2/1994

VIERTE DIMENSION

10. Jahrgang 1994 DM 10,-

Der neue Vertrieb!

ECHT MESSE ZEITMESSUNG auf dem PC

DFÜ

WordWriter '94

Der Dallas DS80C320

OObigFORTH

und, und, und,...

FORTH MAGAZIN
Organ der Forth Gesellschaft e.V.

ultra- und volksFORTH (Diskettenformat)	**Handbuch**		**Disketten**		**Komplettpaket**	
ultraFORTH 3.8 für Commodore C64 (2*5¼"; beidseitig)	C64uF-H	40,00	C64uF-D	25,00	C64uF	55,00
volksFORTH 3.8 für Schneider CPC (2*3" für CP/M)	CPCvF-H	55,00	CPCvF-D	25,00	CPCvF-HD	70,00
volksFORTH 3.80 für Atari ST (3*360K; 3.5")	STvFH	65,00	STvF-D	35,00	STvF-HD	75,00
volksFORTH 3.81.41 für PC (1*360K; 5¼")	PCvF-H	65,00	PCvF-D	10,00	PCvF-HD	65,00
8087-Floatingpointpacket für PC-volksFORTH (nur Diskette: 1*360K; 5¼")					PCFloat	10,00

KK-FORTH (Alle Versionen mit PC-Terminalprogramm und Tools)	**Ergänzung**		**Komplettpaket**	
Allgemeines Handbuch zum KK-FORTH (für alle Versionen, ist im Komplettpaket enthalten)			KKFHB	65,00
KKF V1.2/1 für PC (Zusatzbeschreibung und 2*360K; 5¼")	KKFPC-E	70,00	KKFPC	110,00
KKF V1.2/0 für PC-Einplatinencomputer (V20; SIO0, 32K EPROM; 32K RAM)	KKFV20-E	70,00	KKFV20	110,00
KKF V1.2/0 für EMUF Z80mini3 (84C015; 32K EPROM; 32K RAM)	KKF8415E	70,00	KKF8415	110,00
KKF V1.2/0 für RTX-2000/1a (RTX-Board der FG-e.V. und RTX_EMUF)	KKFRTX-E	70,00	KKFRTX	110,00

Sonstiges		
Mikroprozessor Super8 mit ROM-FORTH (8KByte)	S8Chip	46,00
Handbuch zum Super8-FORTH mit Diskette (PC; 360K; 5¼")	S8HD	34,50
Leerplatine, Bausatzbeschreibung und Bauteile für S8-Bausatz (nur GAL, Quarz und SMD´s)	S8B	57,50
Komplettes Super8-System nach Bausatz aus VD	S8Sys	228,00
FIFTH 2.0 (PC-Diskette 1*360K; 5¼")	FIFTH	10,00
FPC V3.56 (PC-Diskette 4*1.2M; 5¼")	FPC356	25,00
Sourcen zur Vierten Dimension (PC-Diskette; bitte Ausgabe angeben: z.B. 03/94 -> xxxx=0394)	VD-xxxx	10,00
Sourcen zu EFORTH V1.0 für 8098, PC, 68HC11 oder 8051 (PC-Diskette; z.B 8096 -> xx=96)	EF-xx	10,00
DPANS6-Dokumentation	DPANS6	40,00

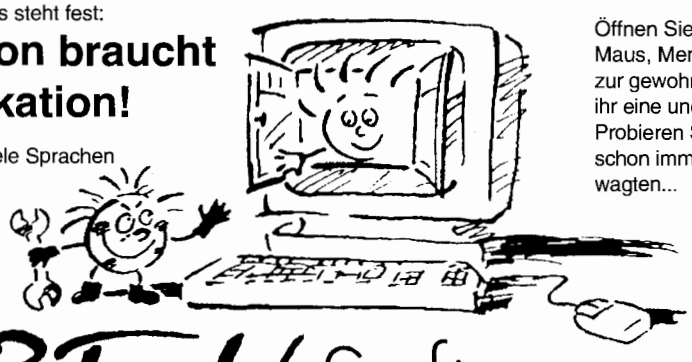
Weitere PD-FORTH-Versionen, Literatur oder andere Diskettenformate auf Anfrage
Alle Preise in DM inklusive 15% Mehrwertsteuer
Preisnachlaß für FG-Mitglieder: 10%
Verpackung, Versandt und Nachnahme: 9,00 DM

Postfach 1173
86406 Mering
Tel. 08233/30524
Fax 08233/9971

Kennen Sie einen widerspenstigen Mikrocontroller?...
Haben Sie sich schon mal mit einem unterhalten?...
Wie dem auch sei, eins steht fest:

Kooperation braucht Kommunikation!

comFORTH spricht viele Sprachen
Intel '86, '51, '96
Motorola 6800,
68HC11,
Zilog Z8, Z80, Z8000
... und lernt gern dazu



FORTecH Software

FORTecH-Software GmbH, Joachim-Jungius-Str. 9, 18059 Rostock, ((0381) 4 05 94 72 oder 71 (fax)

com FORTH für Windows

Öffnen Sie Ihr Fenster - frischer Wind tut gut.
Maus, Menüs und Knöpfchen sind kein Widerspruch zur gewohnten Kommandozeile, sondern verschaffen ihr eine unerwartet Renaissance.
Probieren Sie aus, was Sie bei geöffnetem Fenster schon immer ausprobieren wollten, aber nie zu tun wagten...

Kunden sind mehr als Käufer

Das FORTecH-Team unterstützt Sie durch
- individuelle Beratung und gemeinsame Analyse Ihrer Vorhaben
- Übernahme hardware- bzw. systemnaher
- Software-Entwichlungsleistungen
- Schulung Ihrer Mitarbeiter

-> Sprechen Sie mit uns über Ihre Projekte

Die Inserenten:
VolksFORTH Vertrieb, Klaus Kohl, **Seite 2**
FORTeCH, Software, **Seite 2**
FORTH-Systeme GmbH, **Rückseite**
Holger Dyja, **Seite 22**
Weitere Dienstleistungsanbieter und Produkte finden Sie auf **Seite 17 und 26**

Das Gruppenbild der Forthtagung in Malente

d.c.l.f

von Rafael Deliano

Die Diskussion seit der letzten VD von MAIL 1432 bis 1539 zusammengefaßt:

Am Tag nach Malente taufrisch für alle die nicht dort waren Berichte zur Jahrestagung in Malente. Und Kritiken zur VD 1/94 die in Malente verfügbar wurde.

In MAIL 1436 schlägt Thomas Prinz vor, den "Jupiter-Ace" nochmal aufzulegen. Das Thema wird über 25 MAILs ausgiebigst diskutiert. T. Prinz glaubt ein Z80-Einplatinencomputer der über V24 an einem PC

hängt wäre als Hardware geeignet. Rafael Deliano schlägt als Alternative einen "Video-EMUF" vor. Basierend auf einem RISC-Prozessor mit FORTH und Postcript im ROM zur Ansteuerung eines RGB-Videodisplays. Marcel Hendrix, J. Wilke, Anton Ertl, Bernd Paysan, Ulrich Hoffmann machen sich auch Gedanken zum Thema.

Weitere Vorschläge sind das ROM von X-Terminals oder Acorn-Archimedes neuzu programmieren. Nachforschungen ergeben, daß Klaus Seegebarth an einem FORTH für den Hyperstone-RISC-Prozessor arbeited.

Weitere Beiträge von Bern Beuster zur VD, Dirk Zoller zu PFE. R. Deliano faßt den Stand von Open Boot zusammen wie er sich in c.l.f darstellt. Es heißt jetzt Open Firmware und ist als IEEE 1275 verabschiedet. Es wird wohl Teil der PowerPC-Spezifikation werden. B. Beuster ergänzt anhand seiner Erfahrung mit dem Open Boot auf Sparc, daß Befehlsumfang und Geschwindigkeit ausgezeichnet sind.

Daniel Ciesinger beschwert sich über die Erhöhung der Beiträge und gibt Kommentar zu VD 1/94. K. Seegebarth sucht Interessenten für Kooperation in Sachen Mikrocontroller und Hyperstone.

R. Deliano nörgelt an PD rum, J. Wilke nimmt PD in Schutz. A. Ertl gibt knappen Statusbericht zu ANS-figFORTH, das sich jetzt GNU-Forth nennt. R. Deliano gibt als mögliches Anwendungsbeispiel für Video-EMUF Beschreibung von Bundesbahn-Info-Automat.

Wenn Ihr HP-Deskjet Patronen frißt, sagt Ihnen MAIL 1485 was Sie dagegen tun können. Jack Woehr empfiehlt als Alternative zu ANS-figFORTH sein Jax4th für WindowsNT bzw. Amiga. Tut sich aber schwer mit Deutsch. d.c.l.f wird also auch in den USA verbreited und gelesen.

U. Hoffmann sucht Source für eFORTH in 8051-Assembler statt MASM. Vermutlich noch nicht fündig geworden. R. Deliano berichted über die laut BTX-Recherche lieferbaren deutschen Bücher mit Stichwort FORTH im Titel. Es sind ohnehin nur drei.

Ankündigung, Previews und Postmortem zur "Echtzeit". Immer frisch unmittelbar vor und nach dem Ereignis. Jörg Staben ist unter die EMAILer gegangen und hat nur Probleme damit. R. Deliano würde gerne mehr zu ICR (Intermediate Code for Robots) ISO DraftPpropsal 10562 wissen. Nämlicher spezifiziert wohl eine virtuelle "Kellermaschine" und sollte damit für FORTH interessant sein. Ist bisher aber nicht schlauer geworden.

MAIL 1505 erklärt wie zwischen Hobbyismus die Specs für Röhren verändert und damit die Entwicklung des Radars gefördert hat. Auf weiteren Abwegen folgt ein optoelektronisches Problem an dem R. Deliano nagt. Robert Hoeller und D. Ciesinger knobeln auch mit. Wolfgang Schemmert findet die Lösung. J. Staben kämpft derweil mit ANS. Johannes Teich und D. Zoeller versuchen zu helfen. Staben hat anscheinend nicht genug Probleme und will das Polltool der Mailbox ausprobieren.

D. Zoller will die Entwicklung von Crosscompilern für embedded Controller basierend auf ANS-Forths anleiern. U.Hoffmann, B. Paysan, A. Ertl sagen Unterstuetzung zu. Unterschwellige Rivalitäten zwischen PFE und GNU-Forth. Nächstesmal mehr darüber.

□

IMPRESSUM

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

Herausgeber
Forth-Gesellschaft e. V.
Postfach 1110
85701 Unterschleißheim
Tel.: 089/3173784 oder
Forth-Mailbox Tel.: 089/8714548

Redaktionsleitung, Layout, Satz, Anzeigenverwaltung
Jens Wilke
Streitbergstr. 73 a
81249 München
Tel.: 089/8713533
Fax: 089/8714548

Erscheinungsweise
vierteljährlich

Auflage
1000

Preis
Einzelheft DM 10,-

Manuskripte und Rechte
Berücksichtigt werden alle eingesandten Manuskripte von Mitgliedern und Nichtmitgliedern. Leserbriefe können ohne Rücksprache gekürzt wiedergegeben werden. Beiträge der Redaktion sind vom jeweiligen Redakteur mit seinem Kürzel (s. o.) gekennzeichnet. Für die mit dem Namen des Verfassers gekennzeichneten Beiträge übernimmt die Redaktion lediglich die Presserechtliche Verantwortung. Die in diesem Magazin veröffentlichten Beiträge sind urheberrechtlich geschützt. Übersetzung, Vervielfältigung, Nachdruck sowie Speicherung auf beliebige Medien ist auszugsweise nur mit genauer Quellenangabe erlaubt. Die eingereichten Beiträge müssen frei von Ansprüchen Dritter sein. Veröffentlichte Programme gehen - soweit nicht anders vermerkt - in die Public Domain über. Für Fehler im Text, in Schaltbildern, Aufbauskizzen etc., die zum Nichtfunktionieren oder evtl. Schadhaftwerden von Bauelementen oder Geräten führen, kann keine Haftung übernommen werden. Sämtliche Veröffentlichungen erfolgen ohne Berücksichtigung eines eventuellen Patentschutzes. Warennamen werden ohne Gewährleistung einer freien Verwendung benutzt.

Brief des Direktoriums

Der April ist vorbei und damit auch die jährliche Tagung und Mitgliederversammlung der Forth Gesellschaft. Diese Versammlung hat die FG mächtig aufgemischt: in beinahe allen Ämtern neue Köpfe. Das alte Direktorium hat sich ge- und entschlossen seiner Wiederwahl entzogen. Die VD mußte sich eine neue Redaktion suchen.

Gar nicht so leicht, in einem Verein, in dem sich immer so gerade 10% der Mitglieder zur Mitgliederversammlung schleppen, vier Personen zu finden, die bereit sind, Freizeit und Arbeit zu investieren. In einen Verein, zu dem man eben nicht jede Woche einmal hingeht, der geregelte Unterhaltung mit Schwatzen, Sporttreiben oder Biertrinken bietet. Sollte man meinen.

Aber nichts da. Bereits im Vorfeld hatte sich herausgestellt, daß Jens Wilke, der bisher schon unsere Mailbox betreut hat, die Redaktion der VD wohl gerne übernehmen würde. Und um die drei Direktorenposten haben sich sogar neun Leute bemüht. Damit waren etwa ein Drittel der Versammlungsteilnehmer bereit, Aufgaben zu übernehmen! Ganz abgesehen von denen, die bereits Aufgaben haben, wobei immer mal wieder das Forth-Büro herauszustellen ist.
Und als es dann um die Organisation der Tagung im nächsten Jahr ging, gab es auch freiwillige Hilfsangebote.

Aber warum? Ich zumindest hatte während der ganzen Tagung das Gefühl von Aufbruchstimmung. Und ich denke, das diese Stimmung nicht nur durch das freundliche Ambiente der Tagungstätte hervorgerufen wurde.
Man hatte das Gefühl, daß Forth eine nun schon einige Jahre dauernde Stagnation überwunden hat.

Der Freak in uns allen mußte sich lange von immer neuen Features der Meta-, Down-, Cross-, Up- und Under-Compiler langweilen lassen. Schwierige und anspruchsvolle Dinge, aber nicht Spektakulär. In diesem Jahr gab es Windows an allen Ecken und Enden, statt unterstützter Zielprozessoren waren Slider, Buttons, Customcontrols und bewegte Grafiken das Thema. Anstelle neuer Sprachkonstrukte gab es Tetris. Und statt zu zeigen, wie eine winzige Karte über eine serielle Schnittstelle 'ok' ausgibt, wurde Forth in den Kontext großer Betriebsysteme und Entwicklungswerkzeuge einbezogen.
Hinzu kommt, daß eine Standardisierung immer in Sicht war, aber niemals greifbar wurde. Nun ist sie (fast) da, und jeder spricht davon, daß dieses oder jenes System bereits das eine oder andere Feature des Standards unterstützt.
Unglaublich aber wahr: auf der Tagung lief ein und dasselbe Programm auf ganz verschiedenen Systemen.

Forth hat in diesem Jahr über den Tellerrand gesehen. Niemand hat davon gesprochen, daß man Dinge, die man in C machen kann, in Forth doch viel eleganter lösen könnte. Vielmehr wurde gezeigt, wie sich Forth ideal in andere Umgebungen einbetten läßt und dort seine spezifischen Fähigkeiten voll ausspielt. Dies ist eine neue Form von Selbstbewußtsein. Forthprogrammierer haben keine Scheu mehr vor der vermeintlichen Konkurrenz, sondern beziehen alles, was sie 'draußen' finden können, in ihre eigene Ideenwelt mit ein. So vorgegangen, kann man auch Forth und Multimedia in einem Satz aussprechen. Plötzlich sind grafische Oberflächen für Microcontroller kein Thema mehr.

Dies alles hat uns mitgerissen und uns nicht über PD vs. kommerzielle Systeme oder FIG-Forth vs. Forth83 oder F-PC vs. ZF streiten lassen.

So stehen wir denn nun da, drei neue Direktoren und eine neue Redaktion der VD und hoffen, daß ein Teil dieser Aufbruchstimmung uns unsere Arbeit erleichtert. Denn obwohl die Mitglieder sich gewünscht haben, im nächsten Jahr etwas mehr für die Mitgliedschaft in diesem tollen Verein bezahlen zu dürfen, sind die Kassen leer. Nun müssen wir sehen, wie wir mit Idealismus weiterkommen.

In diesem Sinne,

Euer Jörg

DFÜ mit der Forth-Mailbox

von Zbigniew Diaczyszyn
EMAIL: dia@BBS.FORTH-eV.de

bequem, schnell und sicher

Für die Forth-Gemeinde, die über ganz Deutschland verstreut ist, bietet sich das Medium der elektronischen Nachrichtenübermittlung als ideales Hilfsmittel an, miteinander ins Gespräch zu kommen. Was möglicherweise viele von der Nutzung dieses Mediums abgehalten hat, war die Befürchtung, die DFÜ sei umständlich, fehleranfällig und im Endeffekt recht teuer.

Mittlerweile hat sich aber doch recht viel getan. Die DFÜ ist definitiv dem experimentellen Stadium entwachsen und bietet sich als das Kommunikationsmedium der Zukunft auch für den gelegentlichen Briefeschreiber an, der mit DFÜ-Fachsimpelei nichts zu tun haben möchte.

Die Münchner Forth-Box besitzt nun ein sehr schnelles Modem (14400 baud) mit dem hochoptimierenden Fehlersuchverfahren 42bis. Damit ist die Basis gegeben für die Möglichkeit, auch in der Fernzone noch preiswert Dateien übertragen zu können. Bei 1000 Zeichen pro Sekunde (cps) wären das in der Fernzone abends 40 KB für 23 Pfenning. Persönliche Nachrichten (mails) und Gruppennachrichten (news), etwa aus der Gruppe comp/lang/forth, überschreiten selten diese Größe, zumal sie nur komprimiert über die Telefonleitung geschickt werden.

Das größte Handicap war aber wohl, daß es sehr lästig war, immer online in München anrufen zu müssen. Die Befehle, die eingegeben werden mußten, hatte man nicht immer im Kopf, man mußte ständig die online-Hilfe aufrufen, irrte in den Ordnern umher, und die ganze Zeit über tickte der Gebührenzähler. Hatte man dann sein Nachrichtenpaket in seinen Rechner geladen, mußte man es entpakken und irgendwo speichern. Nach einiger Zeit ging die Übersicht verloren. Welche Nachricht zu dem Thema xyz gibt es? Ohne eine Datenbank ist man bei 4 Megabytes Nachrichten verloren. Für Boxen in anderen Netzen (Fido-, Maus- oder Z-Netz) gibt es Point-Programme, mit denen man bequem offline zu Hause vor seinem Rechner die Post bearbeiten kann. Eine Datenbank und eine Adreßverwaltung sind integriert, so daß sehr schnell eine Nachricht gefunden werden kann oder ein Brief an einen bestimmten Adressaten (user) abgesandt werden kann.

Das ist nun auch in der Münchner Forth-Box möglich! Peter Mandrella hat seinem Allround-Pointprogramm XPOINT ein UUCP-Modul mit allen nötigen Packern und Entpackern beigelegt, und weil die Münchner Forth-Box eine UUCP-Box ist, kann nun auch der die Bequemlichkeit liebende Forth-Freund als Point mit der Forth-Box kommunizieren.

Was hat er zu tun?

Falls er einen DOS-Rechner besitzt, braucht er sich nur das Shareware-Programm Crosspoint (XPOINT) zu besorgen. Zur Zeit ist die Version 3.0 aktuell. Sie ist in allen gutsortierten Mailboxen zu finden und auf der DFÜ-Einsteiger-Diskette von Jens Wilke. Ist man mit dem Programm zufrieden, überweist man dem Autor 50DM. Eine ausführliche Dokumentation und eine detaillierte online-Hilfe lassen einen nie im Stich. Auch unter OS/2 läuft es in einer Dos-Box stabil. Ein Terminalprogramm ist nicht nötig, da XPOINT selbständig die Box anruft. Wenn man mit der Zeitschaltuhr arbeiten möchte, kann man Xpoint auch zu einem automatischen Anruf veranlassen. XPOINT holt sich aus der Box alle neuen Nachrichten, verabschiedet sich von der Box, entpackt die Daten und sortiert alle Nachrichten in die richtigen Bretter. Zu einem beliebigen späteren Zeitpunkt kann sich dann in aller Ruhe das Neue durchlesen und auch eine Antwort schreiben, falls man Lust dazu hat. Dasselbe gilt für das Bestellen von Dateien (filerequest). XPOINT nimmt die Bestellung auf, und

Stichworte:

Mailbox,
Usenet,
Offline-Reader

beim nächsten Anruf kommt mit den Nachrichten auch gleich die Datei. Zur Installation klickt man sich durch die Menüpunkte EDIT BOXEN POINT und füllt das Pointformular aus. Das Paßwort macht man mit Jens Wilke, unserem Sysop, aus.

Bild 1: Die Point-Installation

```
+- dia @ dia ------------------------------+
|                                          |
|  Pointname  dia                Telefon  0898714548
|  Paßwort    ??????????         Login    uucp
|                                          |
|  Upload-Packer       compress -v -b12 $PUFFER
|  Compress-Entpacker  compress -vdf $DOWNFILE
|  Freeze-Entpacker    freeze -vdif $DOWNFILE
|  GNU-ZIP-Entpacker   gzip -vdf $DOWNFILE
|                                          |
|  max. UUCP-g-Paketgröße    4096    [x]   g-Protokoll
|  max. UUCP-g-Fenstergröße     7    [ ]   G-Protokoll
|                                    [ ]   z-Protokoll
|   [x]   variable UUCP-g-Paketgröße [ ]   f-Protokoll
|   [ ]   Ausgangspaketgröße vorgeben [ ]  e-Protokoll
|   [x]   Dateigröße übertragen            |
|   [ ]   Batched SMTP               [ ]   Login mit 7/E/1
|                                          |
+------------------------------------------+
```

Dann wählt man die Menüpunkte EDIT BOX NAMEN aus und setzt die notwendigen Namen ein:

Bild 2: Die Installation der Namen

```
+- dia @ dia ----------------+
|                            |
|  Boxname     FORTH-ev      |
|                            |
|  Username    dia           |
|                            |
|  Kommentar                 |
|                            |
|  Realname    Zbigniew Diaczyszyn
|                            |
|  Domain      .forth-ev.de  |
|                            |
|  Serverdom.  .de           |
|                            |
+----------------------------+
```

Zuletzt ist nur noch die vorgegebene Script-Datei etwas anzupassen:

```
# UUCP.SCR: Netcall-Script für UUCP-Systeme # Die Vergleichstexte sind
nicht case-sensitiv!

       on    relogin send cr
first: timer 5
start: read
       on    "NEW" goto login
       on    timeout goto cr
       goto  start
cr:    send  cr                          # hey, aufwachen!
       goto  first

login: send  $LOGIN cr
loop:  read
       on    "password:" send $PASSWORD cr
       on    "^Pshere" goto ende         # uucico-Startkennung
       goto  loop

ende:  write "^M       ^M"               # ^Pshere löschen
```

So, und das war es auch schon. Von jetzt ab schaut man nur noch zu, wie alles automatisch abläuft, sobald man die Punkte NETCALL EINZELN gewählt hat. Nach dem Anruf zeigt sich schön aufgeräumt die Bretterübersicht, wobei die Bretter markiert sind (groups),. die neue Nachrichten enthalten.

Bild 3: Die Bretterübersicht nach einem Netcall

```
 XPoint  Wartung  Nachricht  NeTcall  Fido  EDit  Config  Zusatz
 Alle  Brief  Textfile  Binä+--------+n: Neues              Tab / Quit
                           | Einzeln      |
  /»Mausinfo               | Alle         |
  /»Mausstatus             | Uhrzeit      |
> /»Netzanruf              | Relogin      |
  /DIA                     | Online       |
  /dia.forth-ev.de!dia     | Bereitschaft |
  /Sysop                   | Timing-Liste |
  /Zbigniew Diaczyszyn     +--------+
  /A/OS2-Programme
> /comp/lang/forth
> /comp/lang/forth/mac
  /FIDO/FRANKEN.OS2
  /FIDO/POB.CLUB
  /FIDO/POB.DEV
  /FIDO/POB.FILES
  /FIDO/POB.MISC
  /FIDO/POB.OS2
  /FIDO/POB.USER
  /FIDO/POBFILES
  /FIDO/PORTALS
  /FIDO/REQUEST.OS2.GER
  /FORTHEV/NEWS
  /INTERNES
  /MAUS/Forth-Prg
  /MAUS/Maus.Info
  /MAUS/Oeffentlich
  /MAUS/Os/2.Maus
  /MAUS/Os2.Prog
  /MAUS/Os2.Prog.G
  /T-NETZ/BINAER/CROSSPOINT
  /T-NETZ/SUPPORT/XPOINT/ALLGEMEINES
  /T-NETZ/SUPPORT/XPOINT/UPDATES
  /Z-NETZ/RECHNER/IBM/OS2
  /Z-NETZ/SPRACHEN/FORTH

F1-Hilfe  F6-Makros  F9-DOS                                  CrossPoint
```

Klicke ich nun in der Brettübersicht ein bestimmtes Brett an, etwa das Brett COMP/lang/forth, so zeigt Crosspoint die soeben empfangenen Nachrichten, wobei das Zeichen ">" anzeigt, daß die Nachricht noch nicht gelesen ist. Ein Klick auf eine Nachricht öffnet sie. Es ist möglich, Textstellen zu markieren, damit sie in der Antwort als Zitat erscheinen (Quote). Selbstverständlich kann man die Nachricht auch drukken oder in eine Datei schreiben. Die Antwort kann in ein beliebiges Brett oder an einen beliebigen Adressaten geleitet werden.

Bild 5: Die Brettübersicht

```
/comp/lang/forth  -  Nachrichten seit dem letzten Netcall
>      747 11.05 bryan@isrc.sandia.gov     Re: Finite state machine compiler
>      763 12.05 ir230@sdcc3.ucsd.edu      feeds to comp.lang.forth
>     1805 12.05 flacy@engin.umich.edu     Re: Finite state machine compiler
>     2000 11.05 jvn@fermi.clas.Virginia.E Re: preposterous boasting
>      444 11.05 jvn@fermi.clas.Virginia.E Re: Starting Forth
>     1916 12.05 aph@oclc.org              Re: preposterous boasting
>      837 12.05 nii@world.std.com         Re: preposterous boasting
>     1339 04.05 gary.chanson@channel1.com Re: Direct vs. Indirect threading
>      569 11.05 dmiller@im.lcs.mit.edu    Re: Starting Forth
>     1335 12.05 gmribeir@david.wheaton.ed Re: preposterous boasting
>      827 12.05 jax@Cygnus.COM            Re: Multitasking Forth?
>     4480 12.05 mikeh@starnine.com        Re: Multitasking Forth?
>     1266 12.05 mikeh@starnine.com        Re: Multitasking Forth?
>     1658 13.05 mikeh@kralizec.zeta.org.a Re: forth on the mac
>      858 13.05 blake@netcom.com          A good (up-to-date) Forth book?
>      810 13.05 anton@mips.complang.tuwie Re: Direct vs. Indirect threading
>     1878 13.05 gmribeir@david.wheaton.ed Re: preposterous boasting
>      993 10.05 lowry@src.honeywell.com   Re: Teaching Forth
>      825 13.05 djohnson@arnold.ucsd.edu  Re: preposterous boasting
>      673 13.05 tanksley@coyote.csusm.edu Re: preposterous boasting
>     1266 13.05 as460@FreeNet.Carleton.CA Re: preposterous boasting and Teachi
>     1274 10.05 flacy@engin.umich.edu     Re: preposterous boasting
```

Bild 4: Der Netzanruf

```
Empfaenger : /»Netzanruf
Absender   : dia@dia.forth-ev.de  (Zbigniew Diaczyszyn)
Betreff    : Netzanruf vom 15.05.94, 07:27
Datum      : So 15.05.94, 07:28
Groesse    : 2282 Bytes
```

```
Netzanruf vom 15.05.94 um 07:27 bei FORTH-ev, 0898714548

Anwahlbeginn : 07:27:22
Wählversuche : 1
Verbindung   : 07:27:57
Connect .... : CONNECT 14400/V42BIS

Connectzeit  :   0:05          Sendepuffer   :       0 Bytes
Loginzeit    :   0:09          Sendepaket    :       0 Bytes
UUCP-Init    :   0:11          Empfangspaket :   22067 Bytes
Sendezeit    :   0:00          Empfangspuffer:   41679 Bytes
Empfangszeit :   0:22
Hangupzeit   :   0:02          Senderate     :       0 cps
-----------          Empfangsrate  :    1003 cps
Gesamtzeit   :   0:49          Schnitt       :    1003 cps

Billigtarif:  0.23 DM

eingehende Nachrichten:     24
ausgehende Nachrichten:      0

---------- Logfile ----------


CONNECT 14400/V42BIS
FORTH-Gesellschaft e.V. Postfach 1110 85701 Unterschleissheim

Netzwerkadresse: BBS.FORTH-eV.de
089/8714548 1200-14400 Bps 8n1

Falls Sie User dieser Box werden
moechten, bitte jetzt NEW eingeben!
Wenn Sie nur mal reinschnuppern
wollen, geben Sie GAST oder GUEST ein!

Username or NEW:

Username or NEW: uucp
Password:
>Shere
«uucico-Aufruf»

---------- UUCP-Logfile ----------

  07:28:19  UUCP connection established
~ 07:28:19  remote protocols: g
  07:28:19  selected protocol ^
  07:28:19  UUCP-g packet sizes: 64/4096
  07:28:19  UUCP-g window sizes: 7/7
+ 07:28:19  HN: remote has data for you
+ 07:28:20  receiving D.BBS.FOR2928 as SPOOL\D-R29280
* 07:28:40  received 22067 bytes, 1103 cps, 0 errors
+ 07:28:41  receiving X.BBS.FOR2928 as SPOOL\X-R29280
* 07:28:41  received 65 bytes, 65 cps, 0 errors
+ 07:28:42  closing UUCP-g connection
+ 07:28:42  hanging up

## CrossPoint v2.93 ##
```

XPOINT hält zur Kontrolle ein Logfile bereit, das sich im Brett NETZANRUF findet. Zu erkennen ist, daß 20 KB in 35 Sekunden übertragen sind bei einer Senderate von ca. 1003 cps. 11 Sekunden dauert es, bis die Box bereit ist, die Point-Befehle zu 1650 cps im Schnitt.

Mit dem Austausch von persönlichen und öffentlichen Nachrichten (news and mails) ist damit der wesentliche Teil der DFÜ-Kommunikation besprochen. Einen Leckerbissen stellt die Übertragung von Binärdateien dar (filerequest and filetransfer). Normale Dateien kann man wie gesagt über die Menüpunkte NACHRICHTEN FILESERVER BESTELLEN saugen, so daß die Forth-Box das umständliche Versenden von Diskettten überflüssig macht. Theoretisch wäre auch die Nutzung von FTP-Servern über ftpmail möglich, aber das ist die Entscheidung des Sysops und des Direktoriums, denn da kommen schnell "Briefe" von ein paar Megabytes Umfang an.

Wer weiterhin im Regen zum nächsten Briefkasten rennen will, vorausgesetzt, er hat noch eine Briefmarke gefunden oder ein Postamt in seiner ferneren Nähe, das auch mehrere Stunden geöffnet ist, so daß man Briefmarken kaufen kann, und wer die Geduld hat, zu warten, bis sein Brief in der VD veröffentlicht ist, der braucht die elektronische Nachrichtenübermittlung nicht. Für alle anderen bietet sich aber nun mit dem UUCP-Modul von Crosspoint die Möglichkeit, bequem, schnell und sicher miteinander zu kommunizieren.

❑

Zeitmessung auf dem PC

von Bernd Paysan

Was macht man, wenn man eine schnelle Uhr braucht (besser als Millisekunden), aber anscheinend keine hat (wie z.B. im PC)? Eine Lösung wird vorgestellt, die eine auf 20µs genaue Zeit liefert.

Das Problem

Zeit spielt bei der Prozeßsteuerung eine eminent wichtige Rolle. Nicht nur bei Wettbewerben wie der "bunten Box" (bei denen auch die Programmierzeit wichtig ist), sondern bei vielen realen Problemen ist eine genaue Zeitmessung wichtig. Mikrokontroller wie der H8 stellen natürlich einen genauen Timer zur Verfügung (siehe VD 1/94, "Pudding mit Sahne"), PCs aber (scheinbar) nicht.

Thomas Beierlein hat in der VD 4/92 eine Lösung vorgestellt, die bei ATs wenigstens ein halbwegs genaues ms ermöglicht. Aber mit ein paar handfesten Nachteilen: Nur ein Task kann auf einmal warten. Damit lassen sich nur höchst simple Probleme lösen. Und was noch schlimmer wiegt: Man hat jetzt zwar einen gut aufgelösten Wecker, kann aber die Uhr nicht ablesen!

Ein bißchen Disassemblieren des BIOS' löst zwar dieses Problem, schließlich muß das BIOS sich ja auch merken, wie lange es noch warten will. Also hatte ich mir kurzerhand einen Interrupt-Handler geschrieben, der die Uhr für FORTH hochzählt, dem originalen BIOS-Handler aber vorgaukelt, er sei noch weit weg vom Alarm und ihn aufruft (damit auf den Interrupt auch richtig reagiert wird). Damit konnte ich erstmal meine Benchmarks endlich ms-genau austesten.

Aber, oh Schreck, oh Graus! Der BIOS-Interrupt schluckte (mit DOS-Extender-Overhead) ca. 1/4 der Rechenzeit! Also wurde der Interrupt eben voll disassembliert und die Quintessenz herausgezogen. Damit schluckte das Biest nur noch 5% der Rechenzeit. Naja. Außerdem hatte die eigene Interruptroutine die unangenehme Angewohnheit, sporadisch außer Tritt zu kommen (wohl hatten einige weggelassenen Codestellen durchaus ihren Zweck). Das mußte auch noch überwacht werden und der Timer bei Bedarf wieder angeworfen werden (genauer wird er dadurch sicher nicht!).

Die Lösung

Die Lösung kam dann aus einer anderen Ecke. Ein paar Komilitonen von mir hatten für ihren Atari Falcon einen Video-Taktgenerator zusammengelötet, der eine höhere Auflösung versprach. Da sie keine echte PLL (phase locked loop) verwendeten, driftete der Takt langsam, aber sicher ab. Also maßen sie die Zeit zwischen 100 Bildschirmwiederholungen und rechneten daraus die notwendige Korrektur aus. Der Systemtimer (beim Atari immerhin 200 Hz) war dazu nicht genau genug, aber beim Atari gibt's ja 4 unabhängige, frei programmierbare Timer (von denen 2 vom Betriebssystem und einer von Spielen gern benutzt wird), die auch deutlich genauer sein können, als die 1024 Hz im AT. Und der 68030 fackelt bei Interrupts auch nicht so lange, also kann man sich das leisten.

Jedenfalls stellte sich heraus, daß auch der 4. Timer alles andere als unbenutzt war. Und hier kam mir die Idee: Jeder Timer hat ja noch einen internen Zähler (der beim System-Timer von 192 auf 0 herunterzählt), den liest man aus und hängt ihn hinten an den System-Timer an (System-Timer mit 192 multiplizieren und 192-<interner Zähler> dazuzählen). Das Problem war damit noch nicht ganz gelöst, da auch der Interrupt vom Bildschirm nicht pünktlich kam, aber das ließ sich mit derselben Methode lösen (die aktuelle Scan-Position aus dem Video-Chip auslesen).

Die Probleme der Lösung

Ganz so einfach wie oben beschrieben ging's natürlich auch wieder nicht. Was, wenn beim Auslesen der beiden Zähler der Timer einen Interrupt auslöst, und zwar genau zwischen dem Auslesen des Interruptzählers und des Timer-internen Zählers? Man liegt völlig daneben. Immerhin kann man die Interrupts abwürgen, beide Zähler auslesen und hinterher fragen, ob genau dieser Timer-Interrupt hängt. Wenn ja, erhöht man den Interrupt-Zähler um 1 und ersetzt den Wert des internen Zählers durch 192. Diese Lösung funktionierte erstmal und lieferte auf dem Atari auf 26µs genaue Zeiten.

Irgendwie wird man diese Lösung auch auf den PC übertragen können. Dessen Standard-Timer, die 18,2 Hz vom 8253, ist für alles andere außer Uhren in einer der oberen Ecken praktisch unbrauchbar. Aber der Timer wird (oder wurde - im Ur-PC) mit einem 4,77 MHz-Takt gespeist. Der wird erst mal durch 2 geteilt, dann in zwei 8-Bit-Zählern durch jeweils 256 (eigentlich frei einstellbar) geteilt. Am Schluß wird er nochmal durch 2 geteilt, das ist dann, was am Interrupt-Ausgang ankommt (das ist natürlich vereinfacht dargestellt, der Takt wird eigentlich erst durch 4 geteilt und dann durch 128, das unterste Bit des ersten Zählers wird nicht benutzt).

Zwar kann man mit dem programmierbaren Teiler auch an eine genauere Uhr kommen (das wurde erst kürzlich in comp.lang.forth vorgeschlagen), damit hat man aber wieder den Nachteil, daß wertvolle Rechenzeit in Interrupts verbraten wird und der Original-Timer auch noch hin und wieder (möglichst alle 55 ms) drankommen soll.

Also habe ich mich mangels dickem Buch zum 8253 drangesetzt und bin dem

internen Zähler durch Trial und Error auf die Schliche gekommen. Der Timer ist auf 4 IO-Adressen von $40 bis $43 gemappt und auf die nächsten 7 durch 4 teilbaren IO-Adressen (bis ausschließlich $60) gespiegelt (in Intel-Schreibweise 40H-43H, aber die Motorola-Schreibweise ist mir lieber).

An $40 sind die zwei 8-Bit-Zähler, die sich abwechselnd (!) auslesen lassen. Man weiß natürlich nie, in welchem Zustand der Chip gerade ist (Murx!), also muß man ihn synchronisieren. Das geht, indem man 0 nach $40 schreibt. Danach liegt zuerst das niederwertige (schnelle) Zähler-Byte, dann das höherwertige an. Im obersten Bit an $43 findet sich dann noch der letzte Teiler, der jeweils beim Übergang von 0 auf 1 einen Interrupt auslöst, beim andern Übergang still ist. Immerhin gibt es jetzt schon drei Stellen, an denen ein Überlauf auftreten kann: Vom schnellen zum langsamen Zähler-Byte, von dem zum letzten Teiler und von dort zum Interrupt-Zähler. Außerdem gibt es auch nirgends ein Bit, das diese diversen Überläufe anzeigt.

Immerhin präsentieren sich damit zwei kritische Stellen: Eine Änderung des höherwertigen Zähler-Bytes und eine Änderung des letzten Teilers. Damit ergab sich eine wesentlich elegantere Lösung des Problems, als die Abfrage eines "pending Interrupt"-Bits. Man liest die kritischen Werte zweimal ein, einmal vor den unkritischen Werten, einmal danach. Hat sich dann etwas geändert, hat man wahrscheinlich nur Müll gelesen und muß nochmal anfangen.

Was eine gute Uhr sonst noch auszeichnet

Nun, jedenfalls hat man jetzt eine Uhr, die im Prinzip bis auf (ungefähr) eine Mikrosekunde genau ist, wenn man den Timer so schnell auslesen könnte. Mehr als einen Tag kann man damit nicht messen, denn um Mitternacht wird der BIOS-Zähler wieder auf 0 zurückgesetzt. Würde man die Genauigkeit voll ausnutzen und einen 32-Bit-Wert produzieren, hätte man nur eine Stunde zur Verfügung.

Immerhin, der Tag ist schon ausreichend. Allerdings sollte eine Uhr nicht um Mitternacht einfach auf 0 zurückgesetzt werden; und wenn, dann sollte sie den ganzen Zahlenraum durchwandert haben (also jeden Wert zwischen 0 und 2^32-1 erreichen). Warum? Absolute Zeiten sind meist uninteressant (sieht man mal von den Uhren in den Ecken und astronomischen Anwendungen ab - diese Leute brauchen eher eine Funkuhr); was den Echtzeitprogrammierer interessiert, sind Zeitdifferenzen. Und für diese Differenzen spielt es nun wirklich keine Rolle, ob es gerade Mitternacht war, oder nicht.

Berechnen möchte man solche Zeitdifferenzen am liebsten mit einem einfachen "-". Aus dem Vorzeichen kann man dann ablesen, ob das Ereignis in der Zukunft oder der Vergangenheit liegt (immerhin hat man dann noch einen halben Tag Zeitdistanz zur Verfügung). Aus diesem Grund habe ich die rohen Ausgangsdaten noch durch $1800B0 geteilt, damit der ganze Tag in die 32 Bit paßt. Durch $1800B0, weil ein Tag offenbar nicht genau 24 Stunden hat, sondern der Timer im Ur-PC offenbar ein klein wenig vorging und man dieses direkt im ROM-BIOS korrigiert hat (da sträuben sich die Haare). Damit ist die Auflösung auch etwa an die Auslesezeit angepaßt. Die Genauigkeit sinkt auf ca. 20 Mikrosekunden, das Auslesen dauert auf meinem 33 MHz 386er ziemlich genau 10 Mikrosekunden.

Für Neugierige gibts's den Source-Code in Listing 1 (alles 32-Bit-Code ab 386 und nur für bigFORTH 386 - eine Anpassung an F-PC sollte erfahrene Assembler-Programmierer auch nicht vor zuviel Arbeit stellen).

Und damit Fans noch sehen können, wo der Vorteil vom Atari gegen den PC liegt (in der Kürze liegt die Würze), dasselbe (diesmal auf 26 Mikrosekunden genau) für den Atari in Listing 2.

Beide Timer liefern eine Rohzeit, die erst in eine gebräuchlichere Einheit umgewandelt werden muß. Da sie beide nicht auf eigene Interrupts angewiesen sind, stehen sie Multitasking und damit mehreren gleichzeitig gestarteten bigFORTH-Systemen nicht im Wege (etwa unter Windows oder mit MultiTOS auf dem Atari).

❑

Listing 1

```
\ exact timer                                              10mar94py

| $40 Constant #timer

Code timer@  ( - timer )  AX push  AX AX xor  CX CX xor
     BEGIN  .b #timer 3 + # in  $80 # AL test  AH 0= setIF
            AL AL xor  .b #timer # out
            .b #timer # in  AL CL mov
            .b #timer # in  AL CH mov
            BEGIN  .b #timer # in  AL CL mov
                   .b #timer # in  AL CH cmp  AL CH mov
            0= UNTIL  GS: $46C #) DX mov  \ GS is DOS-Segment
             .b #timer 3 + # in  $80 # AL test  AL 0= setIF
            AL AH cmp   0= UNTIL  AL AL xor  $17 # AX shl
     .w CX not  $F # CX shl  CX AX add
     $1800B0 # CX mov  CX div  Next end-code
```

Listing 2

```
Code timer@ ( - time )
     BEGIN  $C0 # D1 move  $4BA #) D0 move
            .b $FFFFFA23 #) D1 sub  .1 $4BA #) D0 cmp  0= UNTIL
     6 # D0 lsl  D0 D1 add  D0 D0 add  D1 D0 add
     D0 SP -) move  Next end-code
```

Object Oriented bigFORTH

von Bernd Paysan

Anhand eines Beispiels für die objektorientierten Erweiterung von bigFORTH wird in die objektorientierte Programmierung eingeführt. Das Klassenkonzept, Vererbung und Polymorphismus sowie deren Realisierung werden gezeigt.

Einleitung

Objektorientiertes Programmieren ist für viele Forther etwas neues; obwohl es objektorientierte FORTH-Systeme schon länger gibt ([1], [2], [3] oder [4]). Sowie die bisher bekannten Systeme den einen oder anderen Nachteil haben, so haben auch die Präsentationen oft ein Manko: Ein ausführlicheres Beispiel fehlt. Objektorientiertes Programmieren lernt man aber nicht durch Darstellung der Syntax (genausowenig wie "normales" Programmieren) und leider auch nicht durch Einblick in die Sourcen der Implementierung - auch wenn erst hier genau klar wird, was wie und wann (nicht aber warum) im System passiert.

Die höhere Abstraktion, die objektorientiertes Programmieren erst attraktiv macht, macht solch einen tiefen Einblick weniger nötig. Statt dessen wollen wir uns nun auf die Anwendung konzentrieren. Als Beispiel dient eine kleine Sammlung von Datentypen, wie sie aus der Informatik bekannt sind: Integer, Listen, Arrays und Pointer.

Objektorientierte Programmierung versteckt sich hinter einem Slang, der bei näherer Betrachtung durchaus Ähnlichkeiten mit bekannten Konzepten wie Modularität und der Definition sauberer Schnittstellen erkennen läßt.

Das Klassenkonzept

Der Kerngedanke an der objektorientierten Programmierung ist die Kapselung von Daten und den sie bearbeitenden Prozeduren in ein Objekt. Im Idealfall haben die Prozeduren ("Methoden") eines Objekts alleinigen Zugriff auf dessen Daten und sind somit die einzige Möglichkeit, die Daten zu bearbeiten. Objektorientiertes Programmieren bedeutet also Programmieren von komplexen Datentypen und Operationen auf diese Datentypen; das ist ein fundamentaler Unterschied zur eher prozedural bis funktionalen üblichen FORTH-Programmierung.

Schnittstelle zu einem Objekt sind die Namen der Methoden (Messages) und Parameter, die bei den Messages mitgeschickt werden. Da Objekte auf viele Methoden auch ein Ergebnis zurückliefern, verwendet man für das "message passing" ganz konventionell den Stack und ruft die Methode wie ein FORTH-Wort auf - mit dem Umweg über das Objekt, das die Kapselung handhabt.

Nun wäre es eine ziemliche Verschwendung, für jedes Objekt eigene Methoden neu zu programmieren; vor allem, da viele Objekte einen gleichen oder ähnlichen Aufbau haben und sich nur in den Daten unterscheiden. Solche gleichartige Objekte faßt man zu einer Klasse zusammen. Eine Klasse ist gewissermaßen eine Schablone (oder ein Formular) für ein Objekt; erst durch Instanzierung entsteht aus der Klasse ein reales Objekt mit Platz für Daten und den allen Objekten der Klasse gemeinsamen Methoden.

Da oft an einer Klasse nur kleine Modifikationen vorgenommen werden müssen, um eine neue Klasse zu erhalten, benutzt man die "Vererbung" ("Inheritance"), um eine Unterklasse - ein Derivat - zu erhalten. Zusätzlich benötigte Variablen und veränderte oder neue Methoden werden zur Klasse nur dazugefügt.

Alle Objekte einer Klasse und auch die aller Unterklassen haben ein gemeinsames Message-Protokoll, sie verstehen dieselben Messages und reagieren gleichartig darauf. Die Unterschiede im Detail nennt man "Polymorphismus" - Vielgestaltigkeit. So mag z.B. jedes Graphik-Objekt eine Methode "zeichne Dich" haben, das eine Objekt aber einen Kreis, das andere einen Punkt oder ein Rechteck darstellen.

Legt man viel Wert auf ein gleichartiges Protokoll zu allen Objekten einer Klassenhierarchie, so entwirft man dieses Protokoll getrennt von den einzelnen Implementierungen der Unterklassen und nennt die so entstandene Klasse, die nur Protokoll, nicht aber Implementierung enthält, "abstrakter Datentyp" oder "abstrakte Klasse". Eine solche ist die im Listing 1 vorgestellte Klasse data.

Hier muß noch dazu gesagt werden, daß in bigFORTH alle Klassen letztendlich von einer Vater-Klasse abstammen müssen, von der Klasse OBJECT. Auch sind Klassen und Objekte nicht nur für die Bearbeitung von Daten zuständig, sondern auch für die Erzeugung neuer Unterklassen und Instanzen der Objekte. Deshalb ist eine Klasse ein nur um seinen Datenbereich beschnittenes Objekt, das mit der Methode class eine neue Unterklasse erzeugen kann.

Die Beschreibung einer Klasse besteht aus zwei Teilen: Der Deklaration von Variablen und Methoden, sowie der Implementierung der Methoden. Alle Variablen, alle polymorphen und von außen zugänglichen Methoden müssen deklariert werden; Hilfsmethoden können deklariert werden, müssen aber nicht. Bei der Implementierung werden undeklarierte Methoden automatisch als EARLY (privat) deklariert.

Im Beispiel wird eine Variable names ref von der Größe einer Zelle angelegt, und zwar im privaten Bereich der Klasse, die von außen nicht sichtbar ist, wohl aber vererbt werden kann (entspricht also dem protected: in C++). public:, also öffentlich zugänglich sind die 6 Methoden !, @, ., null, atom? und #, die für Speichern, Auslesen und Anzeigen des Wertes, zur Erzeugung eines "Null"-Objektes, der Feststellung, ob das Objekt unteilbar oder zusammengesetzt ist, sowie der Anzahl an Unterobjekten, falls letzteres der Fall ist.

Die letzten drei Methoden werden schon implementiert, da sie für alle einfachen Objekte gleich sind. Die nicht implementierten Methoden können nicht ausgeführt werden, genauer gesagt, sie führen auf ABORT". Sie müssen in realen Datenty-

Listing 1

```
\ Data structures: data                                    30apr93py

Memory also Forth

object class data       \ abstract data class
        cell var ref    \ reference counter
public: method !        method @        method .
        method null     method atom?    method #
how:    : atom? ( - flag )  true ;
        : #     ( - n )     0 ;
        : null  ( - addr )  new ;
class;
```

Listing 2

```
\ Data structures: int                                     30apr93py

data class int
        cell var value
how:    : !     value F ! ;
        : @     value F @ ;
        : .     @ 0 .r ;
        : init  ( data - )  ! ;
        : dispose  -1 ref +!
          ref F @ 0> 0= IF  super dispose  THEN ;
        : null  ( - addr )  0 new ;
class;
```

pen implementiert werden, wie im Datentyp Integer (siehe Listing 2).

Hier wird in bekannter Manier eine neue Klasse erzeugt und eine (private) Variable value angelegt. Die beiden Methoden ! und @ greifen auf value zu - als Schnittstelle völlig ausreichend. Das F vor den Bezeichnern verhindert, daß die Methoden des Objekts selbst verwendet werden und schaltet statt dessen auf das normale Vokabular um - es werden also die normalen Wörter, die man unter dem Namen ! und @ kennt, verwendet. Auch die Methode . ist einfach zu verstehen.

Zu den Methoden init und dispose muß noch etwas gesagt werden: init wird beim Erzeugen eines Objekts aufgerufen und dient der Vorinitialisierung des Objekts. Hier im Beispiel wird value mit einer auf dem Stack liegenden Zahl initialisiert. dispose entfernt ein Objekt aus der dynamischen Speicherverwaltung. Modifiziert man diese Methode, muß man (anders als in C++) explizit auch die Entfernungsmethode der Vaterklasse aufrufen (mit super dispose). Entfernt wird nur, wenn der Reference Counter 0 oder negativ geworden ist, also kein Verweis mehr vorhanden ist. Andernfalls wird der Reference Counter eben nur erniedrigt.

null erhält hier die Bedeutung, die man schon vermutet: Es legt ein Objekt mit dem Wert 0 (dynamisch) an und hinterläßt dessen Adresse. new ist ebenso wie dispose eine Methode. Ohne zusätzliche Information (also ohne Angabe von Klasse oder Objekt) wird die Methode der aktuellen Klasse verwendet.

Binding: Late oder early?

An dieser Stelle empfielt es sich, etwas über die Vorgehensweise beim Aufruf einer Methode zu sagen. Wieviel kann hier schon zur Compile-Zeit bestimmt werden und was muß bis zur Laufzeit offenbleiben (und sollte dann möglichst keinen Fehler mehr produzieren)?

Sofern, wie bei SUPER DISPOSE klar ist, welche Methode welcher Klasse ausgeführt werden soll, wird man das schon zur Compile-Zeit auflösen und einen direkten Call zur angegebenen Methode compilieren. Das nennt man dann "early binding". Das ist sicher am schnellsten, und auch am einfachsten, erlaubt aber leider keinen Polymorphismus.

Oft ist ja nicht von vornherein klar, welcher Unterklasse das Objekt angehört, dessen Methode man ausführen will. Man muß also die Adresse der Methode zur Laufzeit herauskriegen. Eine Suche im Dictionary kommt aus Effiziensgründen nicht in Frage, auch eine (womöglich gar sequenzielle) Suche über einen numerischen Schlüssel ist nicht das, was man sich unter Laufzeiteffizienz vorstellt.

In bigFORTH steht daher in jedem Objekt an erster Stelle ein Zeiger auf eine Sprungtabelle, in der alle Methoden eingetragen sind. Das garantiert nicht nur eine Aufrufzeit unabhängig von der Menge der Methoden (schließlich soll FORTH ja eine Echtzeitsprache bleiben), es ist auch sehr schnell. Vor allem, weil das Verfahren so einfach codiert werden kann, daß es als Makro direkt in den Code des Aufrufers eingefügt wird.

Was man sich mit diesem Trick verbaut, ist die Mehrfachvererbung, auch "Multiple Inheritance" genannt. Aus mehreren Vaterklassen eine Tochterklasse zu kreuzen ist ohnehin problematisch: Gleichnamige Methoden und Variablen müssen umbenannt werden, die Offsets für die Variablen im Objekt ändern sich (müssen also auch erst zur Laufzeit bestimmt werden) oder der Compiler muß schon vorher Sorge tragen, daß der in der Mischklasse nötig gewordene Platz in seinen Überklassen reserviert ist (ein Space-Time-Tradeoff, der aber mit einem One-Pass-Compiler nicht zu machen ist und selbst in komplexeren Systemen zu fast unüberwindbaren Schwierigkeiten führt).

Ein wichtiger Aspekt ist auch die Echtzeitfähigkeit des erzeugten Codes. Dazu müssen die Ablaufzeiten vom Programmierer bestimmt werden können; es bietet sich also nur ein völlig deterministisches Verfahren zum Erzeugen der Bindings an. Nur dadurch verliert der Programmierer nicht die Kontrolle über das, was er da schreibt. C++ hat diese Eigenschaft nicht und ist deshalb für Echtzeitaufgaben nur begrenzt einsatzfähig.

Objekte als Instanzvariablen

Wie dem auch sei, oft kommt man mit klaren Klassenhierarchien gut aus. Geeignete abstrakte Datentypen erlauben es oft, eine echte

Mehrfachvererbung zu umgehen. Notfalls kann man Mehrfachvererbung auch von Hand und mit dem Editor durch Zusammenkopieren der Sourcen erreichen. Eine selektive Auswahl einzelner Methoden von nahe genug verwandten Klassen geht in bigFORTH auch - bisher hat das auch gereicht.

Was allerdings unbedingt nötig ist, sind Objekte als Instanzvariablen in einem anderen Objekt, und zwar sowohl als Zei-

ger als auch direkt. Das wird beim Beispiel der Listen, die ja nur mit Pointern implementiert werden können, deutlich (siehe Listing 3).

Hier wird erst einmal die abstrakte Datenklasse der Listen angelegt; diese brauchen an Daten sowohl einen Zeiger auf das erste als auch auf den Rest der Liste. Dieses kann auch ein normales Datum sein, es sind also "dot pairs" wie in Lisp erlaub. Müßte der Rest wieder eine Liste sein, wird der Typ nicht angegeben; das erzeugt dann einen Pointer auf ein Objekt der aktuell deklarierten Klasse. LISTS PTR NEXT ginge nicht, da die Klasse LISTS zu diesem Zeitpunkt noch nicht fertig definiert ist und deshalb nicht ausgeführt werden kann.

Neben diesen Pointern braucht man natürlich auch noch ein paar Methoden: Eine Liste kann natürlich auch leer sein, also muß man das abfagen können. Ebenso ist oft ganz brauchbar, das erste Element anzuzeigen (mit ?).

Eine Null-Liste ist die leere Liste, auch "nil" genannt. Da das eine Liste ist, kann sie erst später deklariert werden, also wird eine Vorwärts-Referenz angelegt, die sich bei der späteren Definiton von NIL auflöst.

Leere Listen unterscheiden sich in ihrem Verhalten von anderen deutlich. Sie geben auf EMPTY? immer true zurück, es gibt nur eine von ihnen und die darf natürlich nicht gelöscht werden. Ausgegeben wird sie als ein Klammernpaar.

Von der Klasse der leeren Listen wird nun ein Element angelegt, und die Adresse dieses Elements (die die Methode SELF auf den Stack legt) heißt dann endlich NIL. Sowohl erstes als auch nächstes Element der leeren Liste ist wieder die leere Liste. Damit fällt man nicht auf die Nase, wenn man über das Listenende hinausgeht.

Die Methode BIND erlaubt es, echte Objekte an Objekt-Referenzen zu binden (wie der Name sagt). Der Objekt-Pointer FIRST des Objekts (NIL verhält sich, nachdem er gebunden ist, also genauso wie das Objekt, an das er gebunden ist, (NIL selbst. Interessanter wird das aber erst bei echten Listen (siehe Listing 4).

Eine linked list ist natürlich nicht leer. Beim Erzeugen werden die Referenzen FIRST und NEXT gleich gebunden; entsprechende Objekt-Addressen müssen also schon auf dem Stack liegen. Beim Binden

Listing 3

```
\ Data structures: list                                    17nov93py

forward nil
data class lists
public: data ptr first  data ptr next
        method empty?   method ?
how:    : null  nil ;
        : atom? false ;
class;

| lists class nil-class
how:    : empty? true ;
        : dispose ;
        : . ." ()" ;
class;

| nil-class : (nil
(nil self Aconstant nil
nil (nil bind first
nil (nil bind next
```

Listing 4

```
\ Data structures: list                                    17nov93py

lists class linked
how:    : empty? false ;
        : init ( first next - )
                dup >o 1 ref +! o> bind next
                dup >o 1 ref +! o> bind first ;
        : ?     first . ;
        : @     first @ ;
        : !     first ! ;
        : .     self >o  '(
                BEGIN  emit ?  next atom?  next self o> >o
                       IF ."  . " data . o> ." )"  EXIT THEN bl
                       empty?  UNTIL  o> drop ." )" ;
        : #     next # 1+ ;
        : dispose  -1 ref +!  ref F @ 0> 0=
          IF first dispose  next dispose  super dispose THEN ;
class;
```

Listing 5

```
\ Data structures: string                                  30apr93py

int class string
how:    : !     ( addr count - )
                value over 1+ SetHandleSize
                value F @ place ;
        : @     ( - addr count )  value F @ count ;
        : .     @ type ;
        : init  ( addr count - )
          dup 1+ value Handle! ! ;
        : null  S" " new ;
        : dispose  ref F @ 1- 0> 0=
          IF  value HandleOff  THEN  super dispose  ;
class;
```

müssen auch die Reference Counter der Objekte um eins erhöht werden - es zeigt ja jetzt ein Pointer mehr auf sie. Damit sie aktuelles Objekt werden, werden sie auf den Objekt-Stack geschoben, dadurch wird mit REF ihr Reference Counter angesprochen, und nicht der der Liste. Der Objekt-Stack ist übrigens kein richtiger; nur das oberste Element wird in einem Register gehalten, der Rest auf dem Returnstack.

Die Methoden @, ! und ? beziehen sich, wie zu erwarten, auf das jeweils erste Objekt der Liste; sie werden einfach weitergereicht. Keine komplexe Zeigerverwaltung ist notwendig, der Name der Referenz reicht aus.

Zur Ausgabe der Liste muß man sich durch die Liste durchhangeln. Vor dem ersten Element muß eine Klammer geöffnet werden, ansonsten werden die Elemente durch einen Blank getrennt. Das jeweils erste Element der Liste wird ausgegeben. Ist das nächste Element ein Atom, muß es als dot pair ausgegeben werden, die Liste ist damit zu Ende. Auch zu Ende ist sie, wenn das nächste Element die leere Liste ist. Dann muß nur noch die Klammer geschlossen werden und der Blank vom Stack gelöscht werden.

Verblüffend ist die Rekursion in #, das die Länge der Liste zurückgibt. Es wird einfach die Länge des Rests der Liste bestimmt, das um 1 erhöht und die Sache hat sich. Solange die Liste mit NIL oder einem Atom abschließt, das definitiv die Länge 0 hat, terminiert die Rekursion. Hier zeigt sich erstmals klar der Vorteil objektorientierter Programmierung, der viele IF..ELSE..THEN für Fallunterscheidungen überflüssig macht und so sehr einfache Rekursionen erlaubt.

Beim Löschen einer Liste müssen natürlich beide Teile der Liste und der Knoten selbst gelöscht werden. Auch hier wieder sind keinerlei Fallunterscheidungen nötig und die Frage nach der Abbruchbedingung, die bei Rekursionen gerne vergessen wird, stellt sich gar nicht.

Nun brauchen wir noch Element-Objekte für die Liste. Zahlen haben wir ja schon, Strings wären auch noch schön. Man findet Sie in Listing 5.

Wir leiten die Klasse STRING von INT ab. Deren Instanzvariable VALUE verwenden wir hier als Handle, als Zeiger auf einen verschiebbaren Speicherbereich. Dort ist der String dann als counted String gespeichert. Beim Speichern eines neuen

Listing 6

```
\ Data sturctures: pointer                                    30apr93py

data class pointer
public: data ptr container
        method ptr!
how:    : !     container ! ;
        : @     container @ ;
        : .     container . ;
        : #     container # ;
        : init  ( data – )  dup >o 1 ref +! o> bind container ;
        : ptr!  ( data – )  container dispose  init ;
        : dispose  -1 ref +!  ref F @ 0> 0=
          IF  container dispose super dispose  THEN ;
        : null  nil new ;
class;
```

Listing 7

```
\ Data sturctures: array                                      30apr93py

data class array
public: data [] container
        cell var range
how:    : !     ( <value> n – )  container ! ;
        : @     ( n – <value> )  container @ ;
        : .     '[
                # 0 ?DO emit I container . ', LOOP drop ." ]" ;
        : init  ( data n – )  range F !  bind container ;
        : dispose  -1 ref +!  ref F @ 0> 0=
          IF  # 0 ?DO  I container dispose  LOOP
              super dispose  THEN ;
        : null  ( – addr )  nil 0 new ;
        : #     ( – n )     range F @ ;
        : atom? ( – flag )  false ;
class;
```

Strings muß natürlich die Größe des Speicherblocks angepaßt werden; beim erstmaligen Anlegen muß er überhaupt erst angefodert und beim Löschen natürlich wieder freigegeben werden. Alles andere erklärt sich jetzt hoffentlich ziemlich von selbst.

Sehr brauchbar ist auch eine Pointer-Klasse. Zwar kann man Pointervariablen direkt erzeugen, aber nicht z.B. in eine Liste einhängen (siehe Listing 6).

Analog zur Liste wird eine Pointer-Instanzvariable angelegt (CONTAINER), dazu gibt's noch eine Methode PTR!, mit der man ein Objekt zuweisen kann. Die Methoden @, !, . und # werden an den Container durchgereicht. Die Init-Methode bindet ein übergebenes Objekt an den Zeiger. PTR! gibt zuerst das vorherige Objekt frei und speichert anschließend das neue Objekt. Dabei wird natürlich auch das Reference Counting berücksichtigt.

Löschen eines Zeigerobjekts bedeutet ebenfalls, daß auf das Objekt eine Referenz weniger weist (es also ggf. gelöscht werden muß); anschließend wird der Pointer gelöscht.

Analog zu einem Pointer kann man gleich ein ganzes Array von Pointern einbinden (siehe Listing 7).

Analog zur Methode NEW legt man mit NEW[] ein ganzes Array von Objekten an - die dann auch der gleichen Klasse angehören; also voneinander einen konstanten Abstand haben. Über diesen Abstand wird dann die n-te Objektadresse berechnet (die erste liegt bei 0). Die Array-Variable erwartet also eine Indexnummer auf dem Stack.

Tools und Anwendungsbeispiele

Insgesamt ist das Listenpaket jetzt noch nicht sehr einfach zu bedienen. Ich habe deshalb ein paar kleine Tools geschrieben, die den Umgang erleichtern - aber keineswegs ein vollwertiges Lisp oder sowas daraus machen (siehe Listing 8).

CONS und LIST helfen einem beim Anlegen einer Liste. CONS verknüpft zwei auf dem Stack liegende Objekte zu einer Liste (TOS als next, sollte also eine Liste sein; NOS als first der Liste). LIST faßt ein Objekt zusammen mit NIL zu einer Liste zusammen.

CAR und CDR sollten aus Lisp bekannt sein, sie liefern das erste Element bzw. den Rest der Liste.

PRINT ruft die Ausgabemethode eines Objekts auf.

DDROP schließlich entfernt und löscht ein Objekt.

MAKE-STRING ist der String-Konstruktor, analog zu LIST.

Einen festen String konstruiert $". Als Beispiel, wie man mit diesen Tools eine Liste aufbaut, siehe Listing 9.

Listing 8

```
\ Data structure utilities                                    17nov93py

: cons   ( - 1 )    linked new ;
: list   ( o - 1 )  nil cons ;
: car    ( l - o )  >o lists first self o> ;
: cdr    ( l - o )  >o lists next  self o> ;
: print  ( o - )    >o data . o> ;
: ddrop  ( o - )    >o data dispose o> ;
: make-string ( addr len - o )  string new ;
: $" ( - o )
  state @ IF  compile S" compile make-string exit  THEN
  '" parse make-string ;       immediate
```

Listing 9

```
$" Dies" $" ist" $" ein" list cons $" Test" list cons cons  ok
dup print (Dies (ist ein) Test) ok
pointer : test  ok
test . (Dies (ist ein) Test) ok
test # . 3 ok
```

Listing 10

```
Formale Syntax:
<declaration>:::=
   <parent> CLASS <object>
   {[private: |public: ]<creator> <selector> }
   [HOW: { : <method> <coding> ; }]
   CLASS;

<creator> <selector>:::=
   STATIC <static>|METHOD <method>|EARLY <method>
   <number> VAR <var>|<object> (:|[]|PTR) <instance>

<parent>:::=
   OBJECT|<object>

<creation>:::=
   <object> (:|PTR) <instance>|<number> <object> [] <instance>

<coding>:::=
<word> <coding>|{<instance> }<selector> <coding>|
```

Diskussion

Ein richtig eingefleischter Forther hält objektorientierte Programmierung für unnötig komplex. Oder er behauptet, Forth sei durch CREATE..DOES> ohnehin schon objektorientiert. Letzteres stimmt so leider nicht, denn dort gibt es nur eine Nachricht an ein Objekt: Aufgerufen worden zu sein. Daß (mehrstufiges) CREATE..DOES> natürlich mit Werkzeug der Wahl ist, ein objektorientiertes Forth zu implementieren, darf nicht darüber hinwegtäuschen, daß wesentliche Merkmale objektorientierter Programmierung in Forth nicht vorhanden sind.

Im allgemeinen wird objektorientierter Programmierung als Werkzeug für große bis riesige Projekte vorgeschlagen; seien es graphische Benutzeroberflächen, Datenbanken oder Simulationssysteme. Hier wird versprochen, das Problem überhaupt handhaben zu können. Allerdings geht durch die hohe Abstraktion auch das Gefühl für Resourcenverbrauch zurück - will sagen, die Applikationen werden unnötig groß und lahm.

Allerdings haben hier undurchsichtige Compiler auch ihre Finger im Spiel: Wenn niemand nachvollziehen kann, was für Code generiert wird, ist es nicht verwunderlich, daß langsame und große Programme herauskommen. Der Code, den das OOF für bigFORTH generiert, ist leicht nachvollziehbar; da alle wesentlichen Operationen als Makros compiliert werden, ist man sich auch sicher, daß es mit der Geschwindigkeit stimmt.

Speicher und Rechenzeit sollen im üblichen Forth-Métier Embedded Control ja auch so gut wie möglich eingesetzt werden. Monster kann man sich gar nicht leisten.

Auch gibt es im Embedded Control-Bereich genügend Anwendungen, die nicht sehr komplex sind. Solche Probleme kann man problemlos mit normalem FORTH bewältigen, und das sollte man dann auch. Auch wird kaum ein Forther unbedingt ein Lisp implementieren (wobei es mit dem, was im Beispiel steht, noch lange nicht getan ist). Aber eine Sprache (oder eine Spracherweiterung) sollte man nicht nur an den Beispielen und "toy programs" bewerten, sondern an den Anwendungen.

Und hier existiert unter bigFORTH's OOF tatsächlich etwas, das mehr ist als nur ein "toy program"; ein Chromatographie-System von Ewald Rieger. Dessen Sourcen haben einen Umfang von ca. 200 KB. Das ist keine Aufgabe, die man mit FORTH und einen Z80 machen kann (wobei die Betonung auf dem Z80 liegt). Deshalb läuft das Ganze auch auf VME-Bus-Rechnern mit einem MC 68000. Diese Hardware ist auch Mindestvoraussetzung für bigFORTH.

Natürlich wäre es besser, wenn Ewald Rieger sein System selbst vorstellen würde; mangels entsprechender Hardware (der eines Chemie-Labors) kann ich mir auch kein vollständiges Bild machen. Grob ist die Problemstellung etwa die: Man hat

einen Haufen mit Schläuchen verbundener Pumpen, Ventile, Behälter, die zu trennende Stoffgemenge abgeben und andere Behälter, die die Produkte aufnehmen. Der Trennvorgang muß natürlich überwacht werden, damit die Software auch regeln kann.

Die Zuordnung von realen Objekten (Pumpen, Ventile, Meßgeräte...) zu den Programmobjekten ist bei dieser Problemstellung relativ einfach. Die Objekte werden dann über Pointer genauso verknüpft wie die Geräte mit Schläuchen. Dazu tippt der Benutzer etwa Pumpe1 Ausgang mit Ventil2 Eingang verbinden ein; Pumpe1 und Ventil2 sind Objekte, Eingang und Ausgang sind Instanzvariablen, mit und verbinden zugehörige Methoden. Da sag' noch einer, FORTH sei schlecht lesbar.

Neben der Lesbarkeit, die FORTH den Ruf einer Write-Only-Sprache nehmen könnte, spielt es natürlich eine Rolle, daß gerade der Bezug zur Realität, zu realen Objekten im Bereich der Embedded Control größer ist, als bei anderen Anwendungen. Sei es die Hardware, deren I/O-Register man programmieren muß, oder die Pumpe, die man damit (recht indirekt) steuert. Man hat also hier tatsächlich verschiedene Abstraktionsebenen und das schreit förmlich nach objektorientierter Programmierung.

Natürlich sollte man die graphischen Benutzeroberflächen nicht vergessen, die auch an FORTH nicht spurlos vorübergehen. Derzeit entsteht bei mir auf Basis der hier vorgestellten Spracherweitertung eine Maus&Fenster-Oberfläche für bigFORTH 386 (im Textmodus). Das soll kein Windows-Konkurrent werden, aber immerhin einen mit der Borland-Oberfläche vergleichbaren Komfort bieten. Derzeit gibt es 48 verschiedene Objektklassen, vom einfachen Button bis zum Fenster, Menü, Terminal oder gar zwei Editoren (einer für Blöcke, einer für Text-Dateien). Auch hier sind schon 80 KB Sourcen vorhanden (ohne die Editoren), also durchaus ein praxisgerechtes Produkt, das aber an anderer Stelle vorgestellt werden soll.

Syntax

Yerk und andere objektorientierte Forth-Systeme verwenden eine "Methode Objekt"-Syntax. Damit geht Forth hier wieder mal einen Sonderweg - praktisch alle anderen objektorientierten Sprachen, von Simula an, schreiben "Objekt Methode", evtl. mit einem anderen Trenner als dem Leerzeichen.

Als Vorteil hat man eine vereinfachte Compilation, da die Methode lediglich eine Nummer auf den Stack legt und das Objekt eine immer gleiche Auswertung vornimmt. Erheblicher Nachteil ist dagegen, daß Methodennamen nun global sind: Man kann auf alle Methoden eines Objekts zugreifen (Verlust von Information hiding), und muß sich bei der Wahl der Methodennamen vor Kollisionen hüten. [2] zeigt aber, daß auch die von mir gewählte Syntax eine einfachste Implementierung erlaubt. Wie schon in Dick Pountains OOF werden in bigFORTH die Methodennamen in getrennten Vokabularen verwaltet und sind nur über die Klasse zugänglich - ebenso wie die Objektvariablen. Dadurch ist eine vollständige Kapselung erreicht; Fehler durch falsche Methoden-Bezeichner können schon zur Compile-Zeit entdeckt werden, auf private Methoden kann gar nicht zugegriffen werden.

Damit wären wir am Ende des Artikels angelangt. Weil das OOF für bigFORTH ein recht umfangreiches System ist, konnte nicht alles gezeigt werden. Auch wie das Ganze implementiert ist, bleibt vorerst ein "Geheimnis".

Ich hoffe, daß auch andere kommerzielle objektorientierte Forth-Systeme hier vorgestellt werden (falls es sie noch gibt?); PD-Lösungen haben zwar auch einen Reiz, aber nur ihre Berechtigung, wenn sie sich im Vergleich mit teureren behaupten, (nämlich darin, für "added value" zu sorgen).

Literatur

[1] Dick Pountain; *"Object-Oriented Forth"*; Academic Press 1987

[2] John R. Hayes; *"Object-Oriented Programming for Small Systems"*; E-Mail Distribution

[3] Rick Grehan; *"Yerk kommt zum PC"*; Vierte Dimension 1/1992(14)

[4] Burkhard Golf, Rolf Schönlau, Franz Angenendt, Klaus W. Pleßmann; *"Forth++ (1)"*; Vierte Dimension 1/1993(4)

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

Ingenieurbüro
Dipl.-Ing. Wolfgang Allinger
Tel. (+Fax.) 0212/66811
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

DELTA t GmbH
Tel.: 040/280152-0 (Fax: 280152-90)
Adenauer Allee 54
D-20097 Hamburg

Programmierung von Messgeräten und vernetzen Systemen, Implementation serieller Protokolle, Entwicklung von Spezial-Prozessoren / ASICS.

Diessner Datentechnik
Tel.: 07031/289538 (Fax: 289541)
Furtwanger Str. 9
D-71034 Böblingen

EuroCoCoS-532 - Integriertes Forth-System mit Evaluierungsboard (Hitachi H8/532), Schulungen und kundenspezifische Entwicklungen.

ETA Elektrotechnische Apparate GmbH
Tel.: 09187/10-0 (Fax: 10-397)
Industriestr. 2-8
D-90518 Altdorf (b. Nürnberg)

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

Fortsetzung Seite 26!

ECHTZEIT Programmier-Wettbewerb

Zum vierten Mal wurde während der diesjährigen ECHTZEIT-Messe (14.-16. Juni 1994 im CCH Hamburg) ein ECHTZEIT-Programmier-Wettbewerb durchgeführt. Die Idee geht zurück auf die Real Time Convention 1989 in Anaheim/Kalifornien, wo erstmals eine Austragung auf internationaler Basis stattfand.

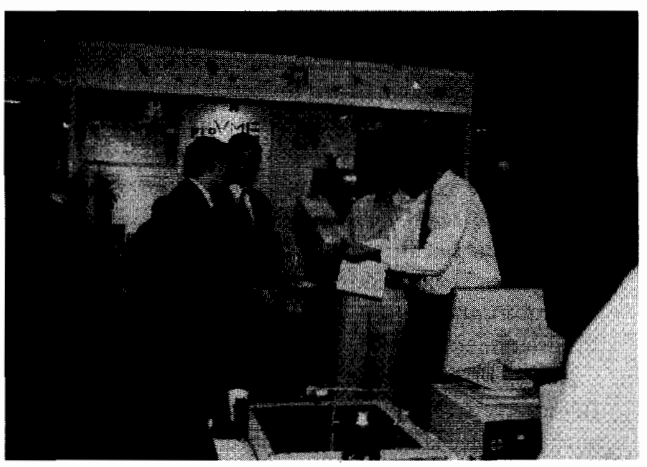
Wettbewerbsziel war es, eine bis zum Beginn des Wettbewerbs unbekannte Echtzeit-Programmieraufgabe schnellstmöglich zu lösen. Dazu standen maximal dreieinhalb Stunden zur Verfügung.

Es ist erlaubt, beliebige Datenverarbeitungssysteme zur Lösung heranzuziehen, im Extremfall auch ein Workstation-Cluster oder ein Cray-Uplink. Auch die Wahl der Software-Werkzeuge ist freigestellt. Das mechanische Modell hingegen, das der Aufgabe zugrunde liegt, darf nicht verändert oder modifiziert werden. Teilnehmen können einzelne Personen oder Teams bestehend aus bis zu drei Personen. Über die Teilnahme wird nach dem Prinzip "first come, first serve" entschieden. Es gewinnt der Teilnehmer oder das Team, das die gestellte Aufgabe als erstes vollständig gelöst hat und die Lösung demonstriert hat.

Können innerhalb der vorgegebenen Zeit nicht alle Teilnehmer eine vollständige Lösung vorweisen, werden die Plätze nach Umfang der vorhandenen Teillösungen vergeben. Es können maximal 10 Teams am Wettbewerb teilnehmen.

Die Aufgabe wurde gestellt vom Sieger des letztjährigen Wettbewerbs in Karlsruhe, Herrn Ulrich Hoffmann, einer der drei Direktoren der FORTH-Gesellschaft.

□



Von links nach rechts:
1.Platz:
FORTecH Software
Dr. - Ing. Egmont Woitzel
Udo Schütz

Die Organisatoren:
Ulrich Hoffmann
Andreas Dobbertim

2.Platz:
Jäger Software
Hubert Jäger
Andreas Kraus

3.Platz:
Forth-Gesellschaft
Jörg Plewe
Jörg Staben

Programmierwettbewerb Echtzeit '94

von Dr.-Ing. Egmont Woitzel

Ein Teilnehmer des diesjährigen Echtzeitwettbwerbs berichtet über die steinigen Wege zum Ziel und das knifflige Modell

Eigentlich handelt es sich nur um einen Zufall, daß die FORTecH Software GmbH an dem diesjährigen Programmierwettbewerb auf der Echtzeit '94 teilnehmen konnte. Verleitet durch die Mitteilung der Messeveranstalter, daß das Teilnehmerfeld auf 15 Teams vergrößert werden sollte, meldeten wir uns erst in allerletzter Minute an - und konnten nur noch den 2. Reserveplatz buchen. Wir erhielten zwar die Ausschreibung, aber mit einer Teilnahme rechneten wir nicht mehr. Unsere "Wettkampfvorbereitungen" beschränkten sich auf ein Minimum, wir löteten der Ausschreibung entsprechend ein Kabel zusammen und packten einen Rechner ein.

Geheimnisvolle Pusteblume

Ganz konnten wir uns dem Reiz des Programmierwettbewerbs aber auch nicht entziehen. Die in der Ausschreibung enthaltene I-O-Beschreibung und der Name "Pusteblume" des Modells gaben genug Raum für jede Art von Spekulation. Mit zwei Ausgängen und fünf Eingängen (aus Modellsicht) und einer geforderten Abtastrate von 1 kHz fiel es in die Kategorie der über ein IBM-Printerport zu steuerndens Modell - was von den Ausrichtern vermutlich auch beabsichtigt wurde.

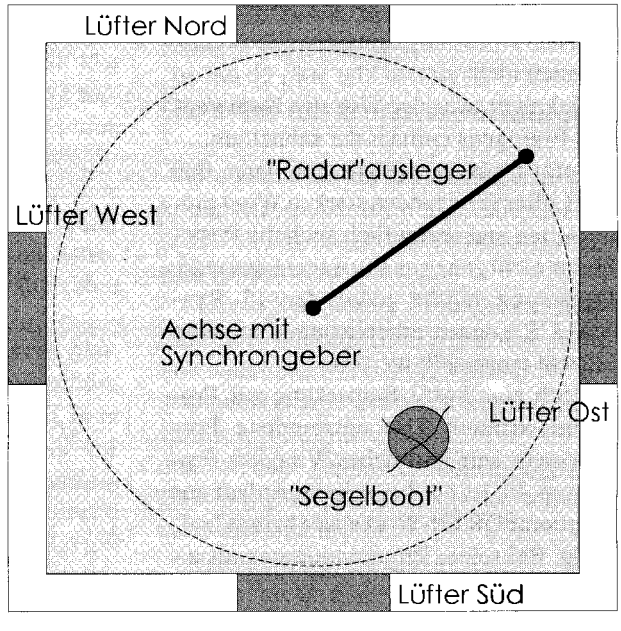
Ganz mysteriös wurde das Modell durch die benötigten "Zutaten". Die Stromversorgung 12 Volt/1,5 A lag im erwarteten Rahmen, aber die Notwendigkeit für eine zusätzliche 9 V-Trockenbatterie und ein Teelicht(!) las sich eher wie ein Plan von Egon Olsen.

Am Mittag des 15. Juni näherte sich unsere Spannung dem Höhepunkt. Nachdem wir von Uli Hoffmann ganz kurz vor Messebeginn davon unterrichtet wurden, daß unser Team auf den Reserveplatz 1 vorrücken konnte, blieb am Wettbewerbstag der 10. Platz tatsächlich leer, so daß es ganz plötzlich richtig ernst wurde. 10 Minuten vor dem Wettbewerbsstart erhielten wir grünes Licht.

Runde Segelboote und stumpfes Wasser

Die "Pusteblume" entpuppte sich als ein ca. 50cm x 50cm großes Brett, auf dem 4 Lüfter und eine sonderbare drehbare Anlage montiert war (siehe Abb.). Die Lüfter ließen sich über 4 der 5 Eingänge einzeln Ein- und Ausschalten. Mit dem 5. Eingang ließ sich der in der ausgeteilten Beschreibung als Radar bezeichnete Ausleger über einen Antrieb im Uhrzeigersinn drehen. Eine Gabellichtschranke unter dem Brett (wo auch der Antrieb zu finden war), lieferte pro Umdrehung einen Synchronimpuls. Bei unserem Modell lag dieser Punkt fast genau im Süden. Der Ausleger bestand aus Messingrohr, das zu einer Art drehbarem Tor zusammengelötet war, welches dicht über dem Boden eine Infrarot-Lichtschranke trug.

Von dem rätselhaften Teelicht wurde in dem Szenarium nur die Blechhülle und nicht die Kerze benötigt. Die Hülle diente als "Schiffsrumpf" für ein "Segelboot", das dreimal im Kreis um die Mittelachse herum gepustet werden sollte. Die Ausstattung des Bootes mit einem Segel war den Teilnehmern überlassen. Wir benutzten wie die meisten anderen Teilnehmer auch zwei gekreuzte Blätter Papier. Als eines der größten Probleme erwies sich die gegenüber der Gleitreibung sehr hohe Haftreibung des Bootes auf der das Wasser darstellenden blauen Folie, mit der das Brett beklebt war. Man konnte das Boot aus dem Stand nur mit einem "Orkan" in Bewegung bringen, der es dann aber "aus dem Wasser" fegte. Dagegen brachte erst ein Stück Tesa-Pack unter dem Boot Abhilfe.



Software-Wirbelwind

Für die Steuerung der Pusteblume verwendeten wir zum Erstaunen einiger der zuschauenden Echtzeit-Besucher die Beta-Version von comFORTH für Windows, dessen grafische Bedienoberfläche uns das nötige Entwicklungstempo erlaubte - nach etwa 2½...3 h hatten wir die Aufgabe gelöst.

Wenn man das in dieser Zeit geschriebene Programm durchsieht, dürfte es mit Ausnahme des benutzten Windows-Multimedia-Timers eigentlich kaum Verständnisprobleme geben. Die Basis für die ersten Versuche bildeten die Befehle zur Steuerung der Motoren, die durch die Worte +M und -M voneinander entkoppelt wurden.

Zur Bestimmung der Winkel-Position des Bootes verwendeten wir das einfachste mögliche Verfahren. In einer in 1-ms-Abständen zyklisch aufgerufenen Interruptserviceroutine (um etwas anderes han-

delt es sich bei dem benutzten Mulitmedia-Timer-Callback im Prinzip nicht) wird die Zählzelle 'W inkrementiert. Die Positionsbestimmung erfolgt dann durch POS im Pollingverfahren. Am Synchronpunkt wird die Zählzelle auf Null gesetzt, bei der nächsten Detektion des Bootes kann der Winkel dann einfach ausgelesen werden. Nachteilig ist bei diesem Verfahren, daß man im "worst-case" fast zwei Umrundungen des Radars abwarten muß, bis der Meßwert vorliegt, das waren bei unserem Modell etwa 4 s.

An der Implementierung des Callbacks kann man übrigens gut sehen, wie einfach die Bedienung eines 'C'-Interface in Assemblercode ist. Normalerweise erwartet das Multmedia-Interface an dieser Stelle eine 'C'-Funktion. Die bei Start und Stop aufzurufenden Windows-Funktionen sind in den Worten +W und -W gekapselt worden.

Etwas mehr Probieren erforderte das gezielte Herumpusten des Bootes. Es zeigte sich, daß man für eine zielgerichtete Bewegung des Bootes nicht nur mit einzelnen Lüftern pusten durfte, sondern sowohl mehrere Lüfter kombinieren als auch das nichtlineare Hochlaufverhalten berücksichten mußte. Als Basisschritt führt PFF einen einfachen Mehrlüfter-Puster aus. Das "um-die-Kurve-Pusten" besteht aus zwei zeitlich gegeneinander verschobenen "gerade-aus" und "nach-rechts"-Pustern. Die Algorithmen können sicherlich auch viel eleganter mit Hilfe von Tabellen und CREATE-DOES>-Konstruktionen implementiert werden, die vierfache Wiederholung fordert das geradezu heraus. In der Hektik des Programmierwettbewerbs, als noch nicht einmal klar war, ob das so funktioniert, war der Weg des Kopierens und Einfügens einfach der schnellere.

Beim Probieren zeigte sich dann, daß alle Lüfter verschieden starken Wind produzierten und vermutlich auch die Pusteblume nicht ganz genau waagerecht stand. Deswegen mußte jeder der 12 Pxx-VALUE's unter erheblichem Probieren einzeln eingestellt werden.

Noch eine kurze Bemerkung zur Programmstruktur. Das entstandene Programm ist kein wirkliches Windows-Programm. Es ist nur bei seinem Aufruf aus dem comFORTH-Workspace heraus lauffähig. Bei seiner Programmierung als ei-

```
\ ============================================================
\ Programmierwettbewerb Echtzeit '94
\ Die "Pusteblume" von Andreas Dobbertin & Uli Hoffmann
\
\ Egmont Woitzel und Udo Schütz
\ FORTecH Software GmbH
\
\ Rechner und Software:
\ PC-AT 486DX33 mit MS-DOS_ 6.0 und Windows_ 3.1
\ comFORTH für Windows (Beta II)
\ ============================================================

\ - Utilities

\NEEDS CODE @:ASM86 1 INCLUDE


\ - Ports

HEX
: PB@ ( ps: ===> 8b )( Pusteblume auslesen )
      379 CP@ ;

: PB! ( ps: 8b ===> )( Pusteblume beschreiben )
      378 CP! ;
DECIMAL


\ - Inputs

HEX
: RADAR? ( ps: ===> ? )( ?t, wenn "Segelboot" detektiert )
      PB@ 40 AND 0= ;

: SYNC? ( ps: ===> ? )( ?t, wenn Synchronposition erreicht )
      PB@ 40 AND 0<> ;
DECIMAL


\ - Outputs

HEX
 2 CONSTANT OST   \ Bitmaske für Windmaschine im Osten
 4 CONSTANT SUED  \ Bitmaske für Windmaschine im Süden
 8 CONSTANT WEST  \ Bitmaske für Windmaschine im Westen
10 CONSTANT NORD  \ Bitmaske für Windmaschine im Norden
DECIMAL

VARIABLE O ( ps: ===> addr )( enthält aktuellen Output )
      O OFF \ initialisieren

: +M ( ps: mask ===> )( schaltet die Motoren der Maske dazu )
      O SET  O @ PB! ;

: -M ( ps: mask ===> )( schaltet die Motoren der Maske ab )
      O RESET  O @ PB! ;

: +R ( ps: ===> )( schaltet den Radarmotor dazu )
      1 +M ;

: -R ( ps: ===> )( schaltet den Radarmotor ab )
      1 -M ;


\ - Winkelmessung

VARIABLE 'W ( ps: ===> addr )( enthält aktuellen Radarwinkel )

LABEL WCB ( ps: ===> addr )( enthält Code für den Windows- )
      ( Multimedia-Timer-Callback, liefert eigenen Offset )
      ( im Callback wird einfach nur 'W inkrementiert )
      ( auf dem Stapel liegen folgende Parameter: )
```

```
      ( wTimerID-16b wMsg-16b dwUser-32b dw1-32b dw2-32b )
      ( diese müssen mit der Rückkehr weggepopt werden )

      \ CPU in Ordnung bringen
      AX PUSH,                \ Register retten
      DS PUSH,
      CS: 'DSEG #) AX MOV,    \ Selektor des Forth-DSEG
      AX DS MOV,        \ als DS verwenden

      \ die eigentliche Callback-Aktion
      'W #) INC,

      \ CPU wieder aufräumen
      DS POP,                 \ Register zurückholen
      AX POP,
      16 # FAR RET,           \ Stack-Frame abräumen
      END-CODE

0 VALUE hEvent

: +W ( ps: ===> )( Winkelmessung aktivieren )
      1 timeBeginPeriod DROP        \ ms-Timer starten
      1 1 CSEG WCB 0, 1 timeSetEvent      \ WCB zuordnen
      TO hEvent ;             \ Handle merken

: -W ( ps: ===> )( Winkelmessung deaktivieren )
      hEvent timeKillEvent DROP           \ WCB abmelden
      1 timeEndPeriod DROP ;        \ Timer stoppen


\ - Positionsbestimmung

: POS ( ps: ===> +n )( Winkel des "Segelboots" bestimmen )
      ( Das Radar muß laufen und das Segelboot sehen! )
      BEGIN  SYNC?  UNTIL     \ warten bis Synchronpunkt
      'W OFF                  \ dort ist Winkel Null
      BEGIN  RADAR?  UNTIL    \ warten auf "Segelboot"
      'W @ ;                  \ Winkel auslesen

\ die in den nachfolgenden Worten benutzten Zahlen wurden
\ online "handvermessen", indem die Lüfter ausgeschaltet
\ wurden, das Seegelboot an die Grenzpositionen geschoben
\ und POS ausgeführt wurde - keine weiteren Umrechnungen
HEX
: SUED? ( ps: +n ===> ? )( ?t, wenn Segelboot im Südwasser )
      DUP 1B4 < SWAP 626 > OR ;

: WEST? ( ps: +n ===> ? )( ?t, wenn Segelboot im Westwasser )
      3FE < ;

: NORD? ( ps: +n ===> ? )( ?t, wenn Segelboot im Nordwasser )
      1B5 626 WITHIN? ;

: OST? ( ps: +n ===> ? )( ?t, wenn Segelboot im Ostwasser )
      3FE > ;
DECIMAL


\ - Drehende Winde

\ Puste-Zeitkonstanten für jede einzelne Richtung und Phase
\ leider lagen die originalen Werte Hand-eingestellt nur
\ im RAM: diese Zahlen sind daher nur grobe Richtwerte,
\ die Werte der einzelnen Phasen waren alle verschieden

500 VALUE PO1     \ Puste nach Osten
300 VALUE PO2
250 VALUE PO3

500 VALUE PW1     \ Puste nach Westen
300 VALUE PW2
250 VALUE PW3
```

genes Applikations-Fenster wäre ein größerer Aufwand nötig gewesen. Da die Worte KEY? und MS nicht in Message-Handlern benutzt werden dürfen, hätte vor allem die Ablaufsteuerung radikal geändert werden müssen. Aber das war ja nicht unbedingt nötig...

PS:

Das Quellprogramm wurde der besseren Verständlichkeit wegen für den Abdruck noch einmal nachbearbeitet. In der Hektik des Programmierwettbewerbs verzichteten wir fast völlig auf Kommentare. Die verwendeten Wortnamen und ihre Codierung wurden jedoch nicht geändert.

□

```
500 VALUE PN1      \ Puste nach Norden
300 VALUE PN2
250 VALUE PN3

500 VALUE PS1      \ Puste nach Süden
300 VALUE PS2
250 VALUE PS3

: PFF ( ps: mask u ===> )( einen u-ms-Puster loslassen )
      OVER +M           \ Lüfter dazuschalten
      MS          \ warten
      -M ;        \ und Lüfter wieder wegschalten

: >NORD ( ps: ===> )( Segelboot von West nach Nord pusten )
      BEGIN SUED        PN1 PFF
            SUED WEST OR      PN2 PFF
            WEST        PN3 PFF
            POS NORD?  KEY? OR
      UNTIL ;

: >WEST ( ps: ===> )( Segelboot von Süd nach West pusten )
      BEGIN OST         PW1 PFF
            OST SUED OR PW2 PFF
            SUED        PW3 PFF
            POS WEST?  KEY? OR
      UNTIL ;

: >SUED ( ps: ===> )( Segelboot von Ost nach Süd pusten )
      BEGIN NORD        PS1 PFF
            NORD OST OR PS2 PFF
            OST         PS3 PFF
            POS SUED?  KEY? OR
      UNTIL ;

: >OST ( ps: ===> )( Segelboot von Nord nach Ost pusten )
      BEGIN WEST        PO1 PFF
            WEST NORD OR      PO2 PFF
            NORD        PO3 PFF
            POS OST?  KEY? OR
      UNTIL ;


\ - Herumpusten

: T ( ps: ===> )( eigentlich nur als Test gedacht, )
      ( tat diese Schleife schon alles nötige )
      TRUE -M
      +R +W
      BEGIN \ einmal im Kreis herumpusten
            >NORD >OST >SUED >WEST
            KEY?
      UNTIL
      KEY DROP
      -R -W ;
```


68HC711E9

Singlechip Entwicklungskit

- **kurze Signalleitungen**, das Modul ist direkt in eine 68HC711E9-Fassung einsteckbar
- alle **40 IO-Leitungen** des 68HC711E9 verfügbar
- **Hintergrunddebugg** über 2. UART mit 57k Baud unabhängig vom 68HC11 Takt.
- **SCI**-Schittstelle für den Anwender frei
- **Debugg-RAM** für 68HC11 schreibgeschützt

Bestückt mit 68HC711E9, 68HC24, 32kByte RAM, HC573, HC00, HC138, COM81C17 und MAX232. Anschluß via 52pol. PLCC-Adapter. Größe 32mm·67mm. Debugger und Handbuch.

68HC711E9 Singlechip Entwicklungskit, kompl.	**499,- DM**
68HC11F1-Board, komplett	**299,- DM**
Debugger MONI11A1(f. A-,E-,L-Typ), Debugger MONI11F1 (f. F1-Typ), Debugger MONI11K4 (f. K-, N-Typ)	**je 99,-DM**
Handbuch zu MONI11.. und TCOM6811	**49,- DM**

Die Software ist auf PC/XT/AT lauffähig. Zu der Software wird der Forth-Compiler TCOM6811(Library v. H. Dyja) mitgeliefert.
Alle Preise incl. Mwst. ohne Porto und Verpackung.

Holger Dyja Naumannstr. 13 10829 Berlin
Tel./Fax. 030 / 784 12 57

Der Dallas DS80C320

von Wolfgang Schemmert
Strahlenbergstr. 123, 63067 Offenbach/Main

"Die Entdeckung des Dallas DS80C320 Mikrocontrollers bereitet mir gewissermaßen so etwas wie eine klammheimliche Freude:

Als ich vor ca. 1 1/2 Jahren anfing, mich für Forth auf Mikrocontrollern zu interessieren, haben mir verschiedene Experten davon abgeraten, so etwas auf einem 8032-Prozessor zu versuchen, der sei eigentlich Forth-untauglich.

Mit dem schnellen 8032-kompatiblen Mikrocontroller DS80C320 von Dallas Semiconductor und meinem subroutinengefädelten Forth-Kern "SST>" erreiche ich mittlerweile eine für 8-Bit-Controller ganz respektable Geschwindigkeit: 3 Sekunden nach dem "Deliano"- Benchmark (voll interaktiv, also keine optimierte Vorab- Kompilation von Forth-Source in Maschinensprache-Runtime).

Trotzdem ein wenig der Reihe nach: Der DS80C320 ist eine pinkompatible Weiterentwicklung des 8032 Mikrocontrollers, verfügbar in 40 Pin DIL, 44 Pin PLCC und Flatpack. Eine Übersicht der wichtigsten technischen Parameter findet sich auch im Elrad-Heft 7/94 S.14. Die Highlights hier nur kurz angerissen: Zweite vollwertige serielle Schnittstelle **onBoard**! Verbesserte (fast für diese CPU-Größenordnung überdimensionierte) Interrupt-Priorisierung, Watchdog-Timer, zweiter "langer" Datenzeiger (der sich vor allem positiv bei der Optimierung des Textinterpreters und bei MOVE-Worten bewährt).

Die übrigen Spezielfunktionen, insbesondere die 3 programmiebaren Timer, entsprechen dem Original 8032. Was ein wenig schmerzt, aber vielleicht ja noch kommen kann: Ein Hauptvorteil der 8032-Familie besteht in der gemessen am Kostenaufwand extrem effizienten Programmorganisation bei kleinen Projekten, die sich überwiegend auf die eingebauten Resourcen beschränken. Dazu gibt es eine hervorragende Palette von funktional erweiterten CPU's wie 80C537, 80C552 ... Gerade wenn man die 2.serielle Schnittstelle des DS80C320 nutzen will, ist seine eingebaute Peripherie auf eine annähernd wertlose Restgöße geschrumpft.

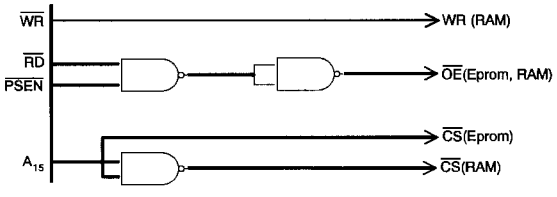
Wodurch wird der 80C320 nun schneller?

Zunächst durch Verbesserung der internen Organisation des Maschinenzyklus: Bei gleichem Quarz schafft er den 1,5 bis 3-fachen MIPS-Wert wie ein Original-8032. Das heißt: 8032 raus aus der Platine, DS80C320 rein, und das ganze läuft (bezogen auf Forth-typische Maschinencode-Strukturen) etwa doppelt so schnell. (Auf den ersten Blick verheißt die Dallas-Werbung eine Verdreifachung des Datendurchsatzes, bei genauerem Studium der Datenblätter fällt jedoch auf, daß insbesondere die bei Forth wichtigen "_jmp" und "_call" Befehle nur 1,5 mal schneller laufen.)

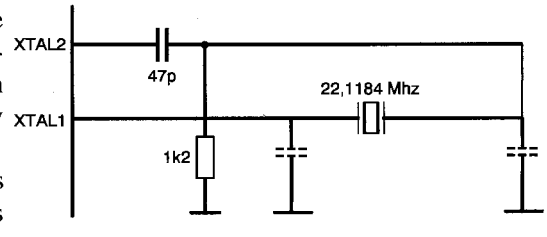
Doch damit nicht Genug der Beschleunigung: Der DS80C320 ist ausgelegt für Quarzfrequenzen bis 25 MHz. Zur Erzielung der Standard-Baudraten muß man sich leider mit einem 22,1184 MHz Quarz zufrieden geben. Gedacht-getan, Quarz bei Conrad geholt und eingelötet. PLEITE. Der Oszillator schwingt auf müden 7,373 MHz. Das Dallas-Datenblatt gibt sich seriös: Quarz: AT-Parallel-Schnitt (woher nehmen?). Da spürt der Mensch plötzlich, daß es nicht völlig nutzlos war, in der Jugend mal Sender gebastelt zu haben: Das war wohl ein Oberton-Quarz, vielleicht einer der auf der 3.Oberwelle erregt werden möchte. Also Hochpaß-Filter einbauen. Mit einem seriös berechneten Filter schwingt der Oszillator leider auf 36,864 MHz. Nach einigem Probieren klappts dann aber mit der Schaltung Bild 1. Sicherheitshalber sollte man aber bei einem Platinenlayout noch die gestrichelten Abgleichkondensatoren mit einplanen und die Werte in Bild 1 nicht für allzu repräsentativ nehmen.

Der schnellere Buszyklus des 80C320 stellt natürlich auch etwas höhere Ansprüche an die am Bus angeschlossenen Peripheriebausteine. Der stolze Besitzer eines Feldwaldundwiesen-RAM-Bausteins bekommt jedenfalls beim ersten Studium des Datenblatts ein leichtes Schaudern, die Anschaffung eines Cache-Rams wird erwogen. Erweist sich als überflüssig. Der Baustein läuft in der Default- Einstellung wunderbar mit einem 100ns-RAM und 150ns-EPROM. Für langsamere Peripheriebausteine lassen sich über ein Special Function Register Waitstates der Pseudo- Harvard- Struktur des DS80C320 programmieren (allerdings nur für den Daten- nicht für den Programmzugriff). Als Resultat bei maximaler Verzögerung wird das Forth allerdings nur ca. 10% langsamer.

Ein weiteres, potentielles Forth-typisches Hardwareproblem sollte nicht unerwähnt bleiben: die Konfiguration des DS80C320 als v.Neumann Maschine. Als ökologisch orientierter GAL-Verächter hatte ich zuvor einfacherweise den /RD und /PSEN- Takt des 80C32 mit einer Widerstands-Dioden-Logik zusammengefaßt. (Siehe A.Roth, Microcontroller Kochbuch, S.5-40). Diese Methode ist bei den optimierten Buszugriffen des DS80C320 zu lahm, es muß eine HCMOS-Logik her (siehe Bild 2 für eine minimale v.Neumann Konfiguration mit 32 k EPROM und 32 k RAM).



Der Baustein wird geliefert von Atlantik-Elektronik, Postfach 1261,82141 Planegg, Tel. 089-857000-0. Abgabe lt. Firmenangabe nur in vollen Verpackungseinheiten (DIL40 = 9 Stck)"



WordWriter 94

- eine fleißige elektronische Schreibkraft

Dr. H.-J. Haase, Kinzerallee 27, 12555 Berlin
EMail: haase@svt.wtza-berlin.de

Vision

Seit Jahren sitze ich am Rechner und bediene ihn. Dabei drängen sich einige Fragen auf. Wieso bedient der Computer eigentlich nicht mich? Die Eingaben erfolgen meist über die Tastatur. Texte schreibe ich Wort für Wort. Nein, nicht Wort für Wort, sondern vielmehr Buchstaben für Buchstaben. Wenn ich mich irgendwo verschrieben habe, dann hagelt es vielleicht eine Fehlermeldung. Dabei müßte diese Maschine die meisten Eingaben inzwischen schon auswendig kennen, sooft wie diese sich wiederholen. Warum hilft mir dieses Gerät nicht, anstatt zu meckern, beim Schreiben?

Es soll da ja schon einige Lösungen geben: Man kann einen Schreibmaschinenkurs belegen und Weltmeister im Tippen werden. Es läßt sich eine Schreibkraft engagieren und diese mit Schonkissen zur Handauflage unterstützen. Weiter sind eine Textverarbeitung und eine graphische Bedienoberfläche einsetzbar, wobei sich der Nutzer an den vielen Ikons, der Sanduhr und den Megabytes erfreuen darf, nachdem er die Handbücher gelesen hat.

Wordwriter

Es könnte auch anders gehen. Ein Forthler nimmt FPC und entwirft ein Programm. Dieses heißt WordWriter und spart bis zu 70 % Tipparbeit. Die Funktionalität von FPC soll maximal erhalten bleiben, also keine neue Variante, die die Funktionstasten vertauscht. WordWriter zeichnet sich durch eine einfache Bedienung aus. Das Menü öffnet sich, sobald die Maus berührt wird. Falls Sie keine Maus haben, geht es auch nur mit der Tastatur. Vor(/ Forth)kenntnisse sind nicht erforderlich. Es existiert eine deutsche Programmversion (Shareware). Umlaute sind kein Problem (Dank Claus Voigt). WordWriter ist einfach, in Minuten mit der Hypertexthilfe am Beispiel erlernbar und besteht hauptsächlich aus den Menüpunkten Lernen, Schreiben und Ende.

Lernen

Damit WordWriter Sie gut unterstützt, kann er lernen. Mit LERNEN liest der Computer das bei Ihnen vorhandene Dokument und lernt die darin enthaltenen Wörter. Gleichzeitig wird eine Analyse durchgeführt und die ausgefilterten Wörter werden mit ihrer Häufigkeit des Vorkommens in eine aktuelle Datei WORTE.SEQ gespeichert. So entsteht ein kleines Wörterbuch. Will man die Ausgabedatei später wiederverwenden, so benennt man sie um. Auf diese Weise sind direkt die Resultate mit oder ohne Berücksichtigung der Häufigkeit lernbar. Vorteile ergeben sich durch die Archivierung kleiner themenbezogener Dateien, in denen die gefilterten Resultate mehrerer Quellen enthalten sind. Das Lernen selbst und die Filterfunktionen sind einstellbar. Alles ist lernbar; Sprachen, Zeichen, Quelltexte und das Verhalten.

Schreiben

Beim Schreiben sucht ihr Computer während der Texteingabe bei jedem Zeichen nach noch vorhandenen Wörtern. Er zeigt je einen Vorschlag von einer noch vorhandenen Auswahl in der unteren Zeile an. Wenn Sie diese Anzeige unberücksichtigt lassen, beenden Sie das Wort wie immer mit der Leertaste. Bei der Auswahl werden drei Kriterien berücksichtigt, die Wortlänge, die Worthäufigkeit und die Reihenfolge der Wörter im aktuellen Text. So werden die zuerst angebotenen Wörter auch von der Reihenfolge Ihrer Eingaben bestimmt.

Betätigen Sie jedoch die Taste <ENTER> wird der noch fehlende Teil automatisch und richtig ergänzt. Sie können sich die Eingabe aller weiteren Buchstaben ersparen einschließlich des Leerzeichens am Wortende. Tritt ein Satzzeichen auf, so kann WordWriter selbst ein Leerzeichen ausgeben und wenn es erforderlich ist, ohne Shifttaste groß schreiben. Wie viele andere Optionen auch, so ist diese Funktion im Menüpunkt Konfiguration ab/anstellbar. Die Taste <BACKSPACE> ergänzt auch, jedoch ohne Leerzeichen. Die Taste <rechter Pfeil> übernimmt jeweils ein Zeichen, <linker Pfeil> setzt mit anderem Wort fort. Die Taste <Pfeil unten> schaltet auf Nachsilben von der Datei NACH.SEQ um. Vorsilben aus VOR.SEQ mischen sich mit den gelernten Wörtern. Die <rechte Maustaste> übernimmt ab ihrer Cursorposition Wort(Teil)e.

Ende

Es wird bye aufgerufen und das System verlassen. Bei fremden Programmen ist es oft nicht einfach, den Ausgang zu finden. Damit sich kein Ballast ansammelt, ist WordWriter beim Start immer wieder neu. Die letzen Einstellungen jedoch bleiben bis zur nächsten Änderung erhalten, auch nachdem die Arbeit mit WordWriter beendet ist oder der Computer neu gestartet wurde. Start, Lernen und andere Operationen laufen sehr schnell ab.

Hilfe

Wenn Sie Fragen haben, WordWriter besitzt eine integrierte (Hypertext-) Hilfe. Die Hilfe erläutert die Bedienung, die Idee, das Wozu und bietet ein Beispiel an. Hört sich das kompliziert an? Mit einem intelligenten Werkzeug müssen Sie nicht stundenlang Handbücher studieren. Es ist ganz einfach. Den WordWriter starten, das Beispiel ausprobieren und dann viel Spaß!

Datei Ansehen

Über das Menü SERVICE können Sie sich die DATEI ANSEHEN oder über das Menü WÖRTER und SCHREIBEN für die eigene Nutzung verändern.

Die Anzahl der Wörter ist aus der Zeilenzahl ZEILE= aktuelle Zeile/Gesamtzahl ersichtlich. Während der Bedienung reagiert WordWriter flexibel auf die Reihenfolge Ihrer Eingaben. Mit der Häufigkeitsanalyse ist feststellbar, wie oft welche Wörter auftreten. Originale, die unverändert bleiben sollen, können so sicher betrachtet werden. Zum Bearbeiten (oder zum Sortieren in WORTE.SEQ) läßt sich

der Browse-Modus direkt in Schreiben umschalten.

Wiederholen Finden

Bekannte Wörter kann man suchen und eventuell ersetzen lassen. Mit Wiederholungen finden ermittelt man unbekannte Wortgruppen z.B. für Sprach- oder Programmpflege. Wiederholungen finden wird sofort nach dem Aufruf des Word-Writer eingesetzt, um mehrmals wiederkehrende Wortgruppen zu finden und in WORTE.SEQ abzulegen, ohne daß diese vorher angegeben werden.

Konfigurieren

Über das Hauptmenü gelangt man in ein Untermenü und kann je nach Verwendungszweck folgende Einstellungen festlegen: Wortlänge, Mindesthäufigkeit, Satzzeichen, Zeichenfilter, Einlesen der Häufigkeit und Ausgabe der Häufigkeit.

Wortlänge

Textanalysen weisen aus, daß einige kurze Worte (der, die, und) besonders oft vorkommen. Fachwörter sind dagegen oft länger. Es wird eine Mindestwortlänge vorgegeben. Kürzere Wörter werden nicht gelernt, so wird der Speicher entlastet. Die Einstellung einer kleineren Wortlänge während des Schreibens bewirkt, daß kürzere Wörter nicht angeboten werden.

Mindesthäufigkeit

Während des Lernvorgangs ändert sich die Worthäufigkeit ständig. Worte, die seltener auftreten als die Mindesthäufigkeit angibt, werden nicht mit in die Datei WORTE.SEQ aufgenommen.

Satzzeichen

Tritt ein Satzzeichen auf, so gibt Word-Writer selbst ein Leerzeichen aus und schreibt auch ohne Betätigung der Shifttaste groß. Falls diese Arbeitsweise unerwünscht ist, läßt sie sich mit der Maus per Ja/Nein Entscheidung abschalten.

Zeichenfilter

Fremde Zeichen anderer Programme und nicht zu Wörtern gehörende Satzzeichen werden während des Lernvorgangs weggefiltert. Die angebotene Liste ist änderbar unter Benutzung der Sonderzeichengraphik mit Ausgabe der zugehörigen Kodenummer.

Einlesen der Häufigkeit

Ist diese Option ausgeschaltet, so werden in Texten vorhandene Zahlen ignoriert. Andernfalls werden die Zahlen als Häufigkeiten interpretiert.

Ausgeben der Häufigkeit

Die Wortliste in WORTE.SEQ ist mit und ohne Häufigkeitsangabe zu haben. Für ein späteres Wiedereinlesen der Häufigkeit ist die Einstellung mit Ja zu wählen.

Worte Umkehren

Es wird die Ausgabedatei REVERSE.SEQ erzeugt, in der die Wörter rückwärts eingetragen sind. So lassen sich z.B. Wortendungen sortieren und dann die Liste nach dem Umbenennen wieder nach vorwärts wandeln.

Export

WordWriter ist eher eine Schreibkraft als eine Textverarbeitung. Dateien werden mit anderen Textverarbeitungsprogrammen als reiner DOS-Text ausgetauscht (keine Bilder). Die .SEQ Dateien des Word-Writer können in .DOC umgewandelt werden. Dabei werden alle Zeilen bis zu einer Leerzeile zu einem Absatz zusammengefaßt, um in der Textverarbeitung formatieren zu können. Dokumente von der Textverarbeitung nutzen:

Der Anschluß üblicher Textverarbeitungsprogamme ist auch als reiner Text einstellbar.

z.B. WORD für WINDOWS
Datei, speichern unter,
TEXT mit Layout (DOS) einstellen.

Maus

Die roten Felder sind mausaktiv. Die linke Taste der Maus wählt aus (wie Enter), die rechte lehnt ab (wie ESC) oder kopiert. Das Hauptmenü erscheint nach dem Aufruf von ww, wenn die Maus bewegt wird. Ziehen Sie die Maus in die Zeile Ihrer Wahl und drücken Sie die linke Taste auf das erleuchtete Feld.

Dateimanager

Tom's Manager dient zur Auswahl der Dateien: Anfangsbuchstabe, Pfeiltasten, ENTER — File Auswahl. Der Aufruf erfolgt selbsttätig. Neu ist hier ebenfalls die Mausbedienung für: ESC Abbruch.

- \ Pfad, Dateifilter eingeben.

EDITOR

Beim SCHREIBEN ist oben links der Einfügemodus voreingestellt. Anklicken wechselt zum Überschreiben oder zum Zeigen. F10 links oben beendet und speichert. MENÜ=ESC rechts unten führt ins Hintergrundmenü und zurück.

HELP=F1 ruft den Hilfetext auf.

Die Pfeile verschieben den Text im Bildschirmfenster. Die Buchstaben auf dem Rand rufen sehr schnell verschiedene Menü's auf:

M - Makro
W - Wiederholung der Tasten/Makro's
S - Sortieren des Inhalts
G - Sprünge
F - Finden
E - Ersetzen
I - Import von Dateien
O - Operationen

Die Tastenkombination Alt-F7 sortiert ab Cursorposition sowohl nach Häufigkeiten als auch die Wörter alphabetisch.

Die Taste CAPS LOCK steuert, ob die Großbuchstaben getrennt berücksichtigt werden.

Soweit für heute die wichtigsten Punkte.

PS. Dieser Artikel wurde mit WordWriter geschrieben.

Er enthält:
45 die
19 Die
Textverarbeitungsprogammen = längstes Wort
63 Wörter mit A/a davon:
40 mit a
2 mit Ab
1 mit Abs (Absatz)

FORTecH Software GmbH
Tel.: 0381/4059-472 (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)

Gräbner-Elektronik
Tel.: (+Fax) 06101/48000
Am Römerbrunnen 11A
D-61118 Bad Vilbel

Wir bieten kundenspezifische Entwicklung von Hard- und Software für 65xx und 68xx. Fertigprodukte: Schrittmotorsteuerung mit Leistungsteil und RS232, DC-Motorsteuerung mit RS232, Leistungsteil und PID-Regelung.
6501-System mit RS232 und FORTH-Compiler.

Dipl.-Ing. Arndt Klingelnberg
Tel.: 02404/61648 (Fax: 63039)
Strassburgerstr. 12
D-52477 Alsdorf (b. Aachen)

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

Lascar Electronics P & V GmbH
Tel.: 07459/1271 (Fax: 2471)
Vordere Kirchstr. 4
D-72184 Eutingen

FORTH COMPUTER TDS-2020 16-BIT CPU H8/532 LOWPOWER - MULTITASKING - PCMCIA - 1,3 ZOLL HARDDISK - DATALOGGER - FOURIER TRANSFORM

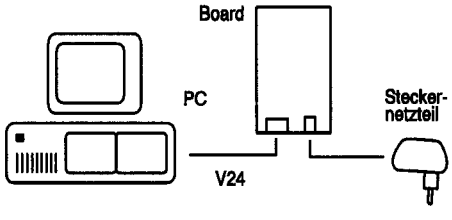
Dipl.-Phys. Wolfgang Schemmert
Tel.: 069/8001208 (Fax: 825957)
Strahlenbergerstr. 123
D-63067 Offenbach

HW: 80C592, H8, 68xxx; SW: Forth, C, Assembler; Entwicklung, Interface, Systemintegration. Speziell Steuerung räumlich verteilter Medieninstallationen, interaktive Benutzer- und Autorensysteme.

Neues vom Microcontroller-Verleih

von Rafael Deliano

Da die Aktivität der Mitglieder ja traditionell nicht sehr hoch ist, entsprach die bisherige Resonanz durchaus den Erwartungen. Eines der drei verfügbaren Boards



war immer unterwegs. Wobei der Kontakt hierbei oft über die Mailbox zustandekam. Zeitweise waren aber auch alle drei Systeme beim Anwender. Der Andrang war jedoch nicht so groß, daß es einer expliziten Warteliste bedurft hätte. Die typische Verweildauer beim Ausleiher beträgt 3 Monate.

Der Gedanke, die Systeme durch Zusatzplatinen mit fertigen Applikationen nützlicher zu machen (vgl. VD4/92), kann wohl nicht realisiert werden. Die Hardwareschnittstellen sind zu unterschiedlich. Und der Aufwand für Neuentwicklung für jedes einzelne System ist zu hoch.

Was jedoch möglich wurde, ist eine Erweiterung der Typenvielfalt. Bisher waren ja nur 2x Super-8 und ein F65 verfügbar. Neu wäre jetzt zum einen der RTX2000 von Frank Stüss. Der kleine Muck von DELTA t zeichnet sich besonders durch vorbildliche Dokumentation aus.

Allerdings konzentriert sich der Verleih normalerweise auf die praxisnahen und unkomplizierten 8-Bit-CPUs.

Von Thomas Prinz stammt eine Leiterplatte mit 8051-BASIC. Als Vorteil gegenüber FORTH bei Meßgeräten wurde mir von den Anwendern die Verfügbarkeit arithmetischer Funktionen wie sin und log dank floating-point genannt.

Von Erich Mann wurden ein New-Micros-68HC11 und ein Rockwell RSC-6511 zur Verfügung gestellt. Beide ähneln vom Konzept her dem Super-8-FORTH.

Die technischen Voraussetzungen wurden einheitlich gehalten und sind ziemlich simpel. Netzteil wird immer mitgeliefert. Was man braucht ist ein Tischcomputer mit V24 und Terminalprogramm sowie ein V24-Kabel. Dieses sollte in einem 9-Pin-Stecker mit DCE-Belegung enden.

Bei einigen Systemen liegen Disketten mit speziellen Terminalprogrammen für die jeweiligen Boards. Diese Software paßt jeweils nur für MS-DOS.

Ab 1.1.95 ist für den Controller-Verleih Thomas Prinz zuständig.

	kleiner Muck	Intel 8051-BASIC	NewMicros 68HC11	Rockwell RSC-6511
CPU	RTX2000	NMOS-8052	68HC11A8	NMOS-R6511
EPROM	16kWord	16k	—	8k ext. ROM
EEPROM	16kWord	—	0,5k	—
RAM	32kWord	16k	—	2k
akku RAM	—	8k	8k	—
I/O	Bus herausgeführt	2x 8255	diverse on-Chip ua. A/D-Wandler	—
Software	volksFORTH (FORTH-83)	BASIC in 8kROM	MAX-FORTH (FORTH-83) in 8kROM	RSC-FORTH (fig-FORTH) in 4kROM

Interessantes aus der Forth Diminsions

von Fred Behringer

Ein wenig neugierig machen sollen die hier besprochenen Artikel. Sie sind vom Autor direkt oder vom neuen Vertrieb erhältlich

5, Vol.15 Januar/Februar 1994, S.21

I Needed It: Mini-Math
Tim Hendtlass,
Hawthorn, Victoria, Australia

"Das Gesetz von Murphy hat viele Erscheinungsformen - Hier ist eine davon: Sobald man jemandem sagt, man habe etwas fertiggestellt, findet man heraus, daß das noch gar nicht der Fall ist." Der Autor spricht von seinem Beitrag "Math - Who Needs It?" in der FD 14/5 vom Januar/Februar 1993 (meine Besprechung in der VD 4/93), wo es um 32-Bit-Festkomma-Arithmetik ging. Sowie er den Artikel fertig hatte, so seine Worte, entdeckte er, daß er eigentlich eine abgespeckte Version mit "nur" 16-Bit-Festkommazahlen viel dringender nötig gehabt hätte: Es ging um Neuronale Netze und die Simulation des menschlichen Denkvermögens. Die Signale im Gehirn laufen mit Genauigkeiten von höchstens zwei bis drei Dezimalstellen ab, aber der Hunger an Speicherplatz ist unendlich groß und Schnelligkeit ist Trumpf.

Der vorliegende Artikel stützt sich in der Theorie auf den obenerwähnten Artikel. Es wird ein ausführlich kommentiertes Paket für 16-Bit-Festkommazahlen angegeben, das unter anderem auch Worte zur Überführung von 16-Bit-Zahlen in 32-Bit-Zahlen und umgekehrt enthält. Das Ganze ist für F-PC gedacht. Alle wichtigen Worte werden einmal in Doppelpunkt-Definition und ein weiteres Mal in 8086/88-Code-Definition gegeben! Es werden Zeitvergleiche angestellt. Der Autor erwähnt ausdrücklich, daß er das Paket der Öffentlichkeit als Public-Domain überläßt.

- 5 Seiten. Sehr zu empfehlen.

2, Vol.15 Juli/August 1993, S.16

Mixed Integer Arithmetic
Walter J. Rottenkolber
Mariposa, California

Der Autor gibt in diesem kurzen Artikel (2 Seiten) Doppelpunkt-Definitionen für acht gemischtganzzahlige Arithmetikoperationen (Zwischenergebnisse mit doppelter Genauigkeit) für F83 (Laxen and Perry) an. Es handelt sich um Operationen, die im Buch "Starting Forth" von Leo Brodie erwähnt werden: M+ M- M* M/ MOD M/ MMOD M*/MOD M*/. Er liefert in zwei Abschnitten eine kurze Diskussion der Pros und Contras zur "floored division" (Rundung - eigentlich Kappung - der vorzeichenbehafteten ganzzahligen Division gegen -ý) und "symmetrical division" (Rundung gegen 0). - Als Einführung, zum ganz schnellen Durchlesen nicht übel. (Zu diesem Thema vergleiche auch die Anmerkungen von Arndt Klingelnberg in der VD 4/93, S.31).

5, Vol.15 Januar/Februar 1994, S.34

Comma'd Output for Forth
Charles Curley
Gillette, Wyoming

Ein kurzer Artikel (2 Seiten), der sich mit der Erhöhung der Lesbarkeit von auszugebenden großen Zahlen (wie z.B. der Meldung über die Anzahl der auf der Festplatte verbliebenen freien Bytes) durch Einfügen von Kommata zur Einteilung in Dreiergruppen beschäftigt. (Interessanterweise ist sich der Autor bewußt, daß er außerhalb Nordamerikas (er sagt "elsewhere") statt des Kommas einen Punkt nehmen muß.) Der Autor arbeitet mit fastForth, einem Forth für 68000-Systeme, seine fünf Doppelpunkt-Definitionen (#,S D,.R D,. ,.R ,.), die die Aufgabe, die er sich gestellt hat, bewältigen, dürften aber übertragbar ("potierbar") sein. Der Artikel ist schon deshalb interessant, weil er eine gekonntkurze Einführung in die Forth-Vorgänge bei der Umwandlung einer auszugebenden Zahl in einen TYPEbaren String enthält: Diskussion von <## #S HOLD SIGN #>.

6, Vol.15 März/April 1994, S.15

Parallel Forth: The New Approach
Michael Montvelishsky
Saransk, Rußland

"Neue Programmier-Paradigmen erfordern neue Programmier-Sprachen. Der Forth Programmierer braucht zu keiner neuen Sprache überzuwechseln. Forth kann beliebig ergänzt und angepaßt werden und schluckt alle Paradigmen." So der Autor. Etwas mehr als eine Seite Text, drei Seiten Listing. Der Autor stützt sich auf Linda und OCCAM und übernimmt in seine Parallel-Forth-Erweiterung unter anderem den PAR-Konstrukt und das Channel-Konzept von OCCAM. Natürlich verwendet er die PAUSE-Umgebung und die im Vokabular USER enthaltenen Multitask-Aufbauten. Sein Paket enthält nur Doppelpunkt-Definitionen und ist für F-PC 3.53 gedacht. Der Autor betont, daß man die Worte leicht auf andere Forth-Plattformen übertragen können sollte, sie aber doch wohl am besten auf einem echten Multi-Prozessor-System verwende. Eben! Wenn das Ganze nur eine Frage der Programmiersprache wäre, hätte man ja den Transputer gar nicht erst zu erfinden brauchen. Der Autor lebt 600 km südöstlich von Moskau. Er bedauert, daß er keinen Transputer habe. Interessant! Ist denn Parallelrechnen nicht in erster Linie eine Frage der geeigneten hardwaremäßigen Reduzierung des Verwaltungsüberbaus auf ein erträgliches Maß? Als Denkanstoß für meine Transputer-Forth-Implementation ist mir dieser Artikel in gewisser Weise willkommen.

4, Vol.15 November/ Dezember 1993, S.7

Sparse Matrices
Rick Grehan
Peterborough, New Hampshire

Mehr als drei Seiten Listing, nicht nur Doppelpunkt-Definitionen, sondern auch ein paar Code-Definitionen (für den Macintosh). Der Autor hatte Probleme der Linearen Optimierung im Sinn und stieß dabei auf Matrizen mit sehr starker Nullenbelegung. Er hat die im Buch "Fundamental Algorithms" von D. Knuth vorgeschlagene Methode zur wirtschaftlichen Abspeicherung dünn besetzter Matrizen leicht abgewandelt: Seine Forth-Implementation verwendet einfach verkettete Listen. Bei einer 50x50-Matrix beispielsweise fängt die Methode an, sich zu lohnen, wenn mehr als 950 Elemente mit Nullen belegt sind. - Interessant sind diese Dinge, und damit auch der vorliegende Artikel, auf jeden Fall. Bei der von mir im Jahre 1981 entwickelten Methode zur Lösung des Nucleolus-Problems bei N-Personen-Spielen durch Zurückführung auf die Lineare Optimierung treten 2 hoch n Nebenbedingungen auf, also ungeheuer große Koeffizientenmatrizen, die aber zur Hälfte mit Nullen belegt sind. Was aber, wenn der erste Simplex-Austauschschritt getan ist? Die Nullenbelegung ist dann weitgehend weg!

3, Vol.15 September/ Oktober 1993, S.26

Terminal Input and Output
C.H. Ting
San Mateo, California

Sechster Teil einer Einführungsserie von C.H. Ting. "Zum Durcharbeiten benötigt man den Zufallszahlengenerator der vorhergehenden Lektion", sagt der Autor. Glossarartig (keine Definitionen) wird zunächst die Funktion der folgenden Worte besprochen: <# # #S #> HOLD SIGN BASE DECIMAL (Worte, die nötig sind, um eine auszugebende Zahl beliebiger Basis in einen formatierten String umzuwandeln). Gutkommentierte Programme zur Ausgabe von Telephonnummern, Tageszeiten (wie beim DOS-Befehl TIME) und Winkelangaben aus der Seefahrt beleben die Erklärungen. Dann wird kurz auf die Forth-Möglichkeit eingegangen, in nahezu beliebigen Basen zu rechnen. ("Programmierer, die etwas auf sich halten, schleppen HP-Taschenrechner mit sich herum und beeindrucken Anfänger damit, daß sie ständig zwischen Hexadezimal- und Dezimaldarstellung hin- und herschalten. Der Forth-Programmierer hat solche Mätzchen nicht nötig. Forth hat das alles, und noch viel mehr, eingebaut.") Jawoll! Interessant, daß Tings Lieblingsbasis die 19 ist. Mit der kann er die Brettpositionen im chinesischen Go-Spiel platzsparend speichern. Es folgt ein Exkurs über Nachrichtenverschlüsselung (ganz kurz). Und dann kommen die eigentlichen Terminal-Input-Output-Worte: KEY KEY? EMIT TYPE EXPECT. Keine Definitionen, nur Erklärung der Funktionsweise. Aber ein kommentiertes Programm zur Ausgabe des IBM-Zeichensatzes (mit den Graphikzeichen). Schließlich ein Liebesbrief mit tanzenden Buchstaben (Programm ohne Kommentar). Gelegenheit zur Erwähnung der Worte ALLOT FILL-TRAILING. Als letztes dann das Wort CONVERT und die Umwandlung eines eingegebenen Zahlenstrings in die betreffende Zahl. Ein kommentiertes Programm eines Zahlenratespiels beschließt den gar nicht so schlechten Einführungsartikel.

5, Vol.15 Januar/Februar 1994, S.7

Rational Numbers, Vulgar Words
Gordon Charlton
Hayes, Middlesex, England

Der Autor verwendet ein "preisgünstiges" Forth (man erfährt nicht, welches) und beschwert sich, daß UM/MOD und D- nicht richtig funktionieren. "Was den Euklidischen Algorithmus betrifft, geht Knuth wirklich in die Tiefe. Für mich sind das jedoch böhmische Dörfer", sagt der Autor. Starke Worte. Dennoch ist der Artikel (9 Seiten) lesenswert. Man erfährt eine Menge über gewöhnliche Brüche und deren Darstellung auf dem Computer (in Forth). (Leute, die "es macht keinen Sinn" sagen und von "ultimativen Lösungen" sprechen, werden sicher versucht sein, "vulgäre Brüche" zu erfinden.) Zur Vereinfachung verwendet der Autor je ein Byte für den Zähler und eins für den Nenner. Er spricht von einem 8-Bit-Forth, meint aber, das lasse sich leicht auf 16-Bit- oder 32-Bit-Forths übertragen. Sein Paket besteht aus 60 Doppelpunkt-Definitionen (keine Code-Definition) und eben so vielen Erklärungen im Glossar. Interessant, wie elegant sich der Euklidische Algorithmus zur Ermittlung des größten gemeinsamen Teilers in Forth ausnimmt: (: GCD (nn — n) BEGIN ?DUP WHILE TUCK MOD REPEAT ;). Daß man dieses Wort beim Umgang mit Rationalzahldarstellungen aber wohl doch tunlichst als Code-Definition faßt, stelle ich, der Rezensent, zur Debatte. Der Autor spricht fast eine Seite lang über Rundung und Genauigkeit. "Die Angabe, Ò ist 22/7, ist recht akkurat (accurate), aber nicht sehr präzise (precise). Die Angabe, Ò ist 29,45450752879436751 ist sehr präzise, aber fürchterlich inakkurat." So der Autor. Hier muß ich, der Rezensent, sagen: "Das kommt mir spanisch vor." Meiner Meinung nach ist die letztere Angabe weder unpräzis noch inakkurat, sie hat mit dem Wert von Ò einfach nicht das Geringste zu tun. Genauigkeit ist ja schließlich ein mehr oder weniger fest definierter mathematischer Begriff. Natürlich bleibt es dem Autor belassen, diesen neu zu fassen. Leid tun mir nur die Übersetzer, die den Unterschied zwischen Akkuratheit (Genauigkeit) und Präzision (ebenfalls Genauigkeit) ins Deutsche übertragen sollen. Der Autor macht recht ordentliche Literaturangaben, die den Leser weiterführen. Über Gleitbruchstrich-Darstellungen (floating slash representation) und Darstellungen mit variabler Basis (von Stelle zu Stelle im Positionsschema) wird allerdings nichts gesagt. Auch wird die umfangreiche Literatur über "Exaktes Rechnen" nicht erwähnt.

□

Brief aus der Provinz

von Friedrich Prinz, Hombergerstr. 335, 47443 Moers

Moers, im Oktober 1994

Ja was ist denn los in der Forthgesellschaft ? Was ist denn mit der VD ? Kommt da nichts mehr ? Das sind die Fragen, denen ich mich hier in Moers stellen muß - und die ich auch nicht beantworten kann. Ruhig ist es auf jeden Fall geworden, selbst in der DFÜ. Fast möchte man meinen, daß die Forther der Republik noch immer im Urlaub sind (...wie machen die das?). Daß es unter diesen Umständen nicht gerade leichter wird, aus den Mitgliedern der Moerser Gruppe neue Mitglieder für die Forthgesellschaft zu ´rekrutieren´, dürfte leicht verständlich sein. Für unsere ´Mitglieder´ stellt sich darum auch vielmehr die Frage, was wir hier in Moers selbst für uns tun - und ein wenig auch für Forth.

Nun, da wäre zunächst aus dem letzten Halbjahr zu berichten, das der Michael Major recht gut mit einem Einsteigerkurs ´gefüllt´ hat, den er unseren ´Neuen´ gewidmet hatte. Unsere ´Neuen´ - das waren die Menschen, die wir im vergangenen Jahr anläßlich unseres ´Tages der offenen Tür´ für Forth interessieren konnten. Heute sind das keine ´Neuen´ mehr für uns, und auch keine Einsteiger mehr. Eine Frau und vier Männer sind ´an uns und an Forth hängengeblieben´. Und diese zählen wir mittlerweile zu unseren Freunden, wozu vor allem die regelmäßigen, allsamstäglichen Kurse mit gemeinsamer Arbeit einen großen Teil beigetragen haben.

Ebenfalls im ersten Halbjahr ´94 habe ich, in einem zweiten Anlauf, einen Forthkurs für die DFÜ aufbereitet und in regelmäßigen Abständen in das Z-Netz abgelegt. Wenn zu Anfang die Resonanz doch recht groß war (36 Rückmeldungen und Interessensbekundungen via DFÜ), ließ sie leider bis zu den Sommerferien mehr und mehr nach - und ist heute auf NULL gesunken. Das hat mich bewogen, die Arbeit an diesem Kurs nicht fortzuführen. Wer arbeitet schon gerne ´für die Katz´?

In einem ´zweiten Kurs´ haben wir uns in der Moerser Gruppe im ersten Halbjahr intensiv mit dem Multitasking auseinandergesetzt. Wir wollten dabei ´ganz unten´ anfangen und zunächst allgemeine Probleme des Betriebsystems und der PC-Hardware diskutieren, über das ZF Multitasking ausprobieren und danach via FORTH/2 und OS/2 moderne Systeme und deren potentielle Leistungsfähigkeiten ´ausloten´. Bei ZF und bei ersten Experimenten mit ZF´s Scheduler waren wir im April diesen Jahres, als sich allgemein die Ansicht durchsetzte, daß ´man eigentlich kaum sinnvolle Anwendungsmöglichkeiten für Multitasking kennt´. Ich hätte dieses Thema gerne bis zum Ende aufgearbeitet...

Dann kamen die Sommerferien, die auch hier in Moers eine 2-monatige Ruheperiode erwirkten. Ältere VD-Leser werden sich erinnern, daß ich an dieser Stelle häufiger von unseren Sommerfesten berichtet habe, die wir alljährlich durchführen, um unseren Familien Gelegenheit zu geben, die Menschen kennenzulernen, mit denen wir die Samstage verbringen. In diesem Jahr haben wir unser ´Sommerfest´ auf ein Schiff verlegt. Am 25.09. waren wir einen ganzen Tag lang auf einem ´Rheindampfer´ unterwegs, woran alle Mitfahrer so großen Spaß hatten, daß sie ähnliche Fahrten möglichst bald in kleineren Gruppen wiederholen wollen. Neben dem Spaß ist vielleicht vor allem die Tatsache bemerkenswert, daß die Fahrt selbst für Alle kostenlos war. Mit den Kursen die wir hier durchführen verbinden sich bescheidene Einkünfte (über die ich mich an dieser Stelle nicht näher auslassen möchte), die aber ausreichen, um eine solche Fahrt finanziell abzusichern.

Inzwischen hat Michael Major die Arbeit mit einem Aufbaukurs begonnen, der die ´Einsteiger´ aus dem letzten Jahr weiterführen wird. Die zweite Hälfte der Samstagnachmittage geht seit dem Ende der Sommerferien für die Graphikprogrammierung drauf. ´Die Moerser´ wollten nach der ´grauen Theorie´ einige entspannende Stunden einfach ´beim Hacken´ verbringen und sich mit Graphiken auf der VGA-Karte im ´Versuch & Irrtum´ auseinandersetzen. So ganz ohne Theorie geht es dabei natürlich nicht ab. Einige von uns haben in den letzten vier Wochen kräftig Mathematik ´nachschieben´ müssen und ich selbst komme immer wieder in Erklärungszwänge, weil meine Freunde verstehen wollen, WARUM z.B. der Bresenham oder ´der Richter´ funktionieren. Das heißt im Klartext, daß sich Spaß und Arbeit zur Zeit die Waage halten - mit leichtem Übergewicht auf der Spaßseite. Der Spaß überwiegt dabei nicht zuletzt deshalb, weil wir u.a. ´den Bresenham´ direkt auf die Hardware der VGA-Karte portieren konnten, was uns die schnellsten Linienaufbauten beschert hat, die wir je zu sehen bekommen haben. So sitzen wir also einmal in der Woche zusammen, ´büffeln´ Mathematik, experimentieren mit der VGA-Karte und ´entwerfen´ neue, schnelle Algorithmen. ´Nebenher´ üben wir mit ZF´s Assembler umzugehen, zeitintensive Mnemonics gegen weniger zeitintensive Sequenzen auszutauschen, das Betriebssystem ´zu umgehen´ und ´das Staunen´ nicht zu verlernen.

Es geht also munter weiter, in Moers. Im ´Rest der Forthgesellschaft´ auch...?

Der neue volksFORTH-Vertrieb

von Klaus Kohl

Seit dem 01.07.1994 habe ich nach Rücksprache mit dem Vorstand der FORTH-Gesellschaft e.V. (genauer Jörg Plewe), Heinz Schnitter vom FORTH-Büro und dem bisherigen Vertriebsbetreuer Johannes Teich den Vertrieb des volksFORTH wieder übernommen. Damit möchte ich meine längere Schaffenspause im Rahmen der FORTH-Gesellschaft beenden und die aktive Verbreitung von FORTH-Material vorantreiben.

Ziel des neuen, als kommerzielle Firma ausgelegten Vertrieb ist schon jetzt:

* Vertrieb unterschiedlichster FORTH-Versionen
- volksFORTH (für C64, Schneider CPC, Atari ST und PC)
- KK-FORTH (für Einplatinencomputer mit 80x86, Z80 oder RTX)
- FPC, FIFTH und weiterer PD-FORTH-Programme
* Lieferung von Sourcelistings in Text- und Diskettenform
- EFORTH (für 8096, 8086, 68HC11 oder 8051)
- Programme aus dieser Clubzeitschrift
- Beschreibung des aktuellen Konzepts für das ANS-FORTH
* Verkauf von Produkten mit und für Super8-FORTH

Da die bisher aufgeführten Punkte auch in der in dieser Zeitschrift veröffentlichten Preisliste angegeben sind, könnte man die Übernahme des volksFORTH-Vertriebes als reine Aufwertung des eigenen FORTH-Vertriebes halten. Wer aber selbst sich mit Vertrieb von (FORTH-)Produkten befaßt, wird schnell erkennen, daß mit dem vorliegenden Preis-Leistungs-Verhältnis, der notwendige Vertriebsbetreuung und den noch folgenden Zielen immer am Rande der Rentabilität arbeiten wird. Ich möchte auch keine Konkurrenz zu den kommerziellen FORTH-Systemen aufbauen, sondern bin gerne bereit, bereitgestelltes Material bei jeder Informationsanfrage oder Auslieferung beizulegen.

Jetzt zu den zukünftigen Zielen des FORTH-Vertriebes:

* Erweiterung des Pools von PD-FORTH-Versionen auch für andere Rechner
* Diskettenservice zur FORTH-Mailbox (alle Sourcen und Bretter)
* Vertrieb einer allgemein verwendbarern Toolbox für FORTH
- Bereitstellung eines standardisierten FORTH-Systems zur Bearbeitung der Sourcen
- Aufbereitung (Test, Dokumentation ..) externer FORTH-Sourcen
- Bereitstellung von FORTH-Artikeln in Diskettenform
* Entwicklung, Herstellung und Vertrieb von Hardware für FORTH
- (Günstige) Einplatinencomputer mit FORTH-Prozessoren
- Sehr günstige Einplatinen mit entsprechenden FORTH-Systemen
- Zusatzhardware für FORTH-Projekte
* Bereitstellung weiterer FORTH-Computer für den uP-Controller Verleih
* Aufbau eines Literaturvertriebes mit
- Büchern über FORTH (Einführung, Applikationen)
- FORTH-Artikelverwaltung und Kopierservice
* Unterstützung der FORTH-Gesellschaft e.V. durch Spenden aus dem Vertrieb

All diese geplanten Ziele des FORTH-Vertriebes sind nicht durch einen Mann durchzuführen. Deshalb möchte ich zum Schluß dieses Forum nutzen und alle Interessierte auffordern, mir bei diesem Projekt zu helfen. Vor allem die Zusendung folgender Daten, Programme und Informationen sind zum Aufbau des Vertriebes nützlich. Ich hoffe, daß zumindest einige "Jäger und Sammler" mit diesem Artikel zur Zusendung von verfügbarem FORTH-Material animiert werden. Soweit es mir möglich ist, werde ich dabei die entstehenden Kosten übernehmen oder dem Sender mit entsprechenden Informationsmaterial einen Ausgleich für seine Arbeit erstatten.

* PD-FORTH-Versionen für unterschiedliche Rechnertypen (MAC, AMIGA, ...)
* Anwendungen für FORTH (möglichst mit entsprechender Dokumentation)
* Quellenangaben (oder falls möglich: Kopien) zu Artikel und Bücher über FORTH
* Informationsmaterial zu FORTH-Produkten (Chips, Rechner, Programme)

Soweit möglich, werden alle Arbeiten des Vertriebes in enger Zusammenarbeit mit dem FORTH-Büro, dem Redaktionsbüro der Vierten Dimension und der FORTH-Mailbox durchgeführt. Dies wird dazu führen, daß alle PD-Programme und FORTH-Sourcen von der Mailbox abgezogen werden können und daß einige der im Vertrieb aufgelaufenen Informationen zu FORTH-Produkten oder Anwendungen als Artikel in der VD erscheinen. Ich kann nur hoffen, daß die Akzeptanz des FORTH-Vertriebes wieder das Niveau früherer Tage, wie z.B. während der Einführung des RTX, erreicht.

Forth-Gruppen regional

Berlin
Claus Vogt
Tel.: 030/2 16 89 38 p
Treffen: nach Absprache

Rhein-Ruhr
Jörg Plewe
Tel.: 0208/49 70 68 p
Treffen: jeden 1. Samstag im Monat im S-Bahnhof Derendorf Münstererstr. 199, Düsseldorf

Moers
Friederich Prinz
Tel.: 02841/5 83 98 p
Treffen: jeden Samstag 14:00 Arbeitslosenzentrum, Donaustr. 1, Moers

Darmstadt
Andreas Soeder
Tel.: 06257/27 44

Mannheim
Thomas Prinz
Tel.: 06271/28 30 p
Ewald Rieger
Tel.: 06239/86 32 p
Treffen: jeden 1. Mittwoch im Monat, Vereinslokal Segelverein Mannheim e. V., Flugplatz Mannheim-Neuostheim

µP-Controller Verleih

Rafael Deliano
Steinbergstr. 37
82110 Germering
Tel.: 089/8 41 83 17

Gruppengründungen, Kontakte

Regional
Stuttgart
Wolf-Helge Neumann
Tel.: 0711/8 87 26 38 p

Fachbezogen
8051 ...(Forth statt Basic, e-FORTH)
Thomas Prinz
Tel.: 06271/28 30 p

Forth-Hilfe für Ratsuchende

Forth allgemein
Jörg Plewe
Tel.: 0208/49 70 68 p
plewe@mpi-dortmund.mpg.de
Karl Schroer
Tel.: 02845/2 89 51 p
Jörg Staben
Tel.: 02103/24 06 09 p

Spezielle Fachgebiete

Anfänger und Wiedereinsteiger
Gerd Limbach
Tel.: 02051/25 51 12 p
Mo.+Di. 20:00 - 22:00

32FORTH (Atari)
Rainer Aumiller
Tel.: 089/6 70 83 55 gp

FORTHchips (FRP1600, RTX, Novix...)
Klaus Schleisiek-Kern
Tel.: 040/2 20 25 67 p

F-PC &ICOM, ASYST (Meßtechnik), embedded controller (H8/5xx//TDS2020, 8051 ... eFORTH...), FUZZY
Arndt Klingelnberg
Tel.: 02404/6 16 48 agp

Gleitkomma-Arithmetik
Andreas Döring
Tel.: 0721/59 39 35 p

HS/Forth (Harvard Softworks)
Wigand Gawenda
Tel.: 030/44 69 41 p

KI (Künstliche Intelligenz), OOF (Object Oriented Forth)
Ulrich Hoffmann
Tel.: 0431/80 12 14 p

Unterricht mit Forth
Rolf Kretzschmar
Tel./Fax: 02401/8 88 91 ap

volksFORTH/ultraFORTH/RTX-FG-Forth/Super8Forth
Klaus Kohl
Tel.: 08233/3 05 24 p
Fax: 08233/99 71 f

Forum: Forth-Mailbox

Forth-Mailbox (BBS)
Jens Wilke (SysOp)
Tel.: 089/8 71 43 52 p
Mailbox 089/8 71 45 48
300-14400 baud (8N1)

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/Anzeigenverwaltung finden Sie im Impressum.

FORTH-SYSTEME GMBH

Postfach 1103
D-79200 Breisach

Telefon (0 76 67) 5 51
Telefax (0 76 67) 5 55

WinFORTH

- UR/FORTH kompatibel
- Windows Funktionen werden voll unterstützt
- Erweiterte Debugging-Hilfsmittel
- Online Windows Hilfe
- Coprozessor Unterstützung möglich
- Software-Gleitkomma-Paket
- Viele Beispielprogramme
- Upgrades von UR/FORTH Systemen auf WinFORTH sind preisgünstig zu erhalten

UR/FORTH

- FORTH-83 Standard
- Für MS-DOS, OS/2, 80386
- Direkt gefädelte Code Implentationen mit dem obersten Stackwert im Register um größtmögliche Ausführungsgeschwindigkeit zu erreichen
- Segmentiertes Speichermodell mit Programm, Daten, Headers und Dictionary Hash Table jeweils in einem getrennten Segment
- Komplett gehashtes Dictionary führt zu extrem schneller Übersetzung
- Mächtige neue String Operatoren (Suche, Extraktion, Vergleich und Addition) sowie einen dynamischen String-, Speichermanager
- Kann mit Objektmodulen, die in Assembler oder anderen Hochsprachen erzeugt wurden, gelinkt werden
- Native Code Optimizer zur direkten Umsetzung in 80 x 86 Code im Lieferumfang

Eprom Emulator

- SRS-II – Serieller-ROM/RAM-Simulator
- Flex ROM plus – Lowcost Eprom-Emulator
- Econo ROM

ModuNORM

CPU-Steck-Module im Scheckkartenformat:

- 8 Bit z.B. 6303
- 16 Bit z.B. V25
- Highspeed RTX-2000/1
- 80C166
- C 167 CAN
- 68332
- Softwareunterstützung durch SwissFORTH
- Thermodrucker und Controller
- LCD Grafik-Controller

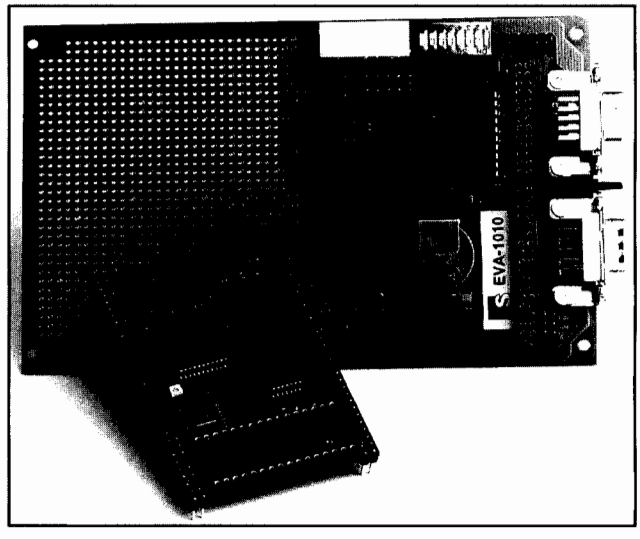
FORTH-83 Metacompiler

Der LMI FORTH Metacompiler wird mit komplettem Quellcode für ein ausführlich ausgetestetes, Hochgeschwindigkeits FORTH 83 Kern ausgeliefert, wobei Sie die Auswahl aus folgenden Zielprozessoren haben:

- 8086/8088
- Z80/HD64180
- 8080/8085
- 68000
- Z8
- 1802
- 6809
- 8096/97
- 68 HC 16
- 78310
- 8031/32/535
- 6303
- 6502
- V25
- 68HC11
- RTX 2000
- 80C166
- 68332

Sie erzeugen schnelle und kompakte Anwendungen, indem Sie Ihre Quellprogramme mit unserem Forth Nucleus zusammenstellen und ihn mit dem LMI FORTH Metacompiler übersetzen.

Scheckkartenmodule



CPU Module für Entwicklungen im Embedded Controller Bereich:

- 80386 EX
- 80C166-CAN
- Evaluations-Boards

Bitte fordern Sie unseren Produktkatalog und die Preisliste an. FORTH-Gesellschaftsmitglieder erhalten bis zu 10% Rabatt (artikelabhängig).