s-int
28  2   \needs Assembler  2 loadfrom ASM8086.SCR                          dx push,    si push,    ax push,             \ Register retten
29  3                                                                    cs seg,   queue # si mov,              \ Queue-Adresse holen
30  4   &02 capacity 1- thru                                           cs seg,   si ) ax mov,          \ Anzahl gespeicherte Zeichen
31  5                                                                    ax dx mov,    ax inc,    $007f # ax and,          \ +1 MOD 128
32  6   5 baud# !   1 port# !                        \ COM2 mit 9600 Baud   cs seg,   ax si ) mov,    dx si add,          \ Zurückschreiben
33  7                                                                    cs seg,   (portaddr #) dx mov,   dx byte in, \ Zeichen holen
34  8                                                                    cs seg,   al 4 si d) mov,        \ und in den Queue schreiben
35  9                                                                    $20 # al mov,    i-ctrl #) byte out,          \ EOI für 8259
36 10                                                                    ax pop,    si pop,    dx pop,    iret,          \ Register holen
37 11                                                                  End-Proc
38 12                                                                | Code rx?   ( -- f ) \ Flag, ob Zeichen im Puffer sind
39 13                                                                    tos push,    queue # di mov,
40 14                                                                    di ) tos mov,    2 di d) tos xor,
41 15                                                                    0<> IF,   -1 # tos mov,   THEN,    NEXT,                End-Code
42
43
44            Screen 2                                                    Screen 10
45
46  0 \ Tabellen für COM-Port                         ( 05.05.96/KK )  \ (rx   rx   = Zeichen aus Queue holen              ( 16.08.93/KK )
47  1   Create baudtab     ( Tabelle mit der Baudrate/100 )                ( Holt Zeichen aus dem Queue, verändert evtl. die Zeiger )
48  2     &1152 ,  &576 ,  &384 ,  &192 ,  &128 ,                          Code (rx        ( -- char )
49  3       &96 ,   &24 ,   &12 ,  &288 ,    &3 ,                           tos push,    si push,
50  4   Create diftab       ( Tabelle mit den Teilern )                  queue 2+ # si mov,
51  5          1 ,      2 ,      3 ,      6 ,      9 ,                     cs seg,   si ) ax mov,
52  6        &12 ,   &48 ,   &96 ,      4 ,  &384 ,                       ax dx mov,    ax inc,    $007f # ax and,
53  7                                                                      cs seg,   ax si ) mov,    dx si add,
54  8   Create porttab     ( Tabelle mit Portadressen )                  cs seg,   2 si d) al mov,    0 # ah mov,
55  9     $03f8 , $02f8 , $03e8 , $02e8 ,                                  ax tos mov,    si pop,    next,
56 10                                                                     End-Code
57 11   Create ileveltab  ( Tabelle mit Interrupt-Level )              ( Holt Zeichen, wartet, bis Zeichen bereit )
58 12        $10 ,    $08 ,    $10 ,    $08 ,                             : rx            ( -- char )  rx? IF (rx exit THEN
59 13   Create inttab       ( Tabelle mit Interrupt-Adresse )             Timeout  BEGIN rx? IF drop (rx exit THEN
60 14        $30 ,    $2c ,    $30 ,    $2c ,                                                  1- DUP 0=                \ Timeout abgelaufen ?
61 15                                                                                        UNTIL DROP -1 ERROR" Receive-Timeout";
62
63
```

```
64              Screen 3                                                Screen 11
65
66  0 \ Variablen für den COM-Port                ( 05.05.96/KK )  \ s-init  = Initialisierung der Schnittstelle   ( 15.04.94/ki )
67  1   2Variable oldivec                    ( Alter Interrupt-Vektor )  | Code s-init   ( -- )
68  2   &50000 Constant Timeout        ( Abbruch nach 50000 Schleifen )      tos push,  ds push,              \ TOS und DS-Register sichern
69  3                                                                          ax ax xor,  (iaddr #) w mov,
70  4 \ Einstellungen                                                          ax ds mov,    s-int # w ) mov,          \ Interuptvektor setzen
71  5   Variable baud#                    ( Nummer der Baudrate )      cs ax mov,  ax 2 w d) mov,        \ und DS wieder zurückladen
72  6   Variable baud                          ( Baudrate / 100 )      ds pop,  (portaddr #) dx mov,  3 # dx add,
73  7   Variable port#                    ( Nummer des Port - 1 )      $80 # al mov,    dx byte out, \ Baudratenregister einschalten
74  8                                                                          2 # dx sub,  (baud #) ax mov,    al ah xchg,          \ Baudrate
75  9 \ Dieser SYSVAR-Bereich ist frei und hat feste Speicheradressen      dx byte out,  dx dec,  al ah xchg,  dx byte out,     \ setzen
76 10   Create (baud        2 allot          ( Teiler für Baudrate )      3 # dx add,  $a06 # ax mov,  dx out,  \ 7bit, noP, +RTS +OUT
77 11   Create (portaddr   2 allot          ( Addresse des Ports )       2 # dx sub,  1 # al mov,  dx byte out,  \ Interrupt erlauben
78 12   Create (ilevel     2 allot                ( I-Maske )      i-mask #) byte in,
79 13   Create (iaddr      2 allot          ( Interrupt-Adresse )     (ilevel #) al and,                  \ 8259 aktivieren
80 14                                                                       i-mask #) byte out,  tos pop,  next,
81 15                                                                     End-Code
82
83
84              Screen 4                                                Screen 12
85
86  0 \ Neue Baudrate oder Port setzen            ( 16.08.93/KK )  \ sioinit  -sioinit  = Initialsierung/Reset     ( 16.08.93/KK )
87  1   : setbaud      ( baud port -- )                                ( Löscht Queue-Zeiger und Initalisiert Baustein/Interupt )
88  2     dup port# !  2*  dup inttab    + @       (iaddr     !          : sioinit      ( -- )
89  3                        dup ileveltab + @   not (ilevel    !          baud# @  port# @  setbaud                  \ Einstellung
90  4                            porttab    + @       (portaddr !          0 (iaddr @ l2@  oldivec 2!          \ Interrupt-Vector merken
91  5     dup baud# !  2*  dup baudtab    + @       baud         !          queue off  queue 2+ off                  \ Puffer löschen
92  6                            diftab     + @       (baud       ! ;        s-init  +dtr ;
93  7
94  8                                                                    ( Blockiert Interupt, Schaltet RTS u. DTR ab )
95  9                                                                      : -sioinit     ( -- )
96 10                                                                       0 (portaddr @ 1+  pc!                \ 8250-Interupt sperren
97 11                                                                       0 (portaddr @ 4 + pc!                \ -RTS/-rts/-out2
98 12                                                                       i-mask pc@ (ilevel @ not or i-mask pc! \ 8259 sperren
99 13                                                                       oldivec 2@  0 (iaddr @  l2! ;        \ Vektor zurücksetzen
100 14
101 15
102
103
104             Screen 5                                                Screen 13
105
106  0 \ Queue und notwendige Befehle              ( 05.05.96/KK )  \ comval@                                    ( 05.05.96/KK )
107  1 ( Datenqueue mit 128 Bytes und zwei Zeiger für Interuptroutine)     Create comval$    &15 allot          \ Platz für "DC-00.000 V  "
108  2 ( Queue+0: Anzahl der gespeicherten Zeichen )
109  3 ( Queue+2: Offset zum nächsten auszugebenden Zeichen )              : com$>val     ( -- val )
110  4   $0084 Constant qlen          ( Bedarf für Queue )                  0
111  5                                                                        comval$ 3 +  1 FOR >r  &10 *  r@ c@ $30 -  +  r> 1+  NEXT
112  6   Create queue  qlen allot  ( Queue )                                 1+             2 FOR >r  &10 *  r@ c@ $30 -  +  r> 1+  NEXT
113  7                                                                        7 -             c@    Ascii -  =  IF negate THEN ;
114  8   $20 Constant I-ctrl              \ 8259-Register
115  9   $21 Constant I-mask               \ 8259-Maske
116 10
117
118             Screen 6                                                Screen 14
119
120  0 \ tx? = Statusabfrage für Zeichenausgabe    ( 05.05.96/KK )  \ comval@                                    ( 05.05.96/KK )
121  1 ( Test, ob ein Zeichen ausgegeben werden kann )                    : (comval@      ( -- val ) \ $7fff bei Fehler; $8000 bei Taste
122  2   Code tx?         ( -- f )          \ f=-1, wenn bereit          BEGIN rx? WHILE rx drop REPEAT                \ Puffer löschen
123  3     tos push,                  \ TOS(=Top Of Stack) zum Datenstack     Ascii D  tx                          \ Startbefehl ausgeben
124  4     (portaddr #) dx mov,    5 # dx add,          \ Statusadresse      0    &10000                          \ Offset, Schleifenzähler
125  5     dx in,                     \ Port zum Register A holen             BEGIN key? IF 2drop $8000 exit THEN    \ Abbruch durch Taste
126  6     tos tos xor,              \ TOS auf 0 setzen                         rx? IF drop rx $7f and  \ Zeichen holen und Maskieren
127  7   \ $1020 # ax and,            \ Ausgabe mit warten auf Handshake                $0d case?
128  8     $0020 # ax and,             \ Ausgabe ohne Handshake                        IF &13 = IF com$>val ELSE $7fff THEN exit THEN
129  9   \ $1020 # ax cmp,            \ dann Test, ob diese Bits gesetzt                over comval$ +  c!  1+
130 10     $0020 # ax cmp,             \ dann Test, ob diese Bits gesetzt                &14 case? IF $7fff exit THEN  &10000 \ Zu Lang
131 11     0= IF, tos dec, THEN,     \ Zeichenausgabe erlaubt ?                      ELSE 1- ?dup 0= IF drop $7ffe exit THEN      \ Neu
132 12     next,                      \ Next kompiliert Next-Makro                   THEN
133 13   End-Code                                                             REPEAT ;
134 14                                                                      : comval@      ( -- val ) \ Mit erneutem Versuch bei Timeout
135 15                                                                        BEGIN (comval@ $7ffe case? 0= UNTIL ;
136
137
138             Screen 7                                                Screen 15
139
140  0 \ (tx  tx  = Zeichenausgabe                 ( 16.08.93/KK )  \ Mini-Terminal                                ( 05.05.96/KK )
141  1 ( Unbedingte Zeichenausgabe direkt in den 8250-Port )               : terminal     ( -- ) \ Terminalmodus
142  2   Code (tx        ( char -- )                                          sioinit
143  3     tosl al xchg,          \ Zeichen zum AL-Register bringen         BEGIN key?  IF  key  3 case? ?exit
144  4     (portaddr #) dx mov, \ Datenadresse in's D-Register                          tx?          IF tx ELSE drop THEN  THEN
145  5     dx byte out,            \ AL ausgeben                                  rx?    IF  rx $7f and
146  6     tos pop,                 \ nächsten Stackwert ins D-Register                  $20 case? IF Ascii . THEN
147  7     next,                    \ Next kompiliert Next-Makro                         $0d case? IF cr ELSE emit THEN  THEN
148  8   End-Code                                                             REPEAT
149  9                                                                       -sioinit ;
150 10 ( Wartet, bis letztes Zeichen ausgegeben wurde )
151 11   : tx             ( char -- )                                      : multimeter  ( -- ) \ Werte vom Multimeter anzeigen
152 12     timeout                          \ Schleifenanzahl                base push  decimal  sioinit  comval$ &14 blank
153 13     BEGIN tx? IF drop (tx exit THEN                                   BEGIN comval@  $8000 <>
154 14           1- ?dup 0=                  \ Timeout abgelaufen ?         WHILE cr  comval$ &14 type  ."  -> "  6 .r  ." mV  "
155 15     UNTIL -1 Error" Transmit-Timeout" ;                             UNTIL key drop  -sioinit ;
156
157
```

LängsteZeilehat 131 Zeichen

TBE	Neue Daten schreiben oder
Interrupt Identification Register lesen	
RS232 Input Change	Modem Status Register lesen

2. FORTH-Terminalprogramm

Eigentlich stellt das Betriebssystem des PC´s über Interrupt 14 einige Funktionen zum Senden und Empfangen von Daten über die RS232-Schnittstelle zur Verfügung. Da aber dies nur bis zu einer Geschwindigkeit von 9600 Baud geht und die Steuerregister nicht besser angesteuert werden können, wird bisher immer auf eigene Routinen zurückgegriffen. Das folgende Listing ist aus einem Hardware-Testprogramm in KK-FORTH entnommen. Urspüngliche Quelle der Kernroutinen war ein Terminalprogramm von Klaus Schleisiek zum volksFORTH-PC. Der hier ebenfalls aufgelistete Beispielscreen (Screen #15) enthält ein einfaches Terminalprogramm TERMINAL , daß empfangene Zeichen ausgibt und Tastatureingaben zur Schnittstelle schickt. Die Baudrate und die Schnittstelle muß vorher mit SETBAUD gesetzt werden. Das ebenfalls vorhandene Programm MULTIMETER wird weiter unten beschrieben.

Anmerkung zum Listing

Die Pegel der Steuerleitungen sind hier für die Kommunikation zu Einplatinencomputer vorbereitet. Ähnlich wie hier bei der Leitung DTR beschrieben, können auch die anderen Steuerleitungen geändert werden. Da der SIO-Baustein selbst keine Auswertung der Handshake-Leitungen durchführt, muß der Ausgabeteil TX bzw. TX? dies übernehmen. Meist kommt man ohne Handshake aus und verwendet nur die Abfrage, ob das nächste Zeichen ausgegeben werden kann. Soll auch noch das RTS berücksichtigt werden, muß das entsprechende Bit (wie in Screen 6 in der ausmaskierten Zeilen) getestet werden. Die Größe des Empfangspuffer ist in Abhängigkeit der Baudrate und der erwarteten Programmzeiten entsprechend groß zu wählen.

Anmerkung zu DOS-Programmen in Windows

Unter Windows 3.1 einschließlich Windows 95 hat sich in der Praxis gezeigt, daß bei sehr schneller bidirektionaler Übertragung manchmal ein Fehler auftritt. Es wird dann ein schon empfangenes Zeichen erst dann dem (Interrupt)-DOS-Programm gemeldet, wenn es das nächste Zeichen ausgibt. Unter Windows NT scheint dieser Fehler aber wieder korrigiert zu sein.

3. Anschluß eines Multimeters

Bei der Hardware vieler Anbieter von Meßsystemen ist eine serielle Schnittstelle zur Einstellung der Parameter oder Abfrage der Daten schon integriert. Das nachfolgende Beispiel verwendet dazu ein z.B. bei Conrad erhältliches Multimeter, daß über diese Schnittstelle den aktuellen Meßwert ausgibt und deshalb z.B. zur Kalibrierung von Analogausgängen genutzt werden kann.

Dabei erwartet das Multimeter das Zeichen “D” und liefert nach der Messung ein String im Format “DC-00.000 V ” zurück. Das hier gelistete COMVAL@ empfängt diesen String, wertet ihn aus und liefert die Spannung in Millivolt zurück. Dabei werden mit den nicht auftretenden Meßwerten -32766 ($7fff) ein Empfangsfehler und mit $8000 ein Abbruch mittels beliebiger Taste kodiert. Da man beim Start des Testprogrammes häufig vergißt, das Multimeter einzuschalten, wird nach einer Timeout-Zeit (hier willkürlich 10000 Schleifendurchläufe) das Startzeichen nochmals ausgegeben.

4. Einsatzmöglichkeiten

Die oben geschilderte Abfrage von Multimeter ist nur eine der Möglichkeiten, die mit Hilfe eigener Schnittstellenprogramme realisiert werden kann. Etwas größere Meßsysteme (leider meist auch entsprechend teuer) können sogar von sich aus Messungen durchführen und liefern nach geeigneter Abfrage ganze Signalverläufe zurück. Oft muß man diese Systeme erst mittels der RS232-Schnittstelle initialisieren.

Leider kenne ich bisher keine A/D-Wandler, der automatisch Signale mit RS232-konformen Timing liefert. Meist sind sie für entspechende RISC-Prozessoren vorbereitet und erwarten einen externen Takt zur Erzeugung des Datenstroms. Doch gibt es einige Schaltungsbeispiele, bei denen die Steuerleitungen der RS232-Schnittstelle dafür “mißbraucht” werden. Neben den Taktsignal wird wie schon in der vorherigen Folge dieser Serie (Centronicsport) die Stromversorgung für das Meßsystem geliefert. Man hat hier sogar den Vorteil höherer Spannungen, da mindestens ±3 bis ±15V von den Schnittstellenbausteinen ausgegeben werden.

5. Literatur

(1) Joe Campbell
C Programmer´s Guide to Serial Communications
Howard W. Sams & Company

(2) Günther Klotz
Bits im Gänsemarsch
c´t 1986, Heft 12, Seite 185ff

(3) Wolfgang Hartung, Michael Felsmann, Andreas Stiller
PC-Bausteine: Der UART 8250 als Tor zur seriellen Welt
c´t 1988, Heft 5, Seite 204ff

(4) Der Interrupt 14 für serielle Schnittstellen
DOS Extra 1´87/88

(5) Thom Hogan
Die PC-Referenz für Programmierer
Microsoft Press

(6) Klaus Dembowski

Forth und der Anfänger

von Thomas Beierlein
Thomas-Mann-Str. 9; D-09648 Mittweida

Einige Themen auf der diesjährigen Forth-Tagung und die aktuellen Diskussionen in de.comp.lang.forth veranlaßten mich, meinen auf der EuroFORTH '95 gehaltenen Vortrag noch einmal herauszukramen und mich mit dem Thema 'Forth und Anfänger' nochmals auseinanderzusetzen.

Stichworte: Anfänger Holon euroForth95

Ein mehrfach aufgeworfenes Thema der Tagung war wieder einmal die Nachwuchsgewinnung. Friederich Prinz plädierte in seinem Vortrag für die Entwicklung eines 'Visual Forth' unter Windows um Einsteigern einen leichten Zugang zu Forth unter einer modernen Bedienoberfläche zu ermöglichen. Thomas Höhenleitner suchte Anregungen wie er einen Forth-Kurs gestalten kann usw.

Wir alle wissen, daß Forth keine der weitverbreiteten Sprachen ist und das es zunehmend schwerer wird, Neueinsteiger für Forth zu gewinnen.

Akzeptanz von Forth

Wenn man über die mangelnde Akzeptanz von Forth spricht, hört man oft Argumente wie

- Forth unterstützt nur Integer Arithmetik,
- Umgekehrt polnische Notation und die zugehörige Stackakrobatik machen Forth zu einer praktisch unleserlichen Sprache und
- Forth ist zu komplex zu benutzen.

Auf die ersten Argumente lassen sich sehr leicht Antworten finden:

- Viele Forth-Systeme bieten Gleitkomma-Arithmetik als Zusatzpaket.
- UPN ist ein effektives Mittel zur Parameterübergabe. Sie erzeugt wenig Laufzeit-Overhead und kann bei geeignetem Faktoring leicht beherrscht werden.
- Unleserliche Programme sind die Folge eines schlechten Programmierstils bzw. schlecht genutztem Faktoring.

Um dem dritten Argument auf die Spur zu kommen, muß man tiefer graben und versuchen, die Natur von Forth selbst zu analysieren.

Was ist Forth?

Forth besteht typisch aus einer Ansammlung von Routinen (Worten), die durch einen Interpreter interaktiv aufrufbar sind. Unter diesen Worten haben einige die besondere Eigenschaft das existierende Forth-System um neue Worte zu erweitern, indem entweder Maschinencode-Befehle oder andere Forth-Worte zu neuen, leistungsfähigeren Routinen kombiniert werden.

Im Unterschied zu anderen Programmiersprachen wird eine Applikation jedoch nicht als selbständiges vom Compiler strikt getrenntes Programm erzeugt, sondern das existierende Forth-System wird auf die oben beschriebene Art und Weise erweitert, bis es den Erfordernissen des zu lösenden Problems genügt.

Die Applikation und das Werkzeug zu ihrer Programmierung bilden ein untrennbares Konglomerat. Worte können gleichzeitig Bestandteil sowohl des erzeugenden Werkzeuges, des Forth-Compilers, als auch der Applikation sein. Ihre konkrete Bedeutung wird erst während ihrer Benutzung festgelegt.

Wozu wird Forth benutzt?

Aus meiner Sicht lassen sich im wesentlichen vier Hauptanwendungsfelder benennen:

- 1. Erzeugung eigenständiger Anwendungsprogramme - Turnkey Applikationen.
- 2. Einsatz in Umgebungen, die theoretisch kaum durchdrungen sind und daher einen hohen Grad an Interaktivität und Flexibilität zur Durchführung technischer Experimente erfordern.
- 3. Erweiterung der gegebenen Forth-Syntax zu einer problemorientierten Sprache, die der zu lösenden Aufgabe besser entspricht. Diese erweiterte Syntax wird schließlich benutzt, um die fertige Applikation zu erstellen.
- 4. Nutzung von Forth als Plattform für die Implementation neuer Sprachkonzepte (Expertensystem, objektorientierte Programmierung usw.).

Die Erstellung von Turnkey Applikationen (Punkt 1) wird relativ selten mit reinen Forth-Systemen gelöst. Grund dafür ist das Grundkonzept von Forth, die Applikation als Erweiterung auf das Entwicklungssystem aufzusetzen. Dieses '*all in one*' führt selbst für ein einfaches 'Hello World' zu *fetten* Programmen von einigen zehn bis hundert Kilobyte Größe, die den gesamten Ballast des Entwicklungssystems mit sich herumschleppen. Schlimmer noch: Um so leistungsfähiger und komfortabler das Entwicklungssystem, desto größer die endgültige Applikation.

Turnkey Applikationen werden daher i.d.R. mit speziellen Forth-Werkzeugen (Metacompilern und Code-Strippern) produziert.

Nutzungsebenen

Die Arbeit mit Forth läßt sich in vier Ebenen mit steigender Komplexität einstufen:

- a) Interaktive Nutzung existierender Worte
- b) Kombination existierender Worte zu neuen Definitionen. Nutzung wie in a) beschrieben oder Compilation zu noch komplexeren Worten.
- c) Deklaration neuer Datentypen mit CREATE ... DOES>, d.h. einerseits Deklarationen, wie der Compiler die neuen Datenstrukturen aufbauen soll, und andererseits Festlegung, wie diese Datenstrukturen später benutzt werden.
- d) Erweiterung des Interpreters und Compilers selbst. Dazu gehören alle Arten von neuen Control-Strukturen, Erweiterung des Interpreters um neue Datentypen oder sogenannte *definer defining words*, die im DOES>-Teil weitere CREATE...DOES> Konstruktionen enthalten.

Diese in d) genannte Fähigkeit - Forth selbst zu erweitern - ist der eigentliche Schlüssel für die Mächtigkeit von Forth. Sie begründet sich gerade in dem oben beschriebenen '*alles in einem*' Konzept. Dieses erlaubt es, das Verhalten des Forth-Systems, auch seines Interpreters und Compilers, auf die gleiche Art und Weise, mit den selben Werkzeugen und zur selben Zeit zu ändern, in der er seine Applikation schreibt.

Und für jeden fortgeschrittenen Forth-Programmierer ist genau dies das eigentliche Element, was für ihn Forth zu *der Sprache* seiner Wahl macht.

Probleme des Anfängers

Für den Anfänger, der gerade eine neue Sprache - Forth - erlernen will, sieht das ganze allerdings anders aus.

Was ist das Hauptziel, wenn man eine neue Programmiersprache erlernt, vor allem wenn es die erste ist? Man will diese Sprache nutzen, um seine Probleme zu lösen.

In vielen Fällen führt dies zu kleinen eigenständigen Programmen mit einem wohl definierten Satz an Aufgaben. Die erfolgreiche Realisierung eines solchen kleinen Projektes ist eine starke Motivation für den Anfänger. Es gibt ihm einen starken Anreiz mehr über die Sprache zu lernen und weitere Projekte zu realisieren.

Auf dem Weg dahin stellen sich dem Anfänger mit Forth einige Probleme.

Ein erstes wurde oben schon erläutert. Es ist für den Anfänger nicht einzusehen (Zurecht, wie ich glaube!) das man so viele Kilobyte Programm benötigt um 'Hello world!' auszugeben oder eine LED an einem Printerport an- und auszuschalten. Bei der heutigen Softwaretechnologie kann das nicht normal sein. Und viele andere Sprachen zeigen, daß dies auch nicht so sein muß. Es war aber bisher auch kaum möglich, ihm ein Werkzeug in die Hand zu geben, mit dem er diese kleinen Applikationen erstellen kann, dabei - wie in Forth üblich - interaktiv bleibt und sich nicht mit den Abgründen von Metacompilation, Codestripping u.ä. herumschlagen muß.

Seine Probleme bei der Realisierung dieser Projekte sind auch nicht die klevere Nutzung des Forth Interpreters sondern solch simple Dinge wie UPN, verfügbarer Wortschatz, Control-Strukturen oder Stackbilanz der einzelnen Worte.

Anders als für uns erfahrene Programmierer ist das Wissen über Arbeitsweise des Interpreters und Compilers, seine Erweiterbarkeit oder seine geschickte Nutzung für den Anfänger relativ unwichtig und bringt eine scheinbar unnötige Komplexität in die Sprache ein. Was für uns ein wichtiger Aspekt von Forth ist, macht aus Sicht eines Anfängers das Erlernen der Sprache unnötig schwer.

Ein Anfänger benötigt eigentlich kein voll ausgebautes Forth-System mit all seinen Features (und hunderten oder tausenden von Worten). Er ist einfach nicht in der Lage dieses zu nutzen. Er wird im Gegenteil von der Vielfalt und scheinbaren Komplexität abgeschreckt. Dies wird auch durch die Arbeit der Moerser Gruppe bestätigt, die ein relativ einfaches Forth-System (ZF) verwenden.

Ein Anfänger benötigt stattdessen ein einfach zu nutzendes Werkzeug, mit dem er sich der Sprache nähern kann und dabei in der Lage ist, kleine, abgeschlossene Applikationen in kurzer Zeit zu erstellen, ohne sich um interne Details kümmern zu müssen.

Zwei Schritte zum Forth

Ich glaube daher, man sollte Forth in zwei Schritten lehren.

In einem ersten Schritt sollten elementare Kenntnisse der Forth-Programmierung erlernt werden. Dazu gehören:

- Variablen, Konstanten und Colon Definitionen
- Control-Strukturen
- UPN und Stack
- Integer Arithmetik
- sinnvolles Faktoring
- inkrementelle Programmierung und interaktives, schrittweises Debugging

Bei geeignetem Werkzeug ist der Anfänger damit in der Lage, nützliche standalone Applikationen zu schreiben.

Der zweite Schritt baut darauf auf und macht mit der Arbeitsweise und den vollen Fähigkeiten eines klassischen '*all in one*' Forth-Systems vertraut:

- Wie arbeitet das Forth-System?
- Verarbeitung des Eingabestrom
- Erweiterung des Interpreters
- Erweiterung des Compilers

Das entscheidende am ersten Schrit ist m.E., daß die anfangs bemängelte Komplexität der Nutzung für den Anfänger beseitigt ist. Er kann Forth fast wie jede andere Sprache erlernen.

Sicher wird es dann immer einen Teil geben, der auf dem Niveau des ersten Schrittes stehenbleibt und den zweiten nie absolviert. Aber ich glaube, auf die vorgeschlagene Weise erreichen wir, daß eine größere Anzahl von Leuten mit den elementaren Programmierkonzepten von Forth vertraut werden. Und damit erhöhen wir sowohl das Potential an guten Forth-Programmierern als auch Akzeptanz von Forth als Sprache überhaupt.

HOLON - Das ganz andere FORTH

von Friederich Prinz; F.PRINZ@MHB.GUN.DE
Hombergerstraße 335; 47443 Moers; 02841/58398

Haben Sie Spaß am Spielen und Experimentieren? Suchen Sie nach einem FORTH zum Lernen und Lehren? Sind Ihnen konventionelle FORTH-Systeme zu unübersichtlich? Benötigen Sie ein professionelles Werkzeug zur Programmentwicklung unter DOS? Ärgern Sie sich schon lange darüber, daß mit konventionellen FORTH-Systemen "HALLO WELT" als TURNKEY-Applikation so um die 70 KByte groß wird? Oder wollen Sie einfach einmal einen etwas anderen Weg bei der Programmentwicklung und Pflege gehen? Dann ist HOLON vermutlich genau das Richtige für Sie!

Stichworte: Holon Forthtagung'96

Als Wolf Wejgaard auf der Jahrestagung der Forthgesellschaft in Mittweida (April 1996) seine "Vorstellung vom idealen Forth Entwicklungswerkzeug" vorstellte, reichten die Reaktionen der Zuhörer von leicht erschrecktem Unwillen über neugieriges Erstaunen bis zur fröhlichen Begeisterung über die Möglichkeiten, die sein HOLON bietet. Immerhin ist ein "auf zwei Rechner verteiltes" FORTH zumindest sehr gewöhnungsbedürftig. Aber mitzuerleben, wie sich jemand mit einer (bewußt) fehlerhaften, laufenden Applikation abmüht, die während ihrer Laufzeit vom Entwickler 'repariert' wird, hat schon einen ganz besonderen Reiz...

Um das vorweg zu nehmen: HOLON ist ein professionelles und kommerzielles Werkzeug zur Programmentwicklung für unterschiedliche Zielprozessoren, insbesondere zur Entwicklung von Codes für "embedded systems". Aber wie jedes andere gut durchdachte Werkzeug ist HOLON selbstverständlich sehr viel ' universeller' einsetzbar. Und was den kommerziellen Status von HOLON angeht, so hat Wolf Wejgaard auf der Weihnachts-CD der Forthgesellschaft eine LITE-Version von HOLON zum Experimentieren und Kennenlernen zur Verfügung gestellt.

Gleichmäßig verteilt ...

Für jeden Programmierer ist es wohl zunächst sehr ungewöhnlich, mit einem "verteilten System" zu arbeiten. HOLON besteht aus einem HOST und einem TARGET, die gleichzeitig arbeiten müssen. Das heißt im Normalfall, daß zwei Rechner mit einem seriellen Kabel (RS 232) über einen der COM-Ports miteinander verbunden sein müssen. Auf einem der beiden Rechner wird ein sogenannter MONITOR gestartet, der das Targetsystem steuert und verwaltet. Auf dem anderen Rechner arbeitet HOLON als der Host des Entwicklers. Auf dem Host findet alle Programmierarbeit statt, deren Ergebnisse vom Monitor "ausgeführt" werden.

Fortsetzung von der letzten Seite

Das 'einfache Werkzeug'

Bliebe nur noch das oben beschriebene Werkzeug zu finden, welches eine einfache Handhabung erlaubt, interaktiv arbeitet und kleine standalone Applikationen erzeugen kann.

Ich habe Wolf Wejgaards Holon-System als ein solches Systrem kennen- und schätzen gelernt. Holon ist ein einfach zu benutzender Metacompiler, der eine interaktive Arbeitsweise bei der Codeerzeugung erlaubt und die Hürden der Metacompilation vor dem Anwender gut verbirgt.

In der Variante als Codegenerator für DOS-Applikationen, als Holon-LT frei verfügbar, ermöglicht es einen einfachen Einstieg in die Grundlagen der Forthprogrammierung.

Wie alle Metacompiler macht auch Holon einige Einschränkungen in der Freiheit der Programmierung. Diese sind allerdings recht gut verborgen bzw. betreffen die im ersten Schritt erlernten Konzepte nicht.

Die geänderte Arbeitsweise in Holon, der Quellcode in einer Datenbank - der zugehörige Browser als zentrales Element der Bedienoberfläche, ist für den 'erfahrenen Forth-Programmierer' nur anfangs ungewohnt.

Ich denke es lohnt, sich ernsthaft mit diesem System auseinanderzusetzen.

❑

Selbstverständlich geht es auch anders. HOST und MONITOR können auf einem Rechner laufen, wenn die entsprechenden Voraussetzungen für die Kommunikation zwischen beiden Systemen geschaffen werden. Und diese Voraussetzungen bieten WINDOWS und/oder OS/2. Host und Monitor werden einfach jeweils in einer separaten DOS-Session aufgerufen. Die unter Windows notwendigen Einstellungen der Sessions beschreibt die Datei README. Unter OS/2 genügt es, diese Sessions zu starten. Alle notwendigen Einstellungen - das Multitasking betreffend - werden von OS/2 implizit vorgenommen. Das Umschalten zwischen Host und Monitor ist danach aus Sicht des Entwicklers nicht mehr, als das Umschalten zwischen Entwicklungs- und Applikationsfenster in anderen Umgebungen.

Der Unterschied zwischen den beiden “Vorgehensweisen”, bzw. zwischen dem “Zwei Rechner System” und einer “ Ein Rechner Multitasking Umgebung” besteht technisch in der Wahl des jeweiligen Monitors. HOLON_LT stellt beide Monitore zur Verfügung, sodaß dem professionell interessierten FORTHer alle für ihn relevanten Testmöglichkeiten offen stehen - und dem Spieltrieb des Hobbyisten keine Grenzen gesetzt werden.

Strukturierte Wortfülle

Der erste Aufruf des Host bringt sofort eine angenehme Überraschung mit sich. Hier muß das Genie nicht mehr das Chaos beherrschen - und wertvolle Ressourcen an das Kennenlernen einiger Tausend Worte verschwenden. Das heißt nicht, daß HOLON nur wenige Worte kennt und womöglich nur über eine geringe “Funktionalität” verfügt. Tatsächlich umfaßt HOLONs Wortschatz mehr als 3.000 Worte. Aber die sind “wohlgeordnet”!

fehlt: Screendump!

ZFs “Wortflut” von mehr als 1.200 Worten haben wir in Moers bereits vor einigen Jahren durch die Einführung von Kategorien und zugehörigen Worten einzudämmen versucht. Durch die “Ausgrenzung” von Worten die z.B. nur vom ZF selbst benutzt werden, konnten wir eine Ordnung in die Vokabulare bringen, die nicht nur Anfängern die Arbeit mit dem ZF ganz wesentlich erleichtert. HOLON geht noch einen Schritt weiter - und kommt ohne jede Ausgrenzung aus!

In drei Spalten werden die bereits vorhandenenen Definitionen, ebenso wie alle dazukommenden, zunächst auf MODULS aufgeteilt. In der Modulspalte finden sich Begriffe wie BASIS, FORTH, DOS&EXT, TASKING usw. Applikationen beginnen mit einem neuen Moduleintrag, z.B. HALLO.

In der Spalte GROUPS werden die Module sinnfällig in Gruppen unterteilt. Für das Modul FORTH existieren Gruppen wie 16-Bit Arithmetik, 32-Bit Arithmetik, Characters, Strings, Logic, Datatypes, FormattedOutput usw.. Und erst diese Gruppen enthalten die einzelnen Worte, bzw. Definitionen - so wie sie in anderen FORTH-Systemen gnadenlos ungeordnet nach dem Aufruf von WORDS über den Bildschirm huschen.

Zwischen Modulen, Gruppen und Worten schaltet der Entwickler mit den Cursortasten hin und her und kann sich auf einfache und effiziente Weise gezielt beim System nach einzelnen Definitionen erkundigen. Welches Wort in welcher Gruppe und in welchem Modul gerade in dem darunter befindlichen Editfenster angezeigt wird, läßt sich einfach an der andersfarbigen Ausgabe (gelb statt weiß) von Wort, Gruppe und Modul ablesen. Selbstverständlich stellt HOLON dabei alle seine Definitionen bis in den Kern hinein zur Verfügung, so daß auch dem Wißbegierigsten keine Fragen offen bleiben. Und selbstverständlich ist Alles zu jeder Zeit veränderbar. Metakompilation ist eine “natürliche Fähigkeit” von HOLON!

Editfenster und Kommentarfenster sind zwei voneinander getrennte Bereiche auf HOLONs Oberfläche. Dabei kann man den schmalen, für Kommentare vorgesehenen Bereich aber eigentlich nicht als “Fenster” bezeichnen, bestenfalls als “Oberlicht”. Die Trennung von Kommentar und Source ist anfangs gewöhnungsbedürftig, so wie das ganze HOLON. Aber der für zwei Textzeilen gedachte Kommentarbereich reicht völlig aus. Tatsächlich lassen sich auch im Source der jeweiligen Definition die bekannten Kommentarzeichen verwenden. Man wird sie aber weniger nutzen. In HOLON gilt die Regel, ein Wort = ein “Fenster”. Editfenster und Kommentarfenster nehmen jeweils maximal 1024 Zeichen auf. Damit wird der Programmierer gezwungen, wieder kleine, modulare Definitionen zu schreiben. Und solch kleine Definitionen kommen erfahrungsgemäß mit wenigen Kommentaren aus. Immerhin wird sich hier mancher, an die sequentiellen Files von ZF und FP-C gewöhnte Programmierer wieder gehörig umstellen müssen. Die Belohnung in dieser Umgewöhnung liegt in einer vielfach schon verloren gegangenen Übersichtlichkeit...

Ganz direkt!

Aber nicht nur der angenehme Eindruck von “Aufgeräumtheit” und Übersichtlichkeit ist neu im HOLON. Ganz sicher vermißt man auch das gewohnte “OK” konventioneller FORTH-Ober-flächen. In HOLON ist eben

der gesamte “Arbeitsablauf” deutlich anders als man es kennt. Die gerade entwickelten Definitionen werden nicht - wie z.B. beim ZF - auf dem System selbst ausgetestet und ausgeführt. Statt dessen wird das gerade in Arbeit befindliche Modul in den Monitor “down-geladen” - und von diesem ausgeführt. Dort lassen sich alle Definitionen in ihrer Arbeitsweise so beobachten, wie man es sich direkter gar nicht wünschen kann - in ihrer geplanten Arbeitsumgebung. Hat man sich einmal mit dieser Arbeitsweise “abgefunden”, wird die Option, in einem “TEST” mit der gewohnten Interpreteroberfläche zu arbeiten, zu einem ungenutzten Angebot.

In HOLON ist nicht die Interpreteroberfläche das zentrale Element der Entwicklungsumgebung, sondern der Editor. Wolf Wejgaard weist hierzu darauf hin, daß sich alle bekannten, konventionellen FORTH-Systeme seit Charles Moore nicht wesentlich verändert haben. Der äußere Interpreter verarbeitet den Eingabestrom, interpretiert oder ruft den Compiler zur Arbeit und nutzt den Editor als “separates” Werkzeug. HOLON nutzt den Editor als zentrales Element der gesamten Ent-wicklungsumgebung. Der Editor startet den Compiler. Und der Compiler ist nur noch mit seiner eigentlichen Aufgabe beschäftigt - mit dem Übersetzen der FORTH-Codes. Das “Einordnen” der Definitionen in Vokabulare, die es in dieser Form im HOLON gar nicht gibt, übernimmt der Editor!

HOLONs Definitionen werden tatsächlich nicht in Vokabularen abgelegt, sondern in einer Datenbank! Auch das ist neu, und ungewohnt, und bringt einen Vorteil mit sich, den man erst dann richtig zu schätzen weiß, wenn man sich einmal damit abquälen mußte, eine beliebige Applikation von soviel “Overhead” wie möglich zu befreien. HOLONs Funktion TURNKEY verwandelt die gewünschte Applikation in eine Datei mit dem Namen TURNKEY.EXE. TURNKEY.EXE enthält nur noch die zur Arbeit der Applikation notwendigen Codes, ohne jeden Overhaed an Definitionsköpfen. “Hallo Welt” wird damit zu einem 1.752 Byte kleinen Executable! Alle Diskussionen um einen Codestripper erübrigen sich damit ebenso, wie auch alle TCOMs.

Fazit

Ob HOLONs Konzepte einen Ausblick auf die Zukunft erfolgreicher FORTH-Systeme geben, müssen letztlich die Anwender entscheiden, die HOLON als eine echte Alternative zu konventionellen Systemen annehmen, oder nicht. Stark gewöhnungsbedürftig is HOLON allemal. Seine Konzepte sind zu “fremdartig”, als daß man sie “mal eben so” vereinnahmen könnte. Das Arbeiten mit HOLON gestaltet sich so ganz anders, als Alles, was man bisher kennt, daß mancher Interessent sich überlegen wird, ob er die Mühen des Umlernens auf sich nehmen will. Vor allem das Konzept des “verteilten Systems” aus Host und Monitor und die damit verbundenen “Umschaltungen” zwischen Entwicklungs- und Ausgabebildschirm wird nicht Jeder annehmen wollen.

Wolf Wejgaard sagt von seinem FORTH, es sei ein System, daß zum Arbeiten entwickelt wurde. Damit betont er den Werkzeugcharakter, den HOLON in hohem Maße hat. Ich sehe in HOLON darüber hinaus ein FORTH, daß ausgezeichnet zum Lernen und Lehren geeignet ist. Die schon gerühmte Ordnung auf der Oberfläche, die Übersichtlichkeit über das Gesamtsystem und über jede einzelne Definition, die Möglichkeit, das System bis in dern Kern hinein zu verändern und zu testen, und dabei sowohl über FORTH, als auch über die jeweilige Maschine und das Betriebsystem alles zu lernen, was es zu lernen gibt, faszinieren mich. Daß HOLON “fast nebenher” noch ohne weiteres Zutun Executables kompilieren kann, die für die allermeisten FORTH-Systeme geradezu phantastischsind, sollte eigentlich ausnahmslos jeden FORTHer dazu bewegen sich

Entwicklung:

HOLON wird entwickelt und vertrieben von
Dr. Wolf Wejgaard, Forth Engineering,
Neuhoeflirain 10, CH-6045 Meggen, Schweiz.
Tel. +41-41-377 3774 Fax +41-41-377 4774
Email: 100012.201@compuserve.com

Systemvoraussetzungen:

Das Entwicklungssystem HOLON laeuft unter DOS.

Versionen / Preise:

HOLON-86 fuer DOS Zielprogramme.
Es handelt sich um ein ungeheuer effektives Entwicklungstool, das wir auch intern zur Entwicklung von industriellen Forth-Applikationen einsetzen. Das Zielprogramm kann interaktiv und inkremental wahlweise in einem separaten DOS-Fenster unter MS Windows und OS/2 oder in einem separaten (z.B. eingebetteten) PC gebildet werden. Preis DM 450.-

HOLON-11 fuer 68HC11 Zielsysteme.
Wird vorerst als Schulversion fuer den Mikroprozessor-Unterricht mit einer funktionsreichen Experimentierplatine geliefert. Eine Entwicklerversion ist in Arbeit. Preise auf Anfrage.

HOLON-LT (Lite) ist eine Freeware Version fuer DOS Zielprogramme, mit nur wenig eingeschraenkter Funktionalitaet (kein Export von Modulen). Sie können hierin die neue Forth-Entwicklungsmethodik HOLON ausprobieren und auch echte (Turnkey-) Applikationen bilden. HOLON-LT ist u.a. erhältlich in der Kieler Mailbox der FG und auf dem Netz u.a. bei ftp://taygeta.com/pub/Forth/Applications/holon.zip sowie auf der Weihnachts-CD`95 des Forthmagazins (Vertrieb Klaus Kohl, s. hintere Umschlagseite).

Weiterführende Literatur:

Wolf Wejgaard, "Not Screens Nor Files But Words", FORML Proceedings 1989, S. 425-430

Wolf Wejgaard, "The Beauty of Separate Systems", FORML Proceedings 1990, S. 269-273

Ein zusammenfassender Artikel (post festum):

Wolf Wejgaard, "A Taste of Direct Programming", EuroFORTH Proceedings 1994, S. 147-150

❑

C auf Stackprozessoren

von Rafael Deliano
Steinbergstr. 37; D-82110 Germering

Sicherlich ist dies in der VD kein populäres Thema. Dafür aber eine Thema, das für die Verbreitung von Stackprozessoren von entscheidener Bedeutung ist. Beim Prozessor MARC4 fehlt der C-Compiler nicht so stark. Am unteren Ende der Preisskala ist es der Anwender gewohnt, in Assembler zu programmieren. Wenn er alternativ dazu irgendeine geeignete Hochsprache bekommt, ist er zufrieden. Spätestens bei 16 Bit-Controllern will man aber in jedem Fall in Hochsprache programmieren können. Und 80 % der Anwender wollen C.

Stichworte: Stackprozessoren C

Halbleiterhersteller meiden Chips die nicht auch C können. Dementsprechend zäh verbreiten sich Stackprozessoren. Und mangels Stackprozessoren verbreitet sich FORTH auch zäh.

Frühe Versuche

Die Frage, ob Stackprozessoren überhaupt C effektiv verarbeiten können, ist weitgehend offen. Es wurden zwar schon diverse derartige C-Compiler angekündigt, aber keiner kam bisher wirklich auf den Markt. In [1] wird die Implementierung von Small-C auf dem NC4016 beschrieben. Small-C wurde als Delta-C von Silicon Composers und Novix Small-C von Novix verfügbar. Dabei wurde aus C-Source FORTH-Source erzeugt die dann von FORTH compiliert wird. Stackframes werden hier über Pseudoregister im RAM angesprochen. Der Framepointer lag auf dem Returnstack. Novix behauptete vom NC6000, er könne native-code FORTH mit 14 MIPS, threaded-code FORTH mit 5 MIPS und C mit 4 MIPS. Der NC6000 hätte sich für C besser geeignet als der NC4016, weil er als zusätzliches Register einen Pointer ins RAM für lokale Variablen hatte.

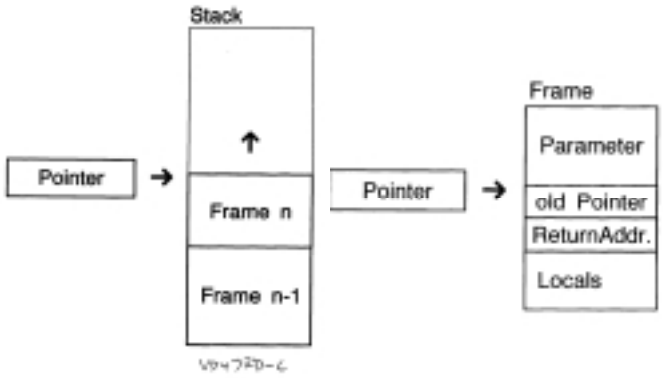
Für den RTX2000 wurde MPE von Harris beauftragt einen C-Compiler zu entwickeln. Er war noch nicht völlig fertig als Harris das Handtuch warf. Gedieh aber bis zu einer Version für 64k Code/64k Daten-Segment, die anscheinend immer noch von MPE erhältlich ist. Auch für den IX1 wurde wieder ein C-Compiler angekündigt. Im IX1 dürften die X- und Y-Register nützlich sein Stackframes im DatenRAM anzulegen.

Stackframes

Wo liegen die Probleme? C ist eine genauso primitive Hochsprache wie FORTH. Es ist oft als “souped up PDP11-assembler” verspottet worden. Es sind also nicht wesentlich leistungsfähigere Grundfunktionen zu verwirklichen. Der am meisten zitierte Unterschied besteht in den Stackframes die C genauso wie andere konventionelle Sprachen (Pascal usw.) zur Übergabe von Parametern auf Unterprogramme verwendet. Dabei werden bei jedem Unterprogrammaufruf auf einem Stack Blöcke von RAM zugeteilt (Bild 1). Diese Blöcke haben, abhängig vom Unterprogramm, variable Länge. In Small-C für den 8086 werden hier untergebracht (Bild 2):

- die Parameter
- die Rücksprungadresse des Unterprogramms
- der Zeiger des letzten Framepointers
- ggf. noch etwas Speicher für lokale Variable

Das Unterprogramm greift über den Pointer mit negativem Offset auf die lokalen Variablen und mit positivem Offset auf die Parameter zu. Wie man den Stackframe aufbaut, bleibt



dem Implementierer freigestellt. Auf konventionellen CPUs wird der Stack im RAM aufgebaut und es wird munter zwischen RAM und Register umgeladen. Das kann man auf einem Stackprozessor auch, wenn er wenigstens noch ein Indexregister hat, das den Zugriff aufs RAM erleichtert.

Probleme

Effizienter wäre es natürlich, den Datenstack direkt zu benutzen. Hier ergibt sich das erste Problem: Ist der

STOIC - Stack Oriented Interactive Compiler

von Rafael Deliano
Steinbergstr. 37; D-82110 Germering

Der Mann hinter STOIC ist Jonathan Sachs. Wer von Sachs noch nie was gehört hat, kennt möglicherweise Lotus 1-2-3. Das Spreadsheet stammt auch von ihm.

Stichworte: STOIC, Jonathan-Sachs, MIT

STOIC wurde von Sachs am MIT entwickelt. Er war dort insgesamt 14 Jahre als Student und später wissenschaftlicher Mitarbeiter. Erste Aufgaben waren Auswertung von Satellitendaten am Center for Space Research. Mitte der 60er Jahre programmiert man noch FORTRAN-Batch. D.h. man gab sein Programm ans Rechenzentrum weiter. Auf Lochkarten. Dort wurde es abgearbeitet. Und nach einigen Stunden kam das Ergebnis zurück. Sachs war mit diesen Arbeitsbedingungen genauso unzufrieden wie Moore und alle anderen.

Der nächste Job war eine Lesemaschine für Blinde in der Cognitive Information Processing Group. Dort programmierte er die Buchstabenerkennung. Hier konnte er bereits in einer interaktiven Umgebung über Terminal auf einer PDP-9 arbeiten.

Das Biomedical Engineering Center schließlich wollte basierend auf 8080 medizinische Geräte entwickeln und braucht Software, die eine vernünftige Programmentwicklung auf derart beschränkter Hardware ermöglichte. Sachs: “So I wrote a language called STOIC, which is a variant of FORTH” [1].

Da mit Regierungsgeldern entwickelt, liegen die Rechte nicht bei Sachs, sondern bei der Hochschule. Diese übergab das System in die PublicDomain. STOIC war 1980 vom Biomedical Engineering Center auf Magnetband erhältlich. Die 8080-Implementierung, es gab und gibt anscheinend nur diese, war auf North Star-CP/M-Systemen lauffähig und belegte dort 14k-Byte Speicher. Eine kurze Beschreibung findet sich in [3]. Der wesentliche Unterschied zu FORTH sind die reichhaltigeren Datentypen. Es gab auch Strings, Arrays, Float (4,5 Digits), und komplexe Zahlen. Als Erweiterungen waren zudem FFT und Grafik vorhanden. Das Magnetband enthielt die ASCII-Source zwar im Blockformat (20 Zeilen/128 Zeichen). Mittels der Strings waren aber auch Textfiles mit Namen implementiert. Was die Einbindung in ein fremdes OS natürlich beträchtlich erleichterte. Durch diese Features war STOIC für viele Anwender um 1980 anziehender als fig-FORTH. Was seine zeitweise Verbreitung erklären dürfte. Genau wie bei fig-FORTH gab es auch rip-off-artists, die die Kommerzialisierung versuchten [3], [4]. Im Gegensatz zu FORTH aber scheiterten. Wodurch STOIC ein reines PD-Produkt blieb, nicht weiterentwickelt wurde und im Mülleimer der Geschichte verschwand.

Um den üblichen Gerüchten vorzubeugen: 1-2-3 wurde nicht in einer Variante von FORTH programmiert, sondern schon damals in C. Das hat Lotus die Portierung erleichtert und zu seinem Erfolg beigetragen. In einigen der komplexeren technischen Aspekte von 1-2-3, dem effizienten Neuberechnen der Tabellen, sind Erfahrungen aus der garbage-collection von LISP eingeflossen.

[1] Susan Lammers “Programmers at Work” Microsoft Press 1986

[2] Jonathan Sachs “An introduction to STOIC” Technical Report Harvard-MIT, 1976/1978, 72 Seiten

[3] Dr.Dobbs Feb 1980 S. 42

[4] Dr.Dobbs Nov/Dez 1979 S.20

Fortsetzung von der letzten Seite

Datenstack tief genug? Beim Novix ja. Da war der Datenstack in externem RAM untergebracht und 256 Worte tief. Bei Singlechip-Stackprozessoren ist er aber meist nur 16 Werte tief. Und natürlich die zweite Frage: Liegen die Daten immer an den 3 obersten Stackpositionen, wenn man sie braucht? Auch dies ist nicht sicher. Wie der “Byte”-Kolumnist Jerry Pournelle einmal schön gesagt hat: ”...FORTH is a kind of assembly language that uses the programmer as a preprocessor.” Der FORTH-Programmierer feilt am Programm zwangsläufig solange, bis beide obigen Bedingungen eingehalten sind. Ob ein Compiler vorhande C-Source gezielt so umformulieren kann, daß beide Bedingungen eingehalten werden, ist nicht sicher. Phil Koopman ist optimistisch. Er ist der Meinung, es sei einfach noch nicht genug Arbeit in optimierende Compiler für Stackprozessoren gesteckt worden. Er arbeitet deshalb immer noch an einem optimierenden GNU C-Compiler.

Eine nüchterne Ansicht dürfte wohl sein, daß man in der Hardware des Stackprozessors vorsorgen muß, wenn man C haben will. Z.B. Stacks die ins externe RAM überlaufen wie beim SC-32. Und tief greifende Stackbefehle wie PICK und ROLL. Dann kann der Compiler jedenfalls immer lauffähigen Code erzeugen. Selbst wenn dieser nicht optimal ist.

[1] D.L.Miller “Stack Machines and Compiler Design” Byte 4/87 S.177

Forth International

von Fred Behringer

Antwort aus England

Von der FIGUK, der englischen Forth-Gesellschaft, habe ich jetzt ein Exemplar der FORTHWRITE zugeschickt bekommen (siehe Inhaltsbesprechung). Sie besteht tatsächlich, wie schon früher berichtet, aus 30 Seiten im Format DIN-A5. Allerdings kommt dieses Format dadurch zustande, daß die Autoren-Vorlagen, die im Format DIN-A4 einzureichen sind, photomechanisch verkleinert werden. Und von "9-Nadel-Druckern" (siehe frühere Wiedergabe eines Berichtes aus Holland) meiner Meinung nach wenig zu sehen. Normale, gut leserliche Schriften, so wie sie die einzelnen Autoren zur Verfügung haben. Ich hoffe, den Kontakt zu unseren englischen Freunden intensivieren zu können.

Die Adresse des Redakteurs lautet: Gil Filbey; 21 Ferncroft Avenue London NW3 7PG .

Fred Behringer

Gehaltvolles

zusammengestellt und übertragen von Fred Behringer

FORTHWRITE der FIGUK (GB)

August 1995

2 Editorial
Gil Filbey

Tip für Anfänger: ?IM <wort> liefert "YES" oder "NO" in Abhängigkeit davon, ob <wort> IMMEDIATE ist oder nicht.

```
: ?IM [COMPILE] ' >NAME 1 - C@ $C0 =
  IF ." YES" ELSE ." NO" THEN ;
```

4 Stack Manipulation
Chris Jakeman

In comp.lang.forth im INTERNET liegen über 100 Beiträge zum Thema "Stackmanipulation". Der Autor bespricht eine Auswahl davon.

21 ANS Forth, Linear Search etc.
Chris Jakeman

Der Autor berichtet unter anderem, daß sein "Educational Forth" (TForth) jetzt fertig ist (Version 1.00).

23 Limit Variables
Chris Jakeman

Der Autor sagt: "Forth-Programmierer haben unbeschränkten Zugang zu ihren Programmen und ihrem Computer. Es gibt jedoch Momente, in denen soviel Freiheit hinderlich ist. Als ich kürzlich Schwierigkeiten bei der Fehlersuche in einem Programm mit vielen Zeigern hatte, gab ich auf und erfand ein neues Wort, LimVar, das dann die Arbeit für mich tat."

29 Honeywell Forth Board
Peter Rush

Ankündigung eines Forth-Informationsdienstes der Firma Honeywell in Bracknell, England. Zugang per Modem. ❑

Forth Dimensions der Forth Interest Group, USA

November/Dezember 1995

6 Forth in the HP100LX M.
Edward Borasky

Die HP-Computer 100LX und 200LX passen wirklich in jede Jackentasche und wiegen nur 300 Gramm. Aber fügen Sie Forth hinzu, und diese Federgewichte werden zum schweren Geschütz. Für 80186/DOS-5.0-Umgebungen sind derzeit eine ganze Reihe von Forth-Paketen erhältlich. Mit Hilfe von Benchmark-Tests und anderem, was der Autor in diesem Artikel beschreibt, können Sie Forth getrost in die Tasche stecken und brauchen nie mehr ohne Forth aus dem Hause zu gehen.

13 Hashing Forth
Xan Gregg

Hashing ist ein Thema, über das derart leicht dahergeplaudert wird, daß sich Neulinge scheuen zu fragen, wie das denn eigentlich geht. Hashing liefert eine schnelle Methode, unsortierte Daten zu durchsuchen - auf Kosten von Speicherplatz. Robert Sedgewick beschreibt Hashing als "Direkter Bezug auf Dateneinheiten in einer Tabelle mit Hilfe von arithmetischen Transformationen an Schlüsselwörtern nach Tabellenadressen." Lesen Sie diesen Artikel und Sie wissen, was gemeint ist ...

17 OOP, Forth, and the Future
Ronald T. Kneusel

Forth ist drauf und dran, die Gelegenheit seines Lebens zu verpassen -wenn es sich nicht ändert. Was man auch immer C++ an Negativem nachsagen kann, C++ wird wegen seiner objektorientierten Fähigkeiten für das mächtigere Werkzeug gehalten. Und die Leute verlangen nach immer mächtigeren Anwendungsprogrammen. Das Paradigma des objektorientierten Programmierens macht das Rennen und C++ ist sein Vorreiter. Aber muß das so bleiben ?

19 RETRY, EXIT, and Word-Level Factoring
Richard Astle

RETRY ist nicht nur einfach eine Marotte irgendeines Forth-Hackers. Der Autor stieß vor etwa 10 Jahren auf RETRY und hat es seither zu schätzen gelernt - mehr noch als RETRYs wohlbekannte Vettern REPEAT, UNTIL und AGAIN. Im Verein mit EXIT ermöglicht RETRY den Aufbau flexibler Ablaufstrukturen, die das Wort als Einheit verwenden.

22 Making Forth Professional
Peter Knaggs

Von Leuten, die sich mit dem Verkauf von Forth-Produkten beschäftigen, hört man immer wieder, daß frühe, vielleicht nicht ganz gelungene Public-Domain-Versuche der Forth-Implementation potentielle Kunden abschrecken. Und Software-Manager sind nur auf Code fixiert. Sie begreifen nicht, daß Forth nicht nur einfach eine andere Programmiersprache ist, sondern daß eine ganz bestimmte Philosophie dahintersteckt. Viele Anliegen der Programmierpraxis der heutigen Zeit, strukturierte Programmierung, Wiederverwendbarkeit, Blibliotheken usw., stehen in Forth schon seit Jahren zur Verfügung und werden dort auch verwendet. Es wird ein Weg aufgezeigt, wie Forth ein Quäntchen seiner wohlverdienten Anerkennung zurückgewinnen kann.

26 Nanocomputer Optimizing Target Compiler: the PIC Library
Jim Hendtlass

Wie im vorausgegangen Teil dieses Artikels (FD XVII/3) angekündigt, wird hier nun eine Bibliothek für die Prozessoren PIC16C71 und PIC16C84 nachgeliefert. In Zusammenarbeit mit dieser Bibliothek akzeptiert der prozessorunabhängige Kern Forth-Eingaben und erzeugt Maschinencode für den PIC. Der Compiler ist für Chips von unterschiedlicher Wortlänge und Architektur verwendbar. Nur die Bibliothek ist von Chip

Forth-Gruppen regional

Berlin
Claus Vogt
Tel.: +30 -782 81 79 p
Treffen: nach Absprache

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
plewe@mpi-dortmund.mpg.de

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

Jörg Staben
Tel.: +2103 -24 06 09 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
Mo.+Di. 20:00 - 22:00

32FORTH (Atari)
Rainer Aumiller
Tel.: +89 -6 70 83 55 gp

FORTHchips (FRP1600, RTX, Novix...)
Klaus Schleisiek-Kern
Tel.: +40 -330 674 p

F-PC & TCOM, ASYST (Meßtechnik), embedded controller(H8/5xx// TDS2020, 8051 ... eFORTH...), FUZZY
Arndt Klingelnberg
Tel.: +2404 -6 16 48 agp

Gleitkomma-Arithmetik
Andreas Döring
Tel.: +721 -59 39 35 p

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

KI (Künstliche Intelligenz), OOF (Object Oriented Forth)
Ulrich Hoffmann
Tel.: +4351 - 712 217 pf

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 98 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