VIERTE DIMENSION

4/1993

9. Jahrgang 1993 4. Quartal DM 10,--

Forth ohne Arbeit

Core-Wars

ANS-News

Let´s Gimpel

FUZZY und Forth

ONLY Forth, ALSO andere!

Aufnehmer im Angebot

Forth von TRIANGLE

68HC11

Board von H. Dyja

FORTH MAGAZIN

Organ der Forth-Gesellschaft e.V.

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 8N1

Redaktionsleitung
Rolf Kretzschmar (rk), (verantwortlich)
Rote Gasse 7, 52499 Baesweiler
(Redaktionsadresse)
Tel/Fax: 02401-88891

Redaktion
Arndt Klingelnberg (akg), Alsdorf
Tel.: 02404-61648 Fax: 02404-63039
Klaus-Peter Schleisiek (kps), Aachen
Tel/Fax: 0241-873462

Layout, Satz, Herstellung
ORGA Sport, Rilkestr. 8, 52477 Alsdorf
Tel/Fax: 02404-61425

Grafik, Illustration, Layout
Rolf Kretzschmar (rolf)

Anzeigenverwaltung
Arndt Klingelnberg
Straßburger Str. 12
52477 Alsdorf
Tel.: 02404-61648 Fax: 02404-63039

Redaktionsschluß
Feb., Mai, Aug., Nov.

Erscheinungsweise
vierteljährlich

Auflage
1000

Preis
Einzelheft DM 10,-, Abonnementpreis DM 50,-, bei Auslandsadresse DM 55,- inklusive Versandkosten

Der Nachwuchs

von Rolf Kretzschmar

Wer sich nicht um den Nachwuchs kümmert, ist bald alleine. Ein Ziel der Forth Gesellschaft e.V. ist es, dafür zu sorgen, daß sich jemand um den Nachwuchs der forthelnden Computernutzer sorgt. Es reicht nicht, jemanden für Forth zu begeistern. Das gelingt oft überraschend schnell. Die Ernüchterung folgt meistens ebenso schnell: keine Bücher, keine Gleichgesinnten, keine Computerzeitschriften, keine Hilfen, keine spektakulären Forth-Produkte. Damit verbunden: wenn schon kein schlechtes Image, so doch eher gar kein Image als ein gutes. Forth ist nicht der BOSS unter den Programmiersprachen, sondern ein NO-Name-Produkt. Aber ein gutes. Das belegt ein interessanter Leserbriefwechsel in der Forth Dimensions, den wir für Wert befunden haben, über die VD auch der deutschsprachigen Forth-Gemeinde vorzustellen.

Nachwuchsförderung: Was tuen wir? Was tuen Sie? Kennen Sie einen Nachwuchsförderer im Umkreis von 100 km? Ich kenne gleich eine ganze Gruppe! Nein, die sitzt nicht in Aachen! Es ist die Gruppe um *Friederich Prinz* in Moers. Wir (*kps, J. Staben, rk*) sind der Einladung zu einer Info-Veranstaltung am 6. November gefolgt, und was mich betrifft, ich zehre noch heute davon: Wenn sich jemand kümmert, dann läßt der Erfolg nicht auf sich warten. Das haben wir mit eigenen Augen gesehen. Das macht Mut. Es lohnt sich, unter diesem Blickwinkel die 'Briefe aus der Provinz' auch in Zukunft aufmerksam zu lesen. Der Bericht über die Novemberveranstaltung in der Provinz zeigt, was möglich ist, wenn man will. Doch ohne ideellen Einsatz an Zeit und Mühen ist das nicht zu leisten. Auch das wurde uns bewiesen.

Für mich ist es kein Zufall, daß *Friederich Prinz* in Moers zu denen zählt, die meinen Apell in der VD 1/93 (Aufnehmer gesucht) am besten verstanden haben: Es ging nicht darum, eine perfekte, ausgefuchste Lösung eines mehr oder weniger interessanten Problems zu finden. Wie zeigen wir es dem Nachwuchs? Das war die Devise! Das bleibt die Devise!

Immerhin: Keiner der vielen Anregungen, die wir in zwei Jahren in der VD gegeben haben, hat auch nur annähernd soviel Resonanz gezeigt, wie der Ruf nach dem Aufnehmer! Das wiederum zeigt, daß auch die Redaktion unserer VD gefordert ist, immer wieder Anregungen zu geben und so zu versuchen, die Kluft zwischen Könnern, Kennern, Künslern auf der einen und dem Nachwuchs auf der anderen Seite zu verkleinern. Der Nachwuchs ist überall. Auch unter den Autoren. Er will gepflegt sein, damit er wächst.

Euer

R. Kretzschmar

Einladung zur Forth Jahrestagung '94

Zur 10. Jahrestagung der Forth Gesellschaft eV laden wir herzlich ein in die Holsteinische Schweiz. Wir treffen uns wieder für ein Wochenende: 15/16/17. April 1994. Direkt vor der Hanovermesse.

Wir haben ein hübsches Tagungshotel in Malente-Neukirchen, einem kleinen Ort gleich bei den berühmten Seen Ostholsteins, gefunden. Das Hotel bietet bis zu 70 Personen im Haus Unterkunft und steht uns samt Wirtschaft und Tagungssaal zur Verfügung.

In diesem Jahr wird der Samstag Abend für das Fest "10 Jahre FG" reserviert sein. Im übrigen Programm finden Sie wieder aktuelle Vorträge und Vorführungen rund um Forth Hard- wie Software. Zwischen den einzelnen Veranstaltungen werden Sie genügend Zeit für eigene Präsentationen in der gemütlichen Lobby haben.

Unser Zeitrahmen wird etwa so sein:

- Freitag 15.4.
  15-18 Uhr Beiträge, Abendessen,
  19-21 Uhr Beiträge.
- Samstag 16.4.
  9-12 Uhr Beiträge, Mittagessen,
  15-18 Uhr Beiträge.
  nach dem Abendessen wird feste gefeiert.
- Sonntag 17.4
  8:30 - 11:30 Mitgliederversammlung der FG. Abschließend noch gemeinsames Mittagessen.

Der Tagungspreis für Mitglieder der Forthgesellschaft wird 260DM sein. Melden Sie sich noch vor dem 1.3.94 zur Tagung an, kostet es nur 240DM.

Darin enthalten sind der Tagungsband, 2 Übernachtungen 6 Mahlzeiten und natürlich das Fest "10 Jahre FG". Auch Teile der Tagung zu buchen ist möglich. Und natürlich können Begleitpersonen auch Unterkunft und Verpflegung ohne Tagung buchen.

Das Haus ist kinderfreundlich, bietet Spielmöglichkeiten, und eine Kinderbetreuung können wir einrichten. Aber bitte - rechtzeitig anmelden!

Die nähere Umgebung des Hotels am See des Dorfes und die ganz nahen großen ostholsteinischen Seen bieten landschaftliche Vielfalt und Abwechslung - zu entdecken zu Fuß, zu Pferd, mit dem Rad oder Boot. Der Kneipp Kurort Malente ist ganz in der Nähe, und der höchste Berg Schleswig Holsteins ist in Sichtweite. In einer Autostunde kommen Sie auch bis in die alte Hansestadt Lübeck oder in die Landeshauptstadt Kiel oder auch an die weiten Ostseestrände.

Anmeldung

Bitte über das

- Büro für die Forth Jahrestagung '94
  c/o M.Kalus
  Plönerstr 24a
  23714 Malente
  Telefon: 04523-4177
  email :
  FORTH94@KBBS.ZER.SUB.ORG
  Bankverbindung für die Tagung:
  Volksbank Eutin, BLZ 213 922 18
  Kontonummer: 1 124 842
  Kennwort: Forthtagung '94
- Call for Papers
  Vorträge und andere Beiträge bitte bis Ende März '94 einreichen. Dann können diese in den Tagungsband aufgenommen werden.
  Wir freuen uns auf dieses Treffen!

Einladung zur ordentlichen Mitgliederversammlung

am
Sonntag, den 17. April 1994 um 8:30
im
Neukirchener Hof
Hauptstraße 10
23714 Malente-Neukirchen

Tagungsordnungspunkte

1. Rechenschaftsbericht des Direktoriums
2. Kassenbericht und Bilanz 1993
3. Aussprache, Entlastung, Wahl des Direktoriums
4. Vierte Dimension
5. Neue Ziele der Gesellschaft
6. Verschiedenes

Ergänzende Tagungsordnungspunkte sind bis zum 1. März über das Forth-Büro dem Direktorium einzureichen.

Die Jahrestagung der Forth-Gesellschaft, Forth'94, findet im Zeitraum vom 15.-17.4.1994 am gleichen Ort statt. Sie endet nach der ordentlichen Mitgliederversammlung, deren Teilnahme kostenlos ist, am Sonntag Mittag. Wir bitten um zahlreiches Erscheinen und mit ausreichend Zeit bis zur Abreise, damit die Mitgliederversammlung beschlußfähig wird. Wie immer stehen auf der Tagesordnung wichtige Themen, die für die Zukunft der Forth-Gesellschaft e.V. entscheidend sind.

Inhalt

Verschiedenes

von Arndt Klingelnberg
Strassburger Str. 12, 52477 Alsdorf, Tel. 02404-61648

Eine Vorstellung des embedded-controller-Modul TDS2020 von TRIANGLE mit 16 bit Hitachi H8/532

Schnell mal eben (und dazu überall) einen 'embedded controller' programmieren ... (?) , eine Applikation mit analogen In- und Outputs, Zählern/Timern, serieller Kommunikation und 8 MB Speicher, Batteriebetrieb über Monate ... (?), Hardware inklusive ... (?). Nun, warum denn nicht! Und das alles komplett günstiger als so mancher kommerzielle TargetCompiler ohne Hardware!

Die Entwicklungsumgebung und die Hardware inklusive des Hitachi Controllers sind Gegenstand dieses Artikels. Der Autor stellte den TDS2020 bereits auf der Forth Tagung in Rostock 1992 vor. Das Board wurde nun mit dem Forth-ROM v.2.13, einem modifizierten TDS2020DV Programm-RAM Modul und zusätzlichen Software-Lösungen aufgewertet. Das neue (englische) Handbuch wurde ausführlicher und besser gedruckt. Anregungen des Autors wurden (fast) alle berücksichtigt. Wo Software oder die Performance verglichen wird, dient der allseits bekannte Standard: F-PC auf PC (bzw. AT) als Referenz.

Stichworte
I²C
F-PC
TDS2020
LowPower
MultiTasking
MEMcard (PCMCIA)
Controller, embedded
Analog-Digital
Forth-Modul
ROM-Forth
fig-Forth
H8/532

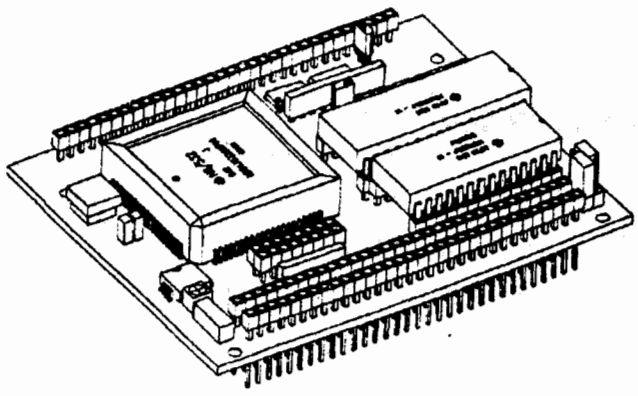
H8, aber nicht 8 bit

Hitachis H8-Serie umfaßt Controller von 8 bis demnächst 64 bit Breite. Motorola bremste zuerst erfolgreich dank eigener Patente den Bekanntheitsgrad speziell in Deutschland. International werden monatlich 30 Mio. H8/xxx abgesetzt. Möglicherweise haben Sie irgendwo einen H8/xxx im Einsatz, wahrscheinlich besitzen Sie zwar keine moderne Schreibmaschine (510), aber vielleicht einen (japanischen) AirBag (520), einen VideoRecorder (3xx) oder einen CompactCamCorder (5xx). Die neuesten Entwicklungen der H8/5xx Reihe arbeiten mit internem Flash Programm Memory und verfügen über einen ISP (intelligent sub processor) oder einen IPU (integrated timer pulse unit) und managen vielleicht bei Ihnen bereits den Motor im PKW.

Diese Produkte wie auch das Controller Modul von Triangle Digital Systems, London gab es zumindest in Deutschland eher als den Controller selbst. (*Anmerkung der Redaktion: in dem CAN-Feldbus-Controller wird ein Hitachi H8/325 verwendet, Vierte Dimension 93Q3*)

Architektur und Befehlsumfang des Prozessors erlauben Hochsprachen (-C- , aber auch Modula2). Hitachi liefert einen RealTime Kernel (einmalig DM14'000) und verschiedene Emulatoren/EntwicklungsSysteme (ca. DM500 bis über DM20'000). Recht schnelle Fuzzy Extensions sind von TogaiInfraLogic zu bekommen.

TDS2020 Computer



Der Befehlssatz ist orthogonal, alle 8 (16 bit) CPU-Register (R0 ... R7) sind (fast) gleich, dazu gibt es, um 1 MB Adressraum zu ermöglichen, noch 4 8 bit page-Register. Task-Wechsel und Interrupts laufen

Universeller Modul

Die folgenden Technische Daten gelten für eine sehr universelle Konfiguration des Moduls, andere Konfigurationen sind möglich, wichtige Informationen zum Prozessor selbst unter 'µC:' in Klammern.

- Daten-/Adressbus non-multiplexed (µP: auch multiplex),
- Adressbus intern 16 bit, extern 8 bit
- 1 MB AdressRaum (PCMCIA Option bis 2*4 MB)
- 1024 byte internal RAM
- 32 kB internal OTP-ROM
- 26 DigitalInOut (inkl. 2 PortsInOut) (teilweise E-clock, teilweise 'intel')
- 3 external InterruptIN
- 65 total interrupts and DTC-vectors
- 8 priority levels
- 3 InputCapture (timer, counter)
- 8 OutputCompare (timer)
- 8 AnalogDigitalConverter (10 bit)
- 3 PulseWidthModulator (8 bit DigitalAnalogConverter)
- 5 dekodierte Chip-Select
- 1 Inter-Integrated Circuit - Bus (IC)
- 1 RealTimeClock
- 2 WatchDogTimer
- 1/2 * 19.6608 MHz (µC: 0.5 ... 16 MHz)
- 2 SerialInOut (µC: 1 synchron or async.)
- 2 IN 2 OUT nach RS232 (für 2* RS232 oder 1* mit HardwareHandshake nutzbar)
- +5 V 0.05 Referenz
- +5V und +8 V / -8 V geschaltet
- 6 V (5.5 V) bis 16 V Versorgungsspannung

Preis: Lascar Electronic bietet für Mitglieder der Forth-Gesellschaft einen Test-Preis von DM795+MwSt, und auch für Nicht-Mitglieder liegt die komplette EntwicklungsUmgebung noch unter der 1 kDM-Schallmauer.

schnell: In einem einzigen Befehl kann eine beliebige Auswahl der CPU-Register gerettet oder rückgespeichert werden: 8 register in 34 cyclen 3.5 µs. Eine vereinfachte Form von 'DMA', ein DTC (Direct Transfer Controller) kann unabhängig von der CPU serielle Inputs oder Ergebnisse der Zähler oder der AnalogDigital Converter (DAC) in den Speicher verfrachten. Die CPU wird dabei zwar angehalten, muß aber nicht leergeräumt werden. Vom PC her unbekannt, aber für SteuerungsAufgaben sehr wichtig, sind effiziente Bit-Befehle. Diese stellt der H8/532 auf MaschinensprachenEbene zur Verfügung, im Forth sind sie sehr schnell in Code oder auch im Forth verfügbar (siehe unten).

Wenn ich oben von 8 MB möglichem Speicher spreche und im folgenden von bis zu 42 Digital Inputs, so ist das nicht der Prozessor (allein), sondern das Geheimnis liegt in einem zusätzlichen CMOS-GateArray, das so einiges an ZusatzFunktionen ermöglicht. So bleiben z.B. neben allen speziellen Funktionen, wie I²C-bus, Timer- und Analog-Schnitstellen, 2 komplette, also 8-bit-breite, Ports übrig.

Viele Wege führen ins ROM

SpeicherAusstattung und Speicher-Nutzung:

- 1 kB µP-intern (Stacks, Flags ...)
- 16 kB Forth ROM (extended kernel), 16 kB SRAM (Application) und/oder (!)
- 32 kB Forth&Application µP-internal-OTP-PROM
- 32 kB SRAM (variable, data ...)
- ≤512 kB SRAM (Option) (extended DATA) und/oder
- ≤8 MB auf 2* PCMCIA MEMcards (Option, FlashROM, SRAM, ROM) (extended DATA).

OTP = *one time programmable*, das Programm kann ganz oder teilweise in den Prozessor hinein gespeichert werden, und zwar auf allen üblichen EPROMern durch Nutzung des mitgelieferten Adapters von 84 pol Gehäuse auf 28pol 27C256-Äquivalent.

Während der EntwicklungsPhase wird typischerweise mit dem erweiterten Forth-Kernel von 16 kB gearbeitet und dazu in einem SRAM die individuelle Applikation hinzu compiliert. Sichert man das SRAM mit einer zusätzlichen Batterie ab, so kann der TDS2020 auch in dieser Konstellation 'normal' betrieben werden. Für

Die Geschwindigkeit des TDS2020

Forth-System:	TDS2020	---- F-PC-ak ----	
Maschine:	TDS2020	PC	AT
Prozessor:	H8/532	V30	386SX
Takt:	9.8 MHz	16 MHz	16 MHz
empty CODE	4	3	3
empty COLON	12	18	22
D+	11	9	6
UM* == U*	11	10	7
UM/MOD == U/	13	13	10
D*/	633	833	640

D*/ arbeitet bei TDS2020 mit 48 bit intermediate, F-PC mit 64 bit. (Zeiten in µs)

Abb. 2

eine feste und natürlich voll ausgetestete Applikation wird man ROM + SRAM auf einem Programmer auslesen und in ein einziges (!) Eprom oder in den Prozessor (OTP !) selbst hineinbrennen. Je mehr vom Programm sich im Prozessor selbst befindet, desto schneller läuft alles, da der Speicherzugriff dann 16 bit breit erfolgt.

Werden kaum Variable oder Daten verwendet oder diese in das extended (langsamere) Memory ausgelagert, kann eine Applikation bis zu 16+45 == 61 kB umfassen.

Motorola- (E-Clock) (!) oder Intelstyle-Peripherie kann angeschlossen werden. Spezielle Low- and High-Level Befehle sind vorhanden.

Die serielle Schnittstelle erlaubt bis 38400 bit/s, die optionale 2. Schnittstelle (Software) bis 4800 bit/s.

Printplatte Größe: 1/2 EuroCard (100 * 80 mm), verschiedene Lötbrücken und Jumper dienen der Konfiguration. Sockel und Stiftreihen zum 'stacken' oder einstecken auf ein Mutter-board. Auch mit DIN 41612 C Steckverbinder lieferbar. Doppelseitig bestückt, teilweise mit SMDs.

Hardware Multiply

Die Geschwindkeit wollte ich gerne austesten und mit der mir und vielen anderen Nutzern bekannten Performance von F-PC vergleichen. Um diese besonders kurzen Zeiten zu messen, nutze ich spezielle Timer Ergänzungen zu F-PC-ak aus. Die Funktionen selbst sind unter F-PC und F-PC-ak gleich schnell. Bild 2

Die Meßgenauigkeit beträgt bei TDS2020 ca. 1 µs, für F-PC-ak (auf 'IBM' unter DOS) ca. 3 µs -0% +3%, diese Toleranzen gelten für einzelne Messungen. Obige Werte beruhen jedoch auf Mehrfachmessungen. Dieses ist bei den unkontrollierbaren Interrupts auf PCs angeraten, um sinnvolle Genauigkeiten zu ermöglichen. Der TDS2020 wurde mit EXTERNEM Forth gemessen, nach dem 'Brennen' der Applikation in den Prozessor erhöht (!) sich die Geschwindigkeit noch.

Die arithmetischen Funktionen sind in allen Fällen CODE, bis auf `D*/` . Diese COLON Definition setzt sich aus CODE und COLON zusammen. Letztere beruht wiederum auf zeitaufwendigen CODE Operationen. Dieses Beispiel stellt sehr deutlich den Geschwindigkeitsvorteil von CODE gegenüber HighLevel dar. Um diesen Vorteil in der Praxis gut nutzen zu können, stellt der Autor daher auch sehr hohe Anforderungen an den Komfort und die Hilfen, die der Assembler unter Forth bereitstellt (siehe unten).

Trotz des niedrigeren Taktes schlägt sich das TDS2020 Modul recht wacker. Auch wenn `D*/` dort eine einfachere Funktion darstellt. Die Multiplikation in Hardware macht den Prozessor recht schnell.

'Gebrochene' Mathematik

Neben den 'normalen' Forth-üblichen Mathematik-Operatoren sind verschiedene Arithmetik-Systeme zuladbar.

- FIXED: 32-bit Arithmetik mit 0 bis 4 FESTEN Dezimalstellen ($FIXED.tds). (z.B: 4 decimal places: -214'748.3647 ... 214'748.3647)
- FRACTIONS (siehe: *Broodie*, 'Starting Forth' 2nd (!) engl. Editon oder Tracy, 'Mastering Forth'): 16 bit-Mathematik mit 5 Dezimalstellen inklusive SQROOT LN SIN ATAN usw. ($TRIAG.tds). Die Operatoren arbeiten im Bereich 2.000 (in Worten: minus bis plus zwei).
- Dann FRACTIONS von *Dr. A. R. Owens* ($MATH.tds), mit Tabellen-gestützen Polynom-Kalkulationen, Exponenten, Logarithmus, die 48 bit-Zahlen erlauben einen single Integer plus double Fraction, gesamt 48 bit: 32767.999999999 .
- Floating Point als getrenntes Software-Paket mit 8 signifikanten Stellen rundet die Möglichkeiten ab.

Je nach geforderter Schnelligkeit, Genauigkeit und möglichem ROM Platz sind also sehr individuelle Forth-typisch angepaßte Lösungen möglich.

Daten-Schaufel

Das CODE Word in Bild 3 schaufelt Daten gruppenweise vom RAM zum Port. Ist das Programm intern

Schneller Transport vom RAM zum Port

```
CODE MOVEOUT   ( addr --      move SIZE bytes, in an array starting at addr,
               (                to a memory address PORT
               (                24 microcycles 2.44µS per loop from on-H8/532
               (                    RAM or PROM
               (                46 microcycles 4.68µS per loop from off-H8/532
               (                    RAM or PROM
    @R7+          R4  MOVI,   ( start addr of string
    SIZE 1- ##    R3  MOVI,   ( length of string
  HERE TEMP !                 ( keep start of loop
    B @R4+        R2  MOVI,   ( fetch one byte from memory
    B PORT ))     R2  MOVO,   ( send it out
  TEMP @          R3  SCB/F,  ( decrement loop ctr & fall through if -1
END-CODE
-2 ALLOT                      ( recover RAM used for variable TEMP
```

Abb. 3

Arbeit für den Interrupt

```
: INTWORK ( Interrupt. One pass every 1/20 second
          ( As many programs as needed can be added
          ( but total execution time should all be
          ( called must be less than 50mS
    0.1 EVERY AAAA    ( do AAAA every 0.1 sec
    1.0 EVERY BBBB    ( do BBBB every second
    60.0 EVERY CCCC   ( do CCCC every minute
    50MS RETURN;      ( return to foreground program
```

Abb. 4

Timer Interrupt zum Piepen

```
: RINGER ( background task; ring bell every second
      COUNTER @ 20 >            ( true every 20 passes through
      IF                        ( counter reached limit
         BELL                   ( ring bell over serial link
         0 COUNTER !            ( restart counter
      THEN  1 COUNTER +!        ( count off this pass
      61440 LATER RETURN;       ( each unit is 814nS, 50mS total, ie: 1/20 sec

: RING ( start background task
      RINGER $FFB0 6 ONE EIS ; ( enable OCI B of timer 3

-89 +ORIGIN ASSIGN RINGER       ( associate RINGER with
                                ( output compare interrupt B of timer 3
```

Abb. 5

Plätzchen backen per Multitasking

```
: WORK       ( -      word to be entered at power-up
       TASKS                          ( create set of inactive tasks
       PREHEAT ACTIVATE DOPREHEAT     ( start up control of pre-heat oven
       COOK ACTIVATE DOCOOK           ( start-up control of cooking oven
       COOL ACTIVATE DOCOOL           ( start-up control of cooler
       KEYBOARD ACTIVATE INTERACT     ( operator interaction
       ABORT ;                        ( maintain interactive Forth on RS232

SET WORK                \ start work at power up ( cool-start )
```

Abb. 6

Schnelles CASE und inverses Array

```
CODE EXEC:      ( n1 -- )  \ execute the n-th word following EXEC:
       @R7+     R3      MOVI,   \ get TOS
       R3               SHAL,   \ 2*
       R3       R5      ADD,    \ IP+(2*n) --> IP'
       0 @R5    R4      MOVI,   \ IP@ --> W
       @R6+     R5      MOVI,   \ return stack --> IP
                                \ we do not want to return to the old word
                \  NEXT 2+ JMP, ?CSP SMUDGE     \ a little slower
       0 @R4    R3      MOVI,   \ @W
       0 @R3            JMP,    \ JMP @W
  ?CSP SMUDGE
  \ Execute the n-th word following the word EXEC:  in a high level
  \ definition. Counting is ZERO-relative. See also  ASSOCIATIVE:

: T-EX  ( n -- )
       0 MAX   4 MIN
       EXEC: DUP DROP 2+ 2* 2 ;

: ASSOCIATIVE:
   <BUILDS       ( N -- )       \ how many values to compare with
      ,
   DOES>         ( N -- INDEX ) \ compare N with list, return index
      DUP @ ( N PFA CNT )   -ROT DUP @ 0 ( CNT N PFA CNT 0 )
      DO   2+   2DUP @ = ( CNT N PFA' BOOL )
         IF 2DROP DROP   I 0 0   LEAVE   THEN
            ( CLEAR STACK AND RETURN INDEX THAT MATCHED )
      LOOP 2DROP ;

  \ Child of ASSOCIATIVE:
  \ Compare n with values in list and return index if n matches.
  \ If no match is made, then the number of entries,
  \ i.e.  max-index + 1  is returned.  This is the inverse
  \ of an index into an ARRAY .
  \ The returned index is ZERO-relative  ( compatible to EXEC: )

5 ASSOCIATIVE: T-A
       1 , 2 , 3 , 5 , 4 ( !!! ) , ( NO  ';' !!! )
```

Abb. 7

eingebrannt und das (dann kleine) Datenfeld auch intern, so läuft alles besonders schnell.

Bytes können mit über 200 kHz Speed hinaus oder herein geschaufelt werden. Das beinhaltet immerhin die Möglichkeit (KÜRZESTzeitig) eine CD stereophon zu sampeln (176.4 kBytes/s). 512 kB SRAM ergeben 4 s 'CD-sound' mit 2*16 bit bei 44.1 kHz SampleFrequenz.

ADC == AnalogDigitalConverter

In dem QuellFile $SPEECH.TDS zeigt *Peter Rush*, wie Sprache mit dem eingebauten A to D converter erfasst werden kann:

- CAPTURE (da n c --
  capture n samples [8-bit] from A-D channel c, & store starting at 32-bit address da.
  Inner loop times:
  28.48uS on-chip program
  21.17uS off-chip program

Wenn denn unbedingt Sprache gespeichert werden muß, so kann das das Wort CAPTURE notfalls immer noch mit 32 kByte/s in eine Memory Karte. Für Telephon Qualität reichen 8 kByte/s aus: also Telephon an A-D anschließen und einen Anrufbeantworter programmieren. Bei einer 2 MB Speicherkarte sind das 2 Minuten Audio.

Den PWM-Ausgang audio-fähig zu machen sollte auch gelingen; das zeigen die PCs, die es ja zu ganz annehmbarem Quäken bringen können, trotz beschränkter Hardware.

MULTITASKING

Der TDS2020 kann sich zu einer wahren Spielwiese für Multitasking und Interrupts entfalten. Die Möglichkeiten gehen weit über das hinaus, was z. B. F-PC bietet (oder was ein PC normalerweise bietet).

- INTERRUPT MULTITASKING ein Interrupt 'swapt' das Forth für einen neuen Task mit einem anderen.
- COOPERATIVE MULTITASKING das traditionelle Forth-

'Round Robin'-Verfahren mit TaskWechseln 'irgendwann' bei PAUSE .

- PREEMPTIVE MULTITASKING hier wird in deterministische Zeitscheiben aufgeteilt. Prioritäten sind besser kontrollierbar.

Ein Beispiel für drei per Ticker-Interrupt eingeklinkte Hintergrund-Tasks, ganz einfach über das Wort EVERY ist in Bild 4 zu sehen.

In Bild 5 wird ein Interrupt von Timer 3 etwas 'aufwendiger' in die Interrupt Tabelle eingeklinkt über das Wort ASSIGN . LATER kontrolliert die Zeitscheiben (Wiederholrate).

Vier gleichberechtigte Tasks PREHEAT COOK COOL KEYBOARD dienen dem Kekse-Backen: und zwar in einem Ofen mit 3 TemperaturZonen: DOPREHEAT DOCOOK und DOCOOL stellen jeweils PID- (oder Fuzzy- ?) Regler-Routinen zur Temperaturkontrolle dar. Und auch die Bedienung darf Eingaben machen oder .... Jede der Regler-Routinen kann vorher unabhängig geschrieben und auch getestet werden. Gerade diese Unabhängigkeit der erst später durch Multitasking verbundenen Operationen vereinfacht die Entwicklung. Einzeln für sich ist es jeweils 'normales' Forth, weit weniger kompliziert als ineinander verwobene Interrupts. Darüberhinaus können auch Resourcen je Task zugeteilt werden. Prinzipiell könnte jeder Task sogar getrennt kompilieren. Aber es ist wohl unsinnig, auch noch MultiUser verwirklichen zu wollen?!

Weitere Beispiele

In F-PC nutze ich gerne die Worte EXEC: und ASSOCIATIVE (letzteres daher in F-PC-ak nun als schneller ;CODE). Der Source-Code in Bild 7 sollte selbsterklärend sein; gleichzeitig gibt er wertvolle Tips, wie man beim TDS2020 mit dem WORD-Register und dem Instruction Pointer umgehen kann. Der ParameterStack-Pointer liegt auf R7, der ReturnStack-Pointer auf R6, der InstructionPointer auf R5 und der Wordpointer auf R4. Es ist ein INdirect threaded Forth, das mit ROM arbeiten kann. Diese Worte nutzen \ , das zum Wortschatz des TDS2020 ergänzt wurde.

Die 'sinnfreien' Test-Wörter:

- T-EX
  führt je nach Zahlenvorgabe (0...4) die Funktionen DUP DROP 2+ 2* 2 aus.
- T-AS
  ist mit 0 1 2 3 4 zu füttern und gibt den Wert addiert mit 1 zurück, Ausnahme bei 3 und 4. (Verhält sich inverses zu einem Array).

Der Assembler ist recht einfach zu beherrschen, zumindest wenn man sich an MOVI, , MOVO, , MOVEIM, und MOVETPE, gewöhnt hat. Diese bedeuten Move-IN, in ein Register hinein, Move-OUT, aus einem Register hinaus, MOVE-IMMEDIATE immediate Values (hier über ##) und MOVE-To-Port-with-E-clock (PC!). Obwohl der Autor von F-PC postfix (MASM) gewöhnt ist, fällt hier der Forth typische ASM-Stil nicht schwer. Es sind eben beim H8/5xx keine problematischen Fälle vorhanden im Gegensatz zu 8086-Assembler (wie in F-PC prefix bzw. ZF). Der Assembler ist HighLevel strukturierbar.

BEGIN, ... WHILE, ... REPEAT,, wie auch IF, , THEN, usw. sind vorhanden und auch an Beispielen erklärt. Auch Code-Proceduren sind (im Gegensatz zu F-PC) erklärt. Sie sparen RAM, und das ist für einen embedded Controller recht wichtig.

Code kann in Colon-Definitionen eingebettet werden. So können Schleifen leicht (nachträglich) getuned werden. Zwischen F>A und A>F kann inline-CODE eingefügt werden (F-PC: INLINE END-INLINE).

Der Assembler ist an vielen Beispielen erklärt. So gelingt es ganz gut, die Programmbeispiele aus dem Manual von Hitachi in Forth Notation umzusetzten. Das mittlerweile vorhandene Pull-Down-Glossary hilft auch sehr und schnell.

Das TDS2020 Glossary ist auch in F-PC-ak als zusätzliches HYPer-File integriert. Nach Umschaltung der HELP-Funktion wird der MouseClick bzw. Alt-H dorthin redirected. Also kann auch dieser Editor mit Vorteil hier genutzt werden.

fig, Umgewöhnen fällt schwer!

Bei dem mitgelieferten Forth handelt es sich leider um ein fig-Forth; unter anderem ein Zugeständnis an den TDS9090, dem 'schwächeren', älteren Bruder des TDS2020 mit 6303Y Prozessor. Der vorhandene Fundus an Quelltexten sollte weitgehend kompatibel sein. Einige Profis sind noch fig-Anhänger. ANS wird diesen 'Knoten' wohl langfristig zerschlagen.

Folgende HIGHlevel Bit-Befehle bietet der TDS2020:

```
BIT       ( n1 - n2 )   n2 has one bit set. This is bit 0 to bit 15
                        according the input number n1.

ONE       ( addr n - )  Set bit n (0 to 7) at byte address addr.

TOGGLE    ( addr b - )  Exclusive or the content of byte at address addr
                        with the byte b

ZERO      ( addr n - )  Clear bit n (0 to 7) at byte address addr.

Zu beachten: die Syntax ist verdreht gegenüber z.B.
!       ( n addr -- )

AnwendungsBeispiele:

PORT1   1 ONE

PORT3   $12 TOGGLE

[ 0 BIT  3 BIT OR  7 BIT OR  12 BIT OR ] LITERAL XFLAG !
( selbstdokumentierendes Zusammensetzen eines 16b-Wortes aus Bits
innerhalb einer COLON Definition )
```

Abb. 8

`Vlist` statt `WORDS` , `IN` statt `>IN`, `R` statt `R@` und geänderte `VARIABLE`, `DO ... LOOP` , `<BUILDS ...DOES>` , `PICK` , um einmal die wichtigsten zu nennen, dann `M/MOD` als `SM/REM` (symmetrical, statt floored: `FM/MOD`).

Dazu bin ich von F-PC gewohnt, GROSS/kleinSchreibung sinnvoll optisch anzuwenden. F-PC erlaubt das bis auf die erste Phase beim Erweitern des Kernels. Für den TDS2020 ist GROSSschrift gefragt, `-FIND` ist CASE-sensitive . Das sind halt Zugeständnisse an 'kleine' Systeme.

Als F83- bzw. ANS-Fan habe ich mir den Luxus erlaubt, ALIASse zu ermöglichen und intensiv zu nutzen, die freien 16kB für Applikationen erlauben das oft. Bei großen Anwendungen fehlt dann allerdings der Platz. Deshalb und weil der `ALIAS` im TDS2020 Forth recht kompliziert ist, werde ich hier auf den Quelltext verzichten.

Bitte schnelle Bits

In Bild 8 sind die HIGHlevel Bit-Befehle aufgelistet. Unübliche Sicherheit und Komfort gibt es auch bei der Eingabe von Literals, wie in F-PC werden HEX mit vorangestelltem $ ($8A1F) eingegeben und CTRL-Chars mit dem CARET ^ (^C) , in Abweichung zu F-PC erfordert ein ASCII einen " ("A , F-PC: 'A').

TDS2020, der DATAlogger

Die Application DATALOG.tds in Bild 9 liest 8 mal am Tag den Port A und die analogen Eingänge 0 und 1. Die Werte werden vorverarbeitet (skaliert bzw. einige Bits invertiert) und mit ZeitStempel im normalen RAM abgelegt. Ein 'unmöglicher' Zeitwert von $FFFF signalisiert das Ende der Messung beim Auslesen des RAM. Bei diesem 'Simpel'-Beispiel würden die reservierten 8 kB für 127 Tage reichen, BatterieBetrieb wäre problemlos möglich, da zwischen den Messungen der Prozessor und das Board in den low power 'Standby'-Modus gelegt werden.

Etwas aufwendiger programmiert würde der Datenbereich in das extended Memory gelegt werden (bis 512 kB SRAM oder MemoryCard). Bei der Verwendung von Flash ist der EnergieBedarf höher und es werden (exakt) 12 V benötigt. Der TDS2020 ist aber für FLASH vorbereitet, es müssen lediglich einige Brücken anders konfiguriert werden. Sinnvoll kann sein, die Werte in einem SRAM zu sammeln und dann blockweise nach Einschalten einer Spannungs-Pumpe in die FlashCard zu übertragen.

Die dem TDS2020 beiliegende Datei $DATALOG.tds erlaubt das Erfassen von 8 analogen (10bit) und 24 digitalen Signalen auf SRAM-DIP oder PCMCIA-MEMcard. Wird alle 30 Sekunden gesampelt, so reicht der 512 kB SRAM Chip über gut eine Woche. Die Kapazität von 5 Mignon-Akkus sollte dafür mehr als ausreichen. Mit 2*2MB SRAM cards reicht es für 2 Monate, aber dann mit 4 Baby-ALKALI-MANGAN.

Für den extended Bereich stehen die Operatoren `E!` , `E@` und `EDUMP` bereit. Sie arbeiten mit Doubles als Adresse (flat, um hier PC-gewohnte seg:offset Anwender 'aufzuschrekken'). Für SRAM PCMCIA-Cards gibt es dann noch das selbsterklärende `?BATTERY` .

I²C

Andere Store- und Fetch-Operatoren bedienen den I²C-Bus: `I2C@` und `I2C!` . Der Stack-Kommentar ist hier etwas komlizierter:

`I2C! ( 8b a1 a2 -- f ),`

das bedeutet: Speichere 8bit an die interne Adresse a1 im Device auf der I²C- Adresse a2. Das Flag f ist FALSE wenn Fehler auftraten, schließlich könnte es Bus-Kollisionen geben oder ÜbertragungsStörungen oder ... .

Ideal lassen sich z.B. über die Chips PCF8574 bis zu 128 digital I/Os ergänzen, und das angesteuert über 2 Pole (dazu +5V und Masse) und auch einige Meter entfernt (nach

Die DATALOG-Applikation

```
\ DATALOG.tds   Ein einfaches low power Beispiel zum DatenSammeln \ akg93sep24
\ siehe  $DATALOG.tds  from  Peter Rush, TDS

INCLUDE $TIMED.tds      \ load precise  NMI-ALARM   ( from RTC )

$81E0   CONSTANT PORTA

: MV?           ( channel -- mv )
        A-D  5000 1023 */ ;     \ scale up to reference of 5.0 V

VDP @ DATASTART ,               \where to start
1FFF ALLOT                      \ reserve some space in RAM
VDP @ 2- DATAEND                \ end
        \ HERE is in ROM , VDP is Variable Dictionary Pointer (in RAM)

: DATALOG       ( -- )
        232OFF                          \ save energy, switch off RS232
        DATASTART
        BEGIN   4 HOUR                  \ set NMI-ALARM after 4 hours
                W@                      \ set system clock from RTC
                @TIME 1125 M/ NIP       \ convert to minutes till 0:00
                        OVER 1+ !       \ store time (minutes)
                @DATE DROP OVER C!      \ store LOWbyte of DATE
                PORTA PC@               \ input of port A
                        $30 XOR         \ toggle some bits
                        OVER 3 + C!     \ store
                0 MV?  OVER 4 + !       \ mV at analog channel 0
                1 MV?  OVER 6 + !       \ channel 1
                $FFFF  OVER 8 + !       \ stop-identifier (time==FFFF)
                ( 8 bytes to store total )
                8 + DUP MEMEND <        \ any memory left
        WHILE   STANDBY                 \ sleep till NMI ( low power )
        REPEAT
        232ON 100 MS                    \ switch on and wait to recover
        DROP BELL                       \ 'Alarm'
        ." memory exhausted" ;
```

Abb. 9

I²C-Specs: <400 pF total).

Die Befehle `@8574 !8574` sind wohl selbst erklärend, als 'Adresse' dient hier die chip Nummer 0...7 . Generische I²C-Befehle sind als Forth Worte vorhanden, Quellcode in Assembler rundet das ganze ab.

Elektroakustik-Liebhaber werden direkt Audio-Chips per I²C ansteuern, oder z.B. auch eine komplette REVOX-HiFi-Anlage.

Aus der Praxis für die Praxis

Man merkt dem Produkt an, daß die Erfahrung aus vielen weltweiten Profi-Applikationen eingeflossen ist.

Die Liebe zum hilfreichen Detail zeigt sich in so kleinen 'Schmakerln' wie dem Ausdruck der ASCII-Order beim Glossary unterhalb von 'A' (wer hat diese SortierOrdnung schon exakt im Kopf). Auch gibt es Hinweise zu Besonderheiten gegenüber Broodies 'Starting Forth', so daß der Beginner nicht direkt beim ersten Problem ein für allemal frustriert kleben bleibt. Oder eben auch auf dem Board selbst die AdressHinweise am ROM/RAM-Sockel aber auch die (Not-) TelephonNr. zu TDS.

Die Extensions, die dann wirklich in der Praxis benötigt werden, sind vorhanden. Es ist eben KEIN Public-Domain System, was in meinen Augen mehr Vorteile denn Nachteile für den wirklichen Anwender hat.

Weitere Beispiele:

Eine Applikation, die zählt, wie oft der Watchdog angesprochen hat oder die Versorgungsspannung ausfiel (jeder Neustart zählt Flags im geschützten RTC-RAM hoch).

Die Ortungsrufe von Fledermäusen wurden mit TDS2020 erfaßt (allerdings externer ADC), gespeichert und teilweise angezeigt oder Geschwindigkeits-/Tonhöhen-transformiert auf CompactCassette analog gespeichert.

Als Beispiel ist sogar ein PID-Regler inklusive KeyboardEingabe und GraphikLCD Ausgabe beigefügt. Der externe Hardware Aufwand ist minimal.

Als LCDs können die vielen Displays, die den HD44780 enthalten (alphanumerisch) oder den HD61830 (character und grahics mode) direct angesteuert werden. Bis 8 (!) LCDs können gleichzeitig angeschlossen werden. Das Keyboard bis zu 8*8 Tasten erfordert einige Dioden zur Entkoppplung, aber auch das wird man nicht 'Aufwand' nennen wollen.

Anschluß eines 'Centronics'-Printers ist leicht möglich. Auch Dreh-Positionsgeber über einen HP-IC und bis zu 4 (!) Stepper (Schrittmotoren). Die Schrittmotoren laufen über Interrupt im Hintergrund. Wirklich sehr komfortabel.

portabel

Ich war kürzlich viel mit dem Zug unterwegs. Statt am Steuer saß ich am doppelt money-effektiven keyboard eines Olivetti MiniNotebook, natürlich stromsparend mit F-PC-ak Editor auf der FLASHcard, der LOWpower-AntiTurboMode konnte gut genutzt werden. Vom seriellen Port lief ein Kabel zum TDS2020, und der wiederum hing am CamCorderAkku. Eine komplette EntwicklungsUmgebung für unterwegs: 2 * GWG (geringwertige WirtschaftsGüter == sofort steuerlich abschreibbar). Bei -C- würde man wohl kaum CODE am 'lebenden' Objekt IN einem ICE entwikkeln, sondern eher AN einem ICE (In Circuit Emulator) sitzen und, wenn man effektiv arbeiten will, kostet das soviel wie ein (Klein?-)Wagen.

Und die (negative) Kritik

Das Applications-SRAM wird beim Power-UP (COLD) bewußt gelöscht, es sei denn, ein Jumper blokkiert die WRITE-Leitung. Das ist nicht ganz das, was der Autor gerne hätte und z.B. von der 6511 Mini-Bee gewohnt ist. Da gibt man dem Rechner Power und er legt los, denn alle Forth WORDS und Variablen-Werte sind wie ehedem im 'Dallas'-geschützten SRAM.

Beim TDS werden die Variablen in jedem Fall gelöscht (außer im extended Memory) und die Words aus dem SRAM sind ge-blankt. Das ist eine Philosophie-Frage. Noch ein paar Diskussionsrunden und auch das mag besser werden. *Triangle//Peter Rush* zeigte sich immer sehr aufgeschlossen für Wünsche.

Zusammenfassung

Kein Briefmarken- oder Cheque-Card-Modul, 'immerhin' eine halbe Euro. Aber eine gut ausgeklügelte Beschaltung. Gestartet werden kann sofort, per beliebigem TerminalProgramm oder besser mit dem beiliegenden PC-(Basic)-Programm mit TSR-PullDown-Glossary. Die Ausstattung in Hard- und Software (und Preis) sollte es auch 'Startern' erlauben, hier mit Erfolg einzusteigen. Etwas Forth muß man natürlich auch selbst beisteuern, und für speziellere Anwendungen kommt man ohne die vielfältigen Innereien des H8/532 bzw. des beiligenden Prozessor- und Assembler-Handbuches von Hitachi nicht herum, man muß eben ab und zu an den Konfigurationsregistern drehen. Der TDS2020 bietet viel Komfort und als 16bit-ter ungewohnt viel Performance (was beides den Programmieraufwand deutlich senken kann). Insgesamt ist das Triangle Produkt eine runde Sache.

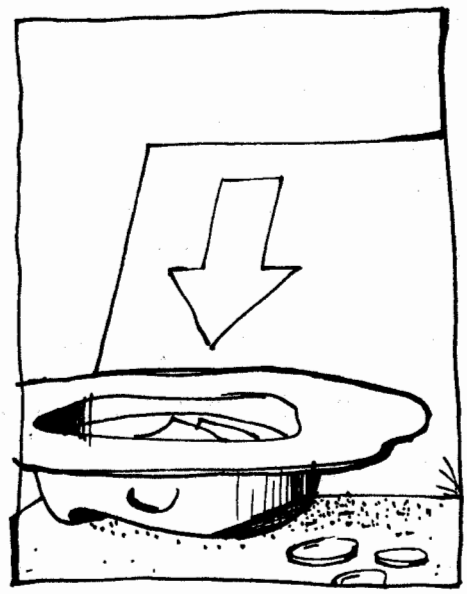
❑

Robotik-News

Das Heft Creeping FORTH IV von *Raphael Deliano* ist an 17 eingetragene Bezieher verteilt. Wie *R. Deliano* berichtet, melden sich zunehmend mehr Interessenten für dieses Projekt-Heft. Auch das Heft IV ist wieder gespickt mit Anregungen und Grundlagen zu den Themen Messen, Steuern Regeln. Einen Schwerpunkt bildet die Sensorik. Bis auf einen Beitrag von *R. Freitag* sind alle anderen des 24 Seiten-Heftes mit RD unterzeichnet. RD hofft, daß sich das ändert. Also, wer bei der Entwicklung eines Labyrinth-Roboters mitmachen will, melde sich bitte bei

Rafael Deliano,
Steinbergstr. 37, 82110 Germering,
Tel. 089-8418317,
Mailbox: 089-8714548 FORTHe.V.
jrd@BBS.FORTH-eV.de

Forth ohne Arbeit?



Die Diskussion ist nicht neu. Immer wieder wird Forth totgesagt, totgeschwiegen und krankgeredet. Autoren, die über Forth schreiben, haben wenig Chancen, in den führenden(!?) Computerzeitschriften veröffentlicht zu werden: "...z.Z. kein Interesse" . Muß Forth auf die Suche nach Arbeit gehen?
In unserer amerikanischen Schwester-Zeitschrift *Forth Dimensions* gab es kürzlich einen diesbezüglichen verbalen Schlagabtausch, den wir für so interessant halten, daß wir ihn den Lesern der VD nicht vorenthalten wollen. Dies vorallem deswegen, weil sich mit Frau *Elisabeth D. Rather* eine Fortherin der ersten Stunde zu Wort meldet. Das zeitraubende und nicht einfache Geschäft der Übersetzung/Übertragung ins Deutsche hat Herr ***Fred Behringer*** (Planegger Str. 24, 81241 München) übernommen. Vielen Dank!

Der Stein
des Anstoßes:

Forth auf der Suche nach Arbeit

Donald Kenney Canton,
Michigan

Ich habe jetzt in Forth Dimensions schon recht viel über den mangelnden Einsatz von Forth in den größeren Entwicklungs- und Programmierungsfirmen gelesen. Als bekehrter Software-Manager hätte ich gern ein paar Gründe dafür erläutert.

Ich will nicht behaupten, daß Software- oder Entwicklungs-Management nach vernunftmäßigen Gesichtspunkten betrieben wird. Nein, es geht vielmehr recht schrullig zu und viel zu viel Beteiligte erwecken den Eindruck, nicht ganz dicht zu sein. Trotzdem sind sie durch die Bank brilliant und selbst auf die Verrücktesten unter ihnen trifft immer noch die Pointe eines alten Witzes zu: Sie sind zwar verrückt, aber nicht dumm. Sie haben gute Gründe dafür, kein Forth zu verwenden.

> Manager haben gute Gründe, kein Forth zu verwenden.

1. Programmieren kann man nicht eben nur mal so auf die Schnelle. Große Systeme erfordern Planung und Zusammenarbeit. Es gibt fast keine Software, die nach ihrer Auslieferung nicht der Wartung, Verbesserung und Fehlerbeseitung bedarf. Und früher als ein paar Wochen vorher vorauszusagen, für welches Vorhaben wieviele Leute nötig sind, ist unmöglich. Wenn sich ein Software-Unternehmen in 22 Dialekten von 13 Programmiersprachen versucht, ist die Wahrscheinlichkeit, zum erforderlichen Zeitpunkt einen hinreichend geeigneten Mitarbeiter an der Hand zu haben, gering. Die meisten Software-Häuser versuchen, mit möglichst wenigen Programmiersprachen auszukommen.

> So sehr ich Forth liebe, bin ich doch auf gar keinen Fall davon überzeugt, daß Forth eine gute Wahl ist, wenn es um die Programmierung großer Systeme geht.

2. Die Wahl der Programmiersprache ist für den Manager bei weitem nicht so wichtig, wie für den Programmierer. Der Manager weiß, daß der größte Teil der verfügbaren Gelder von Bedarfsforschung, Planung, Austesten, Strukturüberwachung, laufenden Unkosten, Einrichtungen, Produktpflege, Handbucherstellung und endlosen Sitzungen verschlungen wird. Das Programmieren an sich verursacht verhältnismäßig wenig Kosten, und da etwas aufzustocken, tut nicht sehr weh. Ich habe die üblicherweise veranschlagten Zahlen vergessen. Sie dürften bei 10% der Gesamtkosten liegen, wenn das auch etwas niedrig angesetzt ist, da dabei die Vorbereitungs- und Testtätigkeiten des Programmierers unberücksichtigt bleiben. Wenn sich jemand für diese Dinge interessiert, möge er in Berry Boehms Buch "Software Engineering Economics" nachschlagen. Jedenfalls bereitet die Wahl einer nicht-optimalen Programmiersprache dem Manager keine allzugroßen Sorgen - zumal dann nicht, wenn die (durchaus realen) Kosten für die Einführung einer neuen Sprache vermieden werden können.

3. Forth ist eine Randsprache. Wenn zusätzliche gut ausgebildete Programmierer gebraucht werden, welcher Manager wird sie einstellen wollen ? Und was werden sie kosten ? Für C, Cobol, Fortran oder BASIC lassen sich sicher am Ort geeignete Leute finden. Der Manager wird sich sogar den Luxus erlauben können, zwischen mehreren bestausgebildeten Kandidaten zu wählen. Für Forth sitzt der nächste Programmierer, den man

bekommen kann, sicher in Fargo in North Dakota. Die Gefahr ist groß, daß man für dessen Leistungen Extraprämien bezahlen muß und ein Vermögen an Reisekostenzuschüssen los wird.

4. Von den Herren in der obersten Etage zusammengestaucht zu werden, ist kein Vergnügen. Warum soll man die Verwendung einer Sprache vorschlagen, von der diese Herren überhaupt noch nichts gehört haben ? Das mindeste, was man dazu braucht, sind gut erfundene Erklärungen. Sicher fällt es den meisten Managern nicht schwer, sich ein geeignetes Märchen auszudenken. Aber warum sollten sie das tun ?

5. So sehr ich Forth liebe, bin ich doch auf gar keinen Fall davon überzeugt, daß Forth eine gute Wahl ist, wenn es um die Programmierung großer Systeme geht. Gewiß, das Programm wird sich glänzend ausnehmen und die einzelnen Teile werden wunderbar anzuschauen sein. Aber der Gesamtentwurf für das System, wenn es den überhaupt gibt, wird aller Wahrscheinlichkeit nach in einen Haufen Flickwerk verwandelt worden sein. Und in der Regel ist er das auch (was im allgemeinen nicht so sehr auf die Unfähigkeit des Systementwicklers zurückzuführen ist, als vielmehr auf die Ungeduld des Managements). Die vielen schönen Programmteile werden höchstwahrscheinlich noch zurechtgefeilt und überarbeitet werden müssen, um ein funktionstüchtiges Gesamtsystem zu ergeben. Das bedeutet, daß jeder mit dem Code eines anderen arbeitet. Forth ist manchmal doch recht schwer zu lesen, und ich möchte nicht wissen, ob die Leute nicht damit enden, daß sie sich gegenseitig mit unüberlegten, ganz unten angesetzten Änderungen am Programm des anderen aus der Bahn werfen. Ich bin mir nicht ganz klar darüber, wie man in einem großem Forth-System Zugriffe auf gemeinsame Daten regeln soll. Nicht, daß ich nicht selbst einen Weg dazu aufzeigen könnte. Aber kann ich sicher sein, daß der auch funktioniert ?

Wenn ich selbst schon nicht genau weiß, wie gut sich diese Ein-bis-zwei-Mann-Sprache bei Acht- oder Zwanzig-Mann-Projekten bewährt, was soll man dann über die durchschnittliche Einstellung der Manager im allgemeinen sagen ? Warum sollten die denn herumexperimentieren, wenn sie genau wissen, daß C oder Fortran das Geforderte leisten ?

Forth hat die Welt nicht erobert - nicht aus technischen Gründen, sondern aus Gründen des Marketings

6. Wenn das Software-Unternehmen vernünftige Kontrolleinrichtungen hat, Steuerung der Konstruktionsabläufe, Biliotheksverwaltung usw., müßten sich die Manager darüber Gedanken machen, wie sie das an Forth anpassen. Es wird sich wahrscheinlich herausstellen, daß sich diese Kontrolleinrichtungen an Forth gar nicht anpassen lassen. Das Unternehmen wird von Grund auf neue Einrichtungen schaffen müssen. Das bedeutet eine Menge Arbeit.

Besteht also keinerlei Hoffnung ? Hat Forth bei Großprojekten nichts zu suchen ?

Keineswegs. Aber die einzigen Stellen, wo Forth eine Chance hat einzudrin- gen, sind Nischen, in denen nichts schon Bestehendes ausreicht oder wo das Mittel, das die Nische bisher ausfüllte, kläglich versagt. Forth hat Marktchancen als Sprache zur Programmierung von Mikrocontrollern oder da, wo nur wenig Speicherplatz vorhanden ist. Es hat Marktchancen als Ersatz für Maschinensprache in einem Umfeld, in dem es nicht auf Höchstgeschwindigkeit ankommt, wo aber die Mittel zur Unterstützung einer gebräuchlichen höheren Sprache fehlen. Es wird ganz bestimmt nicht Tiny-Cs oder Tiny-BASICs da ersetzen, wo das Unternehmen C oder BASIC auch für größere Programmvorhaben verwendet. Vielleicht ließe sich Forth als eine Sprache für Versuchsprojekte verkaufen. Aber Forth wird wohl kaum die eingefahrenen Programmiersprachen ersetzen können. Die meisten sehen nicht ein, wozu eine neue Programmiersprache gut sein sollte. Wenn es schon ADA, hinter dem das gesamte Verteidungsministerium steht, nicht gelang, in die an das Verteidigungsministerium vertraglich geketteten Unternehmen einzudringen, welche Chancen hat dann Forth, von einer freien, an niemanden gebundenen Industrie aufgenommen zu werden ?

Forth hat nur in Nischen eine Chance

Der Schlag
in die selbe Kerbe:
Logik um drei Ecken

Walter J. Rottenkolber
Postfach 1705
Mariposa, California 95338

Donald Kenneys Artikel "Forth auf der Suche nach Arbeit" trifft leider genau ins Schwarze. Auf einer Software-Konferenz im Februar konnte ich von 250 Verkaufsständen folgende Computersprachen-Anbieter ausmachen: Niemand für Forth, Ada, COBOL, Modula 2; ein Anbieter für APL, Lisp, Prolog, Smalltalk; zwei Anbieter für BASIC und FORTRAN, drei für PASCAL, vier für C++ und sieben für C. Zudem war der Großteil der Bibliotheken und GUI-Code-Erzeuger in C gehalten. Die Debugger waren hauptsächlich auf C und C++ abgestellt.

Ich habe eine wirklich gute Geschichte von einem Programmierer in einer großen Software-Firma gehört. Dieser war wirklich davon überzeugt, daß die weite Verbreitung von C daher rührt, daß sich C wie eine High-Level-Assembler-Sprache verhält und für alle möglichen Computer-Plattformen verfügbar ist. Betraut war er aber mit der Aufgabe, dicht gepackten

schnellen Maschinencode für mathematische Funktionen zu schreiben und für diesen dann in umgekehrter Konstruktionsweise C-Routinen zu entwerfen, die als Compilat eben jenen Maschinencode liefern. Die Herren von der obersten Etage hielten das für sinnvoll.

Die Antwort

von kompetenter Seite:

Aufräumen mit einigen weitverbreiteten Mythen

Elisabeth D. Rather
Präsidentin der FORTH Inc.
111 N.Sepulveda Boulevard
Manhattan Beach, California
90266-6847

Ich bin über den Artikel von *Donald Kenney* "Forth auf der Suche nach Arbeit" (Forth Dimensions XV/1) entsetzt. Natürlich habe ich diese Einwände schon oft gehört und ich bin auch der Meinung, daß es für uns alle wichtig ist, die Ängste kennenzulernen, von denen manche Manager geplagt werden. Ich habe aber etwas dagegen, wenn *Mr. Kenney* diese Mythen so ohne weiteres akzeptiert; wenn er annimmt, Leute, die sich so verhalten, gibt es wirklich, obwohl wir uns doch überall vom Gegenteil überzeugen können. Lassen Sie mich seine Einwände Punkt für Punkt durchgehen.

1. Zusammenarbeit, Systempflege, Fehlerbereinigung:

Hinter uns liegen jetzt zwanzig Jahre Zusammenarbeit mit professionellen Forth-Anwendern als Kunden. Die Kunden und wir haben viele Beispiele sowohl von großen Systemen (über die ich weiter unten sprechen werde) als auch von Projekten, deren Entwicklung sich über Jahre hinzog, gesehen. Wir konnten feststellen, daß Forth außerordentlich tragfähig ist. So hat schon unser allererster Kunde, *Bob Barnett* von *Cedar Rapids*, Iowa, ein erfolgreiches Dateneingabesystem auf mehr als sechs verschiedenen CPUs entwickelt, das ihn fast 20 Jahre lang beschäftigte (wir arbeiten seit dem Frühjahr 1974 mit ihm zusammen). *California Municipal Statistics* in San Francisco arbeitet mit einem sehr komplexen Datenbankprogramm, das von *Chuck Moore* 1976 entwickelt wurde und immer noch verwendet wird. Man hat dort eine Menge selbst dazuentwickelt und unsere Hilfe in einigen Spezialfällen in Anspruch genommen, wo es zum Beispiel darum ging, das System an modernere Einrichtungen anzupassen.

Aus jüngerer Zeit ist zu berichten, daß *Federal Express* zwei größere Systeme auf Forth-Basis unterhält, von dem das ältere aus dem Jahre 1987 stammt. Sie haben dort eine Arbeitsgruppe von etwa acht Programmierern, die diese beiden Systeme und daneben auch einige Systeme auf C-Basis warten. Wie sie uns sagten, sind die auf Forth basierenden Systeme bei weitem einfacher zu unterhalten als die C-Systeme. Bei den Forth-Systemen können nachträgliche Änderungswünsche statt innerhalb von Monaten innerhalb nur weniger Wochen erledigt werden.

2. "Die Wahl der Sprache spielt keine Rolle."

Das mag wohl oft die Einstellung der meisten Manager und auch die der Programmierer sein. Und wenn man von, sagen wir, C gegenüber C++ oder Pascal spricht, wird das höchstwahrscheinlich auch stimmen. Wählt man jedoch Forth, so kann man oft sowohl die Kosten wie auch die Bearbeitungszeit (was meist sogar wichtiger ist) um ein Beträchtliches kürzen. Nicht nur, daß die Arbeit schneller vonstatten geht und mit einer kleineren Arbeitgruppe bewältigt werden kann, nein, häufig kommt man auch mit viel kleineren, billigeren Computern aus.

Als wir das Heizungs-, Ventilations- und Air-Condition-System (HVAC) für die GM/Saturn-Anlage in Tennessee entwickelten, wurde uns von der Firma gesagt, daß wir ihnen einige Millionen an Dollar gegenüber mehr konventionellen Methoden eingespart haben. Wir erreichten das, indem wir Software für billige, für industrielle Steuerung bestimmte Z-80-Platinen entwickelten, die von einigen PCs gesteuert wurden. Die Firma mit dem nächstniedrigen Angebot hatte den Einsatz von vielen VAX-Maschinen vorgeschlagen. Bei dem Flughafenprojekt in Saudi-Arabien Mitte der 80iger Jahre haben wir mit 30 Programmiererjahren in 18 Monaten Software entwickelt, bei der ein früheres Team in drei Kalenderjahren mit 100 Programmiererjahren gescheitert war.

Natürlich, wir sind Experten. Aber vor einigen Jahren hat *Cameron Lowe* von *Bell Canada* in einem Zeitschriftenartikel darüber berichtet, daß er und sein Partner ein Projekt in drei Monaten erledigt haben, wo ihre C-Experten drei bis vier Jahre geschätzt hatten - und dabei war das ihr erstes Forth-Projekt.

Solche Unterschiede sind bestimmt nicht als gering einzustufen. Es wird kaum einen Manager geben, der bei einer Einsparung von Jahren und zigtausend oder Millionen von Dollarn nicht aufmerken wird.

3. Verfügbarkeit von Forth-Programmierern und was die kosten:

So, wie das dargestellt wurde, beißt sich die Katze in den eigenen Schwanz. In fast allen Ortschaften gibt es Gruppen von entmutigten Forth-Porgrammierern, die voller Unlust in C programmieren, weil sich ihre zukünftigen Arbeitgeber vor der Verwendung von Forth fürchten, weil es keine Forth-Programmierer gibt.

> Es ist aber auch kein Beinbruch, wenn man Forth nicht beherrscht.

Andererseits ist es aber auch kein Beinbruch, wenn man Forth nicht beherrscht. Dem kann leicht abgeholfen werden. Wir halten hier fast jeden Monat Lehrgänge ab, und wenn mehr als drei oder vier Leute ausgebildet werden sollen, kommen wir auch "ins Haus". Es gibt genügend Ausbil-

dungsstätten, ganz zu schweigen von Büchern und gut dokumentierten Forth-Erzeugnissen, von denen man leicht lernen kann. Unter Umständen ist es für die Firmen das beste, nach wirklich tüchtigen Programmierern (Assembler, C usw.) Ausschau zu halten, die mit dem betreffenden Anwendungsgebiet vertraut sind, und diese dann Forth lernen zu lassen.

So leid es mir tut, muß jedoch auch gesagt werden, daß nicht alle Forth-Programmierer für ein bestimmtes Vorhaben gleichgut geeignet sind.

Die Lesbarkeit der Programme ist in jeder Sprache ein Problem

Wenn Sie es mit einem zeitkritischen industriellen Steuerungssystem zu tun haben, werden Sie mit einem C-Programmierer, der mit dem Prozessor und dem Anwendungsgebiet vertraut ist und dem Sie Forth beibringen lassen, bessere Ergebnisse erzielen, als wenn Sie jemanden nehmen, dessen Forth-Erfahrung sich auf PCs und High-Level-Forth beschränkt. Ja, man kann sagen, ein guter C- oder Assembler-Programmierer mit Forth-Kenntnissen wird bei Verwendung von Forth höchstwahrscheinlich bessere Arbeit leisten, als wenn er in C programmieren würde. (Man lese sich noch einmal das oben Gesagte über *Cameron Lowe* durch - solche Berichte haben wir viele gehört.)

Wenn ich mich mit den Inhabern größerer Forth-Unternehmen unterhalte, frage ich sie gelegentlich, ob sie Schwierigkeiten haben, Programmierer zu bekommen. Obwohl sie alle gern mehr hervorragende Forth-Programmierer hätten, sind sie bei näherem Nachfragen im allgemeinen doch recht zufrieden.

Was die Kosten betrifft, bedenke man, daß es die Gesamtkosten für das ganze Projekt sind, die zählen, nicht der Stundenlohn des Programmierers! Man sollte den bestgeeigneten Programmierer nehmen, der die beste Arbeit zum besten Preis leistet. Wir unterhalten schon lange sehr zufriedenstellende Beziehungen zu Leuten überall im Lande, die es als kostengünstig betrachten, mit uns zusammenzuarbeiten, egal, ob es sich um kleine Ausbesserungen eines schon bestehenden Systems oder um ein größeres Projekt handelt. Wir arbeiten zum Beispiel schon seit Jahren mit *Bob Gherz* zusammen, einem Astronomen an der Universität von Minnesota, der ein auf Poly-FORTH basierendes Teleskopsteuerungsprogramm an drei verschiedenen Observatorien einsetzt. Der ruft uns per Telefon an, trägt ein kleines Problem vor und wir lassen ihm die Lösung mit großer Wahrscheinlichkeit innerhalb weniger Stunden per Modem zukommen. Obwohl er ganze Heerscharen von spottbilligen Studenten höheren Semesters zur Verfügung hat, die nur auf ein Zeichen von ihm warten, findet er es, sagt er, im ganzen gesehen billiger, uns zu rufen.

4. Die Herren von der obersten Etage ...

...würden sich sicher freuen, wenn sie hören, daß Sie einen Weg gefunden haben, ihnen ein schönes Stück Zeit und Geld einzusparen, ganz einfach dadurch, daß Sie eine Methode verwenden, die von vielen führenden Industrieunternehmen Amerikas angewandt wird. Märchen werden nicht benötigt.

5. "Der ursprüngliche Systementwurf wird in einen Haufen Flickwerk verwandelt worden sein."

Ja, muß man da mit einem Seufzer feststellen, das wird er wohl normalerweise auch. Aber selbst Forth-Verleumder geben häufig zu, daß sich Forth wunderbar fürs Programmieren durch fortgesetztes Hinzufügen, Abändern und Verbessern eignet, ein Weg, auf den man sowieso zurückverfällt, wenn der Gesamtentwurf zu wenig durchdacht ist oder überhaupt nicht existiert. Oft ist es leichter, sagen wir für ein Benutzerinterface einfach mal draufloszuprogrammieren, als jemanden zu bekommen, der mit Bleistift und Papier versucht, eine Vorstellung davon zu geben, wie es aussehen sollte. Und wenn dann die einzelnen Entwurfsschritte nochmal überdacht werden, machen sich Interaktivität und Flexibilität von Forth glänzend bemerkbar.

Zu den Großprojekten:

Wir haben es die ganzen Jahre über mit unzähligen Arbeitsgruppen zu 5-20 Programmierern zu tun gehabt. Der Hauptunterschied zwischen diesen und anderen Projekten in anderen Sprachen besteht darin, daß jeder einzelne Programmierer dermaßen produktiv ist, daß man sich auf die Hinterfüße stellen muß, wenn man sichergehen will, daß ihre Produktivität nicht in die falsche Richtung geht. Informationsaustausch und Steuerung des Zusammenspiels sind wichtiger denn je (aber davon kann es ja in keinem Arbeitsvorhaben genug geben). Ist das alles aber ein Grund zur Beanstandung? Hat schon mal jemand einen Manager gesehen, der sich seine Programmierer langsamer oder weniger produktiv wünschte ?

Zudem lösen sich viele Koordinationsprobleme im Zusammenhang mit der Leitung von Arbeitsgruppen von selbst, da man in Forth schneller und mit kleineren Gruppen arbeiten kann.

Die Lesbarkeit der Programme ist in jeder Sprache ein Problem. Dazu gehören auch andere Fragen des Programmierstils. Wir hatten großen Erfolg mit der Aufstellung von innerbetrieblichen Programmiervorschriften, die jeder im Hause befolgte. Aus der bloßen Betrachtung eines in unserem Hause geschriebenen Programmstücks läßt sich seither nicht mehr erraten, wer das geschrieben hat. Und keiner unserer Programmierer braucht sich mehr mit den programmstilistischen Eigenheiten anderer auseinanderzusetzen. Andere größere Erfolgsunternehmen tun dasselbe. Es kommt dabei weniger auf die Art der Programmiervorschriften an als vielmehr darauf, daß sie befolgt werden.

Gemeinsam benutzte Daten und andere technische Gegebenheiten werden in Forth ziemlich genauso behandelt wie in anderen Sprachen auch. Die Details mögen ganz anders sein, die Prizipe sind dieselben.

6. Die Steuerung des Arbeitsablaufs kann nicht angepaßt werden

Meiner Meinung nach werden die Verfahren einfacher. Wir haben zur Lenkung des Zusammenspiels einige Hilfsmittel entwickelt (andere Firmen haben das auch), mit denen wir recht zufrieden sind - obwohl es natürlich immer noch was zu verbessern gibt. *Federal Express* hat ein recht konservatives Firmenverhalten und doch hört man von dort, daß ihre Verfahren gut an Forth angepaßt werden konnten. *Sun Microsystems*, die Forth im *Open Boot* auf jeder SPARC-Station verwenden, setzen UNIX-Werkzeuge ein.

Das Problem bei dem Ganzen ist, daß *Mr. Kenney* diese weitverbreiteten Mythen einfach so hinnimmt, ohne genauer nachzudenken und ohne sie zu prüfen. Sie klingen alle recht vernünftig, es gibt aber viel zu viele Gegenbeweise, um die Mythen als Tatsachen anzuerkennen. Der Forth-Gemeinschaft wird bestimmt kein guter Dienst erwiesen, wenn man dieses Zeug weitergibt, ohne wenigstens mal bei dem einen oder anderen jener Leuten nachgefragt zu haben, die gerade das, was den Mythen nach gar nicht möglich ist, mit großem Erfolg tun.

In welche Richtung wird sich Forth meiner Meinung nach weiterentwickeln ?

In den ersten zehn Jahren des Bestehens der Firma FORTH Inc. programmierte "jeder" in Fortran und keiner hielt eine neue "Hauptsprache" für nötig. Dann programmierte "jeder" kurz in Pascal. Jetzt in C. Und schon taucht wieder eine neue Sprache am Horizont auf: C++ ! Sprachen sind Modeerscheinungen. Sie kommen und gehen (wie andere Moden auch). "Eingefahrene" Sprachen werden immer zunächst nur für irgendeinen speziellen Zweck entworfen, erlangen dann Beliebtheit und werden allmählich auch für andere Zwecke eingesetzt. Forth wurde für einen ganz speziellen Zweck entworfen: Um die Leistungsfähigkeit von Programm und Programmierer, besonders bei Echtzeitanwendungen, zu maximieren. Es gibt einen Haufen Leute, die das immer noch wollen. Forth hat die Welt nicht erobert - nicht aus technischen Gründen, aus Gründen des Marketings: es hat sich kein großes, hinreichend bekanntes Unternehmen dazu entschließen können, viel Geld auszugeben, um Forth bekannt zu machen. Solange das (Wunder) nicht geschieht, wird es immer wieder Firmen mit Phantasie geben, solche wie *AMTELCO* (man rufe sich den in einer kürzlichen Ausgabe erschienenen Artikel von Olaf Meding ins Gedächtnis), die mit Hilfe von Forth ihre Konkurrenz aus dem Felde schlagen und 80% der Marktanteile übernehmen. Hätte Ihre Firma Lust dazu ?

❑

Forth wurde für einen ganz speziellen Zweck entworfen: Um die Leistungsfähigkeit von Programm und Programmierer, besonders bei Echtzeitanwendungen, zu maximieren.

Sprachen sind Modeerscheinungen. Sie kommen und gehen.

Buchbesprechung

von Arndt Klingelnberg

Hans-Jürgen Zimmermann (Hrsg.)
Fuzzy Technologien
Prinzipien, Werkzeuge, Potentiale
VDI Verlag, Düsseldorf 1993,
252 Seiten, ISBN 3-18-401269-7,
DM78.--

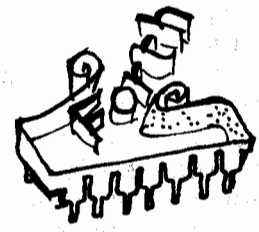
Herausgeber und Mitautor Prof. Hans-Jürgen Zimmermann ist die treibende Kraft in der Deutschen/NordRheinWestfälischen/Aachener Fuzzy Szene. Sein Hauptgebiet ist nicht die Regeltechnik, sondern Operations Research. So bietet das Buch wohltuend eine sehr umfassende Bandbreite der Darstellung rund um Fuzzy. Formeln, Diagramme oder Tabellen bieten sowohl dem Praktiker wie dem (gemäßigten) Theoretiker einen leichten Einstieg. Beispiele aus sehr weitgespannten Bereichen (Beispiele: PKW-AutomatikGetriebes (adaptiver Regler) oder der Einsatz eines Springers an einem Montageband, ProduktionsOptimierung) regen die Lust auf Fuzzy an. Dazu kommen Themen wie z.B: Fuzzy-Petri-Netz, Fuzzy-Historie und zukünftige Potentiale, typische EntwicklungsUmgebungen (alle precompiler-basiert). Es wird dargestellt, wie aus der linguistischen Information über eine Wissensbasis und eine InferenzMaschine eine linguistische Auskunft erhalten wird. Auf 202 ergänzende Literaturstellen wird verwiesen. Ein ordentliches Inhalts- und Sachwortverzeichnis ermöglichen eine sinnvolle Nutzung.

Constantin von Altrock
Fuzzy Logic Band 1: Technologie
Oldenbourg, München, 1993,
247 Seiten Diagramme u. Photos
ISBN 3-486-22673-8, DM49.80

Constantin von Altrock (vom Aachener Institut INForM) erhebt berechtigt den Anspruch, ein Buch für Praktiker gemacht zu haben. Der Schwerpunkt liegt auf Fuzzy Logic, dem Einsatz in der Steuer-/Regeltechnik. Fuzzy Controll wird anhand vieler Fallbeispiele aus Maschinenbau bis Verfahrenstechnik gezeigt, aber auch allgemeinere Anwendungen (wie zB. Datenanalyse zur Kreditwürdigkeit) fehlen nicht. Interessant sind für Regel-Forther solche Punkte wie ein Quelltext (hier in FTL Fuzzy Technology Language), Hinweise zu SpS, Microcontroller-Einsatz oder dedizierte Hardware, Stabilitätsbetrachtungen, Benchmarks, NeuroFuzzy. Auf 162 ergänzende Literaturstellen wird verwiesen. Das Buch geizt nicht mit Bildern, Diagrammen, Bildschirmdarstellungen. Ein ordentliches Inhalts- und Sachwortverzeichnis ermöglichen eine sinnvolle Nutzung. Band 1 ist das preiswerte Einführungsbuch der Reihe (Band 2: Die Anwendungen DM70 , Band 3: Die Werkzeuge DM70 mit FuzzySoftware). Ein effizienter Start hinein in Fuzzy.

68HC11

Noch mehr Forth - 2.Teil



von Holger Dyja
Naumannstr. 13, 10829 Berlin, Tel. & Fax: 030 - 7841257

In dem 1. Teil dieses Beitrags wurden Forth-Entwicklungstools und der Microcontroller 68HC11 vorgestellt. Jetzt wird eine konkrete Realisierung eines Microcontrollerboards dargestellt und anschließend ein Forth TargetCompiler mit einem dazu passenden Debugger beschrieben und die Benutzung anhand von Beispielen erläutert.

Das 68HC11F1-Board

Ausgehend von der Größe des 68HC11F1 wurde ein Board entwikkelt, das mit gesockelten Bauteilen auf engstem Raum soviel Funktionalität wie möglich unterbringt. Gleichzeitig wurde darauf geachtet, daß Erweiterungen nichts im Wege steht. Die technischen Daten entnehmen Sie bitte dem Kasten: Das F1-Board von *H. Dyja.*

Forth-TargetCompiler

Zur Progammentwicklung wurde ein optimierender Forth-TargetCompiler TCOM6811 entwickelt, mit dem es möglich ist, die auf einem Forth-System für 68HC11 Microcontroller geschriebenen Programme optimal zu compilieren.

Als Basis diente der TargetCompiler TCOM von *Tom Zimmer* Version:1.28. Der Aufbau des Compilers erfolgte in 5 Stufen. Zuerst mußte der 68HC11 Assembler/Disassembler geschrieben werden. Da TCOM ursprünglich für einen 8086 konzipiert war, folgte die Umstellung von Intel- auf Motorola-Speicherorganisation. Der Aufbau der Library und die Entwicklung des Optimierers erfolgten quasi zeitparallel.

Für die Routinen der Library ist es erst mal notwendig, den Aufbau des Parameterstacks festzulegen. Um eine optimale Übergabe der Parameter zu gewährleisten, wird das oberste Stackelement (TOS) im Akkumulator D gehalten. Die Entscheidung für das X-Register als Parameterstackpointer war einfach, da fast jeder Maschinenbefehl, der sich auf das X-Register bezieht 1 Byte und einen Maschinenzyklus kürzer ist, als der entsprechende Befehl für das Y-Register. Diese Gegebenheiten haben für TCOM6811 deshalb großes Gewicht, weil jede Stackoperation als MACRO in den auszuführenden Code kopiert wird (mit dem X-Register benötigt man 5 Byte und 11 Zyklen und mit dem Y-Register 8 Byte und 14 Zyklen). Die Library setzt sich aus CODE-, Assembler-MACRO-, COLON- und COLON-MACRO-Definitionen zusammen, wobei für die Entscheidung CODE oder COLON die Komplexität des Wortes ausschlaggebend war, und sie fiel aus Geschwindigkeitsgründen möglichst zugunsten von CODE-Definitionen aus. Für kurze Definitionen bietet sich die Form des MACRO's an, aber im Zusammenhang mit der Entwicklung des Optimierers wurden auch solche Definitionen als MACRO ausgelegt, die sich besonders gut optimieren lassen. Der Optimierer von TCOM6811 optimiert im Gegensatz zu seimem Vorbild TCOM viel kleinere Segmente, sodaß es notwendig wurde besondere Maßnahmen zu ergreifen, die verhindern, daß z.B. der Operand eines OP-Codes mit einem OP-Code selbst verwechselt wird. *Tom Zimmer* hat die Optimierung PeepHole Optimierung genannt (VD93/2).

Z.Zt. ist die Anpassung an den ANS-Standard in Arbeit, mit dem Ziel, die Wortsätze `CORE`, `CORE-EXT`, `DOUBLE`, `STRING` und `EXEPTION` möglichst vollständig zu implementieren. Zur Unterstützung der on-chip Peripherie gibt es Worte

Stichworte

68HC11
Micro-Controller
TCOM6811
Optimierung
Bit-Befehle
Debugger

Das F1-Board von H. Dyja, die Einzelheiten

Die Abmessungen und der elektrische Aufbau sind so gewählt, daß das 68HC11F1-Board sowohl alleinstehend befestigt und betrieben, als auch mit anderen IO-Einheiten kombiniert werden kann. Mit einer **64-poligen VG-Leiste nach DIN41612** besteht die Möglichkeit busorientierte Systeme aufzubauen, wobei das Board direkt in ein Gehäuse eingeschoben werden kann, dessen Schienen auch aus Metall bestehen dürfen. Zusätzlich stehen auf einer **74-poligen Stiftreihe** alle Signale für flexible Erweiterungen zur Verfügung.

Das 68HC11F1-Board verfügt über **zwei 28-polige Sockel**, die wahlweise mit maximal **32kBytes RAM** und **64kBytes EPROM** oder bis zu **2*32kBytes RAM** bestückbar sind.

Besonders flexibel ist das Board durch die **frei verfügbaren, programmierbaren I/O-Chip-Selects**. Der General Chip-Select kann je nach Bestückung für den Anwender als RAM- oder Peripherie-Chipselect genutzt werden.

Um die **Stromsparmodi** des 68HC11F1 optimal zu unterstützen, wurden die Versorgungsleitungen aufgetrennt, was ein selektives Abschalten der Verbraucher ermöglicht. Damit sind batteriebetriebene Geräte einfacher realisierbar.

Zur Erhöhung der Sicherheit des Boards verfügt es über einen **Power-on-Reset**, **Powerfail-Detektor** und einer **Chipselect-Verriegelung** beim Umschalten auf **Standby-Versorgung**. Ein **Watchdog** und ein **Clockmonitor** sind im 68HC11F1 schon integriert.

Das **AD-Wandlersystem** des 68HC11F1 wurde um eine **Referenzspannungquelle** erweitert und mit dem **RS232 Pegelwandler** ist das **SCI(Serial Communiations Interface)** der MCU sofort betriebsbereit.

für den Timer, den AD-Konverter, das EEPROM und zur Bitmanipulation.

Die Arbeitsweise des TargetCompilers

Der Compiler liest die Quelltexte ein, und nach ' : ' wird ein neues Wort im Target-Dictionary aufgenommen. Jetzt passiert folgendes: sind in der neuen COLON-Definition die zu compilierenden Worte als MACRO in der Library vorhanden, so wird das MACRO eingebaut und ggf. mit dem vorangehenden MACRO optimiert. Andernfalls ist es CODE- oder COLON-Definition und es wird untersucht, ob es schon "referenced" ist. In diesem Fall ist die Definition schon im Target vorhanden und es muß nur ein JSR <Adresse> (JSR=JumpSubroutine) eingetragen werden; ansonsten wird solange eine "unresolved chain" (JSR <chain>) aufgebaut, bis die Definition endlich "referenced" ist und alle vorhergehenden "unresolved" JSR's "resolved" werden können. Das "resolve" geschieht normalerweise nach dem ' ; ' der letzten COLON-Definition. Das führt dazu, daß im endgültigen Target-Objektprogramm nur die Definitionen eincompiliert sind, die auch wirklich gebraucht werden. Im Vergleich zu Forth-Systemen ist der Speicherverbrauch von TCOM6811 je Definition größer, da die meisten MACRO's mehr Speicherplatz als ein JSR <Adresse> verbrauchen. Dennoch ist der TargetCompiler im Vorteil, da er mit 0 kByte anfangen kann. Erfahrungsgemäß erreicht TCOM6811 sehr schnell 8kByte CODE und entfaltet seine größte Optimierleistung bis 16 kByte. Höhere Definitionen wird man auf den ersten Blick mit "JSR-threaded"- Forth verwechseln können, da dann kaum noch optimierbare MACRO's aufeinanderfolgen. Dabei sollte aber nicht vergessen werden, daß niedrige Definitionen sehr gut optimiert sind und damit auch alle höheren Definitionen schneller ablaufen.

Als Einschränkungen von TCOM6811 ist auf alle Fälle die fehlende Interaktivität zu sehen, was bei einer endgültigen Applikation nicht ins Gewicht fällt, aber bei der Progammerstellung hilfreich ist. Desweiteren muß sich der Programmierer bei der Entwicklung des Programms immer im klaren sein, daß die Compilation auf dem PC abläuft und dieser zur Compilierzeit nichts vom aktuellen Zustand des 68HC11 wei.

Mit dem TargetCompiler steht gleichzeitig ein Präfix-Assembler zur Verfügung, mit dem die für Controllerapplikationen notwendige Interruptprogrammierung (siehe Listing 2) leicht zu bewältigen ist. Anzumerken ist ferner, daß TCOM6811 über Kommandozeilenoptionen an die vom 68HC11-Typ geforderte und vom Anwender gewünschte Speicheraufteilung einstellbar ist und alle 68HC11-IO-Register (für die Typen A0...N4) als Konstanten vordefiniert sind. Man muß sich also keine Adressen merken.

Die ökonomische Ausnutzung des RAM's wird durch den Datentyp

Abbildung 1: Bildschirmaufbau des Debuggers MONI11

```
  FILE  VIEW  RUN  DATA  TCOM OPTIONS  MISC  HELP
RETURN-stack: 3FB> 00 12 34 56 FF FF FF FF ....
  USER-stack: [2]                          0000 0000
0000 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F ................
0010 10 11 12 13 14 15 16 17 18 19 1A 1B 1C 1D 1E 1F ................
0020 20 21 22 23 24 25 26 27 28 29 2A 2B 2C 2D 2E 2F  !"#$%&'()*+,-./
0030 30 31 32 33 34 35 36 37 38 39 3A 3B 3C 3D 3E 3F 0123456789:;<=>?
WATCH: WORD        0044  XTCNT    :1234
                  Dis-/Assembler-Debugger
                      »HAUPTFENSTER«
                  FORTH-Debugger
: IMAGE_INIT 1.64CYCLES DEF_INIT MAIN
: MAIN       SQW_INIT BEGIN XTCNT @ $0F AND 0= UNTIL XTCNT @ $0F
AND 0= UNTIL XTCNT @ $FFF0 AND 0=

CURR:8123            START:8000
PC:STOP CCR:SXHINZVC A:00 B:00 X:037A Y:0000 S:03FB
   »allgemeine Meldungen«
```

\ ¦	ASM FTH	COM1:(rate)	BTDNAC P	MCU-PGM: STOPPED RUNNING	MCU-MODE: boot test exp single

Glossar Bitmanipulation

```
CSET      ( c-mask c-addr -- )
          setzt die 8 bit Maske c-mask bei c-addr

CRESET    ( c-mask c-addr -- )
          setzt die 8 bit Maske c-mask bei c-addr zurück

CTOGGLE   ( c-mask c-addr -- )
          toggled die 8 bit Maske c-mask bei c-addr

CSET?     ( c-mask c-addr -- flag )
          flag<>0, wenn die 8 bit Maske c-mask bei c-addr gesetzt ist

FLAG
Compilierzeit: ( "name< >" -- )
Laufzeit:      ( -- c-mask c-addr )
          legt eine 1 Bit Flagge bei c-addr mit der Maske c-mask an

MASK&ADDR
 Compilierzeit: ( c-mask c-addr "name< >"-- )
 Laufzeit:      ( -- c-mask c-addr )
          legt die Maskenkonstante mit der 8 bit Maske c-mask und und der Adresse
          c-addr an. Kann leicht mit ' 2CONSTANT ALIAS MASK&ADDR definiert werden.
```

Abb. 2

FLAG gefördert, der eine 1 Bit Flagge anlegt und zusammen mit dem Datentyp MASK&ADDR den Bitmanipulationswortsatz (siehe Glossar Bitmanipulation) abrundet. Mit diesem Wortsatz lassen sich besonders effektiv einzele Bits des 68HC11-IO-Registersatzes einstellen. Die Bitmanipulationsworte nutzen die Bitmanipulations- und BitTest&Branch-CPU-Instruktionen des 68HC11 aus. Im Gegensatz zu der Behauptung des ersten Teils dieses Beitrags, daß die BitTest&Branch- und Bitmanipulations-Instruktionen nur schwer von einem Forth-System ausgenutzt werden können, kann der TargetCompiler aufgrund seiner optimierenden Eigenschaften diese Befehle sehr gut eincompilieren.

Die optimierte TCOMpilierung eines Programms zum Ein- und Ausschalten eines Portpins (siehe Listing 1) ergibt eine Periodendauer von 15µs mit 2 MHz Systemtakt. Mit einem Assembler kann man für dieses Problem 9µs erreichen. Versuche mit dem small-C Compiler (siehe Listing 1a) ergaben 130µs und dem Leser sei es überlassen die Zeiten eines Forth-Systems zu ermitteln.

Die Geschwindigkeit von TCOM6811 läßt sich auch an Floatingpoint Operationen abschätzen. Dazu wurde ein Floatingpoint Modul von R. L. Smith an den 68HC11 und TCOM6811 angepaßt. Mit 4MHz Systemtakt ergaben sich folgende Ausführungszeiten:

- F+ < 0,5 ms
- F* ca. 0,5 ms
- F/ < 0,8 ms
- FLN ca. 1,0 ms

TCOM6811 ist auch sehr leicht auf den 68HC16 zu portieren. Nach dem Überarbeiten des Assemblers konnte ohne weiteres die 68HC11 Library übernommen werden. Der so erhaltene Compiler ist natürlich nicht optimal, da der 68HC16 zu den 68HC11-Instruktionen leistungsfähigere Befehle kennt, die in der vorhandenen Library noch nicht verwendet werden.

Monitor und Debugger

Ausgangspunkt für die Entwicklung des Monitors und Debuggers

Listing zu 68HC11 (Holger Dyja)

```
$08 PortA MASK&ADDR PA3     \ definiert den Port Pin PA3

: TOGGLE_PA0   ( -- )
\F1 $08 DDRA CSET           \ DDR PortA Bit3 auf Ausgang, aber nur beim 68HC11F1
    BEGIN
          PA3 CSET          \ setze PA3
          PA3 CRESET        \ setze PA3 zurück
    AGAIN ;                 \ endlos
Listing 1a:
main()
{ char  *PortA, *DDRA;
  PortA = 0x1001;
  DDRA  = 0x1000;
  *PortA |= 0x08;           /* = $08 DDRA CSET */
  while(PortA==0x1000)
       { *PortA |= 0x08;    /* = PA3 CSET   */
         *PortA &= 0xF7; } /* = PA3 CRESET */
  }
#asm incl c11lib.txt
#endasm

In Listing 1 und 1a soll ein Programm simuliert werden, da zu einem
Zeitpunkt den Portpin PA3 setzt und zu einem anderen Zeitpunkt zurücksetzt.

Listing 2:
\ Beispiel für Interruptprogrammierung:
$80 TMSK2 MASK&ADDR TOV-MSK       \ definiert TOV Interruptfreigabemaske & Adresse
$40 TMSK1 MASK&ADDR OC2-MSK       \ definiert OC2 Interruptfreigabemaske & Adresse
$40 TFLG1 MASK&ADDR OC2-FLG       \ definiert OC2 Interruptflagge & Adresse
$40 TCTL1 MASK&ADDR OC2-TOGGLE    \ definiert OC2-Toggle-Mode
$01 PORTG MASK&ADDR PG0           \ definiert PortG Pin 0
          FLAG      SYNC-FLG      \ definiert eine 1 bit Flagge
          VARIABLE  OC2-RATE      \ Output-Compare2 Periodendauer
          VARIABLE  XTCNT         \ extended Timercounter
$FFDE INTERRUPT-CODE EXT.TIMER    \ CODE-Interrupt-Definition
                                  \ $FFDE ist der Vektor für TimerOverflow-Interrupt
      LDAA # $80                  \ TOV-Interruptflagge zurücksetzen
      STAA TFLG2
      LDX  XTCNT
      INX                         \ extended Timercounter weiterzählen
      STX  XTCNT
      END-IRCODE                  \ Interruptdefinition abschließen, RTI und
                                  \ Interruptvektor installieren
: SQW           ( -- )            \ COLON-Interrupt-Definition
INTERRUPT TOC2-IR
   OC2-FLG C!                     \ Interruptflagge zurücksetzen
   SYNC-FLG CSET?                 \ 2. synchones Rechteck ausgeben?
            IF PG0 CTOGGLE THEN
   OC2-RATE @ TOC2 +!             \ Rechteckzeit setzen
END-INTERRUPT ;                   \ Interruptdefinition abschließen und
                                  \ Interruptvektor installieren
: SQW_INIT ( -- )
   SEI
   TOV-MSK CSET                   \ TOV-Interrupt freigeben
   $80 TFLG2 C!
   $1000 OC2-RATE !               \ OC2-Interrupt freigeben, toggle
   OC2-FLG C!
   OC2-MSK CSET
   OC2-TOGGLE CSET
   $01 DDRG CSET                  \ DDR PortG Pin 0 auf Ausgang
   CLI ;

: MAIN     ( -- )
   SQW_INIT
   BEGIN
        BEGIN XTCNT @
              $0F AND             \ bei 2MHz Systemtakt
        0= UNTIL                  \ wird auf ca. 1s synchonisiert
        .... ( Bedingung 1 )
       IF SYNC-FLG CSET ... THEN  \ 2. synchones Rechteck einschalten
       .... ( Bedingung 2 )
       IF SYNC-FLG CRESET ...THEN \ 2. Rechteck ausschalten
   AGAIN ;
```

MONI11 waren Monitorprogramme für verschiedene Microcontroller, die meist wenig komfortabel ein Debugging des zu testenden Programmes ermöglichen. Bei einigen 68HC11-Typen wird der BUFFALO-Monitor im eingebauten ROM geliefert, der beim Singlestep und bei Breakpoints nur die CPU-Registerinhalte und den aktuellen OP-Code darstellt. Da ich keine Lust hatte, OP-Codes mit Mnemonic auswendig zu lernen, suchte ich nach Möglichkeiten Disassembly, Symbole und Sourcecode beim Debugging verfügbar zu machen. Nach einigen mehr erfolglosen Versuchen Terminalprogramme an die vorhandenen Monitorprogramme anzupassen, ging ich zu dem MONI11 Konzept über.

Die Vorteile des Monitors MONI11 liegen darin, daß sich im 68HC11 nur ein kleines "Kommunikationsprogramm" befindet, das mit einem Hostrechner via RS232-Schnittstelle kommuniziert. Der Monitor und Debugger befinden sich im Hostrechner. Das ist einerseits vorteilhaft, weil der 68HC11 fast vollständig für Anwenderprogramme zur Verfügung steht und andererseits kann der Debugger leicht auf die Symbole zugreifen, da sich die Quelltexte auch im Hostrechner befinden. So kann der Benutzer einfach und komfortabel auf Assemblerebene arbeiten, da beim Disassemblieren, nach Breakpoints und Singlestep die Adressen symbolisch dargestellt werden.

Da es aber oft sehr unübersichtlich ist, Programme auf Assemblerebene zu verfolgen, legte ich bei der weiteren Entwicklung des Monitors darauf Wert, den Forth TargetCompiler optimal zu unterstützen. So ist es möglich, durch die Forth-Worte im singlestep zu gehen. Dabei kann man auswählen, ob man in die Worte hineinspringen oder sie nur ausführen will. Nach jedem Schritt zeigt der Debugger Parameter-Stack, Return-Stack, Memory-Dump und die CPU-Register neu an.

Zusätzlich gibt es noch einen EEPROM- und CONFIG-Register-Editor. Der 68HC11-IO-Registersatz läßt sich komfortabel und übersichtlich bis ins letzte Bit interaktiv einstellen. Schließlich ist es möglich die getesteten Programme mit einem Hilfsprogramm in On-Chip EPROM's und OTPROM's zu brennen (z.B. 68HC711E9 oder 68HC711G5).

Forther werden warscheinlich die Kritik äußern, daß compilierte Programme immer wieder zum Testen in den Controller geladen werden müssen und damit Zeit verschwendet wird. Das kann ich nicht 100%-ig widerlegen. Aber TCOM6811 compiliert sehr schnell die Anwenderprogramme, und wenn man einen 68HC11 mit Baudrate-Quarz (z.B. 14,746 MHz) verwendet, können auch Objekt-Programme von 32kByte mit bis zu 115kBaud rasend schnell geladen werden.

Es gibt z.Zt. 4 Debbugger-Versionen die im Wesentlichen die Typen 68HC11A0..KA4 abdecken. Deren besondere Eigenschaften sollen hier nicht beschrieben werden. Aber der interessierte Leser kann sich gerne von mir zusätzliche Informationen beschaffen.

MONI11 wurde mit TCOM Version:1.28 compiliert.

Die Abbildung 1 deutet den Aufbau der MONI11-Bildschirms an. Im Hauptfenster habe ich versucht das Programm aus Listing 2 darzustellen. Der FORTH-Debugger und der Dis-/Assembler-Debugger im Hautpfenster sind nur alternativ verfügbar.

❑

NEWS +++ ANS-NEWS +++ ANS-NEWS +++ ANS

FPC-Lite (von *U. Hoffmann* für volksFORTH, von W. Allinger weiterverbogen für F-PC) war ein kleines ca. 155 Worte Forth83 dem größeren System für Lern-/Übungszwecke aufsetzt. Nun gibt es auch von U. Hoffmann ein Mini-ANS (ANSI.seq) für F-PC. Hilfreich ist, daß *U. Hoffmann* sich auch die Mühe gemacht hat, die StackKommentare ANS-mäßig aufzubereiten. Da weitgehend ALIAS -e verwendet werden, beansprucht ANSI.seq 'nur' Header-space (Y-space), allerdings wird dadurch auch bewirkt, daß eine DEkompilierung (SEE) weiterhin normales F-PC-Forth ergibt, d.h. diese Lösung übersetzt ins Forth83.

———

Die ak-Version von F-PC (F-PC-ak v.4.20) ist nun auch ab KERNEL schon weitgehend zu ANS-Quelltexten kompatibel. Weitere ANS-Wörter sind in F-PC-min.exe und F-PC-ak.exe enthalten. Zu F-PC-ak wird auch IFDIRECT.seq für [IF] und COLON-NN.seq für :NONAME mitgeliefert. Die frühe Einbindung von ANS-Worten ergibt auch bei DEkompilierung ANS-Worte, auch für normale F-PC (Forth83) Quelltexte (Übersetzung in ANS). Eine (bewußte) Ausnahme bildet POSTPONE , das -- um auf der sicheren Seite zu liegen -- wie gewohnt in COMPILE oder [COMPILE] DEkompiliert. Arndt Klingelnberg weist in diesem Zusammenhang nochmals darauf hin -- es bestanden hier MisTverständnisse -- daß der Kernel und F-PC-min und die hierzu gehörenden Quelltexte aus dem F-PC-ak Paket Public Domain sind (nicht jedoch das gesamte Paket mit den komfortablen Erweiterungen und den erweiterten Hilfetexten.)

———

Auch ist von *A. Klingelnberg* auf der MailBox ein ANS.seq bzw. ANS-S.seq als Public Domain verfügbar, was das original F-PC weitgehend für ANS-Quelltexte tauglich macht. ANS.seq ist mit vielen Kommentaren und Test- SAMPLES: versehen, es vermeidet weitgehend Doppel-Definitionen durch konditionales Kompilieren. Bei harten Inkompatibilitäten wie bei SEARCH and COMPARE wurde die Lösung über eine jeweils spezielles VOCABULARY gewählt, das 'normale' Forth Wort ABORT" -et aus Sicherheitsgründen, und zwar IMMEDIATE -ly ! Das kürzere einfachere ANS.seq definiert eventuell vieles doppelt. Für original F-PC muß eventuell AK-356.seq zur Angleichung der verschiedenen F-PC Versionen zugeladen werden.

———

!!! F-PC widerspricht grundlegenden ANS-Anforderungen !!! (was aber nur selten stört). *Ulrich Hoffmann* führt an (akg übersetze locker aus ANSI.seq):

Der ÄußereInterpreter des F-PC entspricht nicht der ANS-Forth Norm. Ja weit schlimmer, die Grundstruktur verhindert eine Anpassung durch einfache Erweiterungen. Da andere Teile des F-PC Gesamt-Paketes auf dieser Struktur beruhen, wird besser nichts geändert um nicht Pandora's (bug-)Büchse zu beschwören.

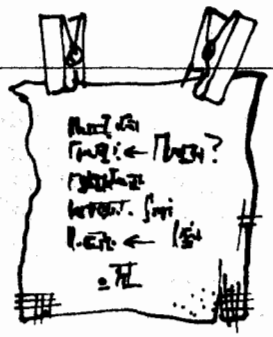
Im Detail: ANS-Forth fordert von dem Wort] , daß es keinerlei Effekt auf den InputStream hat. Das Wort : soll dagegen genau ein 'Token' aus dem InputStream erfassen.

Das ist so nicht gegeben in F-PC, F83, polyFORTH ... , dort ist] selber der Äußere Interpreter und der wiederum wird aufgerufen durch : .

FIG-Forth settzt STATE , um den Status des Äußeren Interpreters aufzuzeigen. Eine EinfachSchleife übergibt an den Interpreter oder den Kompiler. Auch volksFORTH setzt STATE , allerdings als ExecutionVector. So funktionieren] und : im ANS-Forth Stil und sind effizient implementiert.

Auch KEY ist nach *A. Klingelnberg* nicht streng ANS kompatibel. Das gilt für sehr viele (zumal PC-) Systeme.

Aufnehmer im Angebot



von Rolf Kretzschmar

Es ist soweit! Wir können Aufnehmer anbieten. Wenn Sie jetzt nicht sofort wissen, was damit gemeint ist, sollten Sie zunächst den Text im Rahmen (Der Schröder-Aufnehmer) lesen.

Bei einigen Beiträgen, die wir in der VD veröffentlichten, waren wir der Meinung: DAS isses! Darauf werden wir viel Resonanz haben. Doch meistens mußten wir feststellen: Nix iss! Null Resonanz. Jedenfalls keine, von der wir erfahren hätten. Und dann; ein Schuß aus der Hüfte; ein Gedanke ausformuliert zu einem kleinen Apell: Aufnehmer gesucht! (VD 1/93) Siehe da! Es bewegt sich doch etwas in der Gemeinde. Als ich das Problem mit *Ulli Hoffmann* am Telefon besnacke, kommt am nächsten Tag ein Fax mit der Lösung an. Das wiederum erzähle ich dem *Arndt Klingelnberg* und erhalte zwei Tage später gleich zwei Lösung; englisch dokumentiert und mit zahlreichen Sonderfällen, an die ich nicht gedacht hatte. Nicht, daß ich gesagt hätte, ...das brauch ich dringend! Nein, es war eigentlich nur die Ankündigung meines Aufrufes. Und doch mußte die Aufgabenstellung einen starken Aufforderungscharakter haben. Nach Erscheinen der VD ging es nämlich munter weiter: *Wolfgang Schemmert* "konnte es nicht lassen, das Problem zu lösen". Er hat es in C programmiert und fest in sein Terminal-Programm (speziell für die Arbeit mit Mikro-Controllern) eingebaut. *Claus Vogt* schickt eine Lösung für volksForth 3.8 auf PC. *Friederich Prinz* arbeitete gemeinsam mit seinem Bruder *Ulrich* eine Lösung für ZF aus (Siehe: ZF-RECORDER). Dann versucht *Jörg Staben* mich mit seiner Lösung davon zu überzeugen, daß sowas im Zeitalter von Windows und visualBASIC kein Mensch mehr programmieren muß (Siehe: Der Staben-Aufnahmer). *Arndt Klingelnberg* kommt danach mit einer Spezial-F-PC-AK-Lösung rüber: seine Version kann das Geforderte ohne viel Federlesen schon lange...(Siehe: Der Klingelnberg-Aufnehmer) Und dann, ganz spät, läßt *Michael Schröder* von sich hören und seine Lösung sehen: Ein vorbildlich kommentiertes Stück F-PC-Forth.

Über die Resonanz habe ich mich sehr gefreut! Den ersten Preis haben *Friederich* und *Ulrich Prinz* verdient, weil sie mein Anliegen, Anfängern ein gutes Beispiel zu liefern, voll umgesetzt haben. Aber auch der Beitrag von *Michael Schröder* erfüllt voll meine Absicht. Beide Beiträge werden daher als erste in der VD veröffentlicht. Die Mini-Lösungen von *Arndt Klingelnberg* und *Jörg Staben* bekommen den Innovations-Preis und werden auch vorgestellt. Ach ja; ich vergaß zu erwähnen, daß alle Lösungen in die Münchner Mail-Box eingespielt werden und daß ich keinerlei Wertung der Programmierlösung vornehmen werde oder vorgenommen habe. Das war auch nicht mein Anliegen! Ich wünsche mir nur, daß nun von den Adressaten, den Forth-Neulingen, Rückmeldungen eintreffen.

❑

Der Schröder-Aufnehmer `logfile.seq`

```
\ LOGFILE.SEQ  Eingabe-Recorder für die Interpretereingaben   mis
comment:
*******************************************************************************

        In der VD 1/93 stellte Rolf Kretzschmar eine Programmieraufgabe.
        Es sollte ein Modul entwickelt werden, das die Eingaben beim
        Programmieren im Direktmodus in ein Log-File "mitschreibt", um
        besonders Programmieranfängern das Zurückverfolgen von Fehlern
        zu erleichtern. Das Modul sollte folgende Anforderungen erfüllen:

          - das Log-File soll eine normal lesbare Textdatei sein

          - am Anfang und Ende des Log-File soll das aktuelle Datum
            und die Uhrzeit eingetragen werden

          - nach jeder Betätigung der ENTER-Taste soll das Log-File
            aktualisiert werden; das ist besonders wichtig, um auch
            nach einem Systemabsturz die auslösende Eingabe nach-
            vollziehen zu können.

        LOGFILE ist eine Lösung dieser Aufgabe für das Forthsystem F-PC.

        LOGFILE ist Public Domain.

        Autor:
        Michael Schröder, Magdeburger Str. 26, 39387 Oschersleben,
        Tel.: 03949-4086

*******************************************************************************

        LOGFILE verändert die Funktion des (äueren) Forth-Interpreters
        indem es dem "deferred word" EXPECT eine neue Bedeutung zuweist.
        EXPECT erwartet eine Benutzereingabe von der Tastatur. Seine
        Funktion muß also nur soweit geändert werden, daß
            1) die ursprüngliche expect-Funktion ausgeführt wird, die
               die Tastatureingabe abfragt und
            2) diese Eingabe in das Log-File geschrieben wird, bevor sie
               an den Interpreter weitergegeben wird.

        Das macht die Definition LOG-EXPECT.

*******************************************************************************
```

ZF-RECORDER

von Friederich & Ulrich Prinz
Homberger Straße 335, 47443 Moers

In der VD 1/1993 (Aufnehmer gesucht!) rief *Rolf Kretzschmar* dazu auf, einen automatischen Protokollierer für Eingaben auf der Interpreterebene einer Programmiersprache zu schreiben. Hier ist eine Lösung für ZF!

In der VD 1.'92 hatte *Rolf Kretzschmar*, für die diversen FORTH-Systeme nach einem LOG-File verlangt. Hintergrund dieses Ansinnens ist, daß Anfänger oft Probleme haben, ihre Schwierigkeiten zu beschreiben, weil ihnen einfach noch zu viele Voraussetzungen fehlen, um grundsätzliche Fehler und Irrtümer erkennen zu können. Der versierte Forther, der längst 'über aller Trivialität' steht (was aber zum Beispiel auch die Autoren durchaus nicht vor trivialen Fehlern schützt :-]), kann die Fehler des Anfängers meist nicht nachvollziehen. Hier würde ein LOG-File, also ein Protokoll aller interaktiven Versuche auf der Oberfläche des Interpreters, in beiden Richtungen viel Positives bewirken können. Der versierte Forther könnte einfach nachlesen, wo sich der Anfänger 'verrannt' hat. Der Anfänger könnte aus mancher Session des versierten Forthers probate Arbeits- und Vorgehensweisen auslesen.

Ulrich und ich haben uns deshalb, unmittelbar nachdem wir *Rolf's* Ruf in der VD gelesen hatten, zusammengesetzt und diskutiert, wie ein solches 'Protokoll' in 'unser' ZF-Forth zu implementieren wäre. Die grundsätzlichen Überlegungen, die letztlich zu der noch vorzustellenden Version unseres "RECORDERS" geführt haben, wollen wir an dieser Stelle kurz aufzeigen, weil auch das bereits einen Einblick in die Vorgehensweise für einen wenig versierteren Forther bietet. Zudem zeigen die folgenden Lösungsansätze auf, daß selbst ein vergleichsweise triviales Problem wie das angestrebte Protokoll durchaus auf ganz unterschiedliche Weisen gelöst werden kann.

Zunächst wollte ich eine Lösung auf der Basis der in der VD veröffentlichten Serie über die 'UHR' anstreben. Ein TSR sollte im Hintergrund der Dinge harren, die sich da auf der Interpreteroberfläche, sprich: auf dem Monitor, tun. Das wäre eine Lösung voller 'technischer Rafinessen' geworden, die den deutlichen Nachteil hätte, wieder einmal nicht verstanden zu werden. Die Definitionen zum LOG-File sollen aber zuallererst ein Werkzeug für Anfänger sein, auch wenn sich letztlich daraus eine Reihe weiterer Werkzeuge 'basteln' lassen. Ein Werkzeug für Anfänger soll aber nach unserem Verständnis auch in seiner Funktionsweise von diesen verstanden werden können.

Unser nächster Gedanke war, daß sich die Lösung im Interpreter INTERPRET befinden müsse. Schließlich soll das Protokoll alle interaktiven Arbeiten auf der interpretierenden Oberfläche des Systems mitschreiben. Folglich gibt es gar keinen Grund, das 'System zu verlassen' und von einem TSR aus diese Arbeit erledigen zu lassen. `INTERPRET` ist (nicht nur in ZF!) eine `DEFER`ed Definition, die sich jederzeit auf andere Definitionen 'umbiegen' läßt. Eigentlich könnte zur Laufzeit des Protokolls, also wenn die Arbeit auf der Oberfläche mitgeschrieben wird, der Interpreter durch eine spezialisierte und angepaßte Version ersetzt werden. In diese Richtung

Stichworte

ZF
LOG-File
File-Handling
Ausgabeumleitung

Listing zu ZF-Recorder (F. & U. Prinz) `recorder.seq`

```
\ --- RECORDER.SEQ ------------------------------------------------------------
\
COMMENT:

        System   :      ZF-Forth
        Maschine :      80386/33
        Autoren  :      Ulrich Prinz, Friederich Prinz
                        Forthgruppe Moers
        Status   :      Public Domain

      <RECORDER.SEQ> stellt eine LOG-Datei bereit, in die (fast) alle
      interaktiven Eingaben, sowie die Antworten des FORTH-Systems,
      mitgeschnitten werden können.

      Wenn RECORDER.SEQ geladen wurde, schalten die Worte <LOGON> und
      <LOGOFF> den 'Mitschnitt' ein, bzw. aus.
        Während LOGON alle Aktionen mitschreibt und in der Datei
      ZFFORTH.LOG sichert, wird die Anzeige der <Statuszeile> ausdrücklich
      abgeschaltet. Die Statuszeile wird im normalen Betrieb nach jedem
      Druck auf die ENTER-Taste aktualisiert, weil in ihr u.a. die aktuelle
      Stacktiefe angezeigt wird. Dies würde aber dazu führen, daß die
      Statuszeile generell nach jeder Eingabe ebenfalls in der LOG-Datei
      gesichert würde. Soll die Statuszeile trotzdem zwischendurch in der
      LOG-Datei erscheinen, genügt ein Aufruf von <STATON>. (Abschalten mit
      <STATOFF> nicht vergessen !)

         <DEBUG>-Läufe im LOGON empfehlen sich nicht. Während einer DEBUG
      Session wird im oberen Bildschirmbereich ständig der Quelltext des
      zu untersuchenden Wortes angezeigt, weil sich dieser Quelltext vor
      allem bei 'NESTINGs' verändert. Auch diese Aktualisierungen werden
      in der LOG-Datei mitgeschrieben, was zu einem sehr schnellen Anwachsen
      der Datei führt.

      Wenn die LOG-Datei direkt mit dem ZF-Editor weiter bearbeitet werden
      soll, empfiehlt es sich darauf zu achten, daß diese Datei cirka <60
      KByte> nicht überschreitet.

      Der Aufruf des Editors SED schaltet den Mitschnitt ausdrücklich LOGOFF.
```

Der Klingelnberg-Aufnehmer

Eine sehr einfache und zudem universelle Art des LOGs sollte nicht ganz vergessen werden, auch oder gerade weil sie nur in F-PC-ak möglich ist, und zwar ab dem Public Domain Kernel. F-PC-ak erlaubt den Output gleichzeitig auf zwei Handles. Und zwar schaltet `OUT>prn&con` gleichzeitig auf die Console und den Printer. Das Printer-Handle kann natürlich auf einen Filenamen gesetzt werden. Vorteil ist, daß das alles problemlos interaktiv eingegeben werden kann. Je nach Sachlage ist es wünschenswert oder auch kritisch, das alles, wirklich alles (!) mitgeschrieben wird. Beispiel:

```
pfile logfile.log
out>prn&con
.date .time
...
...
.date .time
out>con
pclose
```

oder:

```
...
CON-ALSO? ON
PRINTING ON
...
...
PRINTING OFF
...
```

Das ist ideal, um fortlaufend auch Protokolle von dem zu bekommen, was man am Bildschirm verfolgen möchte/muß. So existieren zur Meta-Compilation eine Liste kernel.lst der für Info und Debugging wichtigen Adress-Label. So und (fast) nur so kann z.B. `>NEXT` über die Hyper-Hilfe gefunden werden.

Der Staben-Aufnehmer

Die Lösung in fünf Schritten:

1. WINDOWS starten...
2. Den Makrorecorder - das ist die kleine Filmkamera aus dem Zubehör - mit den üblichen Einstellungen aufrufen und mit MAKRO/AUFZEICHNEN starten...
3. WinFORTH starten. Fortheln, (englische) Hilfe anfordern, sich vertippen, Programme kompilieren, 0 1000 ERASE tippen...
4. Über ALTAB zum Rekorder wechseln; MAKRO SPEICHERN wählen...
5. Zum Ansehen:
   Mit gedrückter SHIFT-Taste (!!) MAKRO/EIGENSCHAFTEN anwählen - da ist die Ursache für den Systemabsturz!

Fortsetzung des Schröder-Aufnehmers

logfile.seq

```
        Nur die beiden Worte 'LOG-ON' und 'LOG-OFF' sind zum Gebrauch
        durch den Anwender bestimmt:
          'LOG-ON' erzeugt oder öffnet das Log-File, schreibt die
          Startmeldung mit Datum und Uhrzeit hinein und installiert das
          Wort 'LOG-EXPECT' in das "deferred word" EXPECT. Die ersten
          drei Zeilen von 'LOG-ON' prüfen, ob die Protokollfunktion schon
          eingeschaltet ist. Wenn man 'LOG-ON' in die Datei F-PC.CFG ein-
          trägt, wird die LOG-Funktion automatisch bei jeder F-PC-Sitzung
          gestartet.

          'LOG-OFF' schreibt eine Protokollende-Meldung in das Log-File
          und restauriert den ursprünglichen Zustand des Wortes EXPECT.
          Auch 'LOG-OFF' prüft in den ersten Zeilen, ob die Protokoll-
          funktion überhaupt abgeschaltet werden muß. 'LOG-OFF' wird auch
          automatisch mit dem Wort 'BYE' ausgeführt, falls man es vergißt
          einzugeben.

*****************************************************************************

        Hinweise zur Benutzung:
          1) Das Log-File wird fortlaufend geschrieben, d.h. das Protokoll
          der jeweils jüngsten Sitzung wird an das Ende des Files angehängt.
          Wenn das Log-File zu gro wird, sollte man entweder das File
          insgesamt löschen oder mit einem Texteditor die älteren Einträge
          entfernen.
          2) Wer LOGFILE ständig in sein System aufnehmen will, sollte FPC
          unbedingt bei ausgeschalteter Log-Funktion "fsaven", z.B. so:

                FLOAD LOGFILE FSAVE NEWF LOG-ON

          sonst enthält EXPECT bereits beim Systemstart die veränderte
          Funktion.

*****************************************************************************

        Die folgenden zwei Direktiven '\GR' und '\EN' haben zwar eigentlich
        nichts mit der Aufgabe zu tun, erleichtern aber generell das
        Schreiben mehrsprachiger Programme. Die Direktive '\GR' (german)
        wird dabei allen deutschsprachigen Strings vorangestellt und '\EN'
        (english) dementsprechend den englischsprachigen. Ganz einfach
        kann die Programmversion für eine bestimmte Sprache compiliert
        werden, indem die zugehörige Direktive auf 'TRUE' und die andere
        auf 'FALSE' gesetzt wird. Die Voreinstellung setzt '\GR' auf
        'TRUE', so daß die deutsche Version compiliert wird.

*****************************************************************************
comment;

ONLY FORTH DEFINITIONS

TRUE  DIRECTIVE \GR
FALSE DIRECTIVE \EN

ALSO HIDDEN DEFINITIONS

CREATE crlf$    2 c, 13 c, 10 c, \ ein Datei-Zeilenende, CR+LF,
                                 \ als "counted" String

\gr CREATE log-start$ ," \ Protokoll der F-PC Sitzung am "
\en CREATE log-start$ ," \ protocol of the f-pc session on "

\gr CREATE log-end$   ," \ Ende der F-PC Sitzung am "
\en CREATE log-end$   ," \ end of the f-pc session on "

HANDLE log-file                 \ Handlestruktur des 'log-file' reservieren
       log-file !HCB FPCLOG.SEQ                 \ Dateinamen darin eintragen

defer expectorg         \ Platz für Kopie der originalen expect-Funktion
torg  \ expectorg bekommt den Inhalt von EXPECT

: log-expect    ( a1 n1 --- )  \ a1 = Adresse des Eingabepuffers (TIB)
                               \ n1 = maximale Länge der Eingabe
       over >r                                 \ a1 auf Returnstack sichern
       expectorg                               \ originales EXPECT ausführen
```

Fortsetzung auf Seite 34

haben wir dann auch erste Versuche unternommen - die wir aber ebenso schnell eingestellt haben.

Auch das temporäre Austauschen des Interpreters zählt nicht mehr unbedingt zu den Dingen, die wir gerne Anfängern und/oder Quereinsteigern als 'einfache Lösung' zumuten wollen, obwohl dieser Weg nachweisbar funktioniert und dem 'versierten' Forther einige 'technische Feinheiten' entlockt, die einfach Spaß machen.

Der dritte, und letztlich realisierte Ansatz ist der Gedanke, da alle Reaktionen auf dem Bildschirm letztlich über die Worte EMIT und QTYPE vektorisiert werden (In anderen Systemen mag vor allem QTYPE anders heien...). Dabei ist EMIT für die Ausgabe einzelner Zeichen zuständig, und QTYPE regelt die Ausgabe von Zeichenketten. Um eine 'Sitzung' auf der Interpreteroberfläche zu protokollieren, brauchen nur diese beiden Worte zur Laufzeit des Protokolls auf erweiterte Definitionen 'umgebogen' zu werden. Die notwendigen Erweiterungen sind relativ einfach und müssen lediglich einige Definitionen aus dem Moerser Kurs zum Dateihandling beiinhalten. Wie der nachfolgende Quelltext zeigt, wird die eigentliche Arbeit des Protokolls mengenmäig bei Weitem von den Hilfsroutinen übertroffen, die das Öffnen und Schließen der LOG-Datei steuern, oder die LOG-Zeile erzeugen und verwalten.

Nun, LOGON und LOGOFF funktionieren - ZF bietet seinen Freunden ab sofort die Möglichkeit, komplette Sitzungen an der Oberfläche mitzuschreiben. Damit sind zunächst die Anforderungen erfüllt, die *Rolf Kretzschmar* in der VD (s.o) definiert hat. Aus einem solchen Protokoll lät sich aber mit ein wenig Phantasie noch eine Menge mehr herausholen. Zum Beispiel ein regelrechter "Definitionsrecorder" ließe sich hieraus bauen. Vielleicht nehmen die Leser der VD diesen Gedanken auf, oder bringen eigene Ideen dazu. Auf das LOG-File aufsetzende Applikationen könnten in der Zukunft Forth um ein mächtiges Werkzeug bereichern, aus welchem Anfänger und 'Fachleute gleichermaen Nutzen ziehen können.

❑

Fortsetzung des Listings zu ZF-Recorder (F. & U. Prinz)

```
      SED macht selbst intensiven Gebrauch von den Worten EMIT und QTYPE, so
      daß ein paralleles Arbeiten von LOGON und SED zu Problemen führt.
      Deshalb ist SED am Ende der Definitionen selbst auch noch einmal
      'umgebogen' worden.

COMMENT;

\ --- Arrays, Konstanten und Variablen -----------------------------------

        CREATE Log.Zeile        76 ALLOT
                                13 C, 10 C,          \ CRLF anhängen
               Log.Zeile        76 ASCII - FILL      \ mit "-" auffüllen

\              Das 'SystemInfo' kann und SOLL vom jeweiligen Anwender
\              verändert werden. Zum Beispiel der jeweilige Name gibt
\              Aufschluß über den Urheber der mitgeschriebenen Texte.

        CREATE SystemInfo  ," [ ZF-Forth LOG-File ]" \ warum nicht den
               SystemInfo  COUNT                     \ eigenen Namen ???
               Log.Zeile   3 +  SWAP CMOVE           \ Info > Log.Zeile

\              konstante Adresse auf die LOG.Zeile ersparen später
\              unnötige Rechnereien ...

               Log.Zeile  53 +
      CONSTANT Datum                  \ feste Offsets auf die Log.Zeile

               Log.Zeile  65 +
      CONSTANT Zeit

               Log.Zeile  28 +        \ Adresse in der Log.Zeile, ab der
      CONSTANT Log.Status             \ der 'Status' eingetragen wird

               Log.Zeile  76 +        \ ab dieser Adresse steht ein CRLF
      CONSTANT FCR                    \ für Dateien ...

        CREATE geoeffnet   ," (geöffnet   )"    \ das sollte immer in die
        CREATE geschlossen ," (geschlossen)"    \ LOG.Zeile hinein ...

\              zur Arbeit mit der LOG-Datei selbst, sprich zum Dateihandling,
\              empfehlen wir den entsprechenden Kurs der Moerser Forthgruppe
\              aufzuarbeiten...

        CREATE LOG.Name ," ZFFORTH.LOG" \ Name der LOG Datei

        HANDLE ZFFORTH.LOG              \ Handle der Log Datei ...
               LOG.Name ZFFORTH.LOG
               $>HANDLE                 \ ... mit dem Namen auffüllen

\ ------------------------------------------------------------------------

\              Dtol (DatumToLog) und Ztol (ZeitToLog) 'brechen' die von
\              GETDATE und GETTIME gelieferten 32-Bit Werte auf, und
\              konvertieren die Zahlen durch Addition von 48 nach ASCII

: Dtol ( -- )                           \ Datum als ASCII Zeichenkette
    GETDATE                             \ in die Log.Zeile schreiben
    0 256 UM/MOD SWAP
    10 /MOD 48 + Datum      C!
            48 + Datum  1 + C!  ASCII . Datum  2 + C!
    10 /MOD 48 + Datum  3 + C!
            48 + Datum  4 + C!  ASCII . Datum  5 + C!
    10 /MOD 10 /MOD 10 /MOD
            48 + Datum  6 + C!      48 + Datum  7 + C!
            48 + Datum  8 + C!      48 + Datum  9 + C! ;

: Ztol ( -- )                           \ Zeit in ASCII Zeichenkette
    GETTIME                             \ umwandeln
    SWAP 0 256 UM/MOD
    10 /MOD 48 + Zeit       C!
            48 + Zeit  1 + C!  ASCII : Zeit  2 + C!
    10 /MOD 48 + Zeit  3 + C!
            48 + Zeit  4 + C!  ASCII : Zeit  5 + C!
          0 256 UM/MOD
    10 /MOD 48 + Zeit  6 + C!
```

Fortsetzung des Listings zu ZF-Recorder (F. & U. Prinz)

```
            48 + Zeit  7 + C!  DROP ;

\              die eigentlichen Arbeiten werden von (LOG.EMIT) und
\              LOG.TYPE erledigt. Im ZF-Forth System vektorisieren alle
\              Bildschirmausgaben letztlich auf die Definitionen (EMIT)
\              und QTYPE. Wenn LOGON aktiv ist, werden diese beiden
\              Definitionen zeitweise durch die nachfolgenden Definitionen
\              ersetzt.

: (LOG.EMIT) ( -- )
     DUP DUP DUP                          \ keine Steuerzeichen, außer CR & LF
     31 > SWAP 13 = OR                    \ alle Zeichen ab ASCII 32 einschl.
          SWAP 10 = OR IF SP@ 1 ZFFORTH.LOG HWRITE DROP
                       THEN
     PRINTING @ IF DUP PEMIT #OUT DECR THEN (CONSOLE) ;

: LOG.TYPE ( A1 N1 --- )
     ?DUP
     IF  PRINTING @
         IF      (TYPE)
       ELSE    2DUP ZFFORTH.LOG HWRITE DROP      \ TYPE in die LOG Datei
               #OUT @ 2DUP + 79 MIN #OUT !
               #LINE @ 24 MIN DUP #LINE ! VTYPE
       THEN
   ELSE DROP
   THEN    ;

: LOG.Oeffnen ( -- )
    ZFFORTH.LOG HOPEN                     \ LOG Datei öffnen
     0<> IF ZFFORTH.LOG HCREATE DROP      \ Wenn LOG nicht existiert, dann
       THEN                               \ wird die Datei erzeugt.
    ZFFORTH.LOG ENDFILE
    ZFFORTH.LOG MOVEPOINTER               \ 32-Bit Dateizeiger an das Dateiende
    Log.Zeile 78 ZFFORTH.LOG HWRITE       \ Log.Zeile in LogDatei schreiben
    DROP ;

: LOG.Schliessen ( -- )
    Log.Zeile 78 ZFFORTH.LOG HWRITE DROP  \ Log.Zeile in LogDatei schreiben
    ZFFORTH.LOG HCLOSE DROP ;             \ Datei schliessen

\ ---------------------------------------------------------------------------

: LogOn ( -- )
    STATOFF                                \ Statuszeile abschalten
    ['] (LOG.EMIT) IS EMIT                 \ Ausgaben umvektorisieren
    ['] LOG.TYPE IS TYPE                   \           ""
    Dtol Ztol                              \ Datum und Zeit zur Log.Zeile
    geoeffnet COUNT Log.Status SWAP CMOVE  \ Status geöffnet
    LOG.Oeffnen ;                          \ LOG-Datei öffnen

: LogOff ( -- )
    FCR 2 ZFFORTH.LOG HWRITE DROP          \ CRLF in die LogDatei schreiben
    ['] (EMIT) IS EMIT                     \ Ausgabendefinitionen wieder
    ['] QTYPE  IS TYPE                     \ restaurieren
    Ztol                                   \ Zeit neu auslesen
    geschlossen COUNT Log.Status SWAP CMOVE \ Status 'geschlossen'
    LOG.Schliessen                         \ LOG-Datei schließen
    STATON ;                               \ Statuszeile wieder einschalten

\              Der Editor kann nicht aufgerufen werden, wenn sich das
\              System im LOGON befindet. Der nachfolgende 'Patch' schaltet
\              deshalb beim Aufruf von SED sofort wieder in den 'normalen'
\              Betriebsmodus.

DEFER  Editor                              \ Ein Dummy für den Editor wird
       ' SED IS Editor                     \ mit dessen CFA belegt

WARNING OFF
: SED ( -- )                               \ Redefinition von SED ...
       LOGOFF                              \ Mitschreiben ausschalten,
       Editor ;                            \ und den Editor wie gewohnt
WARNING ON                                 \ aufrufen.
```

Transputer gefällig?

Fred Behringer berichtete uns in einem Brief folgendes über seine Arbeit am Transputer-Forth:

"Mein Transputer-FORTH-System macht Fortschritte: DBBUG, SEE (Discompiler), HELP (On-Line-Hilfe), DIS (Disassembler, der auch in SEE für Codedefinitionen wirkt und Konstruktionen wie ASM[...code...]FORTH verarbeitet), ASSEMBLER, DUMP und alle anderen fürs Austesten so wichtigen Hilfen funktionieren schon lange bestens. Momentan ist der Gleitkommateil dran.

Mit ." wort1 wort2 wort3 ... " $EXECUTE kann ich beliebig vom Transputer aus im Host-FORTH-System herumwirtschaften. Mit SAVE-SYSTEM wird das Transputer-FORTH-System als Datei abgespeichert, mit LOAD-FTP wird es vom Host aus wieder in den Transputerraum geladen. Mit adl ad2 SAVE name wird ein beliebig langer Bereich im Transputer-RAM als Datei abgespeichert, mit ad len LOAD name wird eine Datei vom Laufwerk ins Transputer-RAM nach ad geladen. Natürlich alles 32 Bit breit. Der T800 kann's ja. Dazu brauche ich kein LINUX und kein OS/2.

Zu alldem brauche ich natürlich einen SERVER (vom Hostsystem aus), einen Assembler (vom Hostsystem aus) und einen Metacompiler (vom Hostsystem aus). Diese Dinge habe ich schon vor drei Jahren entwickelt. Sie funktionieren seither bestens. Das Transputer-FORTH-System wird entwicklenderweise laufend neu metacompiliert. Am Metacompiler selbst habe ich schon seit etwa einem Jahr nichts mehr zu verbessern. Irgendwann in einer späteren Version werde ich dann den Metacompiler gleich für das Transputer-FORTH-System schreiben."

Interessenten an Transputern wenden sich an:

Fred Behringer,
Planegger Str. 24, 81241 München

Bücher-Kiste

Arndt Klingelnberg verfügt noch über Restbestände des (fast) vergriffenen DDR Buches von *Jochen Bonitz*: "Der 16-Bit Mikroprozessor des ESER-PC". Das Buch vermittelt ohne unnötigen Balast genau die Informationen, die sinnvoll sind, um sich bei einem 'ibm' PC z.B. mit Hilfe des F-PC in die Welt der schnellen CODE Worte hineinzuarbeiten. Mit DM29.80 ist das Buch sehr günstig. Nur noch wenige Exemplare des mehrfach erwähnten Forth Buches unseres FG-Mitgliedes *Gert-Ulrich Vack* (Programmieren mit FORTH, kurz: der Vack) können zum Preis von DM42.-- bei *Arndt Klingelnberg* bestellt werden. (Der Preis, der in der ELRAD 8/93 angegeben wurde (DM15,--), ist nach Mitteilung durch den Verlag falsch!)

Krieg der Kerne

von Jörg Plewe
Haarzopfer Straße 32, 45472 Mühlheim, Tel.: 0208 - 497068

In der Zeitschrift SCIENTIFIC AMERICAN erschien 1984 ein Artikel mit dem Titel 'Corewars' von *A.K. Dewdney*. Damals als Spiel gedacht und betrieben, sind die Kerne die Urväter aller Viren.

Kürzlich stieß ich beim Durchblättern der Zeitschrift 'Spektrum der Wissenschaft - Computer-Kurzweil II' auf den Artikel 'Krieg der Kerne'. Diese Zeitschrift ist ein Zusammenfassung von Artikeln über die Computerei, die aus dem amerikanischen Magazin 'SCIENTIFIC AMERICAN' stammen. Obwohl sich irgendwo in meiner organischen Datenbank eine Erinnerung regte, las ich den Artikel mit viel Interesse. Nachdem ich mit einigen Leuten darüber gesprochen hatte, stellte ich fest, daß es vielen so ging wie mir: man hatte einmal davon gehört oder darüber gelesen. So richtig 'warum und wozu und wie genau' wußte aber niemand. Hier noch etwas Aufklärung zu betreiben ist ein Ziel dieses Artikels. Die anderen Ziele sind der Ausdruck meiner Betrübtheit, nicht mit einem professionellen Computer, sondern nur mit einem Spiele-Rechner arbeiten zu können und die Beschreibung der sich daraus ergebenden Schwierigkeiten bei der Portierung von Forth-Quelltexten.

Corewars

Über den Ursprung dieses Krieges herrscht Unklarheit. Es existieren hierzu verschiedenste Anekdoten. Es wird erzählt, daß in frühen Computertagen ein durchgegangenes Assemblerprogramm, daß sich selbst modifizierte und verbreitete, dadurch wieder eingefangen wurde, daß man ein zweites hinterherschickte, um den Irrläufer zu zerstören. Wahr oder nicht wahr - diese kurze Geschichte gibt schon sehr treffend den faszinierenden Gedanken dieses Spieles wieder: zwei kleine Programme streifen durch die dunklen Weiten des Computerspeichers und versuchen, sich gegenseitig den Garaus zu machen. Wie für jeden anderen Krieg, gibt es auch hier so etwas wie die Genfer Konventionen - strenge Regeln der Kriegsführung. Zunächst einmal ist das Kriegsgebiet eingegrenzt. Die Gegner bewegen sich ausschließlich in dem sogenannten Core, einem Speicherbereich festgelegter Größe. Damit keines der Progamme daraus entkommen kann, sind die beiden Enden des Speicherbereiches miteinander Verbunden. Die Gegner können sich nun beliebig weit und beliebig schnell bewegen, ohne wirklich besonders weit zu kommen. Eben so wie der berühmte Hamster in dem berühmten Laufrad. Zum Zweiten sind die Waffen gegeben. Das bedeutet, daß die Kämpfer in einer einheitlichen Sprache geschrieben sein müssen. Um hier eine gewisse Überschaubarkeit zu wahren und da das Spiel immerhin aus der Computersteinzeit von 1984 stammt, wurde hierfür keine objektorientierte, logikgestützte 3D-Multimediasprache, sondern ein sehr einfacher Assemblerdialekt gewählt: der REDCODE. REDCODE enthält in seiner ursprünglichen Fassung nur neun Befehle:

`MOV`	Übertrage
`ADD`	Addiere
`SUB`	Subtrahiere
`JMP`	Springe
`JMZ`	Springe, wenn Null
`JMG`	Springe, wenn größer
`DJZ`	Vermindere; Springe, wenn Null
`CMP`	Vergleiche
`DAT`	Data-Anweisung

Später sind noch weitere Anweisungen hinzugekommen, mit denen der Programmlauf aufgeteilt oder bestimmte Speicherzellen mit Schreibschutz belegt werden können. Diese sollen hier aber zunächst außer Betracht bleiben. Es gibt drei verschiedene Adressierungsarten: unmittelbar, direkt und indirekt. Diese Adressierungsarten möchte ich hier nicht erklären, da es sie zum Einen in jeder Assemblersprache gibt und sie zum anderen in den Beispielen später intuitiv deutlich werden. Alle Adressen im REDCODE sind relativ zum aktuellen Programmzeiger. Ein Programm hat also insbesondere keine Möglichkeit, seine absolute Position im Speicher festzustellen. Mit dieser Sprache werden nun zwei Programme verfaßt, die an eine zufällige Position im Core geladen werden. Es ist selbstverständlich, daß sie sich dabei nicht überlappen dürfen. Anschließend wird abwechselnd von jedem Programm eine Instruktion ausgeführt. Diese Aufgabe übernimmt der 'Memory Array Redcode Simulator', kurz MARS. Er läßt die Programme 'laufen'. Dies geht nun so lange, bis es einem der beiden Programme gelingt, dem jeweils anderen einen nicht ausführbaren Code vor die Füße zu legen. Das kann zum Beispiel eine Null sein. In diesem Fall bricht MARS sofort ab und benennt den Sieger. Gelingt es keinem Programm, seinen Gegner um die Ecke zu bringen, bricht MARS nach einer festgelegten Zahl von Instruktionen ab und wertet die Partie unentschieden.

Kampfprogramme

Wie sehen nun solche Kämpfer aus? Es haben sich verschiedene Standardprogramme gebildet, gegen

Stichworte
Corewars
LMI-Forth
F-PC
F68K
Portierung
Viren

die sich neue Kämpfer erst einmal durchsetzen müssen, bevor man sie in einen Wettkampf schicken kann. Hierbei gibt es kleine, aggressive Programme, die unintelligent aber höllisch gefährlich sind, und klügere, die lieber nachgeben. Es gibt Programme, die sich selbst reparieren können und solche, die zu Ihrem Schutz Wachen aufstellen. Der Phantasie sind wie immer keine Grenzen gesetzt. Ein ebenso kleines wie gefährliches Programm ist DWARF:

```
DAT  -1
ADD  #5,-1
MOV  #0,@-2
JMP  -2
```

Die erste Zeile definiert eine Variable, die mit dem Wert -1 initialisiert wird. Die zweite Zeile addiert den Wert 5 (Adressierungsart: unmittelbar) zur Speicherzelle mit der Adresse -1 (Adressierungsart: direkt). Steht der Programmzähler in dieser zweiten Zeile, ist die Adresse -1 genau die gerade mit DAT definierte Variable. Diese enthält nun also den Wert 4. In der dritten Zeile wird der Wert 0 an die Adresse geschrieben, die in der Speicherzelle mit der Adresse -2, das ist wieder die Variable, steht. Diese Adressierungsart ist indirekt. Das Programm schreibt jetzt also eine Null an die Adresse 4. Diese 4 zählt aber relativ zur DAT-Zelle! Die Null landet damit unmittelbar hinter dem Programm selbst. Anschließend erfolgt der Rücksprung in die ADD-Zeile. Dieses einfache Programm schreibt also in Abständen von fünf Zellen Nullen in den Speicher. Wenn die Größe des Core durch 5 teilbar ist, wird es sich selbst auch niemals selbst treffen, da es selbst nur vier Zellen belegt! Ein weiteres, noch einfacheres Programm ist IMP. IMP ist mit DWARF fast nicht zu erlegen.

```
MOV  0,1
```

Das Programm kopiert seinen einzigen Opcode unmittelbar vor sich selbst und führt sich damit im nächsten Schritt wieder selbst aus. IMP bewegt sich also durch den Speicher. Dies zusammen mit seiner Minimallänge machen IMP fast unverwundbar. Allerdings kann IMP niemals wirklich aus eigener Kraft gewinnen! Wenn IMP auf seinen Gegner aufläuft, patcht er diesen ebenfalls zu einem IMP. Danach laufen eben zwei IMPs parallel durch den Speicher bis zum Erreichen der maximalen Schrittzahl. Dann erkennt MARS auf unentschieden! IMP ist für einen Sieg also auf Selbstmörder angewiesen. Diese zwei Beispiele sollen hier erst einmal genügen.

Katzenjammer

Nach dem Studium des Artikels kam ich zu dem Schluß, daß es eine nette Beschäftigung für einen Sonntagnachmittag sein muß, einen REDCODE-Assembler und einen MARS in Forth zu implementieren. Ich hatte mir auch schon nebenbei so einiges überlegt, wie man sowas anlegen könnte und wie später die Darstellung sein wird. Adrenalin floß reichlich und der Elan schoß über alle Grenzen. Leider habe ich meinem Freund *Jörg Staben* bei einem unserer häufigen Treffen davon erzählt. Dieser nun ist im Gegensatz zu mir einer dieser überglücklichen PC-Besitzer. Er sagte: "Schön, schön. Aber warum nimmst Du nicht das hier?" und zerrte mit dem Commander ein File seiner Wunderplatte hervor. Es stammte von einer 'User Contribution Disk' der FIG und trug den unheilvollen Namen `COREWARS.SCR`. Mein Elan brach schlagartig zusammen: wieder einmal war ich zu spät gekommen und vom heldenhaften Implementator zum schnöden Portierer degradiert. Ein Blick in den Screen-Editor offenbarte, daß es wirklich eine Corewars-Implementation für LMI-Forth war. In diesem Augenblick war auch *Wolfgang Allinger* mit einem Tragbaren mit PC-Forth zur Stelle. `'1 LOAD'` und der Krieg der Kerne konnte schon beginnen. Sogar sechs edle Kämpen fanden sich im Quelltext. Ich überlegte, wie ich wohl den Sonntagnachmittag verbringen würde. Als Herkunft dieses Programms ist in Screen 0 die Firma LMI angegeben. Der Autor ist nicht namentlich bestimmt, so daß man wohl ruhig einmal auf *Ray Duncan* tippen darf. Jedenfalls kopierte ich es auf eine Diskette und zog von dannen.

Portierung

Natürlich wollte ich dieses Programm auch auf meinem eigenen Rechner (Atari TT) unter meinem Forth-System F68K laufen sehen. Als 'Migrationspfad' wählte ich LMI - F-PC - F68K. An meinem Arbeitsplatz portierte ich das System ins F-PC; selbstverständlich in der Mittagspause! Hierbei gab es nur einen einzigen Fallstrick, über den F-PC-Profis wahrscheinlich nicht stolpern würden: im Original wurde eine Sprungtabelle mit Hilfe der eckigen Klammern angelegt:

```
[ MOV ADD SUB JMP JMZ
JMG DJZ CMP DAT ]
```

Dies sah vernüftig aus und wurde zunächst einmal intuitiv beibehalten. Aufgrund des etwas wirren Speichermodells von F-PC geht es aber so nicht! Der Drei-Tasten-Griff ist gefragt. Ich habe es dann einfach mit ' und , erledigt und mich dann später belehren lassen, da es dazu im F-PC ein Spezialkonstruktion gibt:

```
EXEC: MOV ADD SUB JMP
JMZ JMG DJZ CMP DAT ;
```

Im Wesentlichen wurde bei der Portierung ins F-PC aber nur die Bildschirmausgabe vereinfacht, so daß sie auf einem normalen ASCII-Terminal funktioniert und keine DOS-Spezialitäten mehr benötigt.

F68K

Die Portierung auf meine Maschine versprach etwas mehr Ärger. Immerhin war der Übergang von einem 16-Bit- auf ein 32-Bit-System zu schaffen. Außerdem suchten die 5-bytigen REDOPCODES (oder OPREDCODES? Auf jeden Fall 1 Byte Opcode und jeweils 2 Byte für Operanden) auf einer Maschine, die ziemlich ungehalten auf ungerade Adressen reagiert, ganz offensichlich Streit. Es wurden zunächst 16-Bit-Speicheroperatoren für ungerade Adressen geschaffen. Diese wurden schlicht mit dem Suffix 'cw' für CoreWars versehen. Das ist einfach. Die neuen Operatoren mußten dann an all den Stellen eingesetzt werden, an denen entweder REDCODE erzeugt oder in den Core zugegriffen wird. Das Auffinden dieser Stel-

len war das eigentliche Portierungshindernis, da in einem 16-Bit-System der Zugriff auf einem REDCODE-Operanden ZUFÄLLIGERWEISE mit dem gleichen Operator durchgeführt werden kann wie der Zugriff auf eine gewöhnliche Variable. 1984 war wahrscheinlich die Portierung auf andersbittige Maschinen nicht unbedingt ein Hauptaugenmerk! Es wäre sicher besser gewesen, Corezugriffe auszufaktorisieren und auch die Umrechnung relative Coreadresse -- physikalische RAM-Adresse in diesen Zugriffen implizit durchzuführen, anstatt an einigen Stellen explizit zu wandeln. Portierbarkeit wäre damit sofort gewährleistet gewesen, da nur die Zugriffe anzupassen gewesen wären. Der Autor hat es anders gemacht, ich habe es nicht geändert. Eine weitere Schwierigkeit war die Tatsache, daß F68K aufgrund seiner strengen Trennung von Code und Daten keine einfache Möglichkeit kennt, von der Adresse einer Variablen zu deren Header- oder Codeadresse zurückzufinden (es gibt also kein BODY). Deshalb mußte das Programm noch so umgestellt werden, daß es anstelle der Adressen, die von den mit CREATE erzeugten REDCODE-Programmen geliefert werden, mit deren '-Adressen zurechtkommt. Dies allerdings war sehr einfach.Statt

```
GEMINI MORTAR GO
```

im LMI oder F-PC heißt es nun im F68K

```
' GEMINI ' MORTAR DOWAR
```

oder einfacher

```
DOWAR: GEMINI MORTAR
```

Trotz all dem war die Portierung innerhalb eines Nachmittages abgeschlossen, was einerseits für die Sprache Forth und andererseits auch für das System F68K spricht.

Was könnte man noch machen?

Wie immer wäre es natürlich auch bei diesem Spiel schön, den Spielverlauf grafisch verfolgen zu können. Um die Sache nicht unnötig kompliziert und unportabel zu machen, sollte man vielleicht auf einfache Zeichengrafik zurückgreifen. Dabei wird das Spielfeld auf den Bildschirm abgebildet. Jede Zelle entspricht einem Zeichen. Bei einer Coregröße von 1000 Zellen könnte man dazu ein Rechteck von 40x25 Zeichen auf dem Bildschirm ausersehen. Nun gibt es für jedes Programm zwei Informationen, die darstellungswürdig sind: sein Programmzähler und die Lage der Zellen, auf die schreibend zugegriffen wird. Mein Vorschlag hierzu: für das eine Programm verwende man große und kleine 'x', während das andere sich der 'o's bedient. Das sollte ein hübsches Bild à la Gobang ergeben. Eine weitere Modifikation betrifft den Befehlssatz. Dieser ist nämlich inzwischen erweitert worden. Die fehlenden Befehle `PCT` (Protect) und `SPL` (Split) wären da z.B. noch nachzutragen. Auerdem existieren inzwischen erweiterte Adressierungsarten, die z.B. ein postdecrement erlauben. Für eine wirklich vollständige Implementation von REDCODE wird man wohl um den 'Core War Standard' nicht herumkommen. Wo man den bekommt, weiß ich aber auch nicht. Weiterhin hat man die Möglichkeit, den 'Core War Newsletter' zu abonnieren oder der 'Internationalen Gesellschaft für den Krieg der Kerne' beizutreten. Also auch für den typisch deutschen Vereinsmeier ist gesorgt.

Der Mühe Lohn

Um nun auch noch einmal zu zeigen, was man für den Portierungsaufwand nun auch als 68000-Benutzer zu sehen bekommt, sei hier einmal eine vollständige Schlacht zwischen den Kampfprogrammen GEMINI und MORTAR dargestellt. GEMINI besteht aus einer Schleife, die das Programm 100 Zellen nach vorne kopiert und dann dorthin verzweigt. MORTAR ähnelt dem oben bereits vorgestellten DWARF, nur daß es seine Nullen in einer Folge von Fibonacci-Zahlen in den Speicher schreibt. (Abbildung)

MORTAR bezwingt GEMINI übrigens regelmäßig! Auch BIGFOOT, der GEMINI sehr ähnlich ist, hat gegen MORTAR kaum eine Chance. IMP und NOWHERE bezwingen trotz ihrer scheinbaren Passivität MORTAR dagegen fast immer, da dieser sich nach 1145 Schritten selbst erlegt. Aber auch der einfachere DWARF ist gegen den raffinierten MORTAR nicht selten erfolgreich. Der absolute Sieger steht also noch gar nicht fest. Und wer sich auch gerne theoretisch mit solchen Problemen auseinandersetzen möchte, der fühle sich durch folgendes Zitat aus dem erwähnten Artikel motiviert: "Goetsch, Mauldin und Milazzo haben MORTAR analysiert und festgestellt, da ein Programm mit mehr als zehn Anweisungen selbstheilend sein muß, um MORTAR zu schlagen. Andererseits kann sich kein Programm mit mehr als 141 Anweisungen schnell genug reparieren, um einen Angriff von MORTAR heil zu überstehen."

Alles klar??

❑

```
dowar: gemini mortar
COREWARS for F68K
(c) 1986 by Laboratory Microsystems, 1992 JPS
Size of CORE: 1000 locations
Maximum number of instructions to be executed:
2000
GEMINI loaded at location 23
MORTAR loaded at location  0
 0   23: DAT    0              0: MOV #  0, @  7
 1   24: DAT   99              1: MOV    5,  4
 2   25: MOV @ -2, @  -1       2: MOV    5,  4
 3   26: CMP   -3, #   9       3: ADD    2,  4
 4   28: ADD #  1,    -5       4: JMP   -4
 5   29: ADD #  1,    -5       0: MOV #  0, @  7
 6   30: JMP   -5              1: MOV    5,  4
 7   25: MOV @ -2, @  -1       2: MOV    5,  4
 8   26: CMP   -3, #   9       3: ADD    2,  4
 9   28: ADD #  1,    -5       4: JMP   -4
10   29: ADD #  1,    -5       0: MOV #  0, @  7
11   30: JMP   -5              1: MOV    5,  4
12   25: MOV @ -2, @  -1       2: MOV    5,  4
13   26: CMP   -3, #   9       3: ADD    2,  4
14   28: ADD #  1,    -5       4: JMP   -4
15   29: ADD #  1,    -5       0: MOV #  0, @  7
16   30: JMP   -5              1: MOV    5,  4
17   25: MOV @ -2, @  -1       2: MOV    5,  4
18   26: CMP   -3, #   9       3: ADD    2,  4
19   28: ADD #  1,    -5       4: JMP   -4
20   29: ADD #  1,    -5       0: MOV #  0, @  7
21   30: JMP   -5              1: MOV    5,  4
22   25: MOV @ -2, @  -1       2: MOV    5,  4
23   26: CMP   -3, #   9       3: ADD    2,  4
24   28: ADD #  1,    -5       4: JMP   -4
25   29: ADD #  1,    -5       0: MOV #  0, @  7
26   30: JMP   -5              1: MOV    5,  4
27   25: MOV @ -2, @  -1       2: MOV    5,  4
28   26: CMP   -3, #   9       3: ADD    2,  4
29   28: ADD #  1,    -5       4: JMP   -4
30   29: ADD #  1,    -5       0: MOV #  0, @  7
31   30: JMP   -5              1: MOV    5,  4
32   25: MOV @ -2, @  -1       2: MOV    5,  4
33   26: CMP   -3, #   9       3: ADD    2,  4
34   28: DAT #  0
Result:  MORTAR wins! ok                        ❑
```

ONLY Forth, ALSO andere!

Fast eine Buchbesprechung

von Wolfgang Schemmert
Strahlenberger Str.123, 63067 Offenbach Tel 069-800 12 08, Fax 069-82 59 57

Nach der Lektüre dieses Artikels werden Sie Aspekte des Forth aus der Sicht der Informatik besser einordnen können. Zudem werden Sie auf drei interessante Bücher aufmerksam gemacht.

Mensch soll nicht zuviel lesen, sonst kommt mensch nur auf dumme Gedanken. Ich jedenfalls habe mich von einigen Büchern aus der großen weiten Welt des Compilerbaus verführen lassen, die explizit auf Postfix-Notation und Forth-ähnliche Stackmaschinen eingehen. (Das Wort "Forth" kommt übrigens in keinem der Bücher vor.) Um es gleich vorweg zu nehmen: von meiner ursprünglichen Hoffnung, in diesen Büchern Tricks zu finden, um Forth "besser" oder wenigstens "anders" implementieren zu können, ist nichts übrig geblieben. Trotzdem, für mich als Nicht-Informatiker war die Lektüre sehr aufschlußreich, um Forth im Spektrum der Programmiersprachen besser wiederzufinden.

R. Mak stellt ohne tiefere theoretische Erörterungen nach und nach modulweise den kompletten Source-Code für einen Pascal-Interpreter vor; dazu den Source-Level-Debugger und Compiler (alles in C geschrieben). Ich habe nicht nachgeprüft, ob das Programm tatsächlich funktionsfähig ist. (Der Sourcecode ist an die 100 Druckseiten lang. In den USA soll es irgendwo für 25$ eine Diskette geben.) Auf jeden Fall ist es eine der seltenen Möglichkeiten, so ein Teil mal von innen zu betrachten; dazu noch in einer fast didaktisch aufgemachten Form. Was den strukturellen Aufbau betrifft, war ich überrascht von der Ähnlichkeit des Pascal-Interpreters mit einem Forth System. Hauptunterschied: Im "äußeren Pascal-Interpreter" herrscht deutlich mehr "Gewurschtel" als im Forth System, eben weil Forth speziell auf die einheitliche Behandlung aller Sprachelemente hin gezüchtet ist, Pascal hingegen für jeden Daten-, Funktions- und Strukturierungstyp eine spezielle Behandlungsroutine braucht. Andere Eigenschaften, die Forther manchmal Forth exklusiv zuschreiben, wie Interaktivität, Erweiterbarkeit, inkrementelle Kompilation, würden zwar noch etliche Verbesserungen am Code von *Mak* erfordern, wären aber ohne grundsätzliche Schwierigkeiten auch in Pascal (ebenso C, C++, Modula2) realisierbar. (*Das wollen wir aber nicht weitersagen, sonst macht das noch einer und wir brauchen Forth nicht mehr...rk*)

Die beiden nächsten Titel sind eher theoretisch orientierte Lehrbücher für Informatik-Studenten. In der industriellen Software-Produktion sind handgeschriebene Compiler überholt. Man beschreibe die Syntax der zu entwickelnden Programmiersprache in der allgemein als Meta-Standard anerkannten "Backus-Naur-Form", speise dies ein in einen Compiler-Generator und lasse sich den gewünschten Compiler berechnen. Die kreative Arbeit liegt eher in der sinnvollen Gestaltung neuer Programmiersprachen. Dies Verfahren hat Ähnlichkeit mit Forth-Metacompilation, allerdings mit dem wesentlichen Unterschied, daß beim Metacompiler Forth in seiner eigenen Terminologie beschrieben wird. Damit die zukünftigen Informatiker später beim Füttern der Compiler-Compiler wenigstens andeutungsweise verstehen, was sie da meta-tun, sind in den Lehrbüchern auch weniger up-to-date Verfahren zur manuellen Konstruktion von Compilern und Interpretern beschrieben und teilweise als Übungsprojekte ausgeführt. Leider sind beide Bücher gegenüber konsumorientierten Lesern recht geizig. Viele Fragen, die er gerne mit dem Erwerb des Buches gelöst hätte, werden ihm als Übungsaufgaben aufgebrummt.

Ronald Mak,
Writing Compilers & Interpreters, an applied approach
Wiley, 1991

Alfred V.Aho, Ravi Sethi, Jeffrey D.Ullman,
Compilerbau
(2 Bände, in deutscher Übersetzung)
Addison-Wesley (D) 1992

Thomas Pittman, James Peters,
The Art of Compiler Design
Prentice-Hall International Editions, 1992

Kaum anders als beim `QUERY` und `WORD` eines Forth-Systems wird beim Pascal Compiler der Sourcecode zunächst einer lexikalischen Analyse unterworfen, d.h. in Token zerlegt. Neue Token werden in die Symboltabelle eingetragen. Das Konzept der Symboltabelle rüttelt ein wenig am Mythos des Forth Dictionary: Name des Bezeichners als String; Beschreibung der Attribute: bei Forth i.w. `Immediate` und `Restrict`, bei Pascal vor allem Deklaration von Datentypen, Priorität von Operatoren (also alles an Information, das nötig

Stichworte
Compiler-Bau
Pascal, C
Metacompilation
Stackmaschine
Parser

ist, um bei Auffinden des Symbols in der Symboltabelle korrekt durch das Gewurschtel des Pascal-'Textinterpreters' zu navigieren); Verweis auf semantische Aktion (Eintrag des vom Symbol zur Runtime auszuführenden Parameter oder Code); Verwaltung der Symboltabelle (Link, meistens als binärer Baum organisiert).

Im nächsten Schritt erledigt sowohl bei Forth als auch bei den anderen Programmiersprachen der Parser die Syntax-Analyse. Das ist bei Pascal vor allem eine hierarchische Analyse, d.h. die verschachtelten Statements müssen in verdaubare Häppchen vereinfachter logischer Struktur zerlegt werden, aus denen der Compiler später Maschinenanweisungen konstruieren kann. Eine dabei häufig angewandte Strategie ist die Umformung der Pascal Syntax in Postfix Notation, die dann z.B. im oben beschriebenen Pascal Interpreter von einer virtuellen Stackmaschine ausgeführt wird. (Eine andere ebenfalls beliebte Strategie zerlegt den Sourcecode in "Tripel", d.h. vereinfacht ausgedrückt werden Ketten von Zwischenergebnissen aus jeweils zwei Operanden und einem Operator abgespalten.) Eine solche Zwischenform des Programmcodes wird von vielen Pascal-Compilern explizit erzeugt, weil sie gute Angriffspunkte für Optimierer bietet.

Der Parser stellt sich als interessantester Reibungspunkt bei der Unterscheidung zwischen Forth und anderen Programmiersprachen heraus. Schließlich bestimmt die Syntax die formale Struktur einer Sprache und damit auch ihre operativen Ausdrucks-Möglichkeiten. Pascal, C und ähnliche Programmiersprachen weisen einen höheren syntaktischen Organisationsgrad auf als Forth. Andererseits sind sie hinreichend einfach konstruiert, daß sie den Regeln einer sogenannten LL(1)-Grammatik genügen. Das heißt, der Tokenstrom wird linear von links nach rechts bearbeitet, wobei immer vor dem aktuell bearbeiteten Token ein Token prädiktiv, als Ablaufsteuerung vorweg eingelesen wird. Damit kann die Sprache deterministisch (d.h. logisch eindeutig ohne "Trial and Error") in allen Verschachtelungen, Klammern, Prioritäten usw. aufgelöst werden. Praktisch erreicht wird das mit dem Verfahren des "rekursiven Abstiegs": Deutet das vorweg gelesene Token eine Operation höherer Priorität an - etwa * oder (- dann werden die bisherigen Zwischenergebnisse auf dem Stack gemerkt und zunächst das Konstrukt höherer Prioriät nach dem gleichen Verfahren geknackt. Genau dies ist auch ein Algorithmus zur Umwandlung beliebiger "Sätze" von Pascal Syntax in Postfix Notation. (LL(1)-Grammatik und rekursiver Abstieg ist die einfachste Methode zum Parsen von Pascal und C, deswegen auch beliebtes Lehrbuchbeispiel. Maschinelle Parser-Generatoren benutzen komplexere Verfahren, damit sie für möglichst allgemeine Sprachkonstrukte einsetzbar sind.)

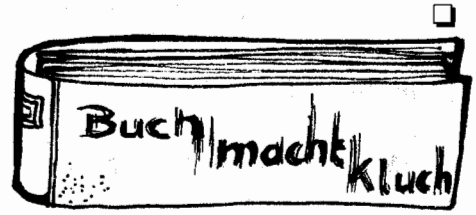
Der Parser stellt sich als interessantester Reibungspunkt bei der Unterscheidung zwischen Forth und anderen Programmiersprachen heraus.

Der Parser eines Forth-Systems genügt dagegen einer LL(0)-Grammatik. Die Token werden ebenfalls von links nach rechts bearbeitet, aber ohne Prädiktion. Allgemeiner: die Token tauschen während das Parsens keine Information miteinander aus. Das bestimmt wesentliche Eigenarten von Forth. Dabei macht Forth folgende Ausnahme: Worte, die einen (-- ; <string>) im Stackdiagramm haben, setzen irgendwie das Wort TOKEN in Aktion, das dann präemptiv tätig wird. Konkret ist das der Fall bei: ." (: CONSTANT VARIABLE VOCABULARY TASK .(ABORT". Die Genialität von Forth besteht in der Einfachheit und Eleganz, praktisch keine Syntax außer einer linearen Reihung zu benötigen. Der Code des äußeren Interpreters gestaltet sich dadurch extrem kompakt. Die interaktive Arbeit mit Forth ist schneller und komfortabler, weil die Deklaration von Datentypen und Parametrisierung von Funktionen bereits in den aufgerufenen Symbolen=Worten enthalten ist - spart Tipperei am Terminal. Dafür zahlen Forther ihren Preis: Einige Arbeit des Pascal Parsers müssen Forth Programmierer selber im Kopf leisten. Der hermetische, unflexibel vom syntaktischen Konzept diktierte Sprachstil herrscht über die Reihenfolge der ausgesagten Worte. Das erschwert in vielen Fällen "human engineering". (Damit meine ich, daß Mensch beim Benutzen von Maschinen ein Recht auf historisch gewachsene oder einfach in der Gesellschaft verbreitete Denkschemata hat, auch wenn diese vom Standpunkt der Maschinenlogik weniger genial sind. Die kritische Auseinandersetzung damit ist ein anderes Bier. Sie sollte aber nicht künstlich vor die Erledigung von Alltagsproblemen gestellt werden, sonst könnte ein Haß auf das Bessere dabei rauskommen.) Aus sprachlicher Kargheit entstehen Paradoxien: Ausgerechnet beim konstituierenden Operator des Forth-Denkens, dem COLON, schreibt Forth zwingend Prefix-Schreibweise vor. Versuche, diese Probleme mit den Mitteln von Forth zu lösen, führen hintenrum Prädiktion und rekursiven Abstieg ein (siehe Artikel von *B.Paysan* in 4D, Heft 4/91).

Neben dem aus Tokenizer und Parser bestehenden Front-End benötigt der Pascal-Compiler ein Back-End, d.h. Generierung des Maschinenprogramms aus dem Zwischencode. Der Forth-Compiler endet bei der Erzeugung des token-gefädelten Codes.

Wo dieser die Form einer als Maschinensprache kompilierten Subroutinen- Fädelung annimmt, ist mit Forth eine Laufzeit-Effizienz zu erwarten, wie bei einem Pascal- oder C-Programm mit sehr starker Fragmentarisierung des Codes, d.h. vielen aus wenigen Zeilen bestehenden Funktionen.

❑



Fuzzy und Forth

von Dr. Birgit Steffenhagen
Ingenieurbüro Steffenhagen, Fasanenweg 12, 18184 Roggentin

In diesem Beitrag bietet Ihnen die Autorin neben einem kleinen Exkurs in die Fuzzy-Theorie einige Information zum KI-Paket des comForth.

Zunächst möchte ich eine kurze Definition bzw. Beschreibung des Begriffs Fuzzy-Theorie, der inzwischen zu einem Modewort geworden ist, liefern, ohne dabei gleich zu sehr ins Theoretische abzugleiten.

Die Fuzzy-Theorie (auch unscharfe Logik genannt) ist ein Teil der Systemtheorie, die eine Beschreibungsweise nutzt, die weder streng deterministisch noch stochastisch ist. Sie orientiert sich an der Beschreibung von Erscheinungen mit Hilfe der Umgangs- oder Fachsprache und dem Problemlösen des Menschen. Während die traditionelle Logik Ausdrükken oder Aussagen genau einen von zwei Wahrheitswerten zuordnet, werden in der unscharfen Logik mehr Werte verwendet (etwa alle Werte zwischen 0 und 1), um Ausdrücke semantisch interpretieren zu können, die weder eindeutig wahr noch eindeutig falsch sind. Die Fuzzy-Logik kann genutzt werden, um sowohl ungenaue bzw. vage Informationen als auch unscharfes Wissen, das z. B. in Form von unscharfen Regeln vorliegt, zu beschreiben.

Die Fuzzy-Theorie kann auf verschiedenen Gebieten eingesetzt werden. Einige Einsatzfelder sollen im folgenden genannt werden:

- Lösung regelungstechnischer Probleme auf vielen Gebieten, vor allem verfahrenstechnische Prozesse.
- Fuzzy-Klassifikation zur Analyse von Prozeßzuständen
- Meßwerterfassung und -analyse
- Speicherprogrammierbare Steuerungen sowie Prozeßleitsysteme
- Erfassung und Analyse von akustischen Signalen als eine Methode, die zur Untersuchung mechanischer Systeme, aber auch in Bereichen wie Produktsicherung und Fertigungsüberwachung eingesetzt wird.
- Gebiet der technischen Diagnostik - frühzeitiges Erkennen von Materialverschleiß und Fehlersuche in technischen Systemen.
- Mustererkennung
- Bildverarbeitung
- Erkennen von Signalverläufen
- Spracherkennung

In Rostock beschäftigen wir uns mit dem Einsatz der Fuzzy-Theorie auf regelungstechnischem Gebiet. Hier wäre das Schlagwort Fuzzy-Control zu erwähnen. Da dieses Wort auch oft mißverstanden wird, möchte ich auch hier eine kleine, verständliche Begriffklärung einschieben:

Bei der Nutzung der Fuzzy-Theorie in der Regelungstechnik wird nicht das technische System modelliert und dann ein Regler entworfen, sondern die Art, wie ein menschlicher Prozeßführer reagiert. Das Wissen über den Prozeß und die Regelung selber bilden hier eine Einheit und werden oft in sogenannten unscharfen Regeln formuliert. Fuzzy-Control ist nicht notwendigerweise mit unscharfen Daten verbunden, sondern mit unscharfen Steuerungskonzepten, die es gestatten, das formulierte Wissen des Prozeßbedieners möglichst direkt, d. h. in Form unscharfer Begriffe, zu übernehmen. Diese Steuerungskonzepte können sowohl scharfe als auch unscharfe Daten verarbeiten. Ein Fuzzyregler stellt ein in der Prozeßautomatisierung eingesetztes Echtzeit-Expertensystem dar, das zur Darstellung qualitativer Größen die Fuzzy-Logik verwendet. Dabei muß nicht notwendigerweise immer der ganze Regler mittels Fuzzy beschrieben werden, sondern es können auch sinnvolle Teilkomponenten oder eine Umschaltung zwischen einem Fuzzy- und einem konventionellen Regler realisiert werden.

Heute sind auch eine Reihe von Werkzeugen bekannt, um Fuzzy-Systeme, vor allem auf dem Gebiet der Steuerungstechnik, zu entwickeln. Sie lassen sich in 4 Gruppen unterteilen [Siemens 1991].

- Geschlossene Shells (sie sind eigentlich nur zu Übungszwecken geeignet)
- Dedizierte Systeme für spezielle Hardware (sie sind sehr leistungsfähig, schnell und professionell, aber auf spezielle Hardware beschränkt und enthalten nur sehr einfache Methoden der Fuzzy-Theorie)
- Shells nach dem Precompilersystem (vor allem für Informatiker)
- On-line-Entwicklungssysteme (Entwicklung, während das System läuft; sie erfüllen die Anforderungen des Ingenieurs am besten)

Stichworte

Fuzzy
Messen - Steuern - Regeln

Gradulation

rk. War vor einigen Jahren noch das Wort Fuzzy in Computerzeitschriften ebenso selten zu finden wie Frauen im deutschsprachigen Raum, die sich mit Forth beschäftigen, so hat sich das Verhältnis in letzter Zeit deutlich verändert: Quellen über Fuzzy gibt es inzwischen zuhauf...! Um so mehr freuen wir uns, auf die Leistung von *Birgit Steffenhagen* aufmerksam machen zu können: Sie hat a) ihren Doktorhut mit der Note *magna cum laude* (sehr gut) erlangt und b) in ihrer Doktorarbeit mit dem Titel: "Integrierte Softwareumgebung zur Realisierung Intelligenter Regelungssysteme" dem Gebiet der Fuzzy-Theorie, deren Einsatz für regelungstechnische Aufgabenstellungen sowie der Implementierung eines Fuzzy-Systems auf der Basis von Forth einen breiten Raum gewidmet. Wer Interesse an dieser "Forth-Dissertation" hat, kann sie zum Selbstkostenpreis von DM 30,- beim Ingenieurbüro Steffenhagen, Fasanenweg 12, 18184 Roggentin erhalten. Hier bekommt man dann etwas Hintergrundwissen, das zwar in wissenschaftlicher Form, aber -wie Sie glaubhaft versichert- nicht zu theoretisch aufbereitet wurde. Herzlichen Glückwunsch von der Redaktion!

Das für das Basissystem comFORTH entwickelte KI-Paket, welchen u. a. eine Fuzzy-Komponente mit Grafikunterstützung enthält, läßt sich in die letzte Gruppe einordnen. Forth bietet hier den unschätzbaren Vorteil interaktiv entwickeln und testen zu können sowie weiterhin konventionelle Sprachelemente nutzen zu können. Das Fuzzy-Modul innerhalb des KI-Paketes stellt unscharfe Inferenzmechanismen, ein Tool mit häufig verwendeten Zugehörigkeitsfunktionen, Modifikatoren und Verknüpfungsoperatoren, Arbeit mit Mustervariablen sowie die Darstellung von unscharfem Wissens mittels Prädikatoperationen zur Verfügung.

Weitere Moduln des KI-Paketes sind die Listenverarbeitung, als grundlegenste Wissendarstellungs- und -verarbeitungsform, Backtrakkingmechnismen für unbestimmte prozedurale Verzweigungen, Mustervariablen und entsprechende Unifikationsalgorithmen, Regelcompiler, boolsche Inferenzmechanismen, Verarbeitung von "scharfen" Prädikatsymbolen, sowie eine Erklärungskomponente, die aus Geschwindigkeits- und Speicherplatzgründen abschaltbar ist.

❑

Zeitschriften

Elrad 94-01 und 94-02 .

94-01 IEEE-488/GPIB/-Karten im Test
AHDL-Share-Ware Programm
'easy-ABEL' für PLDs
PIC-16C Programmiergerät

94-02 PLD-Software im Test
Den Forth-fähigen 68HC11 'MOPS' gibt es jetzt auf Mini-Board.

CAN you ? Elektronik plus 1993/6

In VD 93Q3 wurde Hard- und Software zum CAN-FeldBus vorgestellt. Mehr zu der ganzen Bandbreite der CAN-Möglichkeiten finden Sie im obigen Heft. Neben allgemeinem BasisWissen auch CSUL (CAN-SLIO-User-Langungage), physical layer, hardware und Applikationen.

Franzis Verlag, München
DIN A4, 130 Seiten, DM28.--

Weitere Fuzzy-Infos

Das **KI-Paket zum comForth** gibt es zur Zeit als PC-Variante. Eine cross- und downcompilerfähige Version zur Arbeit auf verschiedenen Mikrocontrollern ist bis Mitte des Jahres 1994 geplant. Wer mehr über dieses Paket, das auch als UR/FORTH-Variante verliegt, wissen möchte, kann sich an untenstehende Adresse wenden. Eine Demo-Diskette ist auch verfügbar.

Die **UR/FORTH-Version** ist ebenfalls seit Anfang des Jahres mit ausgebautem Fuzzy-Modul und Grafikunterstützung zu haben. Sollte jemand ab Februar 1993 eine alte Version (ohne die Worte FUZZY-SET: FUZZYMEMBER CRISPMEMBER sowie ohne Grafikunterstützung) erworben haben, kann er bei unten stehender Adresse die aktuelle Version kostenlos und die Dokumentation sowie das Grafikpaket zum Selbstkostenpreis von jeweils 100 DM gegen Einsendung der Originaldisketten erhalten.

An der **Universität Rostock**, Fachbereich Elektrotechnik, Institut für Automatisierungstechnik läuft eine Vorlesung zum Thema "Technische Expertensystem und Fuzzy-Control". Ein Praktikum für das Fach Regelungstechnik, das den Studenten diese Problematik auch praktisch näher bringen soll, ist in Vorbereitung.

Für weitere Anfragen zum Thema Fuzzy und technische Expertensysteme steht Frau *Dr. B. Steffenhagen* unter den angegebenen Adressen zur Verfügung. Dort sind auch die Demodisketten erhältlich.

Ingenieurbüro Steffenhagen
Fasanenweg 12,
18184 Roggentin

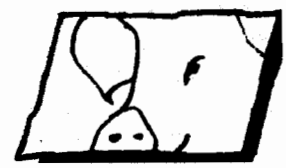
Wer Fragen zu Aktivitäten sowie der Ausbildung an der Uni hat, schreibe bitte an folgende Adresse:

Universität Rostock
Fachbereich Elektrotechnik
Institut für Automatisierungstechnik
Frau Dr. B. Steffenhagen

Die negativ-unendlichen Tatsachen des Bodens

von Arndt Klingelnberg

Straßburger Straße 12, 52477 Alsdorf, Tel. 02404 - 61648



Die letzte SchweineKiste (und zwar die Super-Korrigierte) zeigte die Funktion `/round-abs` (zumindest heißt sie bei mir so, und einen besseren Namen haben wir noch nicht gefunden). Sie leistet eine um Null herum symmetrische Division, also nicht wie Forth83-üblich

```
  -5 2 /MOD       ==> 1 -3
   5 2 /MOD       ==> 1  2
```

bzw.

```
  -5 2 /          ==> -3
   5 2 /           ==> 2
```

sondern:

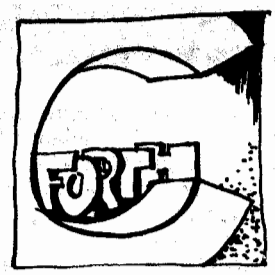
```
  -5 2 /round-abs ==> -3
   5 2 /round-abs ==> 3
```

wobei `/round-abs`, wie oft gewünscht, bei 0.5 aufrundet. ANS unterscheidet `FM/MOD` ('floored' mit MODUL) und `SM/REM` (symmetrisch mit Rest). FLOOR engl. für Fußboden; gemeint ist: nach Negativ-Unendlich hin rundend bzw., genauer, kappend. Gerundet wird NICHT. In der Qualitätskontrolle können sich Rundungsfehler zu horrenden Fehlern aufaddieren bzw., die Produktion unnötig schief legen. Die `XOR 0<` Funktionsfolge ermittelt auf sehr effiziente Weise das Vorzeichen des Ergebnisses, die Absolutwerte werden weiterverarbeitet. Der Modul dient dazu, eine eventuell notwendige Aufrundung zu bewirken. Das Flag wird, 'Thinking Forth'-gemäß, als -1 (oder 0) weiterverarbeitet, so wird ein `IF THEN` Jump umgangen. Eigentlich mag ich diese von *Leo Brodie* dargestellte Methode nicht, aber für die Schweinekiste eignet sie sich gut. TRUE-Flag und `/MOD` sind abhängig vom Forth-Dialekt: "If you have seen one Forth, you have seen ONE Forth!".

❑

Let's Gimpel

Nr. 722



von J² Staben/Plewe, U.Hoffmann
Hagelkreuzstraße 23, 40721 Hilden

Der Blick über den 'C'aun ist immer lohnend. Die Probleme, die dort auftauchen, können auch für den Forther eine Herausforderung sein. Lesen Sie selbst...

Was zum Teufel ist Gimpeln??

Kurz gesagt: die pfiffige Kampagne der amerikanischen Software-Firma GIMPEL, die auf diese Weise auf vor allem sich selbst, ihr Programm LINT und die Tücken eines C-Compilers aufmerksam machen möchte. Gimpel Software stellt in ihren Anzeigen in loser Folge eine ganze Reihe von Problemen im Umgang mit C und einem C-Compiler vor, die dem nicht so geübten C-Programmierer doch ernste Kopfschmerzen bereiten können. Ein "Gimpel" ist jeweils als kurze Programmsequenz formuliert, in der es Fehler oder Ungereimtheiten zu finden gilt.

Witzigerweise beziehen sich die C-'Bugs' von PC-LINT, dies ist der Produktname, nicht auf einen speziellen Compiler, sondern allgemein auf C. Das macht die Sache so interessant - die Probleme sind also zwischen verschiedenen C-Compilern übertragbar, zumindest innerhalb einer Plattform wie dem PC.

In der hier vorgestellten Aufgabe (Abb. 1) geht es um das Anlegen und Initialisieren eines Feldes namens "a" - nur nicht mit 0-Bytes, wie man es gewohnt ist, sondern mit den Werten 0 ... Feldgrenze. Ein Feld von 10.000 Werten ist also mit 0 ... 9.999 vorbesetzt.

Ein Klacks in C, genauso wie in Forth!? Oder doch nicht? Wenn Sie glauben, der C-Quelltext sei richtig - probieren Sie's aus!

Stichworte

F-PC
Datenstrukturen
Arrays
Felder
Programmierfehler
Compilersicherheit

C Bug 722
Initialisieren eines Feldes

```
/* !! Fehlerhafter SourceCode !! */

#include <stdio.h>

int a[1000];

void   main()
{
       int i;
       for (i = 0 ; i < 10000; i++);
       a[i] = i;
}
```

Abb. 1

Nun Forth!

Nachdem Sie gesehen haben, wie tückisch ein C-Compiler sein kann, folgt nun das Gegenstück in Forth. Dank Interpreter und F-PC im Handumdrehen entwickelt, so hatte ich es mir gedacht. Leider saß ich viel, viel zu lange an diesem kurzen Codestück; daher ist mein erster Eindruck: Das taugt alles nichts!

Das erste Problem, das auftrat, war das Aufhängen des Rechners durch das Überschreiten der Feldgrenzen beim Füllen des Feldes. Nun hätte ich mir doch eine Speicherprüfung gewünscht. Ärgerlicherweise gelang es mir auch nicht auf Anhieb, die `DO...LOOP` so zu gestalten, daß ! und ? vernünftig arbeiteten.

Wie eine Speicherprüfung aussehen könnte? Vielleicht merkt man sich via `HERE` den Bereich jenseits des Feldes:

```
here Constant LIMIT
```

und kann dann später gezielt die CTRL-ALT-DEL-bringende Schleife verlassen:

```
dup limit >= IF
                  2drop
                  undo
        exit THEN
```

Nachteilig ist bei diesem Verfahren, daß nur dieses eine Feld abgesichert wird. Ein zweites Feld benötigt dann ein `LIMIT2`, ein drittes Feld benötigt..., weil halt `HERE` sich immer verschiebt. Ach so, wenn Sie das Programm mit einem Feld für 10000 Zahlen mehrfach laden, überschreiten Sie natürlich den verfügbaren Speicherplatz ... und Forth steht still. Vor diesem Effekt soll Sie dann `EMPTY` schützen, das an dieser Stelle seinen Dienst besser versieht, als `ANEW` - warum, weiß der Himmel?!

Beim Ausfaktorisieren von `@addr` hatte ich das "`i`" vergessen; so hängte sich das F-PC ruckzuck auf - dagegen hätte ein `?stack` geholfen, aber wer das kennt, merkt auch, daß ihm ein Wert auf dem Stack fehlt.

Weiter so ?!

Schaut man sich den Forth-Quelltext an (Bild 2), so fallen die unmotivierten `CELLS`-Anweisungen auf. Dümmsterweise ist Forth eine Sprache mit impliziter Typvereinbarung -

Lieber Forth statt C ?

```
empty   \ not ANEW

defined cells nip not
        #IF  ' 2* Alias cells   #THEN


10000 Constant fieldsize

Create a  fieldsize cells allot


: @addr    ( i -- addr )
           cells  a + ;

: init   ( -- )
         fieldsize 0
         DO   i dup @addr !
         LOOP
         ; init
\s
```

Abb. 2

wenn man nix sagt, dann umfaßt auf einem Standard-PC der Grundtyp für Zahlen sechzehn ganze Bit; das ist dann in neuForth (sprich:ANS): Eine Zelle `CELL` . So nimmt man in Forth das `@` "fetch" zum Lesen eines 16Bit-Zeigers, `c@` liest nur 8 Bit, `2@` dagegen 32 Bit. Klare Sache, der Prefix verrät dem Programmierer - nicht dem Compiler -, wie groß der jeweilige Zahlenwert ist.

Aber - wenn Sie nun Wörter wie `ALLOT` oder `ERASE` einsetzen, dann schließen Sie aufgrund eines fehlenden Prefixes "c" oder "2" messerscharf auf einen 16Bit-Zugriff. Daneben! Diese beiden Wörter arbeiten nur mit schlappen acht Bit, müssen also zum richtigen Arbeiten erst mit `CELLS` auf das korrekte Format gebracht werden.

Andere Sprache, andere Probleme

`ALLOT` ist also eigentlich ein `CALLOT` und `ERASE` eigentlich ein `CERASE`. Das "normale" `ALLOT` würde dann als

```
: allot ( n -- )
 cells callot ;
```

definiert; `ERASE` ginge es genauso und Forth wäre wieder ein Stück weiter. Soll jetzt nun eine eMail zum Standards Team geschickt werden oder leben wir einfach damit oder *gibt es einen Weg, diese Stelle zuverlässig abzusichern?*

Übrigens: sollten Sie mit dem C-Programm nicht zurecht gekommen sein - dann ist dies eine gute Gelegenheit, in der Werkzeugkiste Ihres C-Compilers herumzukramen.

Meist ist es die WATCH-Funktion, die da weiterhilft.

Oder man greift halt zu einem *gekauften* Fremdprogramm wie PC-LINT. Sie übergeben den fragwürdigen Quelltext an LINT, der daraufhin eine statische Analyse des Queltextes durchführt.

LINT meldet dann - natürlich innerhalb der integrierten Entwicklungsumgebung - mit einer Auswahl unter hunderten von speziellen Fehlermeldungen seine mehr oder minder begründeten Zweifel an Ihrem Machwerk an.

Stellt man nun Forth und C nebeneinander, hat jede Sprache ihre sprachspezifische Eigenheiten und Probleme:

- C als typprüfender Compiler kennt über die Deklaration `int a[]` den Zahlentyp INTEGER für das Zahlenfeld "a"; intern werden so an allen benötigten Stellen die richtigen Zugriffe ausgewählt. Dafür fallen andere Probleme erst bei der Programmausführung auf, zu deren Lösung weitere Programmierwerkzeuge hinzugekauft werden müssen.
- **Forth** bietet einen kleinen Laufzeitvorteil, da das Feld schon während des Kompilierens initialisiert werden und diese Initialisierung mit einem `SAVESYSTEM` festgeschrieben kann. Für das Behandeln von Programmierfehlern können - mangels Anbietermasse - keine Fremdprodukte eingesetzt werden; dafür ist die Sprache Forth selbst Werkzeug genug, um mit zusätzlichen Sicherungen gegen Programmierfehler ausgestattet zu werden.

Abschließend der Blick ins Bücherregal, wo auch *M.Tracy's* "Mastering Forth" [1989] steht. Darin wird ab S.66 ff. alles über Felder und

Die Lösung in Forth mit range check (U. Hoffmann)

```
\ cell-array, optional range check  by U.Hoffmann
/*
(c) 1993 Ulrich Hoffmann, Graf-Spee-Strasse 38, D-24105 Kiel
*/

defined cells nip not
        #IF  ' 2* Alias cells   #THEN


Variable range-checking    range-checking off

: unchecked-array       ( n -- )
    Create cells allot    \ ( cell0 | cell1 ... celln-1 )
    Does>  ( i -- addr )
           swap  cells +  ;
 \  ;code  pop BX  pop AX   add AX, AX   add AX, BX   1push   end-code


: checked-array ( n -- )
    Create dup ,
           cells allot  \ ( size | cell0 ... celln-1 )
    Does>  ( i -- addr )
           dup @ rot  tuck u>
           IF ( index ok )  1+ cells +  EXIT THEN
           cr ." index out of range in array " over body> >name .id
           cr ." indexrange is 0 to " over @ 1- u.
           cr ." actual index is " . cr
           true abort" out of range"
           ;

: cell-array ( n -- )
   range-checking @ IF  checked-array  ELSE  unchecked-array  THEN ;

\ ----------------

range-checking on

10000 cell-array a

: main ( -- )
   10000 0 DO  I I a !  LOOP ;

: show ( -- )
   10000 0 DO  I a ?  tab ?cr LOOP ;

\ ----------------
\ Like Turbo Pascal? Here's their weird range checking syntax :-) !

: (*$R+*)   range-checking on ;
: (*$R-*)   range-checking off ;
```

Abb. 3

Tabellen gesagt. Den gleichen Sachverhalt hat *A.Winfield* [1983!] in "The complete Forth" ab S.25 ebenfalls sehr schön dargelegt; solche Datenstrukturen dürften also seit zehn Jahren eigentlich kein Thema mehr sein. Das dies dennoch der Fall ist, beweist *H.-W. Beilstein* [1987] in "Wie man in Forth programmiert", wo es ab S.281 falsch zu lesen ist. Danach empfindet man die Ausführungen von *G.-U. Vack* in "Programmieren mit Forth" [1990] ab S.214 richtig wohltuend, die zwar anspruchsvoll, aber dafür richtig und umfassend sind.

❑

Fundsachen

Ein HandBuch heißt so, weil es gerade noch in der Hand gehalten werden kann.

— — —

Herzlichen Glückwunsch zu Ihrem neuen Computer. Wir danken Ihnen, daß Sie die besten Jahre Ihres Lebens opfern wollen, um eine unterentwickelte Technik zu verstehen. ... Das erste Kapitel das Handbuches erklärt, wie sie mit dem Kapitel das Handbuches umgehen müssen, das erklärt wie sie mit dem Rest des Handbuches umgehen müssen. Bevor Sie diese Kapitel lesen, machen Sie sich bitte mit dem nächsten Kapitel vertraut, das klarstellt, warum das erste Kapitel wichtig war. Wenn Sie das Handbuch gelesen haben, sind Sie zwar nicht in der Lage, Ihren Computer zu verstehen, aber erheblich älter.

Dr. h.c. Brian Eno

Kein Whiskey!

Die Redaktion hatte 0.0 Rückantworten zu "Bug, Prompt, EC-Geld-Automat und Whiskey" (VD 93Q1 p.38), die Redaktion dankt und trinkt Whiskey und Schampus alleine.

Die Abhilfe war bereits aufgeführt: Enthält das Forth-System Zahlen als Literale (-1, 0, 1, 2) so sollte vor einem Prompt die VariableDPL immer auf -1 gesetzt werden.

Die Erklärung zum 'Vorfall' fehlte: Nach der (zweiten) Eingabe, und zwar von 4, verarbeitet das Forth-System 2621.44 (da war ein Druckfehler), da DPL den Wert 2 enthält und die 4 als HighWord eines 32-bit Wertes aufgefasst wird (LowWord == 0).

262144.=$40000.

Die Lösung fällt mit F-PC-ak besondes leicht, da es einen PullDown-Stack enthält, eben auch für DoubleNumbers.

Fortsetzung des Schröder-Aufnehmers von Seite 21

```
        write-only  log-file hopen drop       \ Log-File zum Schreiben öffnen
        log-file endfile log-file movepointer \ an's Ende anhängen
        r> span @   log-file hwrite drop      \ Eingabe schreiben
        crlf$ count log-file hwrite drop      \ Datei-Zeilenende dazu
        log-file hclose drop                  \ Log-File schlieen
        ;

ALSO FORTH DEFINITIONS

: log-on ( --- )
        @> EXPECT ['] log-expect = ?exit \ Test, ob Log-Funktion schon aktiv ist
        write-only log-file hopen 0<>        \ Log-File zum Schreiben öffnen
        IF log-file hcreate                  \ oder neues File erzeugen...
\gr     abort" Fehler beim Erzeugen der Log-Datei!" THEN \ bei Fehler Abbruch
\en     abort" error while creating logfile!" THEN
        log-file endfile        log-file movepointer    \ an's Ende anhängen
        log-start$ count        log-file hwrite drop    \ Meldung schreiben
\gr     D.M.Y                                           \ deutsches bzw.
\en     M/D/Y                                           \ engl. Datumsformat
        GETDATE FORM-DATE COUNT log-file hwrite drop    \ Datum eintragen
        SPCS 1                  log-file hwrite drop    \ 1 Leerzeichen
        GETTIME FORM-TIME COUNT log-file hwrite drop    \ Zeit eintragen
        crlf$ count             log-file hwrite drop    \ Zeilenende schreiben
        log-file hclose drop                    \ Log-File schlieen
        ['] log-expect is expect                \ Bedeutung von EXPECT ändern
        ;

: log-off ( --- )
        @> EXPECT ['] log-expect <> ?exit    \ Test, ob Log-Fkt. schon aus ist
        @> expectorg is expect               \ ursprüngl.EXPECT restaurieren
        write-only log-file hopen drop       \ Log-File zum Schreiben öffnen
        log-file endfile log-file movepointer        \ an's Ende anhängen
        log-end$ count          log-file hwrite drop \ Ende-Meldung schreiben
        GETDATE FORM-DATE COUNT log-file hwrite drop \ genau wie LOG-ON
        SPCS 1                  log-file hwrite drop \
        GETTIME FORM-TIME COUNT log-file hwrite drop \
        crlf$ count             log-file hwrite drop \
        crlf$ count             log-file hwrite drop \ noch eine Leerzeile
        log-file hclose drop                         \ Log-File schließen
        ;

HIDDEN DEFINITIONS

: log-bye ( --- )       \ automatisches 'Log-Off' beim Tschü-Sagen
        defers byefunc log-off ;

' log-bye is byefunc

ONLY FORTH DEFINITIONS
```

❑

Tips und Tricks

Ein Profi meldet sich zu Wort

von Dipl.-Ing. Wolfgang Allinger
Brander Weg 6, D-42699 Solingen 11, Tel.: 0212-66811, Fax: 0212-66811

Das wollen wir lesen: Profis berichten aus ihrem Alltag mit Forth. Es folgen Tips, Tricks und Erfahrungen; gewonnen bei der Arbeit mit Forth für Embedded Controllern.

Ein- und Ausgabe Formatierung

Ich habe einige Jahre gebraucht, um einen guten Ansatz zu finden, mit dem man die Probleme bei kleinen Systemen mit begrenzter Kommunikationsmöglichkeit angeht. Ich schrieb einige Routinen um Zahlen auf 7 Segment-Anzeigen auszugeben; schrieb andere Routinen, um Tastaturen zu lesen und benutzte serielle Schnittstellen für die Kommunikation während der Entwicklung. Dann schrieb ich weitere unterschiedliche Formatierungs-Routinen für 7 Segment-Anzeigen, testete sie interaktiv mit Parametern auf dem Stack über die serielle Schnittstelle und so weiter und so fort. Dann brauchte ich weitere Formate, die Formatierungs-Routinen wurden erweitert, getestet usw. Für jede Modifikation brauchte ich Testroutinen und Teststubs tief im Kernel. Ich machte viele Fehler, einige davon kamen erst nach vielen Wochen zum Vorschein.

Als ich aber einen Abkömmling des Spinnmaschinen-Controllers (siehe: Forth spinnt, VD 1/93) so umprogrammieren mußte, daß er als Steuergerät für einen sogenannten Cutter (Maschine zur Herstellung von Wurstmasse) funktionieren konnte, fand ich heraus, daß es viel leichter ist, einen 7 Segment-Treiber zu haben, der wie `EMIT` und `TYPE` arbeitet, aber nur die Zeichen umwandelt und an den 7 Segment Controller Baustein schickt, die er ohne Murren verträgt und die anderen Zeichen wegschmeißt. HEUREKA !! Das wars !!! Nun brauchte ich bei Bedarf nur noch `EMIT` und `TYPE` nach `7EMIT` und `7TYPE` mit einem Wort `7SEG` umbiegen und nach getaner Arbeit die alten Vektoren mit `CONSOLE`, `COM` oder anderen FORTH-83 Worten zurückstellen.

Ich baute ein Wort `%7OUT` das das 7 Segment Equivalent zu `OUT` ist. `7OUT` setzt die Adresse der 7 Segmentanzeige nach `%7OUT` und `EMIT` und `TYPE` springen nach `7EMIT` und `7TYPE`, solange `%7OUT` > -1 ist. Nun kann ich Standard-Forth-Formatierungs-Worte benutzen und brauche mich nicht mehr mit speziellen Formatierungsworten herum zuschlagen. Vor allem kann ich alles über `CONSOLE` austesten, schalte dann mit `7OUT` um und und mit `-7OUT` ab und bin fertig!!!

In Zukunft werde ich das auch für die Tastaturen von Embedded Controllern machen. So kann ich dann `NUMBER` und alle anderen Input Worte von Forth benutzen. Dazu muß `?TERMINAL` und `KEY` vektorisiert werden, und fertig ist die Laube.

Hört sich nach einer guten Idee an, aber ich habe sie noch nirgendwo gesehen. Für mich ist das nun klar wie Kloßbrühe! Umso mehr wundert es mich heute, daß ich mindestens 5 Jahre gebraucht habe, um darauf zu kommen!

Dump und Anzeigen

Der nächste Tip ist: Schreibe sofort alle möglichen Dump-Routinen für spezielle Datenstrukturen und Anhäufungen von Variablen; man braucht sie irgendwann sowieso! Aber wenn man sie schreibt, während man seine Datenbasis entwirft, hat man noch alle speziellen Formate, Umwandlungen, Sonderfälle.. im Sinn, und man kann diese Worte gleich beim Testen der Daten gebrauchen. Zu einem späteren Zeitpunkt verheddert man sich leicht bei einigen besonders skalierten Variablen oder merkwürdigen Strukturen. Es kostet dann sehr viel mehr Zeit!

Ein weiterer Tip zu derartigen Programmen: mach sie möglichst nett und zeig sie Deinem Kunden. Möglicherweise findet er diese Ausdrucke brauchbar für sich und erweitert den Auftragsumfang um nette Anzeigeprogramme. Das bringt zusätzliches, leicht verdientes Geld.

Simulation des Targets

Simuliere Dein Programm auf einem PC, nimm `GOTOXY` oder irgend etwas ähnliches, um an eine bestimmten Platz des Bildschirmes zu schreiben, der selten von anderen Programmen benutzt wird. Wenn Du spezielle Ausgabetreiber wie `7EMIT` usw. hast, setzte Conditional Compilation Flags ein, um die Teile bei Bedarf scharf zu machen, die auf den PC Bildschirm schreiben.

So kann man sehr leicht auf dem PC interaktiv testen und braucht später nur noch auf die speziellen Target-Definitionen per Flag umzuschalten um nur noch diese zu testen. Das spart sehr viel Zeit! Für diese Conditional

Listing zur Simulation des Targets

```
0 CONSTANT ccALL ( 0= TARGET, >0 TEST environment )
ccALL .IF
: 7EMIT ( c -- ) \ emit char to 7 segment controller \ HOST
            ?XY ROT 24 %7OUT @ 20 + GOTOXY
            EMIT %7OUT 1+!
            GOTOXY ;
  .ELSE
: 7EMIT ( c -- ) \ emit char to 7 segment controller \ TARGET
             d 7 3 $P7SEG C! %7OUT 1+! ;
  .THEN
```

Abb. 1

Compilation Flags benutze ich den Namen ccALL. 'cc' erinnert mich an Conditional Compilation und ALL ist eh mein Logo. Mit LMI metacompilern, besonders den 2.xx Versionen, muß man unbedingt den HOST Teil vor dem TARGET Teil definieren oder es gibt merkwürdige Ergebnisse. Ich benutze ccALL auch, um zwischen HOST dummys und TARGET Code Definitionen umzuschalten (Bild 1).

In diesem Beispiel sieht man auch, daß ich weitgehenst die Namens Konventionen aus *Leo Brodies* Buch 'Thinkink Forth' benutzte, die ich für meinen Bedarf etwas erweitert habe, siehe Bild 2.

Weiterhin benutze ich Prefixes wie %% $$ und && vor Namen die größere Werte als 16bit ansprechen. Worte in GROSSBUCHSTABEN sind die globalen Worte, Kleinbuchstaben Worte werden für locale Worte, Primitives und Runtime Worte benutzt...

(Die Klammerritis in F-PC und F-83 ist ein Greul, bloß weil in der Vergangenheit einige Schlamper aus Faulheit oder Bequemlichkeit keine Unterscheidung zwischen UPPER Case und LOWER Case gemacht haben, plagt man sich heute damit rum.)

Prüfung auf STACK Veränderungen

Mache einen Stack Check in der Haupt-Endlos-Schleife und breche sie ab, wenn die Stacktiefe verändert wurde. Es macht nämlich keinen Sinn, in einer Endlosschleife den Stack atmen zu lassen. So kann man leicht feststellen ob was auf dem Stack hinterlassen oder etwas geklaut wurde.

Stell Dir den Test eines komplizierten Benutzerdialogs vor, bei dem nach Gebrauch von Funktion 4711 und zweimaligem Drücken der Ziffertaste 5 und danach einem Backspace das System crasht. Aber dies kommt nur vor, wenn Funktion 4711 2mal nach Funktion 10 aufgerufen wurde. Nach langer Zeit findest Du, daß der allererste Aufruf von Funktion 10 irgendwas auf dem Stack hinterlassen hat und 3 Stunden später crasht das System. (So ein ähnlicher Sch... ist mir ein paarmal passiert, nun hat sowas keine Chance mehr.)

Stenographische Ausgaben

Baue eine Flagge in Dein System ein, das Stenografische Ausgaben auf der seriellen Schnittstelle freigibt, wenn der Bediener eine Taste drückt und gib allen Tasten einen Spitznamen möglichst mit einem Zeichen. zB. ENTER erzeugt ein 'e', UP-ARROW ein '^', Zahlen eine Ascii Ziffer und so weiter. Gib den Stenomode frei, und man braucht nur einen Drucker mit serieller Schnittstelle, um alle Tastendrücke mitzuschreiben. Man braucht selbst nicht mehr alle Tastenreihenfolgen zu notieren. Außerdem hilft es gegen böse Buben, die Dein System zum Absturz bringen, Dir aber nicht erzählen (wollen) wie Sie es gemacht haben (verlängert so schön die Zeit bis zur Abnahme...). Ein versteckter Drucker bringt es dann zu Tage. (*..oder ein guter Aufnehmer. Tip der Redaktion*)

Einige nützliche Worte

Ich mag noch Worte wie L L+ L-, welche den current, next oder previous Screen eine Screenfiles anzeigen.

```
: L   SCR @     LIST ;
: L+  SCR @ 1+  LIST ;
: L-  SCR @ 1-  LIST ;
```

Das Wort .SH (Bild 3) zeigt mir einen horizontalen HEX Stack Dump. Ich setze es mit dem Editor innerhalb eines Wortes, was ich testen will und schalte es mit +SH und -SH bei Bedarf ein und aus.

Frage

Nun noch ein letzter Tip: Befrage Deinen Kunden über alles, was nicht klar ist! Frage auch nach Dingen die scheinbar klar und selbstverständlich sind! Ich kenne ein Sprichwort: Es gibt keine dummen Fragen, nur dumme Antworten.

Dein Kunde hat möglicherweise eine Insider Sprache, die Worte in einem anderen Kontext benutzt. Versuche diesen Slang zu verstehen und benutze diese Ausdrücke bei Deiner Applikation. Dies ist in Forth besonders mit besonderen CONTEXTs einfach. Das Ergebnis wird ein DICTIONARY sein, was eine anwenderspezifische Sprache enthält.

❑

Namenskonventionen

Type	Form	Beispiel
variables	%variable	%7OUT
constants, besondere Adr. im internen Memories von speziellen Processoren wie 8051...	$constant	$P7SEG
arrays, tables ...	&array	&RESULTS

Abb. 2

Das Wort .SH

```
: .sh     ESC? DROP BASE @ >R HEX DEPTH ?DUP
          IF  ." ( --"       \ "(" didn't work w/ MC2.2
              0 DO
                   DEPTH I - 1- PICK
                   0 <# # # # # # BL HOLD #> TYPE
              LOOP ."  )"
          ELSE ." <empty>"
          THEN CR R> BASE ! ;
VARIABLE %SH
: .SH     %SH PERFORM ;
: +SH ['] .sh %SH ! ;
: -SH ['] NOOP %SH ! ;
+SH
```

Abb. 3

Brief aus der Provinz

von Friederich Prinz
Homberger Str. 335, 47443 Moers

Unseren 'Tag der offenen Tür' haben wir hinter uns gebracht; mit weitaus größerem Erfolg, als wir zu hoffen gewagt hatten. Mehr als 60 Besucher haben unser Angebot genutzt, von denen 37 (!) völlig 'forthfremd' waren. Wir haben uns im Nachhinein gefragt, warum 'alles so gut geklappt' hat, und meinen heute, das hätte daran gelegen, daß wir auch diesen Tag, der eigentlich eher ein 'FORTH-Fest' war, einfach 'forthig' angegangen sind. "Just doit" hieß unsere Devise. Wir haben bei den Vorbereitungen nicht viel diskutiert und 'herumgemacht', sondern einfach getan was notwendig war - und wir hatten Erfolg. Immerhin haben sich bereits am ersten Samstag nach diesem Fest 5 Interessenten an einem Einsteigerkurs gemeldet. Und schon am nächsten Samstag ist diese Zahl auf 11 (!!!) Personen gestiegen. Wobei wir aber zugeben müssen, daß von den elf 3 'Wiederholer' sind, die für sich entschieden haben, FORTH noch einmal 'von Grund auf' zu lernen. Einen großen Teil unseres Erfolges verdanken wir aber auch der vielfältigen Hilfe, die uns schon bei den Vorbereitungen zuteil geworden ist. Das Moerser Arbeitslosenzentrum hat uns zum Beispiel weitere Räume zur Verfügung gestellt. Die Forthgesellschaft e.V. hat uns mit reichlich Material zur FG und zu FORTH unterstützt, ebenso wie die Firma FLESCH Forthsysteme aus Breisach. Rolf Kretzschmar's 1*1 m großes Lampenfeld durften wir ausstellen, und ein ganz besonderer Anziehungspunkt war die Lampenfeld 'Montage' von Klaus-Peter Schleisiek, die ein aus 7 DisCo-Feldern montiertes Gesicht mit forthigen Grüßen gezeigt hat. Allen 'Supportern' unseres Festes sei an dieser Stelle ganz herzlich gedankt.

In der Moerser Forthgruppe 'tut sich was'. Bei uns geht es gut voran. Wir hoffen, daß sich von den 'Neuen' der Eine oder die Andere (wir haben auch wieder eine mutige Frau in unseren Reihen) für eine Mitgliedschaft in der FG entscheiden wird. Wir drängen aber nach wie vor Niemanden dazu und stellen den Spaß an FORTH und am gemeinsamen Lernen immer noch über die Ziele der FG. Neben dem Einsteigerkurs, der diesmal von Michael Major angeboten und gestaltet wird, bietet Friederich Prinz ein sehr interessantes 'Seminar' zum Thema Multitasking und Interprozeßkommunikation an. Dabei sollen im Verlauf des nächsten halben Jahres grundsätzliche Probleme aufgezeigt werden, die MSDOS bei der Arbeit in diesem Themenkreis mit sich bringt. Dem werden zum Abschluß die 'Features' moderner Betriebssysteme gegenübergestellt werden, unter deren Regie Multitasking und IPC 'ganz einfach' zu realisieren sind. Hier wird FORTH/2 zum Zuge kommen, das sich allmählich über die Moerser Grenzen hinaus verbreitet.

Selbstverständlich sind die Moerser FORTHer über die Aktivitäten in ihrer Gruppe hinaus damit beschäftigt, FORTH zu 'verbreiten' und zu größerer Popularität zu verhelfen - entsprechend der Satzung der Forthgesellschaft. So haben wir zum Beispiel Kontakt zu einer Firma in einer Nachbarstadt aufgenommen, die über eine Mailbox einem Kundenstamm in der gesamten BRD Support zu ihren Softwareprodukten bietet. TRICOM EDV und ACOS vertreiben mit großem Erfolg ein Produkt zur Erstellung von Planungen im Bereich von Großprojekten. Die gerade im Aufbau befindliche Mailbox dieser Unternehmen bietet seit kurzem ein 'Brett' zum Thema FORTH/2 an, das von uns gepflegt wird. Eine erste Reaktion ist die Anfrage eines Kunden der Firma TRICOM, der sich auf diesem Wege für FORTH interessiert und über die Moerser Gruppe Kontakte zu FORTHern sucht (und gefunden hat).

Forth Dimensions Artikel

Besprechungen von Fred Behringer, Planegger Str. 24, 81241 München

Math - Who Needs It?

5, Vol.14 Januar/Februar 1993, S.27
T. Hendtlass, Hawthorn, Australia

"Der Titel dieses Artikels wurde bewußt als in zweierlei Richtung interpretierbar gehalten. Wie man auch immer zur Mathematik im allgemeinen stehen mag, Arithmetik (zumindest das) braucht früher oder später jeder in seinen Programmen. Das, was jedem, der sich Forth von einer anderen Sprache her nähert, sofort auffällt, ist der Umstand, daß das Forth-Grundsystem anscheinend nur einen einzigen Typus von Zahlen direkt anbietet, 16-Bit-Ganzzahlen. Blickt man jedoch genauer hin, dann sieht man, daß dem nicht ganz so ist, daß vielmehr ..." Der Autor führt diesen Gedanken weiter fort, nach dem Motto: Im Forth-Grundsystem braucht fast gar nichts definiert zu werden, da der Anwender ja beliebig viele weitere Forth-Programmteile hinzuladen kann, eigene Entwicklungen oder übernommene Pakete. Und an "Mathematik" sollte man nur genau soviel einführen, wie man benötigt. Ganzzahlen erfordern weniger Aufwand, zeitlich und vom Speicher her gesehen, als doppeltgenaue Ganzzahlen, diese weniger als Festkommazahlen, und letztere wiederum weniger als Gleitkommazahlen. Der Autor bietet eine ganze Sammlung von gut kommentierten Arithmetik-Paketen an: 93 Doppelpunktdefinitionen. (Ein wahrer Schatz!) Einiges stammt aus seiner Feder, anderes von anderen (gut kenntlich gemacht). 32-Bit-Ganzzahlen, 32-Bit-Festkommazahlen, 32-Bit-Gleitkommazahlen aus verschiedenen Quellen, trigonometrische Funktionen auf der Grundlage von 32-Bit-Gleitkommazahlen ("Zen slide rule" von Nathaniel Grossman). Außerdem wird das fast ganz in Assembler geschriebene 48-Bit-Gleitkomma-Paket SFLOAT aus F-PC kurz besprochen. Dessen Programm wird im Artikel (natürlich) nicht wiedergegeben. Mir fällt auf, daß im ganzen Artikel keine einzige Code-Definition zu finden ist. Das wird aber in einem Abschnitt (26 Zeilen in einer von zwei Spalten) begründet: Portierbarkeit, Lesbarkeit, Korrigierbarkeit usw. Der Autor meint, daß ein großer Teil des Zeitgewinns bei maschinencodeprogrammierten Gleitkommazahlen darauf zurückzuführen ist, daß das Überlaufbit bei Addition und Subtraktion in Doppelpunktdefinitionen erst "künstlich" erzeugt werden muß. (Eben! Und noch viel mehr! - Warum muß eigentlich F+ portierbar sein ? Ich habe ja gar keine Veranlassung, F+ in einem Anwendungsprogramm zu zerpflücken. Und wenn ich es sowieso als Ganzes nehme, ist es mir doch egal, ob als CODE F+ ... NEXT END-CODE oder als : F+ ... ; (?)) Die in den Programmen dieses Artikels enthaltenen Forthworte werden im Text ausführlich diskutiert. In drei Tabellen werden Zeitvergleiche angestellt.

Fazit: Ein gründlicher, ausführlicher (14 Seiten), informativer, in das eigene System sofort übernehmbarer, sehr empfehlenswerter Artikel.

Numbers

5, Vol.14 Januar/Februar 1993, S.18
C.H. Ting Forth-Kurs, 3. Folge

Ein Artikel aus einer Serie nur für Einsteiger. Nicht für Umsteiger. Kein Bezug auf andere Sprachen. Es ist nicht klar, welches Forth gemeint ist. `.S` wird als "Anzeige der obersten vier Stack-Einträge" erklärt. FORTH 83 kann es also nicht sein. Dort bedeutet `.S` die nichtdestruktive Anzeige des gesamten Stackinhalts (vergleiche grünes Zech-Buch). Es wird der Umgang mit ganzen Zahlen (Integern) eines 16-Bit-Forths anhand von Währungsumrechnungen demonstriert. Dann die Operationen `+ - * / */` und `.S` (siehe oben). Dann `MOD 1+ 1- 2* 2/ ABS` und `NEGATE`. Dann `DUP SWAP OVER ROT` und `DROP`. Kein `TUCK`, kein `NIP`. Keine Doppelpunkt-Definitionen, kein Maschinencode. Nur verbale Beschreibungen. Als Beispiele werden-Fläche, Mittelpunkt und Umfang eines Rechtecks berechnet. Die Relationen > < und = und auch `0= 0<` und `NOT` werden im Abschnitt "Logische Operationen" besprochen. Als Wahrheitswert `TRUE` wird jede Zahl ungleich 0 betrachtet (meiner Meinung nach gefährlich). Als Beispiel für "logische Operationen" und Sprünge wird eine `IF-ELSE-THEN`-Konstruktion angegeben. Keine Erklärung der Funktionsweise dieser Konstruktion und der Vertraktheit der Bezeichnung (man denke an den Ausweg `ENDIF`). Schließlich wird die `DO-LOOP`-Konstruktion anhand eines Beispiels (Multiplikationstafel) erklärt. Was passiert, wenn Anfang und Ende der Schleife gleich sind (vergleiche `?DO` in FORTH 83), wird nicht gesagt.

➪

Game of Life

1, Vol.15 Mai/Juni 1993, S.30
C.H. Ting Forth-Kurs, 4. Folge

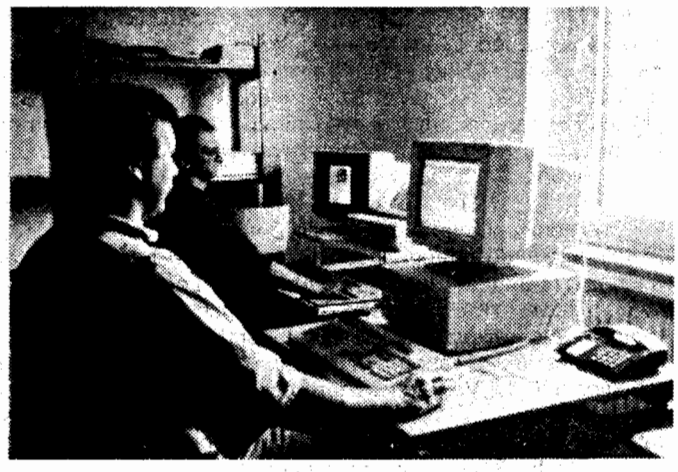
"Anhand von zwei ziemlich komplizierten Beispielen soll die Funktionsweise der in den letzten drei Artikeln besprochenen Forth-Befehle gezeigt werden." Kalender von 1950 bis 2050 auf Seite 30, Conways "Game of Life" ab Seite 33. Beide Programme werden in voller Länge mit Kommentaren gebracht. Aus einer Bemerkung geht hervor, daß Ting F-PC im Sinn hat. Die an sich interessante Besprechung des Kalender-Problems beschränkt sich auf den Kalender. Die Funktionsweise der Forth-Worte wird nicht erklärt. Das Programm enthält eine geschachtelte `IF-ELSE ... IF-ELSE...THEN...THEN`-Konstruktion und Eakers CASE. Die Besonderheit von Eakers CASE, die elegante Schreibweise für eine an sich undurchsichtige geschachtelte IF-THEN-Konstruktion, wird nicht erwähnt. Der Kalender-Algorithmus ist der astrophysikalischen Fachliteratur entnommen. (Hinweis wird gegeben.) Es werden ein paar Worte über `S>D D+ UM*` und über den Return-Stack nebst `>R R> R@` gesagt. Die Besprechung beschränkt sich auf das, was man sonst in den "Glossaren" findet. Das "Game of Life" enthält 7 spieleigene FORTH-Worte. Es ist erstaunlich und interessant, daß man dieses Beispiel für Evolutionsstrategien mit so wenig Aufwand erledigen kann. (Es lebe FORTH!) Beide Programme sind zur Einführung für Anfänger gedacht. Der "Anfänger" wird sich schwer tun. Eigentlich interessant - für Nichtanfänger, aber eben auch für Anfänger - sind die dahinterstehenden Algorithmen. Die kurze Beschreibung des "Game of Life" - nicht die der Forth-Worte, die gar nicht gegeben wird, die des Algorithmus - ist beeindruckend. Abschließend erwähnt werden die im Life-Programm "neu auf- tretenden" Forth-Worte `CONSTANT C@ C! CMOVE PAD AND OR` . Ich habe das Life-Programm spaßeshalber mal für mein Transputer-Forth-System (das ich wohl F-TP nennen werde) abgetippt: Läuft prima. Da es nur Forth-83-Standard-Worte verwendet, lief es ohne Änderungen sofort auch auf dem Turbo-Forth-System für IBM-Kompatible (JEDI - Marc Petremann). - Und da es die Bildschirmausgaben mit `EMIT` "bewältigt", läuft es auf beiden Systemen gleichlangsam ab. Die eigentliche Rechenzeit kann beidemal vernachlässigt werden.

Bemerkung: Die folgende Population hat (als Ganzes betrachtet) einen Lebenszyklus von 14 Generationen, d.h., die 14 nachfolgenden Generationen sind alle voneinander verschieden, die 15. hat wieder das Aussehen der ersten: 3 SPACES, ##, 14 SPACES, ##, 14 SPACES, ##, 14 SPACES, ##, 14 SPACES, ##, 11 SPACES . Das alles in einer Zeile, und davon 24 Zeilen.

Das Besondere an diesem Beispiel ist, daß sich nur die Spalten von- einander unterscheiden. In jeder Generation haben alle Zeilen gleiches Aussehen.

Frage: Ist es möglich, die Gesamtheit aller Populationen (für einen 80x24-Bildschirm) zu beschreiben, die einen Lebenszyklus von 14 Generationen haben ?

❑



Tag der offenen Tür in Moers

Eine Idee, die von Michael Major 'geboren' wurde, wächst allmählich zu konkreteren Strukturen aus. Wir beabsichtigen eine Art Forum für einen regional begrenzteren Kreis zu installieren. Eines der größten Probleme der Forthgesellschaft ist unbestreitbar, daß in ihr kein Vereinsleben im üblichen Sinne stattfinden kann. Dazu leben die einzelnen Mitglieder zu weit auseinander. Zusammenkünfte sind schwer zu organisieren und gerade einmal im Jahr 'mit Inhalten zu füllen'. Uns schwebt ein 'Forum' vor, das zusätzlich zur Jahrestagung der FG zwei bis drei Mal im Jahr stattfinden und gezielt die FORTHer im Großraum Rheinland, Aachen und Niederlande ansprechen soll. Wir stellen uns vor, daß zwei bis drei 'forthige' Wochenenden zu jeweils ganz speziellen Themen die FORTHer in diesem Großraum einander ein wenig näher bringen könnten. Es gibt so Vieles zu diskutieren, so viele Anregungen und Ideen, die man austauschen könnte - das sollten wir nicht brachliegen lassen. Auf Dauer können wir das allerdings sicher nicht alleine leisten. Deshalb sind an dieser Stelle alle FORTHer aus dem angesprochenen Raum aufgefordert, sich in Moers zu melden und ihr (hoffentlich vorhandenes) Interesse zu bekunden.

Zum Schluß wollen wir noch etwas 'loswerden', was auf den ersten Blick vielleicht weniger selbstverständlich ist, als es sein sollte. Einen 'Tag der offenen Tür' zu organisieren, hat unseres Wissens unsere Gruppe erstmalig in der FG versucht. Es wäre schön, wenn andere lokale Gruppen Ähnliches probieren - und ähnlichen Erfolg damit haben. Damit der Erfolg sich ein wenig leichter einstellt, sind wir, wie das unter FORTHern üblich ist, selbstver- ständlich bereit, mit Rat und Tat zu helfen. Wir geben gerne weiter, was wir selbst bei der Organisierung gelernt haben. Und wir helfen gerne auch bei 'ganze profanen' Dingen, wie zum Beispiel beim Druck von Plakaten etc..

In diesem Sinne stürzen wir uns wieder in unsere eigene Arbeit und grüßen die FORTHer der FG vom 'linken' Niederrhein...

Forthgruppe Moers

Endlich: Rad erfunden!

Angeregt durch den Staben-Artikel 'Sl-nuß' (VD 3/93) schreibt *Arndt Klingelnberg* sich einige Gedanken mit gewohnt kritischem Unterton von der Seele:

Das alte Problem "ES WAR ALLES SCHON MAL DA" wird -- obwohl eigentlich (über)lebenswichtig -- ein Problem bleiben, zumal eine Lösung keinen Lebensunterhalt verspricht, und selbst die, die das Problem immer wieder anprangern, einige einfache BasisRegeln nicht einhalten.

Die für uns zugänglichen Quellen (F-PC und Forth-e.V.) fördern folgende Sinus Sourcen zutage:

- Tshapes.seq in TCOM
- sin-zech
  \ sin-zech SINUS aus GRAFIK-ANBINDUNG (ZECH F83) dhd 23jan89
  \loadscreen Grafik-Paket fnk 31jan89
- GRAFIK-ANBINDUNG IN volksFORTH (frei nach ZECH: " FORTH 83 ")
- sincos.seq = dragon.scr 21feb88 ck volksFORTH auf dem Atari ST

Vielleicht sollten wird auch mal Namen-IDs sammeln, wer um Gottes Willen ist nun *ck dhd unf fnk* ? Vielleicht auch eine Anregung für die Mitglieder-Liste. Ansonsten sollten wohl Dateien, die in die PublicDomain geschickt werden auch mal einen Voll-Namen tragen.

Weiterhin finde ich es gut, wenn QuellHinweise auch im Source nicht vergessen werden, und zwar auch mit dem Autor. So hätte ich gerne in Jörgs SIN und Quelltext selbst TSHAPES.seq gefunden und

eben auch einen Hinweis auf ODS == *Oliver Shape* ,die eigentliche Sinus-Quelle. Werden PublicDomain Quellen genutzt, so sollte deren Herkunft klargestellt werden, das halte ich für eine grundlegende Verfahrensart im PD-Bereich.

Der Dateinamen sollte immer auch in der Source selbst, und wenn das nicht, so doch zumindest im Ausdruck angegeben werden. Wie kann sonst etwas wiedergefunden werden? Die sinnvolle Wahl eines Dateinamens ist leider auch ein Kritikpunkt, aber es gibt Möglichkeiten mnemotechnisch sinnvolle Namen zu finden, und zwar eben auch Namen, die doch nicht hochgradig ambivalent oder trivial sind. Ein völlig blödsinniger Name ist in diesem Zusmmenhang read.me .

Nicht nur *Tom Zimmer* hat es uns vorgemacht: Dateiname und informative Überschrift in die erste Zeile eines Quelltextes. Beim Blocksystem wurde das schließlich auch von vielen auf Screen 0 in der obersten Zeile gemacht. `INDEX` oder ein Ausdruck mit F-PC (oder auch anderen print utilities) verarbeiten diese erste Zeile.

Aber auch bei den 'keywords' zu den Artikeln: selbst da kommen von den Autoren fast nie und wenn, dann oft unzureichende Beiträge. Wie sollen wir denn da weiterkommen, wenn jeder das Neu-und-Neu-Erfinden provoziert. Ich weiß, daß die letzten 5% einer Software Entwicklung 95% der Zeit benötigen, aber so vieles so roh auf den PublicDomain TummelPlatz geworfen, das schreckt ab.

Wenn es Forth nicht ermöglicht, daß Informationsaufbereitung/kommprimierung (professionell) gemacht wird, weil es nicht verkäuflich ist -- leider -- und soVIEL kostenlose Zeit mag keiner aufbringen -- ich höre allzugerne das Gegenteil -- kommt keine saubere Arbeit heraus. Forth liegt wohl unterhalb einer kritischen Mindestmasse, zumindest was die Sorgfalt einiger angeht.

Und noch ein Anmerkung. *Jörg Staben* fragt, warum er den TaschenRechner für SINUS holen muß. Nun er muß nicht, weder für den Sinus noch für die MwSt.. Ganz einfach gehts in F-PC-ak, und zwar in der 'Saurier'-INSTANT version. Ein Maus-Click im ForthMenü oder interaktiv `CALC` ruft den Floating Point Calculators auf. Ein Sinus von 12 ergibt sich nach Eingabe von 12.0 -- ja, es ist Forth und ein eingeschlossenes Decimalzeichen __.0 ist (noch) notwendig -- dann über die CursorTasten xDEG»» und SINx angelickt: 0.2079117 . Genauer wird es, wenn ein 80x87 vorhanden ist und ABACUS.seq auf F-PC geladen wird (0.2079116908).

Der 'alte' `ABACUS.seq` von Dr. Ting liegt nun in einer verbesserten Version v.1.0ak3 vor, dazu `aFLOAT.seq` , das automatisch sFLOAT.seq ('S'oftware) oder fFLOAT.seq ('F'ast bzw. genau bzw. 87) lädt. sFLOAT.seq und fFLOAT.seq sind jetzt sehr sehr kompatibel zueinander. Auch sind (fast alle) ANS FloatingPoint Worte eingeflossen, dazu verbesserter Zahlen-In und Output. Dazu einige wichtige BUGfixes (meinen Dank an *Peter Tröster* = PTR), das Ergebnis soll ja doch zuverlässig sein. Dazu direkt eine Frage/Bitte. Natürlich handelt es sich nicht um den SuperPlusCalculator. Wer hat Lust, dem ganzen (noch mehr) mehr HP-36 ... HP-41 Verhalten beizubringen, die Voraussetzungen sind nun da.

Benchmark

Leserbrief von Arndt Kingelnberg zum Artikel S.30 VD3/93

Straßburger Straße 12, 52477 Alsdorf

Die Daten für F-PC auf 386SX 20 MHz :

- BENCH
  15 Byte (CODE-space, 129 gesamt)
  0.55...0.61 s
- XENCH
  (mit VALUES statt vars)
  0.49...0.55 s
  (ca. 10% schneller)

Es wurde das Aufrufen verschiedener Funktionen gezählt (ohne Gewehr):

- 256 * LOOP I
- 8193 * CONSTANT + UNconditional-JMP (ELSE-->THEN)
- 8066 * 1-
- 16259 * ! @ short-literal 16b-literal rot swap
- 16400 * DROP
- 32k * AND VAR CONDITIONAL-JMP (IF & UNTIL)
  \ 8300 * NICHT verzweigend
  \ 24k * verzweigend (dazu noch 256 * LOOP)
- 48k * DUP

Kritik:

- zu viel: `AND DUP`
- etwas viel: Verzweigender conditional-JMP
- etwas wenig: NICHT-verzweigender cond.-JUMP (`TRUE IF ... THEN` oder `TRUE WHILE...`)
- (fast) kein `LOOP I * /MOD 2* >R R@ R>`
- Bit-Befehle sind für embedded-controll ganz wesentlich, wenn auch nicht standardmäßig programmierbar. Sie sollte in KEINEM sinnvollen Benchmark fehlen.
- Das viel zu viele `DUP` benachteiligt alle Forth Systeme ÜBERgebührlich die TOS nicht im Register halten (z.B: F-PC). Das besonders schnelle `DROP` des F-PC kann das nicht wieder ausgleichen. Das ist praxisfern!
- Da Besonderheiten bei `>R R> R@` vorkommen können, sollte soetwas nicht fehlen. `DO__LOOP`s sind nun auch häufig genug an der Tagesordnung, und wenn, dann auch meist besonders laufzeitKritisch.
- Etwas simpel-Mathematik ist immer nötig (`* /MOD */`), `2*` wird auch oft genug benötigt.

Postiv war der Hinweis auf den Kompromiss zwischen Schnelligkeit zu Speicherverbrauch und der Hinweis, daß Takt und Takt je nach µP etwas ganz anderes bedeutet.

Die Normierung auf (langsame) EPROM Zugriffszeiten halte ich heutzutage nur für bedingt sinnvoll, zudem war die Normierung kaum (begreifbar) erklärt.

Im englischen gibt es einen weitverbreiteten Spruch: Benchmarks don't lie, but liars do benchmarks. Das ist eine Abwandlung von Winston Churchill's Aussage: Benutze nie eine Statistik, die Du nicht selbst 'ausgewertet' hast.

Moerser Mail Support

Die MHB (engl.: BBS; sprich: Mail-Box) in Moers (02841-57325) stellt allen Forthern in Zukunft eine eigene 'Support Area' zur Verfügung. Einfach die MHB anwählen, als User FORTH eingeben (Passwort wird nicht erfragt) und solange <enter> drücken, bis der kurze Vorspanntext abgelaufen ist. Der Anrufer wird automatisch in die Area SUPFORT/FORTH geführt, wo er sich in den drei Verzeichnissen /ZF /TCCM und /FORTH2 bewegen kann. In allen drei Verzeichnissen liegen jeweils die Versionen der drei Systeme, die aktuell von der Forthgruppe Moers genutzt werden. Die Systeme sind komplett und zunächst als sehr umfangreiche *.EXE Dateien verpackt, die sich nach dem 'Downladen' beim ersten Aufruf selbst entpacken.

Die drei Verzeichnisse werden von der Moeser Gruppe gepflegt und aktualisiert.

Vor allem zum ZF und zum FORTH/2 werden dort in Zukunft sporadisch immer wieder Demos und neuere Versionen aufgeladen werden. Diese 'Updates' werden dann als einzelne, kleine Dateien eingespielt, so daß nicht jedes Mal eine Datei von mehreren 100 KByte via DFÜ kopiert werden muß.

Die MHB ist mit 1200 bis 14400 BAUD anwählbar.

Mistgeschick...

Friederich Prinz möchte auf diesem Weg folgende Nachricht an alle **FORTH/2**-Interessenten von *Mike Warot* verbreiten:

Hello Fritz,

(...) I do have a problem, however. My car was stolen, and I had 2 letters from users in germany in it at that time. The car has since been recovered and repaired, however the letters are missing. I do not know who the senders were. I would like to send these persons the disks which they are owed.

Could you please post a notice asking them to send their addresses and preferred disk size? (Email would be ok!) I'm not worried about the money, I just want to get these people what they are owed. I trust that I won't be flooded with false claims. ; -)

Thank you.

Mike Warot - MikeWarotI@aol.com

Anmerkung: Mike Warot ist via Internet - also auch über die Box der Forth-Gesellschaft - unter der angegebenen Adresse zu erreichen. Erfahrungsgemäß antwortet Mike innerhalb von 48 Stunden.

Forth-Gruppen regional

Berlin
Claus Vogt
Tel. 0+30 - 2 16 89 38 p
Treffen: nach Absprache

Rhein-Ruhr
Jörg Plewe
Tel. 0+208 - 49 70 68 p
Treffen: jeden 1. Samstag im Monat im S-Bahnhof Derendorf
Münsterstr. 199, Düsseldorf

Moers
Friederich Prinz
Tel. 0+2841 - 5 83 98 p
Treffen: jeden Samstag 14:00
Arbeitslosenzentrum, Donaustr. 1
Moers

Darmstadt
Andreas Soeder
Tel. 0+6257 - 27 44

Mannheim
Thomas Prinz
Tel. 0+6271 - 28 30 p
Ewald Rieger
Tel. 0+6239 - 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.:0+89 - 8 41 83 17

Gruppengründungen, Kontakte

Regional

Stuttgart
Wolf-Helge Neumann
Tel. 0+711- 8 87 26 38 p

Fachbezogen

8051 ... (Forth statt Basic,e-FORTH)
Thomas Prinz
Tel. 0+6271 - 28 30 p

Forth-Hilfe für Ratsuchende:

Forth allgemein

Jörg Plewe
Tel. 0+208 - 49 70 68 p
plewe@mpi-dortmund.mpg.de

Karl Schroer
Tel. 0+2845 - 2 89 51 p

Jörg Staben
Tel. 0+2103 - 24 06 09 p

Spezielle Fachgebiete

Anfänger und Wiedereinsteiger
Gerd Limbach
Tel. 0+2051 - 25 51 12 p
Mo. + Di. 20:00 - 22:00

32FORTH (Atari)
Rainer Aumiller
Tel. 0+89 - 6 70 83 55 gp

FORTHchips (FRP1600, RTX, Novix ...)
Klaus Schleisiek-Kern
Tel. 0+40 - 2 20 25 67 p

F-PC & tCOM, ASYST (Meßtechnik), embedded controller (H8/5xx//TDS2020, 8051 ... eFORTH...), FUZZY
Arndt Klingelnberg
Tel. 0+2404 - 6 16 48 agp

Gleitkomma-Arithmetik
Andreas Döring
Tel. 0+721- 59 39 35 p

HS/Forth (Harvard Softworks)
Wigand Gawenda
Tel. 0+30- 44 69 41 p

KI (Künstliche Intelligenz), OOF (Object Oriented Forth)
Ulrich Hoffmann
Tel. 0+431 - 80 12 14 p

Unterricht mit FORTH
Rolf Kretzschmar
Tel./Fax 0+2401 - 8 88 91 ap

UUCP (FORTH ... per eMAIL)
Andreas Jennen
Tel. 0+30 - 3 96 52 27 ap

volksFORTH/ultraFORTH/RTX-FG-Forth/Super8Forth
Klaus Kohl,
Tel. 0+8233 - 3 05 24 p
Fax. 0+8233 - 99 71 f

Forth-Vertrieb

volksFORTH (PC, ST, C64) / F68K (68000) / ...
Johannes Teich
Ettaler-Mandl-Weg 19
82418 Murnau
Tel. 0+8841 - 29 43
jgt@bbs.forth-ev.de

Forum: Forth-Mailbox

Forth-Mailbox (BBS)
Jens Wilke (SysOp)
Tel. 0+89 - 8 71 43 52 p
Mailbox 0+89 - 8 71 45 48,
300-2400 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.

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

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

FORTH-SYSTEME GMBH

Postfach 1103,
79200 Breisach

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

Telefon Schweiz:
(055) 53 65 55

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

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

SRS II

- Serieller ROM Emulator
- Unterstützung folgender Bausteine:
  27256, 27512, 271000, 27010, 27020, 27040
- Minimale Zugriffszeit 100 ns
- Maximale Baudrate 115.200 bits/s
- Highspeed Interface als Option
- Gleichzeitiger Zugriff von Host und Zielprozessor
- Zusätzliche serielle Schnittstelle über den ROM-Sockel
- Intel-Hex, Motorola-S oder ASCII/binär Formate werden unterstützt
- Der SRS II ist nur 157 x 94 x 36 mm groß
- SRS63 kompatibel

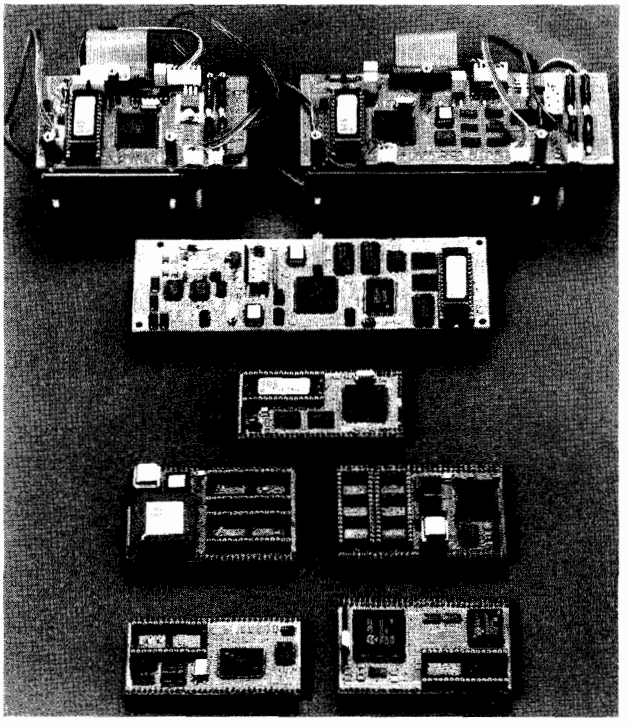
FORTH-83 Metacompiler

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
- 78310
- 8031/32/535
- 6303
- 6502
- V25
- 68HC11
- RTX 2000
- 80C166

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

ModuNORM



CPU-Steck-Module im Scheckkartenformat:

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

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