1987

Vierte Dimension
Volume III/Nr.1

VIERTE DIMENSION

THURAU
MASCHINEN
STANDARD
MENUETECHNIK
GEMSELECTBOX

FORTH MAGAZIN

5,00 DM

Programm '87

der Hamburger Gruppe

der Forth Gesellschaft e.V.

Friedensallee 92, 2000 Hamburg 50

Jeweils ab 16:00 in der Fachschule für Fernsehtechniker, Eimsbüttelerstr. 64-66, 3.Stock Hinterhaus.

28. Feb	U.Hoffmann:	Metacompilation
28. Mar	K.Schleisiek:	Nachrichten Verwaltung auf dem PC "Westentaschen-dpa"
25. Apr	R.Heuer:	Assoziative Systeme
23. Mai	M.Pauck:	Computer-Grafik

-- Sommerpause --

22. Aug	L.Wellhausen:	Robotersteuerung
26. Sep	H.Blum:	Prozessteuerung in der Chemie
24. Okt	J.Storjohann:	Numerik und Floating Point
28. Nov	N.N.	

-- Winterstarre --

euer lokaler Koordinator

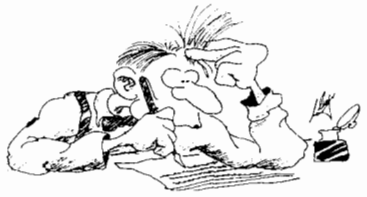
Bernd Pennemann, Steilshooper Str. 46, 2000 Hamburg 60, Tel. 690 05 39

AN ALLE

Besten Dank
an alle
aktiven und passiven
Förderer des Forth
1986

Es ist 1987.
Und es ist wieder so weit.
An alle Mitglieder der Forth Gesellschaft:
Beiträge für dieses Jahr
JETZT
überweisen.
Formular im Heft.

INHALT



Anleitung für Autoren
Das FORTH MAGAZIN 'Vierte Dimension' veröffentlicht originale Arbeiten, Berichte und Bibliographien, die in Bezug zur Programmiersprache FORTH stehen. Manuskripte können an das Büro der Forth Gesellschaft geschickt werden, oder direkt an die REDAKTION des Forth Magazins, zZ.:
Michael Kalus, Präsidentenstr.40, D-5830 Schwelm

Die Arbeiten sollten folgendes enthalten:

1. TITELSEITE mit dem Titel, Autor(en) und Institut(en) oder Bertieb(en), in denen sie angefertigt wurden.
2. Eine ZUSAMMENFASSUNG von 50-100 Worten, möglichst in deutscher und englischer Sprache. Die Zusammenfassung sollte Absicht, Methoden, Ergebnisse und Schlußfolgerungen der Arbeit enthalten und für sich genommen bereits verständlich sein.
3. Etwa 5 SCHLÜSSELWORTE. Um einen Index leichter erstellen zu können, werden diese mit der Zusammenfassung in beiden Sprachen wiedergegeben.
4. TEXT. Wenn möglich sollte der Text in klassischer Form aufgebaut sein, dh in einer kurzen Einleitung über das Ziel der Arbeit informieren, Materialien und Methoden hinreichend genau wiedergeben, über die Entwicklung der Ergebnisse oder Systeme berichten, diese diskutieren und Schlüsse ziehen.
5. DANK für Hilfen oder Rat, technische Mitarbeit, Materialien sollte in einem eigenen Abschnitt am Ende der Arbeit ausgesprochen werden.
6. LITERATURHINWEISE und QUELLEN sollten auf einer eigenen Seite getippt sein, um sie für das Layout gesondert verkleinern zu können, sollten alphabetisch nach Autoren geordnet und durchnummeriert sein. Im Text beziehen sich die Nummern in runden Klammern auf diese Liste. Zeitungsartikel sollten mit den Namen und Initialen von allen Autoren, dem vollen Titel des Artikels, dem Namen der Publikation, der Rubrik, dem Erscheinungsjahr und der Nummer der erste und letzten Seite des Artikels genannt werden. Bücher sollten mit dem Namen des Autors, dem vollen Titel, Ausgabe, Erscheinungsort, Herausgeber und Jahr genannt werden.
7. ILLUSTRATIONEN sollten auf das unbedingt Notwendige beschränkt werden und keine Daten enthalten, die besser in Tabellenform wiedergegeben werden können. Diagramme und Kurvenverläufe sollten als Schwarz-Weiß-Zeichnungen kopiergeeignet sein. Bilder sollten ebenfalls Schwarz-Weiß, scharf und kontrastreich sein. Anleitungen für das Layout oder Anmerkungen zum Text sollten auf einem aufgelegten Transparentblatt und nicht auf der Vorlage selbst gemacht werden. Alle Illustrationen sollten auf ihrer Rückseite dem Text entsprechend nummeriert sein.
8. Jede TABELLE sollte auf einer eigenen Seite getippt sein, eine Nummer und eine Überschrift tragen und im Text angesprochen werden. Jede Spalte sollte einen Namen tragen. Senkrecht gestellte Beschriftung möglichst vermeiden.
9. Der QUELLCODE sollte auf einer eigenen Seite getippt sein, eine Seiten- oder Screen-Nummer und eine Überschrift tragen und im Text angesprochen werden. Die Kommentare zum Code sollen zeigen, WAS der kommentierte Abschnitt ausführt, also den Sinn wiedergeben. Bei besonderen Programmiertechniken kann der Kommentar auch das WIE näher beschreiben oder begründen. Bitte stets mit einem kräftigen Farbband drucken. Bitte nur Listings von getesteten Programmen einsenden. Programmfehler fallen auf den Autor zurück. Der Ausschuß zur kritischen Besprechung von Forth Code kann die Progamme nicht alle austesten.

Die Arbeiten sollten in doppelter Ausfertigung eingereicht werden. Verwenden Sie eine normalgroße, nicht zu dünne Schrift. Verwenden Sie möglichst ein frisches Farbband. Verwenden Sie nicht mehr als 80 Spalten und 72 Zeilen auf einer DIN A4 Seite. Die Beiträge werden überarbeitet, um die Kommunikation zwischen Leser und Autor effektiver zu machen und um Mehrdeutigkeiten zu vermeiden. Wenn ausgedehnteres Edieren nötig ist, erhält der Autor vor der Wiedergabe den Beitrag zur Korrektur und Zustimmung zu Verbesserungsvorschlägen zurück. Die Layouts werden nicht mehr zur Prüfung durch die Autoren vorgelegt. Autoren erhalten 5 kostenlose Exemplare des FORTH MAGAZIN. Auf Wunsch auch mehr, falls der Vorrat reicht. Manuskripte können per DFü (300Bd,ASCII) übergeben werden.

IMPRESSUM

Titel: FORTH MAGAZIN 'Vierte Dimension'.
Zeitschrift der Mitglieder der Forth Gesellschaft eV.
Herausgeber: Forth Gesellschaft eV.
Forth: Klaus Schleisiek und die Mitglieder des Review-Boards sowie alle namentlich genannten Autoren.

Redaktion: Michael Kalus, Tel: 02336-82204
Präsidentenstraße 40, 5830 Schwelm
Anzeigenleitung: Arndt Klingelnberg, Tel: 02404-61648
Straßburgerstr.12, 5110 Alsdorf
Erscheinungsweise: Ein Heft je Quartal.
Redaktionsschluß: Der mittlere Quartalsmonat.
Auflage: 500 Stück europaweit.

Druck: NN

Das Forth Büro

Forth Gesellschaft eV, Friedensallee 92, 2000 Hamburg 50

Tel.: 040-3904204

Postgiroamt Hamburg, Kto: 563211-208, BLZ 20010020

EDITORIAL

"Wenn Pascal wie ein Holzbaukasten mit vorgeformten Bausteinen ist, dann ist Forth wie ein Haufen Holz und ein Satz Tischlerwerkzeuge."

Forth ist kein Monolith, sondern eine Modul-Sammlung, mit der man zahlreiche Programmieraufgaben vortrefflich lösen kann. Forthler versuchen immer wieder, aus ihrer Programmiererfahrung so etwas wie 'den' Grundwortschatz herauszufiltern. Uns sie bauen stets ihre Systeme aus zu 'ihrer' idealen Programmierumgebung. Und so finden sich zwei Strömungen im Forth: Die einen suchen radikal jede Spur von 'Overhead' zu tilgen und wollen das optimierte Minimalsystem. Die anderen gestalten komplizierte Systeme in Megabyteräumen zur komfortablen Umgebung aus. Nun, man könnte sagen: Jedem wie es ihm beliebt. Doch da ist das Problem der menschlichen Natur: Unstetigkeit. Keine Firma wird sich in sensiblen Bereichen von der Präsenz nur eines Programmierers abhängig machen! Und arbeiten mehrere in einem Bereich, brauchen sie einen gemeinsamen Grundwortschatz. Also 'Standards'. Das Problem beginnt aber gerade hier.

1979 und 1983 wurden die Erfahrungen mit Forth gesichtet und Standards formuliert. Weitere vier Jahre sind vergangen - wird das 'Forth Standards Team' wieder tätig werden? Gibt es neue Erkenntnisse, neue Erfahrungen? Die Megabyte-Adressräume wollen bedient sein. Ich bin gespannt, was da kommt.

Charles Moore's eigenes Forthsystem, das CMForth, obschon Silicon geworden im Deltaboard, wird sicher keinem Standard folgen. Doch er könnte erneut Maßstäbe setzen durch sein Streben nach dem kleinstmöglichen Wortschatz überhaupt, dem 'echten' KERNEL des Forth. Lets keep it simple'

Wie soll das bloß weitergehen ?

An dieser Stelle wurde eine in loser Folge erscheinende Artikelreihe zur GEM-Programmierung in Forth begonnen [1,2]. Die einfachen Strukturen sind damit leider weitgehend abgegrast. Die geplanten Artikel werden sich mit Dialogboxen und Menüzeilen beschäftigen. Zur Herstellung dieser Boxen wird aber das Programm "RCS" (Resource Construction Set) als Hilfsmittel benötigt. Dieses Programm ist im Entwicklungspaket zum Atari ST und im Megamax-C-Compiler enthalten. Sollten alle Leser ein RCS besitzen, so können die oben erwähnten Artikel relativ kurz sein. Anderernfalls besteht auch die Möglichkeit, diese Boxen und Menüzeilen als Listing einzutippen, allerdings erfordert das relativ viel Zeit. Es hätte aber den Vorteil, daß man dabei lernt, wie die Strukturen aufgebaut sind. Sofern unser Editor mitspielt, würde ich diese Lösung bevorzugen.
Ich bitte alle (!!) Leser, die diese Artikelreihe lesen und abtippen wollen, zwischen den unten angegebenen Möglichkeiten abzustimmen.

[1] "Alert-Boxen unter volksFORTH83 auf dem Atari ST", Bernd Pennemann, Vierte Dimension, Vol. II/Nr.4, S.19-23
[2] "Die Fileselector-Box unter volksFORTH83 auf de Atari ST", Bernd Pennemann, in diesem Heft.

(Fragebogen auf der nächsten Seite)

* * *

Question: Do you think the fifth generation could be done using Forth? Chuck Moore: Oy! I think Forth is the only possibility. Even stronger, Forth is the only SDI possibility. The chance of convincing them is zero. So pray for disarmament. (Aus einer CompuServe Konferenz mit C.H.Moore)

* * *

Johns Hopkins Correction

Dear Editor,

We would like to point out a factual error in Glen Haydon's article, "The Multi-Dimensions of Forth" (VIII/3). The article, in describing several hardware Forth engines, states that we at Johns Hopkins University's Applied Physics Laboratory "...have taken the basic design of the Novix 4000 device and expanded it to a thirty-two bit processor on a chip." It is true that we have designed a single-chip, thirty-two bit Forth processor, but it is in no way related to the Novix processor. Our processor was independently designed based on our experience with a microprogrammed bit-slice Forth engine our group designed for the Hopkins Ultraviolet Telescope, a part of the ASTRO Space Shuttle mission.

The Novix processor and our processor are radically different in both architecture and implementation. The Novix chip achieves high performance by connecting to external memory via three buses, one for fetching instructions and two for accessing the parameter and return stacks. Our processor uses a more conventional single bus, but caches the top sixteen elements of both the parameter and return stack on chip. Our architecture was influenced by RISC research and has only two instruction formats. The Novix design is implemented in a CMOS gate array. We did a full custom implementation of our design in four-micron SOS CMOS, which is suitable for high radiation spacecraft environments. We are currently reimplementing the architecture in three-micron bulk CMOS and will be finished in the second quarter of 1987. Papers describing the full details of the processor and architecture have been submitted to the 1986 FORML Conference.

Martin E. Fraeman
John R. Hayes
Robert L. Williams
Thomas Zaremba
Johns Hopkins University
Laurel, Maryland

cms:

---- schnippel -- schnippel schnippel schnapp -- schnippel dapp--

( ) Ich habe dieses RCS, mir muß nur gesagt werden, was ich tun muß, um die Beispiele zu reproduzieren.

( ) Ich habe kein RCS und will auch nicht lange Listings abtippen. Daher möchte ich, daß mir die benötigten Files gegen einen Unkostenbeitrag auf eine Diskette kopiert werden.

( ) Ich will alles selber machen. Druckt die Listings ab und ich tippe sie ein. Dabei verstehe ich wenigstens, wie die Strukturen des GEM aufgebaut sind.

( ) Ich finde diese Reihe großen Quatsch. Sie sollte abgebrochen werden.

Ihre Meinung senden Sie bitte schriftlich oder telephonisch so schnell wie möglich (innerhalb eines Monats) an :

Bernd Pennemann Steilshooper Str.46 2000 Hamburg 60 Tel. 040-690 05 39

NACHRICHTEN

*** Berichtigungen und Hinweise zu VD86 II/4:
Gedruckt wurde - anders als im letzten Impressum ausgewiesen - wieder in Hamburg. Billiger und - zu meinem großen Bedauern - so schlecht wie früher. Nun gut, wenn die Vereinskasse nicht mehr hergibt, kann man nichts machen. Die Schallmauer für den Einstieg ins Profilager bei der Produktion der Hefte liegt um 2500,-DM/Ausgabe allein für die Fertigstellung bis zur Druckfolie. Die Papierkosten für den Druck einer höheren Auflage und der Versand fallen dann nicht mehr so ins Gewicht. Alles in allem könnte die FG beim derzeitigen Mitgliederstand sich gerade eben ein solches Blatt mit vier Ausgaben im Jahr leisten, wenn es nicht noch andere Aktivitäten gäbe, wie das FORTH BüRO und den TREE. Schade... (mk)

*** Berichtigungen und Hinweise zu VD86 II/3:
Die Redaktion möchte bemerken, daß mit dem Heft VD86II/3 gezeigt wurde, was man machen könnte, wenn die Mittel da wären. Wer sich also bessere Heftqualität wünscht: Bitte nicht meckern, sondern neue Mitglieder für die Forth Gesellschaft werben und Spenden für die Sache der Public Domaine des Forth besorgen. Alles Klar? --- Dem Drucker, Herrn Lüdemann in Wuppertal, möchte ich für die exzellente Arbeit und extrem schnelle versandgerechte Herstellung der Auflage VD86 II/3 auf diesem Wege herzlich danken. Übrigens, in seinem Betrieb steuert FORTH diese und jene Maschine... (mk)

Im Artikel 'Queues in Forth' von B.Pennemann enthält der Quelltext einen Fehler: Die Definition auf Zeile B muß QFULL? (und nicht qempty?) heißen, und die Definition auf Zeile C muß QEMPTY? (und nicht qfull?) heißen. Also die Namen einfach austauschen. (Wie konnte das passieren??? der säza)... (bp/mk)

*** VolksFORTH-83 3.8
Neben C64 und ATARI 520ST sind die CP/M Versionen bereits im Umlauf, die Schneider CPC 464/664 Version eingeschlossen. Kostet 45,- DM mit Handbuch. Gibts bei B. Pennemann, Steilshooperstr.46, 2000 Hamburg. Im Großraum Rhein-Ruhr zu bekommen in der Buchhandlung Finke, Kipdorf 32, 5600 Wuppertal 1, also mitten in W.-Elberfeld.

*** National Forth Convention USA 1986
Bald eintausend Leute versammelten sich im November letzten Jahres im neuen Handels- und Kongresszentrum von Santa Clara in Californiens Silicon Valley. Thema waren die Fortschritte der Forth-Forschung und kommerzielle Unternehmungen basierend auf Forthprogrammen. Dabei lag das Interesse eindeutig bei der Entwicklung im Bereich der Forth Maschinen. Und diese scheinen wie Pilze aus dem Boden zu schießen. Angepeilt ist der Bereich Robotics. Typische Eigenschaften der Forthmaschinen: Implementationsaufwand klein, vergleichsweise wenig Gatter und extrem schnell. Was will man mehr: Simpel im Umgang, billig und schnell.
Die größte Stückzahl Forthchips hat bis heute sicherlich Rockwell ausgeliefert - R65F11 und F68HC11 Chips. Ihr R65F11 ist dabei noch nichts weiter als ein 6502 mit Ports und Forth im Rom, wenn auch auf einem Chip (Und damit so etwas wie der gute alte AIM65 mit fig-Forth damals - ein echtes Arbeitspferd unter den Singelboard Computern jener Jahre; noch häufig anzutreffen, übrigens). Die Forthworte des R65F11 sind damit der heimliche 'Defacto Standard' - so gesehen ist Forth-79 bzw fig-Forth heute aktueller denn je. Forthsysteme mit Zilogs super Z-8 wurden ebenso diskutiert.
Novix's NC4000 hingegen ist eine echte Forth CPU. Vorgestellt in Europa bei der EuroFORML 1985 machte dieser Chip im letzten Jahr Karriere zb auf dem DeltaBoard, offenbar trotz der Bugs im Chip. Und weitere Hersteller machen von sich reden: die Hartronics Maschine, ein

32Bit-Forth Chip der Johns Hopkins Universität und der multi-stack writeable instruktion set computer (WISC) von Haydon Enterprise.

Das es die Sprache SMALL-C, geschrieben in Forth für den Novix NC4000, bereits gibt, sei am Rande erwähnt. Mit einem solchen Board im IBM-PC laufen Anwendungen 6 bis 7 mal schneller ab. (Warum es mit C unter Forth machen, wenn es mit Forth direkt gleich 40 mal schneller geht?) So viel dazu für heute. (mk)

* * *

Thomas Prinz *6930 Eberbach a/N*
FORTH - KOPIEN *Ad.-Stifter-Str. 2*
Tel.:(0 62 71) 2830

Stand : 08.01.87 L I T E R A T U R - V E R Z E I C H N I S

Fig - FORTH Listings für 6502 , 6800 , 6809 (e)
FORTH - 83 Standard Beschreibung (e)
euroFORML Conference 1985 Treffen auf Burg Stettenfels (e)
1984 FORML Conference Proceedings (e)
1984 Rochester FORTH Conference Real-Time-Sytems (e)

FORTH Dimensions Vol. 5 No. 1 1983 (alles -e-)
" " 2 "
" " 3 "
" 6 " 5 1984 S. 9 - 11 How to learn FORTH
S.13 - 16 Simple Modem I/O - Words
" " 6 " S. 7 - 8 Learning FORTH

Dr. Dobb's Journal Vol.6 Is. 9 Nr. 59 1981 September Spezielle FORTH-Ausgabe (alles -e-)
" 10 " 60 1981 Oktober FORTH - Teil
" 64 1982 Februar FORTH - Teil
" 71 1982 September Spezielle FORTH-Ausgabe
" 83 1983 September Spezielle FORTH-Ausgabe

DEPARTMENTS

FORTH DIMENSIONS
Vol. VIII, No.5

DEPARTMENTS

FORTH DIMENSIONS
Vol. VIII, No.4

LESERBRIEFE

Klaus-Peter Schleisiek
An den Finkenweiden 38
D-5100 A A C H E N
Tel.: 0241-872 272

13. Januar 1987

G E G E N D A R S T E L L U N G

Der in der VIERTEn DIMENSION II/4 erschienene Artikel "Die Kunst, Namen zu geben" stammt **n i c h t** von mir ! Es handelt sich dabei um einen lächerlichen Verschnitt, den die Redaktion ohne mein Wissen oder Einwilligung angefertigt hat.

Der Original-Beitrag hat den Titel "FORTH - Namens-Konventionen oder: dem FORTH am Zeug geflickt" und behandelt folgende Punkte:

	List-Ref.#
Einleitung	
Zur Person des Autors; Zweck des Beitrags	
Verschiedenen Konventions-Vorschläge	
Stack-Operationen ohne Pfeile	#1
Fragezeichen bei interaktiven Eingaben	
Umschalt-Worte mit Kennzeichnung	#2
Präfix-Kennzeichnung mit Doppelpunkt	#3
Klare Worte und symmetrische Logik für die Ablauf-Steuerung	
Ersatz für IF, ELSE und THEN	#4a
Ersatz für NOT IF	#4b
logisch symmetrischer Ersatz für WHILE und UNTIL	#5
artgleiche CASE-Struktur	#6
Ergänzung zu den Vorschlägen	
Aussprache-Vorschläge	
Listing nach Referenz-Nummern	

Wer das Original mit der Fälschung selbst vergleichen möchte kann es gerne bei mir anfordern.
(Bitte mit adressiertem Freiumschlag (DM 1,30))

Bitte das Beleg-Exemplar der Erscheinungsnummer nicht zu vergessen!
Den Abdruck des Original-Artikels oder seiner Derivate verbiete ich.
Mit gleicher Post sende ich meine Beschwerde an die Direktoren der FGeV.

Höflichst

Schleisiek

(Schleisiek)

Antwort auf die Gegendarstellung

Lieber Klaus-Peter Schleisiek, nachdem ich Deinen Brief vom 13.1.87 gelesen hatte, mußte ich erst einmal luftholen. Nun gut. Nachdem mein erster Zorn verraucht ist, antworte ich Dir.

In der Sache hast Du - zugegeben - recht. Hiermit entschuldige ich mich in aller Form.

Im Heft VD86II/2 hatten wir den Beitrag "IF THEN gestürzt". In Deinem Beitrag waren dazu einige Überlegungen enthalten. Andere denken auch über Kontrollstrukturen nach. Diese Strömungen hätte ich gern in weitere Artikel umgesetzt, nur war das für die VD86II/4 nicht zu schaffen. Deshalb habe ich zunächst nur einen Teil Deines ursprünglichen Artikels 'verarbeitet', den über die Namensgebung - ich fand gut, wie Du das machst. So kam es, daß ich daraus einen eigenständigen Artikel gestrickt habe. Die Form hat sich dabei - nungut, sehr - geändert, aber die Idee ist doch wohl erhalten geblieben, und Dein Name steht darüber, weil es Deine Idee war. Schade, daß Dir die Verarbeitung so garnicht gefallen hat. Machs gut! (Kalus)

* * *

CMOVE und Umgebung

Bei meinem ersten 6800 fig Forth bin ich als 'Novize' mit CMOVE schrecklich eingegangen. Der zu verschiebende Block war zerstört, nur die stumpfsinnige Wiederholung des ersten Bytes bis zum Ende des Blocks war geblieben. Dann las ich, daß es bei manchen Forthversionen ein umgekehrt schiebendes Wort gibt. Aber von einem intelligenten MOVE las ich nichts. Wieso ist CMOVE so debil? Der 'Dreck-Effekt' des CMOVE wird übrigens vom FILL im 6809 fig Forth ausgenutzt!

Andreas Soeder, Seeheim-Jugendheim

FORTH MAGEAZIN: CMOVE dürfte in allen Forthsystemen in Code geschrieben sein, denn es ist Geschwindigkeit gefragt. Diese Routine schiebt oder kopiert Strings, Heaps, Buffer, Blocks usw. Dabei treten oft keine Überlappungen der Speicherbereiche auf, und es ist somit nicht wichtig, ob von niedrigen zu hohen Adressen oder umgekehrt bewegt wird. Nur das Tempo ist gefragt. CMOVE hat daher die größtmögliche Geschwindigkeit, die der jeweilige Prozessor für solche Bewegungen zu bieten hat, ein Minimum an Zyklen ohne Prüfung der Richtung. CMOVE ist ein echtes Forthwort: kurz, ballastfrei und ungemein nützlich. CMOVE bewegt die Bytes beginnend mit der niedrigen Adresse. Überlappen die Speicherbereiche, kann man nur zu den niedrigeren Adressen hin gefahrlos 'verschieben'. CMOVE> ist für die andere Richtung da. Es bewegt beginnend mit dem höchsten Byte. Bei Überlappung kann man zu höheren Bereichen hin gefahrlos verschieben. Will oder muß man es dem Programm selbst überlassen, welche der beiden Routinen genommen werden soll, dann setzt man MOVE ein. MOVE klärt zunächst, wohin die Reise gehen soll und bewegt dann die Bytes mit der 'richtigen' Routine. Der folgende Code stammt aus dem F83 von Laxen & Perry, CP/M 2.2 mit 8080 CPU.

```
0 \ Block Move Memory Operations
1 CODE CMOVE      (S from to count -- )
2    IP>HL   B POP  D POP  XTHL
3    BEGIN   B A MOV  C ORA  0= NOT WHILE
4       M A MOV  H INX  D STAX  D INX  B DCX
5    REPEAT  B POP  NEXT  END-CODE

6 CODE CMOVE>   (S from to count -- )
7    IP>HL   B POP  D POP  XTHL
8    B DAD H DCX   XCHG   B DAD H DCX    XCHG
9    BEGIN   B A MOV   C ORA   0= NOT WHILE
10      M A MOV H DCX D STAX   D DCX B DCX
11   REPEAT   B POP   NEXT   END-CODE

14 : MOVE   ( from to len -- )
15   -ROT   2DUP U< IF   ROT CMOVE>  ELSE   ROT CMOVE   THEN ;
```

Leaky Sieve

Berichtigung der Berichtigung

Da haben wir den Salat! Im Heft VD8ć II/3 haben wir voller Vertrauen in 'Forth made in USA' den Code von Terry Noyes zur Berechnung von Primzahlen wiedergegeben. Und nun stellt sich heraus, daß auch er nicht ganz richtig lag. Stephen Brault klärte auf.
Hier nun Noyes' Berichtigung seiner Berichtigung.

Noyes' Sieve

```
decimal
8192 constant size

variable flags  size allot

: primes ( - primes)              ( does the primes once)
        flags size 01 fill        ( initialize the array)
        0                         ( prime counter)
        size 0                    ( range/2 of numbers to do)
        DO
         flags i + c@   ( see if prime already)
          IF
          3 i + i + dup i + size <      ( don't go too far)
           IF
             size flags + over flags + i +      ( range of nums to tag)
             DO
              0 i c! dup          ( tag numbers as non-primes)
             +LOOP
           THEN
           drop 1+       ( drop the i used for +loop, increment prime count)
          THEN
        LOOP ;

: sieve
        COUNTER                   ( start counting )
        10 0 DO primes LOOP       ( perform 10 iterations )
        TIMER                     ( stop counting )
        CR . ." primes"           ( print the number of primes )
         9 0 DO DROP LOOP ;       ( clean-up stack )

CR .( Type 'sieve' to execute this benchmark program ) CR
```

Forth Sieve. Uses pointer arithmetic to calculate the number of primes from zero through 16383. To save space and time, it only needs to work with the 8192 odd numbers.

Leaky Sieve

Mr. Ouverson:

In the process of optimizing the sieve benchmark, Terry Noyes has unwittingly rejected a superior algorithm and departed from the de facto benchmark standard. The sieves Mr. Noyes calls "corrupt" are not corrupt at all. They correctly count 1899 primes from 3 to 16383. The **FLAGS** array represents only odd integers, and only odd multiples of primes are "flicked." Fortunately, the Noyes version is easily modified to incorporate the better algorithm, and the resulting version finds the 1899 primes slightly faster than the unmodified version finds 1028.

Stephen Brault
Chandler, Arizona

Mr. Ouverson:

I would like to retract my April letter to you (*Forth Dimensions* VIII/2) and live in shame for the rest of my life.

I had thought that the 0 - 8192 loop in all sieve benchmarks meant they were calculating the number of primes from zero to 8192. Not so. A few weeks after the letter was sent to you, someone pointed out that these sieves were actually finding primes in the range of zero to 16,000+ by looping through the 8192 *odd* numbers in that range.

Oh.

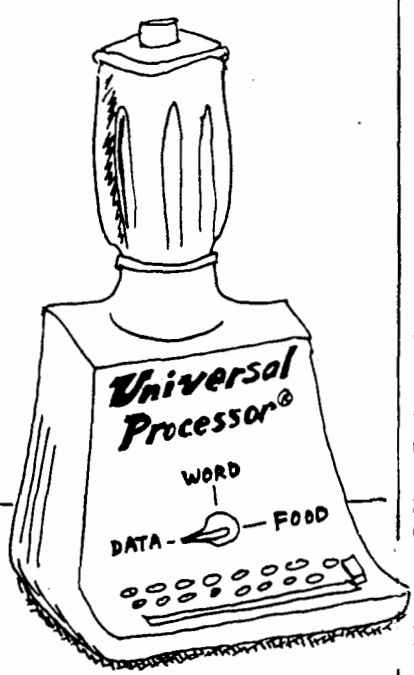
Fortunately, we use identical code to benchmark other Forth systems, so they also received the five percent speed improvement resulting from using the wrong sieve program. I've enclosed the proper Forth sieve with this letter.

Living and Learning,

Terry Noyes
Palo Alto Shipping Company
Menlo Park, California

„ daran denken"

keept it simpel !



An overgeneralized solution.

Rockwell Single Chip Forth Bugs (Fortsetzung Leserbriefe)

Der Forth-Einchipper ist eigentlich ein 6502 Prozessor-System mit RAM, Ports und Forth im ROM an Bord. Forth dient dabei als 'Betriebssystem'. Die Version R65F11 kommt im 40-Pin Gehäuse und bietet 2 freie Ports, die Version R65F12 kommt im 64-Pin Gehäuse und bietet 5 freie Ports. Im Port-A ist jeweils eine serielle Schnittstelle untergebracht, KEY und EMIT wirken über diese. Floppyanschlüße sind vorhanden und werden in gewohnter Weise per BLOCK unterstützt. Ich baue den Chip für Meß- und Steueraufgaben ein.

Das RSC-Forth hat einige Eigenarten. Aus dem AIM-Rockwell Forth für den 6502 stammend, ist es wie dieses eine Mischung aus fig-Forth und 79-Standart-Forth. Die ersten Versionen hatten einige Fehler. Diese wurden aber schnell von Rockwell im Handbuch dokumentiert. So die Probleme des ADMP, FORMAT, IROVEC, I, NOT und CREATE. Bis Version 1.5 traten diese auf, mit V1.7 wurden sie korrigiert. Im Handel ist praktisch nur diese korrigierte Version.

Aber es gibt noch weitere Fehler. So wurde im HWORD der gleiche Fehler wie im CREATE noch mal gemacht. Es wird nicht geprüft, ob bei der Übertragung der Wortköpfe ins HEADER DICTIONARY eine CFA auf die Seitengrenze zu liegen kommt. Dies ist eine Fehlerbedingung für die 6502CPU und bringt sie ganz durcheinander. Daher stürzen Worte, die im normalen Dictionary problemlos gelaufen sind, nun nach dem headerless Compilieren aus 'unerfindlichen' Gründen manchmal (!) ab. Abhilfe: HWORD redefinieren, in das Wort eine Prüfung auf Seitenende FF einbauen. Und es gibt noch weitere Tücken, doch Abhilfe ist möglich. Ich verwende den Baustein zur Zeit oft. Bei Gelegenheit möchte ich eine Liste der Fehler und der Korrekturen zusammenstellen. User, die jetzt schon mehr wissen wollen, können mich gern anrufen.
Adolf Krüger, Schwelm. Tel: 02336-82045

* * *

EINE LöSUNG

Hier haben Sie meine Version für die Aufgabe Nr.2 (VD86 II/4, S.7).
Marc Petremann, 17 allee de la Noiseraie, F-93160 Noisy de Grand, Frankreich. Tel: (0033) 1-4656 3367 6.Feb.87

FORTH MAGAZIN: Besten Dank für die Ausarbeitung. Ich hoffe, genügend Leser beherrschen Französisch. Ansonsten ist der Forthcode in diesem Beitrag auch so verständlich.

* * *

DIE AUFGABE NR.3

Im letzten Heft ging es um Eingabeverfahren. Die neue Aufgabe behandelt die Ausgabe. Die Aufgabe ist wieder als Glossar formuliert. Viel Spa3 beim knobeln.

Glossar

GELDBETRAG (d n x y --)
Druckt den Betrag d mit Vorzeichen und n Vorkommastellen und Komma und zwei Nachkommastellen und Währungsangabe rechtsbündig auf die Zeile x in der Spalte y. Der doppeltgenaue Betrag d liegt in hundertsteln der Währung vor.
Die Eingabe: 123456789 (Pfennige) 10 12 15 GELDBETRAG <ret>
bewirkt in Zeile 12, Spalte 15 die Ausgabe: ..+1234567,89 DM

FORTH

Dans ce propos, les diverses manières de développer une fonction en 83-Standard seront exposées avec pour exemple la création du mot INPUT.

Il existe de nombreuses manières de définir la fonction INPUT en langage FORTH, tout dépendant du niveau de sécurité souhaité et de la nature du résultat obtenu. La première manière, la plus simple, consiste à reproduire la ré-entrance de l'interpréteur externe de FORTH:

```
: DINPUT ( -- d)
QUERY 32 WORD NUMBER ;
```

De ce mot découle son homologue INPUT délivrant un nombre 16 bits:

```
: INPUT ( -- n)
DINPUT DROP ;
```

On pourrait se satisfaire de cette première approche. Mais tout programmeur sait que les utilisateurs ont une fâcheuse tendance à confondre les lettres O et les chiffres 0, quand ce n'est pas les doigts à coté des touches ou carrément l'activation de RETURN avant d'avoir entré un nombre. Or, nos deux premières définitions ne fonctionnent pas correctement si le nombre tapé n'est pas explicitement un nombre. De plus, si on entre un nombre 16 bits (sans point décimal), c'est un entier qui est déposé sur la pile de données. De même, si on entre un entier double précision, se sera un entier 32 bits qui sera déposé sur la pile.

Etant donné qu'il n'est pas question d'exiger de la part de l'utilisateur (qui lui n'en a cure de votre programme, et il a raison car il n'est qu'utilisateur...) de taper:

```
1234567.
```

quand il tapera instinctivement:

```
1234567
```

ce qui bien entendu ne rentre pas dans un format 16 bits. On va donc substituer le mot NUMBER? au mot NUMBER dans la définition de INPUT.

```
: INPUT ( -- d fl)
QUERY 32 WORD
NUMBER? ;
```

L'exécution de INPUT délivre le nombre double précision tapé en réponse à INPUT précédé d'un flag booléen. Ce flag booléen est vrai si la réponse est un nombre, elle est fausse dans les autres cas:

```
réponse à INPUT: 55    empile    55   0   -1
réponse à INPUT: 55.   empile    55   0   -1
réponse à INPUT: TRUC  empile    0    0    0
réponse à INPUT: <ret> empile    0    0    0
```

Encore une fois, cette solution serait satisfaisante avec le petit arrangement suivant:

```
: INPUT ( -- d)
QUERY 32 WORD NUMBER? DROP ;
```

Mais on pourrait se servir utilement de ce petit drapeau booléen pour faire réexécuter l'entrée du nombre tant que celui-ci n'est pas valide:

```
: INPUT ( d --)
BEGIN
   QUERY 32 WORD NUMBER? NOT
WHILE
   2DROP
REPEAT ;
```

Là c'est nettement plus sécurisé. Le petit malin qui prendra le clavier pour un défouloir à phalanges risque fort de rester bloqué avec des entrées numériques du style:

```
12ER4TRE56773JHIGJ67657676GHG76765
```

Mais à toute solution peut venir se greffer un problème. Si votre écran contient des informations, ce genre de manipulation risque de vous obliger à refaire le décor si le défoulement de l'individu appelé utilisateur vient empieter sur l'affichage dont la signification peut être vitale pour la suite des opérations. On prévoiera donc d'effacer et de faire revenir à la position de départ le curseur, ceci tant que l'information demandée n'est pas un nombre:

```
: INPUT ( -- d)
#OUT @
BEGIN
   DUP #OUT ! QUERY 32 WORD NUMBER NOT
WHILE
   2DROP DUP #OUT@ SWAP -
   DUP BACKSPACES DUP SPACES BACKSPACES
REPEAT
ROT DROP ;
```

Ca commence à s'étoffer, non? Bien entendu, votre utilisateur obstiné à confondre les 'Os' et les 'zerOs' ne court plus le risque de se répandre en essais successifs au travers de votre ..... (tableur, traitement de texte, base de données: <- rayer la mention inutile). Mais par contre, rien ne l'empêche, histoire de voir et de vous embèter, d'essayer un nombre du genre:

```
12345676789012445678901234567890134567890913345667
```

Forth n'appréciera guère (*j'ai moi-même planté un serveur* visiblement pas au point avec ce truc, puis j'ai tapé EXP 'LIST', ce qui m'a donné le mode d'emploi, un peu à la manière de HELP avec dBASE...).

On peut envisager de limiter le nombre de caractères à taper:

```
: #INPUT ( u -- d)
#OUT @
BEGIN
   DUP #OUT ! TIB OVER EXPECT
   SPAN@ TIB ! BLK OFF >IN OFF 32 WORD
   NUMBER? NOT
WHILE
   2DROP DUP #OUT@ SWAP -
   DUP BACKSPACES DUP SPACES BACKSPACES
REPEAT
ROT DROP ROT DROP ;
```

Ne vous étonnez pas si le remplacement de QUERY par EXPECT nous oblige à rallonger la définition. Ce complément provient du contenu de la définition de QUERY et est listable par SEE QUERY.

Je vous encourage vivement à procéder à vos propres expériences. Bien entendu, n'hésitez pas à vous servir de DEBUG pour visualiser la trace de l'exécution de votre fonction INPUT.

Maintenant, notre fonction #INPUT est inviolable et sécurisée. Pour vous en convaincre, essayez:

```
DARK 10 10 AT .( ! !) 11 10 AT 4 #INPUT CR
```

Tout ce que vous taperez entre les deux points d'exclamation et qui ne sera pas constitué de quatres chiffres sera obstinément refusé.

Attention, la fonction AT est vectorisée et n'est peut-être pas initialisée correctement sur votre système.

Les définitions précédemment décrites sont spécifiques au FORTH 83-Standard et tournent sur les versions MSDOS et CP/M.

Cet exemple peut certainement être amélioré et adapté:

- lecture à un format spécifique, du genre INPUT-USING

```
" ==/==/==" INPUT-USING
```

De même, le nombre déposé sur la pile peut devenir un nombre 16 bits, flottant ou fractionnaire en fonction du contenu d'un pointeur paramétré par une application spécifique; pensez à la vectorisation... ■

DAS WORT DES MONATS #3

Von Klaus Schleisiek, Hamburg

Im Forthgrundwortschatz stehen Worte wie COUNT EXPECT TYPE zur Verfügung, um einfach strukturierte Stringpuffer zu benutzen. Diese einfachen Puffer sind abgezählte Zeichenketten (counted string), sie werden nur ganz gelesen oder geschrieben (1). Für viele Anwendungsfälle reicht dies auch aus. Daneben werden aber des öfteren Puffer gebraucht, die zeichenweise sowohl gelesen als auch beschrieben werden können. Ein typisches Anwendungsbeispiel ist das Terminalprogramm von B. Pennemann (2). Er benutzt dabei einen zirkulär implementierten Puffer, genannt "Queue" (3). Hier stelle ich eine simple Alternative dazu vor.

Mein Zeichenpuffer behält die Form eines 'counted string'. Der Pufferanfang hat eine feste Adresse, hier liegt auch das 'countbyte' das angibt, wieviel unmittelbar nachfolgende Zeichen (Bytes) zur Zeichenkette gehören. Der Puffer wächst in Richtung steigender Adressen. Gelesen wird der Puffer vorn, und hinten dran wird geschrieben. Das Zeichen, welches gelesen wurde, wird danach wirklich entfernt. Dazu rutscht der String um ein Zeichen nach vorn und wird um ein Zeichen kürzer. Wird ein Zeichen dazugeschrieben, kommt dies einfach hinten dran. Der String wird ein Zeichen länger.

In einem solchen Puffer braucht es gegenüber der 'klassischen' Queue etwas länger, bis ein Zeichen gelesen worden ist. Doch finde ich es angenehm, daß er sich sonst wei ein gewöhnlicher gecounteten String verhält, der sich mit den üblichen Forthworten bearbeiten läßt. Er kann z8 mit COUNT TYPE angezeigt oder auch durch EXPECT aufgefüllt werden. Damit werden die hier definierten Worte APPEND und DETRACT auch bei anderen Gelegenheiten als der Zwischenspeicherung von Zeichen nützlich.

Weil es sich um einen gecounteten String handelt, ist die maximale Groesse des Puffers auf 255 Bytes begrenzt. In der Praxis ist das meißt genug.

```
: more?   ( addr -- n )   c§ ;
: ?string ( n -- n )      dup 255 u> abort" overflow" ;
: append  ( char addr -- )
  dup c§ 1+ ?string  over c!   dup c§ + c! ;
: detract ( addr -- char )
  dup c§ 1- dup 0> and  over c!
  count >r  dup count -rot  swap r> cmove ;

( Beispiel: Implementation einer Queue)
Create Queue  0 c, 255 allot
: q§   ( -- char )
  Queue more? IF  Queue detract exit  THEN  Queue 1+ c§ ;
: q!   ( char -- )  Queue append ;
```

(1) Forth-83 Standard, A Publication of the FORTH Standards Team, August 1983, Mountain View Press Inc., darin:
Definition of Terms, S.10, "string, counted"
Required Word Set, S.25-40
(2) Bernd Pennemann, Queues in Forth,
Forth Magazin 'Vierte Dimension' II/3 (1986), S.27-30
(3) Bernd Pennemann, Ein Terminalprogramm in Forth,
Forth Magazin 'Vierte Dimension' II/2 (1986), S.20-32

volksFORTH beim Sechstagerennen in Bremen

Norfried Mann / Dietrich Weineck

Im Januar war in der Stadthalle in Bremen wieder Sechstagerennen. Zu diesem Radsportereignis waren sie alle gekommen, die großen Namen in diesem Geschäft - von Dietrich Thurau über Danny Clark bis zum Tour de France Sieger Laurent Fignon. Bis zu 13.000 begeisterte Zuschauer pro Abend gaben die Kulisse zu diesem einmaligen Spektakel. Doch eins war in diesem Jahr neu: Zur Auswertung des Rennens wurde erstmals ein Computer, ein ATARI ST, eingesetzt. Das Programm dazu wurde unter volksFORTH entwickelt.

Die Vorbereitungen hatten Mitte letzten Jahres begonnen. Die Punktrichter brauchten Unterstützung bei ihrer Arbeit, die Dokumentation der Rennen sollte verbessert werden . Mit dem Vorsitzenden des Bremer Radsportverbandes, Wilfried Gehrken, wurde die traditionelle Papier-und-Bleistift Arbeitsweise der Renndokumentation und die Bewertungsregeln mit den zugehörigen Berechnungen durchgegangen. Nebenbei wurde die Firma PS-Data schonend darauf vorbereitet, daß sie eine komplette Computeranlage im Januar kostenlos zur Verfügung zu stellen hätte.

Zwei Gesichtspunkte standen ganz oben an beim Entwurf: Unter allen Bedingungen, auch bei plötzlichem Stromausfall, mußte Datensicherheit gewährleistet sein; mit dem Neustart des Programms mußte der letzte Rennstand unmittelbar wieder verfügbar sein . Das Zweite: Fehlbedienungen, in der Hektik einer spannenden Jagd nicht immer zu vermeiden, sollten so weit wie möglich abgefangen werden.

Ursprünglich sollte die Applikation unter dbMAN, einem dBase ähnlichen Programm, entwickelt werden. Doch dies erwies sich schon bald als äußerst unbefriedigend. Die Arbeitsgeschwindigkeit ließ sehr zu wünschen übrig, die Druckeranpassung - Pressemitteilungen sollten direkt vom Computer erstellt werden - wurde zum Problem, da z.B. deutsche Umlaute nicht verfügbar waren usw. Und dann: Keiner in der siebenköpfigen Wertungsmannschaft hatte auch nur die geringste Ahnung von Computern. Kommandozeilen stellten sich als völlig unzumutbar heraus.

Bald schon wurden alle Versuche in dieser Richtung eingestellt. Der Komfort einer GEM--Oberfläche, Anwenderführung über Buttons und Hilfstextfenster erschien als mögliche Rettung, die Anwenderebene simpel, schnell begreifbar und vor allem sicher gegen Fehlbedienungen zu gestalten.

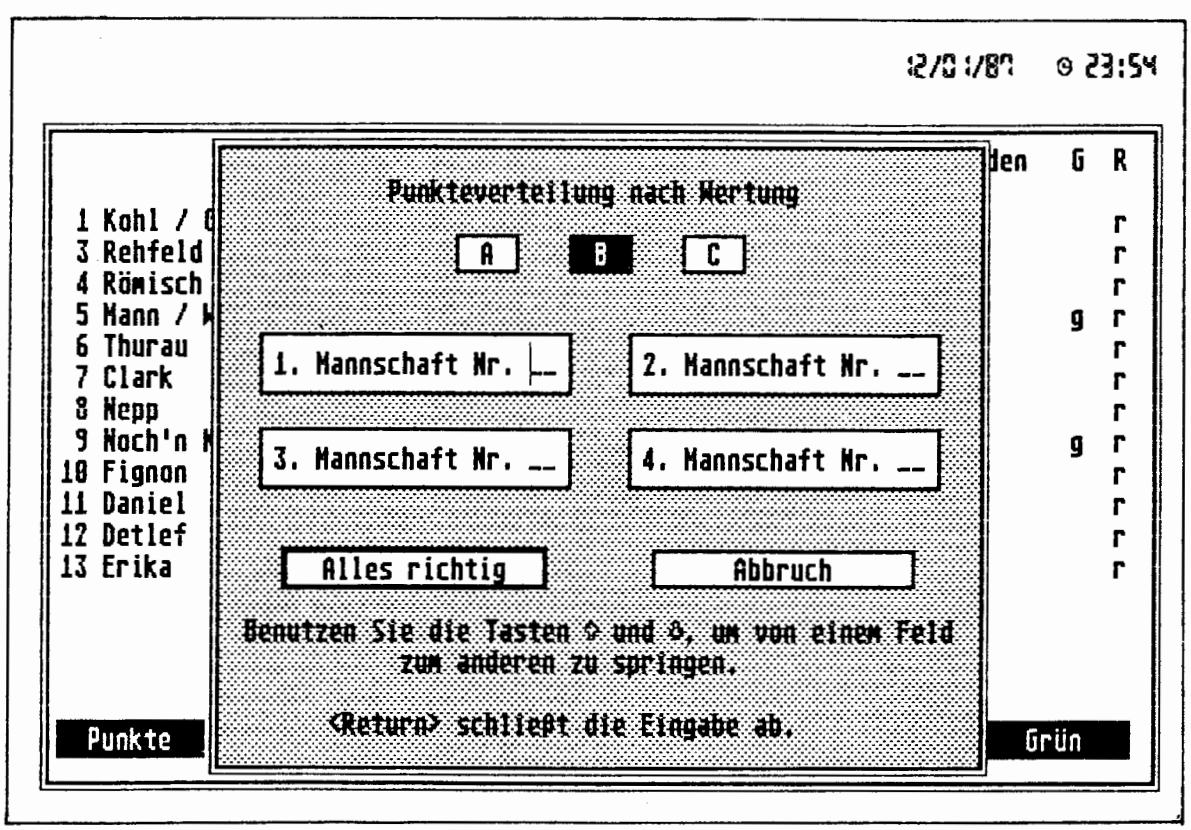
Als Programmiersprache wurde volksFORTH gewählt, dies vor allem aus zwei Gründen. Einerseits sind die Entwicklungszyklen in FORTH sehr kurz, d.h. der Dreischritt Editieren, Kompilieren, Linken schrumpft auf einen Bruchteil der bei anderen Sprachen nötigen Zeit, zum anderen gibt es wohl keine andere Entwicklungsumgebung auf dem ATARI, die ein derart komfortables Debugging ermöglicht wie volksFORTH.

Und es wurde auch deutlich, daß volksFORTH - obschon public domain - keinen Vergleich mit 'professionellen' Entwicklungssystemen zu scheuen braucht: Die GEM--Bibliothek erleichtert den Zugang zu diesem komplexen Teil des ATARI-Betriebssystems, der komfortable Editor beschleunigt die Arbeit, die Verfügbarkeit aller Systemquelltexte erlaubt die leichte Anpassung z.B. des Drucker-Interface. Nebenbei erwies sich der Blockmechanismus von FORTH als geradezu ideal für diese Anwendung. Alle Daten stehen nach einmaligem Einlesen im RAM, trotzdem ist eine hohe Datensicherheit erzielbar, da alle geänderten Daten (und nur diese!) grundsätzlich sofort auf Diskette zurückgeschrieben werden.

Herausgekommen ist eine Applikation für den Rennstand, in der der Rennablauf festgehalten wurde, die die komplizierte Bewertung der Runden- und Punktgewinne vornahm. Jederzeit war der Rennstand auf dem Bildschirm ablesbar, fertige Pressemitteilungen gab es 10 Sekunden nach einer Jagd, wozu sonst gut 10 Minuten herumgerechnet werden mußte. Alles hübsch anzusehen in vielen Fenster, Anleitungen und Hilfsboxen. Natürlich mit einer Zeitanzeige, wozu hat man schließlich Multitasking?

Im laufenden Rennen zeigte sich, was bei den Vorarbeiten an Details dann doch übersehen worden war. Jetzt wurde es für die Programmierer spannend. Das Programm wurde der laufenden Entwicklung im Rennstand angepaßt, neue Wünsche der Jury praktisch über Nacht eingearbeitet. Wieder zeigte sich, wie schnell man in FORTH durch einen Entwicklungszyklus hindurchkommt. Und das Vertrauen der Punktrichter in das Computerprogramm wuchs zusehends. Wurden zu Beginn noch alle Ergebnisse mißtrauisch von Hand nachgerechnet, kehrte sich dies Verhältnis bald um.

Daß die Rennauswertung im nächsten Jahr wieder mit Hilfe des Computers vorgenommen wird, steht fest. Erweiterungen sind geplant, die z.B eine ständige Anzeige des Rennstandes auf einem Terminal im Pressezentrum beinhalten sollen. So steht nach einer spannenden Rennwoche neben dem Team Thurau/Clark ein weiterer Sieger fest: volksFORTH.



(Hardcopy des Atari Bildschirmes)

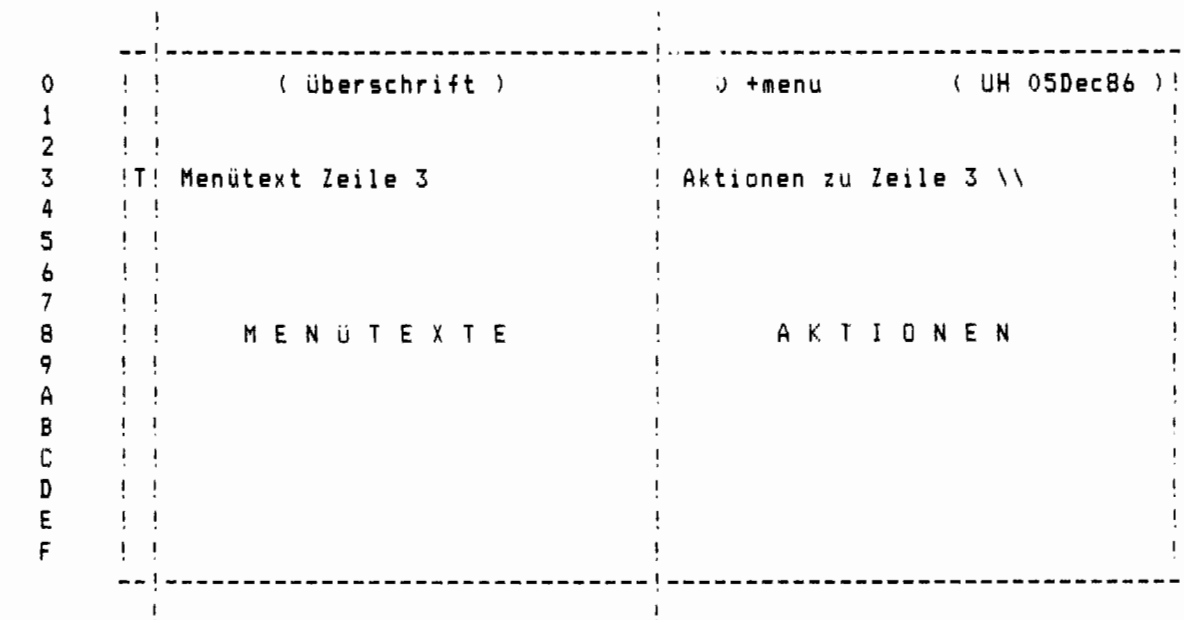
Menüs in FORTH

Ulrich Hoffmann, Kiel Dezember 1986

Das Erstellen von Menüs, also Listen anhand derer ein Benutzer Programmteile auswählen soll, gestaltet sich meist als zeitraubende und unangenehme Routinearbeit. Wünschenswert ist es daher im Besitz eines universellen Verfahrens zu sein, das diese Arbeit erleichtert.
Vorgestellt wird ein Konzept, zur Verwirklichung von Menüs in FORTH mit Quelltext im FORTH-83 Standard.
Die Menuetexte und deren entsprechende Aktionen werden in FORTH-Screens gesammelt. Sie liegen auf dem Massenspeicher, verbrauchen daher auch keinen Platz im Dictionary und können nach Belieben mit Hilfe des normalen Text-Editors verändert werden, ohne die Applikation neu übersetzen zu müssen.

Der prinzipielle Aufbau der hier vorgestellten Menüs sieht so aus:

Ein Screen mit 16 Zeilen á 64 Zeichen. Die linke Hälfte enthält den Menütext,



die rechte Hälfte enthält die Aktionen, und zwar in jeder Zeile die entsprechenden Aktionen für den Menütext der linken Zeilenhälfte. Die erste Spalte jeder Zeile enthält den Wert der Taste, bei der diese Zeile angewählt werden soll. Zum Auswählen (MENU) wird nun die linke Hälfte des Screens angezeigt (SHOW), und dann das Drücken einer Taste abgewartet. Nun wird in der 1. Spalte von oben nach unten nachgesehen (beginnend mit Zeile 1), ob dort die gedrückte Taste zu finden ist (SELECT). Wenn ja, dann soll die zweite Hälfte der entsprechenden Zeile interpretiert werden (CHOICE). Dies geschieht indem die beiden Interpreter-Zeiger BLK und >IN auf den Anfang des zu interpretierenden Textes gesetzt werden, und der Forth-Text-Interpreter die Arbeit verrichtet. BLK enthält die Nummer des momentan interpretierten Blocks (0 für Eingabe von der Tastatur) und >IN den Versatz bezüglich des Anfangs dieses Blocks.
In der zweiten Hälfte der Zeile 0 steht typischerweise ein 0 +MENU, was diesen Screen als Menü kenntlich macht, die linke Zeilenhälfte soll als Kommentar aufgebaut sein.
Wird nun dieser Screen mit LOAD von einem Wort angesprochen, so wird der Text auf ihm interpretiert, zuerst also 0 +MENU. Es veranlaßt nun den oben beschriebenen Vorgang (Anzeige, Auswählen, Zeigersetzen) für diesen Screen. Da wir in der Interpreter-Schleife sind, wird dann aber wie gewünscht an der neu bestimmten Position weiter interpretiert. Am Ende der Aktionen steht meist das Wort \\, das den Rest des Screens zu Kommentar erklärt. Es gibt dann nichts mehr zu interpretieren, der Text-Interpreter wird verlassen und das ursprüngliche Wort weiter ausgeführt.

Auch zusammengesetzte Menüstrukturen lassen sich leicht erzeugen. Dafür gibt es im wesentlichen zwei Möglichkeiten:

1) Baumartig organisierte Menüs, bei denen nach Abschluß eines Untermenüs wieder in das Hauptmenu zurückgekehrt werden soll, werden mit LOAD aufgebaut, da LOAD jedes mal die Interpreter-Zeiger >IN und BL rettet und den Text-Interpreter erneut aufruft.

2) Netzartig organisierte Menüs, bei denen jederzeit zwischen beliebigen Menüs gewechselt werden soll, werden mit MENU aufgebaut, da hierbei nur die Interpreter-Zeiger gesetzt werden.

In gewisser Weise ist also LOAD vergleichbar mit einem "jump to subroutine", MENU mit einem "jump" und \\ mit "return from subroutine" für Menüs.

Zum Abschluß nun ein kleines Beispiel und der Quelltext, hier in standardgemäßem Forth-83:

```
    Page No  1              volksFORTH83 der FORTH-Gesellschaft eV      MENU.SCR

                          1                                                2

0 ( Menu ) Forth-83                          ( UH 05Dec86 ) ( Menu - Shadow )                              ( UH 05Dec86 )
1 decimal   64 Constant c/l             16 Constant l/s      Definiere Konstanten. In den meisten Systemen sind einige von
2   c/l l/s * Constant b/blk        c/l 2/ Constant c/hl     ihnen schon vorhanden.
3 80 c/hl - 2/ Constant margin
4
5 : show ( scr -- )   b/blk 0  DO cr margin spaces           Zeige die linke Haelfte von Screen scr.
6   dup block l + c/hl -trailing type c/l +LOOP drop space ;
7
8 : select ( key scr -- +line )   block under l/s l         Suche in der ersten Spalte von Screen scr nach dem Zeichen key.
9   DO c/l + 2dup c@ = IF leave THEN LOOP swap drop swap - ;
A
B : choice ( scr  -- )   key over select c/hl + >in ! blk ! ;   Lies Tasten und setze Interpreter-Zeiger entsprechend.
C : menu   ( scr  -- )   dup  show  choice ;                    Rufe Screen scr als Menue auf.
D : +menu  ( +scr -- )   blk @ + menu ;                         Rufe den Screen, der +scr Screens entfernt ist, als Menue auf.
E : haeh?  (      -- )   blk @ choice ;                         Wiederhole die Frage fuer dieses Menue nocheinmal.
F : \\     (      -- )   b/blk >in ! ;                          Ueberspringe den Rest des Screens beim Interpretieren.

                          3                                                4

0        ( Beispielmenue )      0 +menu          ( UH 05Dec86 )        ( Untermenue )       0 +menu          ( UH 05Dec86 )
1                               haeh?                                                       haeh?
2 t -> Text sehen               .( HAUPTMENUE ) 0 +menu             t -> Text sehen         .( UNTERMENUE )  0 +menu
3
4 n -> netzartige Menues        1 +menu
5
6 b -> baumartige Menues        1 +load    0 +menu
7
8 f -> Forth                    quit                                f -> Forth              quit
9
A e -> Ende                     .( Ende Hauptmenue) \\              e -> Ende               .( Ende Untermenue) \\
B
C
D
E
F Deine Wahl:                   haeh?                               Deine Wahl:             haeh?
```

Glossar für Menüs Seite 1

+menu (+scr --) "plus-menu"
Wählt das Menü an, das relativ +scr Screens von dem momentanen Screen entfernt ist. Ähnlich wie +LOAD und +THRU

b/blk (-- n) "bytes-per-block"
Eine Konstante, die angibt, wieviele Zeichen in einen Forth-Screen passen. Nach dem Forth-83 Standard sind dies 1024 Zeichen.

c/l (-- n) "characters-per-line"
Eine Konstante, die die Breite eines Forth-Screens angibt. Im allgemeinen 64 Zeichen.

c/hl (-- n) "characters-per-half-line"
Eine Konstante, die die halbe Breite eines Forth-Screens angibt.

choice (scr --) "choice"
Erwartet die Auswahle aus dem Menü scr. Setzt >IN auf die zweite Zeilenhälfte der entsprechenden Menüzeile und BLK auf den Wert scr.

haeh? (--) "wie-bitte"
Wiederholt die Auswahl auf dem momentanen Menü.

l/s (-- n) "lines-per-screen"
Eine Konstante, die die Höhe eines Forth-Screens angibt. Im allgemeinen 16 Zeilen.

margin (-- n) "margin"
Eine Konstante, die angibt, wie breit der rechte Rand sein muß, damit ein halber Forth-Screen auf einem 80 Zeichen breiten Bildschirm zentriert ausgegeben wird.

menu (scr --) "menu"
Zeigt das Menu scr an und ermöglicht die Auswahl eines Menüpunktes. BLK und >IN werden dann auf die neu zu interpretierenden Worte gesetzt.

select (key scr -- +'line) "select"
Sucht in der ersten Spalte von Screen scr mit Zeile 1 beginnend von oben nach unten nach dem Zeichen key. Liefert den relativen Versatz des entsprechenden Zeilenanfangs zum Anfang des Screens. Wird key nicht gefunden, so wird verfahren, als ob es in der letzten Zeile gefunden worden wäre.

show (scr --) "show"
Gibt die linke Hälfte des Screens scr zentriert auf dem Bildschirm aus.

\\ (--) "skip-screen"
Ignoriert den auf dieses Wort folgenden Text bis zum Ende des Screens.

Ulrich Hoffmann, Kiel Dezember 1986

Die "Haeh?"-Falle

Bernd Pennemann, Hamburg

Ein kurzes Programm spart manchmal viel Tipparbeit. So auch das hier vorgestellte. Es erweitert die Tastaturabfrage des *volksFORTH83*, sodaß bei Drücken der UNDO-Taste (auf dem Atari ST) die letzte von der Tastatur aus eingegeben Zeile erneut angezeigt wird.
Bei Tippfehlern reagiert das volksFORTH83 bekanntlich mit "Haeh?" (statt "MSG # 4" oder "SYNTAX ERROR" in anderen Forthsystemen). In diesem Fall drückt man einfach UNDO und korrigiert den Fehler. Dafür kann man wie immer die Cursortasten, Insert etc. verwenden.
Diese Erweiterung funktioniert immer, wenn eine Zeile mit QUERY eingelesen wird, wie es z.B. der Kommando-Interpreter tut.
Sie macht etwas trickreichen Gebrauch von den Variablen #TIB und SPAN . Daher kann man Screen # 1 auch nur auf dem *Atari ST* verwenden. Er sollte aber auch auf allen *CP/M-Computern* funktionieren, da die Tastaturabfrageworte dort einfach abgeschrieben worden sind.
C64-Besitzer brauchen diese Erweiterung eigentlich nicht, denn dort gibt es den Bildschirmeditor. Wer dennoch nicht auf sie verzichten will bzw. ein anderes volksFORTH besitzt, muß den Code auf Screen #2 verwenden, der für Systeme ohne die Möglichkeit, die Eingabezeile zu editieren, bestimmt ist.

```
 volksFORTH-83     FORTH-Gesellschaft eV (c) 1985/86 we/bp/re/ks  LAST_LIN.SCR  Seite 1

                          0                                                2

0               ## LAST_LIN.SCR ##                  bp 1jan87 \ extend keyboard loop für alle anderen              bp 1jan87
1
2 Dieses Programm erweitert die Tastaturroutine um eine Funktion.  Onlyforth
3  Die letzte eingegebene Zeile wird bei Drücken von UNDO wieder   cr  .( Drücke jetzt die Taste, die die  UNDO-Funktion auslösen s
4  angezeigt und kann erneut editiert werden.                      oll :)
5 \\
6 Tastenwert für die UNDO-Taste                                    ! key Constant $undo
7
8 NEWDECODE       pos ist die Länge der eingegeben Zeile           : newdecode     ( addr pos1 key -- addr pos2 )
9    Es gibt nur UNDO, falls noch keine Tasten gedrückt wurden       over 0= IF
0       Drucke Zeile aus, setze Länge neu und fertig                   $undo case? IF   drop #tib @   2dup  type  exit  THEN  THEN
1    Altes Decode ausführen                                          ??decode ;
2
3 ersetze ??... durch die 'richtigen' Worte, die auf Deinem        Input: keyboard   ??key ??key? newdecode ??expect [
4    Rechner vorhanden sind, also z.B.   C64key  C64key? ....      keyboard  save
5

                          1                                                3

0 \ extend keyboard loop für ATARI ST and CP/M          bp 1jan87                                                      bp 1jan87
1 \ mit command line editor
2
3 Onlyforth
4 ( $6100 Constant $undo          \ 1 für Ctrl-A Taste auf CP/M    Wert der Undo-Taste
5
6 : newSTdecode   ( addr pos1 key -- addr pos2 )                   newSTdecode
7    over 0= IF                                                      UNDO geht nur, falls noch keine Tasten gedrückt wurden
8      $undo case? IF  at? )r )r                                     Teste auf UNDO : Falls gedrückt : Rette Cursor
9                      over #tib @   dup span !  type                   drucke TIB aus und setze Eingabelänge neu
0                      r) r) at   exit  THEN   THEN                     setze Cursor zurück ; fertig
1    STdecode ;                                                    Kein UNDO : Weiter wie immer
2
3 Input: keyboard       STkey STkey? newSTdecode $Texpect ;        Neuen Tastatureingabevektor definieren
4
5 keyboard save                                                    Systemeingabe umlenken          FOr immer !
```

Die Fileselector-Box unter volksFORTH83 auf dem Atari ST

Bernd Pennemann, Hamburg

In dem folgenden Artikel wird die Handhabung der Fileselektor-Box, die das Betriebssystem des Atari ST zur Verfügung stellt, besprochen. Ein Beispiel demonstriert die Einbindung der Box in den Editor des volks-FORTH83.

Schlüsselworte : Fileselectorbox, Atari ST, GEM, volksFORTH83

Die Box

In einem früheren Artikel wurden die Hilfsmittel, die das GEM (Graphics Environment Manager) auf dem Atari ST zur Verfügung stellt, erwähnt [1]. Zu den vordefinierten Objekten des GEM gehört die Fileselektor-Box. Das Bild zeigt, wie diese Box aussieht.
Sie dient der Auswahl von Files. Der Benutzer kann diese Box auf verschiedene Arten beeinflussen.
Er kann

- -) durch Anklicken einen Filenamen aus dem Directoryfenster in das Auswahlfeld kopieren.
- -) durch Klicken in den grauen Bereich des Sliders oder Ziehen des hellen Bereichs das Fenster über das Directory bewegen.
- -) durch Anklicken eines Subdirectories oder des Schließsymbols das Indexfeld ändern.
- -) durch Eingabe von der Tastatur aus sowohl das Indexfeld als auch das Auswahlfeld ändern. Dabei haben die Tasten <Esc> <Backspace> <Delete> <Return> sowie die Cursortasten eine besondere Funktion.

Die Box sollte in GEM-unterstützten Programmen verwendet werden. Sie hat den Vorteil, daß man leicht Subdirektories manipulieren kann. Außerdem wird dem Benutzer eine Liste der Wahlmöglichkeiten präsentiert. Die Auswahl der Laufwerke ist mit dieser Box allerdings sehr unbequem. Um einen maximalen Bedienungskomfort zu erreichen, sollte die Box mit sinnvollen Vorbesetzungen der Felder erscheinen. In der Regel wird das Indexfeld das aktuelle Laufwerk, das aktuelle Subdirectory und ein "passendes" Wildcard enthalten (s.u.). Das Auswahlfeld kann z.B. das zuletzt ausgewählte File enthalten. In Applikationen, die mehrere Arten von Files mit Hilfe der Box auswählen, sollte für jede Art von Files eine eigene Vorbesetzung existieren, da Benutzer oft verschiedene Typen von Files in verschiedenen Subdirectories halten.

Beim Aufruf der Box werden zwei Strings mitgeliefert, einen für das Indexfeld und einen für das Auswahlfeld. Im Bild sind das die Strings "A:\GEM*.SCR" und der leere String gewesen. Nach Drücken von "OK" wird er Inhalt der beiden Felder in die Strings zurück übertragen, sie enthalten dann das ausgewählte File und Laufwerk sowie das Directory, in dem sich das File befindet. Um mit diesen Informationen das File öffnen zu können, müssen die Strings erst in für das GEMDos geeignete Stücke zerlegt werden. Der Indexfeld-String besteht aus

bis zu drei Teilen, nämlich :

[Laufwerk ":\"] [evtl: Subdirecories "\"] [Wildcard]

Das Wildcard bestimmt, welche Files des Directories überhaupt in dem Fenster präsentiert werden sollen. Im Bild sind das nur Files, die auf .SCR enden, also Files, die Screens für das volksFORTH83 enthalten. Dieser Wildcard wird nach Aufruf der Box nicht weiter benötigt und in unserem Programm einfach weggeschmissen. Laufwerk und Subdirectory werden mit Hilfe der Worte SETDIR und SETDRIVE an das GEMDos weitergereicht. Anschließend kann auf das File unter seinem Namen zugegriffen werden.

Das Programm

Das folgende Listing bindet die Fileselektor-Box in den Editor ein. Bei Aufruf des Wortes ED wird die Box gemalt, der Benutzer kann ein File auswählen und nach Anklicken von "OK" das File editieren. Es wird immer der Screen 1 des Files präsentiert. Das Wort ED ruft das Wort BOX_USE auf. BOX_USE entspricht in der Wirkung völlig dem Wort USE , daß im volksFORTH83 vorhanden ist. Der entscheidende Unterschied besteht darin, daß man bei BOX_USE nicht den Namen eintippen muß, sondern stattdessen mit Hilfe der Fileselector-Box ein File durch Anklicken auswählt. Diese beiden Worte sind zur Benutzung, d.h. zum Eintippen von der Tastatur, bestimmt. Die anderen Worte stellen die primitiven Funktionen dar, die für den Benutzer nicht so interessant sind.

Für eigene Applikationen, die nur ein File öffnen, daran herummanipulieren und es danach wieder schließen, ist das Wort GETFILE auf Screen 6 gedacht. Es hat den Vorteil, daß es kein Forthfile kompiliert. Die Erklärung des Unterschieds zwischen einem Dos- und einem Forthfile würde den Rahmen dieses Artikels sprengen; bitte schauen Sie in das Handbuch [2].

In den Worten OPEN_FILE und GETFILE wird der Suchpfad des Fileinterfaces (siehe PATH in [2]) mit PATHES OFF auf die Länge Null gesetzt. Damit der Pfad aber nicht verloren ist, wird seine Länge mit PATHES PUSH gerettet. Diese Maßnahmen sind erforderlich, damit der gerade ausgewählte File nur in dem ausgewählten Directory und nicht im gesamten Suchpfad gesucht wird. Andernfalls könnte es nämlich passieren, daß im Suchpfad ein File gleichen Namens aber in einem anderen Directory gefunden wird! Andererseits soll der Pfad nicht vollständig gelöscht werden, damit die VIEW-Funktion des Editors die zu suchenden Files auch finden kann.
Wie schon in [1] erwähnt, muß vor dem Aufruf von GEM-Funktionen, zu denen auch die Fileselektor-Box gehört, GRINIT ausgeführt werden. Soll die Applikation beendet werden, muß vorher ein GREXIT ausgeführt werden. Da der Editor auch eine GEM-Applikation darstellt, brauchen wir uns bei ED um dieses Problem nicht weiter zu kümmern. Im Wort GETFILE kann das jedoch erforderlich sein.

Eine Definition der Worte $ADD und $SUM findet man in [1] oder [2]. Hier wird noch eine zusätzliche Funktion implementiert, nämlich das Anfügen von einzelnen Zeichen an einen String. Dies geschieht mit Hilfe von ADDCHAR .

[1] B.Pennemann, Alert-Boxen unter volksFORTH83 auf dem Atari ST, Vierte Dimension II/4 (1986) S. 19-23

[2] B.Pennemann et.al., Handbuch für das volksFORTH83, 2.Auflage vom 18.12.1986, Selbstverlag (c) 1986, Hamburg

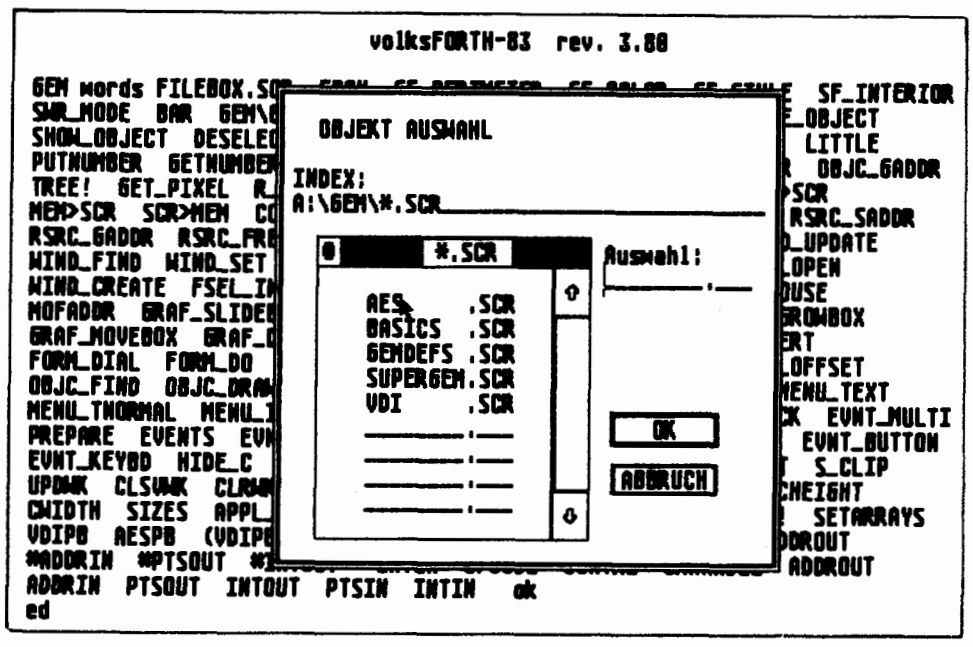


Bild 1 (Hardcopy des Atari Bildschirms)

Die Fehler

Die Fileselektor-Box des GEM hat natürlich auch Fehler. Mir sind zwei bekannt :

-) Beim Editieren des Indexfeldes durch Texteingabe und anschließendes Drücken von "OK" liefert die Routine den Wert für "CANCEL" statt für "OK". Man muß erst den Balken rechts von der Liste der Files anklicken.

-) Wenn man nach Editieren des Indexfeldes den grau schattierten Bereich oberhalb des Fensters anklickt, um das GEM zu veranlassen, die Files entsprechend dem neuen Wildcard zu präsentieren, wird das Wildcard auf "*.*" gesetzt.

volksFORTH-83 FORTH-Gesellschaft eV (c) 1985/86 we/bp/re/ks EDBOX.SCR Seite 1

0

```
0              ## EDBOX.SCR ##                    bp 26dec86
1
2 Dieses File erweitert den Editor um eine Fileselectorbox, die
3  nach Aufruf des Wortes ED erscheint.
4
5 Die Behandlung der Box sollte beispielhaft alle Eventualitäten
6  abdecken und kann direkt in eigene Applikationen eingebaut
7  werden.
8
9
10
11
12
13
14
15
```

1

```
0 \ Loadscreen                                   bp 26dec86
1
2 Onlyforth
3
4 \needs GEM                    include GEM\AES.SCR
5
6
7 Create charhold   1 allot
8
9 : addchar       ( c -- )       \ add char to  string
10    charhold c!   charhold 1 $add  ;
11
12
13 1 4 +thru
14
15
```

2

```
0 \ select the right files                       bp 26dec86
1
2 : getsubdir     ( -- adr len )  \ get actual subdirectory
3    here   [ Dos ]  getdrive 1+  getdir ?diskabort
4    here scan-name ;
5
6 : default       ( adr -- )      \ initialize  fileselectorbox
7    [ Gem ] inpath off  inpath $sum !
8    [ Dos ] getdrive Ascii A + addchar   Ascii : addchar
9            getsubdir ?dup
10           IF   $add  Ascii \ addchar  ELSE  drop  THEN
11   count  $add  0 addchar ;
12
13
14
15
```

3

```
0 \ strip drive and path from INPATH             bp 26dec86
1
2 : strip_wildcard ( -- )          \ get rid the file-wildcard
3   [ Gem ] inpath count +
4   BEGIN  dup c@  Ascii \  -
5   WHILE  1-   REPEAT             1+ off ;
6
7 : set_disk                       \ set drive and directory
8   strip_wildcard          inpath 1+
9   dup 1+ c@ Ascii :  =
10   IF dup c@              [ Dos ] setdrive  2+  THEN
11   [ Dos ]                getdir ?diskabort ;
12
13
14
15
```

7

8

```
                                                 bp 26dec86
Durchsuche nur FORTH und ONLY   ; ich hätte auch
  Gem also Dos also   schreiben können, aber durch explizites
 voranstellen (s.u.) der Vokabulare wird klar, wo bestimmte
 Worte zu finden sind...

ADDCHAR          entspricht $ADD , jedoch für einzelne Zeichen
```

9

```
                                                 bp 17jan87
GETSUBDIR       Das aktuelle Subdirectory wird durch Aufruf
 der GEMDOS-Funktion geholt und bei HERE abgelegt. Die Länge
 wird mit SCAN-NAME festgestellt.

DEFAULT         Initialisiert die Fileselektorbox. Das
 'Index'-Feld wird mit dem aktuellen Laufwerk, dem aktuellen
 Subdirectory und einem Wildcard-String, dessen Anfangsadresse
 auf dem Stack liegt, initialisiert. Der Inhalt des 'Index'-
 Feldes befindet sich ab  INPATH  im Speicher!
```

10

```
                                                 bp 26dec86
STRIP_WILDCARD                  schneidet den Wildcard-String
  hinten wieder ab. Der Rest enthält dann nur noch Laufwerk und
  evtl. Subdirectory.

SET_DISK                        Wertet INPATH aus. Der Wildcard-
  String wird abgeschnitten, das evtl. vorhandene Laufwerk
  gesetzt und schließlich, falls vorhanden, das Subdirectory
  gesetzt.
```

volksFORTH-83 FORTH-Gesellschaft eV (c) 1985/86 we/bp/re/ks EDBOX.SCR Seite 2

4

```
0 \ open box and select a file                          bp 26dec86
1
2 : do_selector   ( -- f )        \ select a file, set direct.
3   [ Gem ] fsel_input   1 -  dup ?exit  set_disk ;
4
5 : (use          ( adr len -- )  \ like USE , but takes a $
6   dup #tib !  tib swap cmove  )in off
7   use  ;
8
9 : open_file     ( -- f )        \ select a file for future use
10   ' *.SCR' default     do_selector    dup ?exit
11   [ Dos ] pathes push  pathes off      \ equiv.  'PATH ;'
12   [ Gem ] insel  count   (use  open ;
13
14
15
```

5

```
0 \ main word : display box, use choice                 bp 17jan87
1
2 \ : desktop       ( -- x y w h )  \ Größe des desktops
3   0 0 [ Gem ] cwidth &80 *   cheight &25 * ;
4
5 \ : savescreen    ( -- )  [ Gem ] desktop scr)mem1 ;
6 \ : restorescreen ( -- )  [ Gem ] desktop mem1)scr ;
7
8 : box_use       ( -- )          \ query a filename
9   savescreen
10      [ Gem ] show_c  open_file  hide_c
11   restorescreen
12   abort' dann nicht !' ;
13
14 : ed           ( -- )                          \ edit screen 1
15   box_use  capacity 1-  1 min  1 ;
```

6

```
0 \ fileselector for Applications                       bp 26dec86
1
2 file selected_file
3
4 : getfile       ( -- )          \ query a file
5   selected_file  [ Dos ] close
6   ' *.SCR' default
7   [ Gem ] show_c   do_selector  hide_c
8           abort' dann nicht!'
9   [ Dos ] pathes push  pathes off      \ equiv.  'PATH ;'
10  [ Gem ] insel count 1+   isfile@
11          [ Dos ] filename  swap   cmove
12  open  ;
13
14
15
```

11

bp 26dec86

DO_SELECTOR malt die Fileselector-Box und wertet INPATH aus. Wird CANCEL gedrückt, so ist das gelieferte Flag TRUE !

(USE wie USE , jedoch wird der Name des zu benutzenden Files auf dem Stack erwartet. Da es keine primitive Form von USE gibt, muß der String erst in den Text Input Buffer (TIB) kopiert werden, von wo ihn USE einliest.

OPEN_FILE Selektiert mit Hilfe der Fileselector-Box Laufwerk, Subdirectory und File. Sucht und öffnet das File. BLOCK LIST V L etc. beziehen sich auf das File.

12

bp 1jan87

DESKTOP liefert Lage und Größe des Bildschirms in Pixeleinheiten. (Normalerweise 0 0 640 400)

SAVESCREEN kopiert den Bildschirminhalt in einen Buffer ..
RESTORESCREEN .. und wieder zurück.

BOX_USE Vor Aufruf der Box wird der Bildschirm gerettet, da das GEM nach Löschen der Box das von ihr belegte Rechteck in neutralem Atari-Grau ausfüllt. Wurde CANCEL gedrückt, so wird abgebrochen.

ED Selektiere ein File; Rufe Editor mit Screen # 1 (falls er existiert) auf.

13

bp 26dec86

In Applikationen gibts nur ein Forthfile. nämlich dieses...

GETFILE ordnet mit Hilfe der Fileselectorbox dem Forthfile SELECTED_FILE ein Dosfile zu. Es wird auch gleich aufgemacht.

DER FORTH MARKT

Vor einem Jahr noch dachte ich, die Hersteller von Forth Systemen würden dazu übergehen, einfache Grundsysteme 'public domain' abzugeben und dazu leistungsstarke Module für alles mögliche zu verkaufen. Es ist anders gekommen. Die Anbieter in USA haben statt dessen 'aufgerüstet'. Angepriesen werden Systeme die schneller, größer, vielfältiger und teurer sind.

So kostet MACH2 für den Mac jetzt 100$ gegenüber 50$ für MACH1 vor einem Jahr; das Handbuch wuchs dafür von 200 auf 500 Seiten und die Unterstützung der Maschine und Ihrer sonstigen Software soll jetzt 100%ig sein. MACH2 für OS-9 Operating System kostet 500$ incl. Linker von Forth an C und umgekehrt. *** Harvard Softworks HSForth für IBM-PC stieg von 280$ auf 395$, jetzt werden Handbücher dabei ausdrücklich mitgeliefert. Und auch hier die gleiche Entwicklung - Forth wird zur kompletten Programmierumgebung ausgebaut. Alle Wünsche werden erfüllt: Es gibt single step, trace, decompile und dissassemble, syntax checking assembler & optimiser, 9 digit software floatingpoint und blitzschnelles 18 digit 8087 math pack; dazu complete music, sound effects & graphics support, effective string funktions, disk flexibility mit traditionellen Forth Screens sektorisiert oder in Files oder in Files von freiem Format, alle mit full screen editor; man bietet I/O redirection und natürlich linearen Adressraum Megabyteweise; und dann auch noch 79 und 83-Standard gleichzeitig. Nun denn, da darfs schon mal was mehr kosten, gell? Soll auf allen MS-DOS Maschinen laufen. *** LMI hat die Palette der unterstützten CPU's erweitert und unterscheidet zwischen einem interaktivem Entwicklungssystem und dem Metacompiler zur Erstellung von Applikationen; dazu werden Crosscompiler angeboten für 8080, Z80, 8086, 68000, 6502, 8051, 8096, 1802 und 6303. Preise werden dabei vorsichtshalber gar nicht mehr genannt. *** Stabil geben sich MVP (175$), BryteForth (100$), MCA (280$) und last not least PolyforthII (immer noch astronomisch). Polyforth informiert, es sei das einzige Forth für realtime Applikationen.

Dazu gekommen sind jetzt die Anbieter von Forth Maschinen und Forth Chips. Es dreht sich viel um den Novix NC4000P, der auf diversen Karten angeboten wird. Chuck Moore persönlich ist dabei. Er und seine 'Computer Cowboys' bieten ein Board für 400$ an. Das Deltaboard von Silicon Composers liegt jetzt bei 495$; ihre 'plug in the IBM-PC' Karte PC4000 kostet dagegen 995$. (mk)

DER KOMMENTAR

Flotte Sprüche begleiten diese Forthentwicklung. Harvard Softworks bringt es auf den Punkt: "Public Domain Software mag billig sein, aber Ihre Zeit ist es nicht. Sparen Sie nicht am falschen Ende. Nehmen Sie das Beste. Nehmen Sie es Jetzt!" Nichts gegen Werbung. Klappern gehört zum Handwerk. Bedenklicher stimmt mich eigentlich die Neigung der Hersteller, ihre Systeme zu den berüchtigten eierlegenden Wollmilchsäuen auszubauen. Das steht einem wichtigen Satz des Forth entgegen, der da lautet "Keep it simple!". Und die Transparenz der Systeme ist auch hin, nirgens ein Wort darüber, daß der Sourcecode - gar noch kommentiert - offengelegt würde (Ausnahme: MVP). Diese undurchsichtigen Alleskönner sind mir unsympathisch. Public Domain ist mir lieber. Ich bitte um die Zusendung von reichlich schonungsloser Kritik! (Michael Kalus)

Aus FORTH DIMENSIONS, Volume 8, Number 4, November/Dezember 1986

A Forth Standard?

Glen B. Haydon
La Honda, California

What is a standard language? Natural languages evolve. Only after a word is used with a specific meaning for a period of time do dictionary editors consider including it. Many words have multiple meanings. Many definitions include examples of their use. Some words become obsolete or archaic. Languages are dynamic. They cannot be set in concrete. There is no such thing as "standard language." Dictionaries only record current usage.

Forth does not differ from any other language. It is evolving. That is the way Charles Moore designed it. He changed his kernel and application utilities almost daily. Many of you are aware that he includes a meta-compiler with most of his applications so he can easily recompile his kernel. It will be interesting to see what direction he takes now that he has cast his kernel in the Novix 4000 chip.

Before going any further, I would like to make a distinction between a kernel and a functional language. The Forth kernel is, in essence, the emulation of a hardware processor. The Novix 4000 is the implementation of a kernel in hardware. On the other hand, Forth as a functional language is built upon a kernel. It utilizes its extensibility to develop an operating system, compiler directives and utilities to solve problems. The functional language is a bridge between application requirements and the kernel. The beauty of Forth is the ease with which the necessary and sufficient functions can be added to a kernel.

The kernel usually includes between sixty and seventy hardware-related functions. There is little problem identifying these, but in actual hardware it has becomes obvious that some of the emulated functions are not optimal. Some of the problems were not anticipated by anyone.

The best example of a problem is the **DO LOOP** structure. The original fig-FORTH implementation requires a range in reverse order. What did the emulation do when a range crossed the boundary of a signed number? Considerable error checking was added to the **LOOP** function in the 79-Standard definition. This proved to be a real boat anchor for speed nuts. This problem was addressed again in the 83-Standard and was improved. In the Novix 4000 the function was replaced by **FOR NEXT**. This function takes a count and decrements it to zero. The hardware requirements for speed dictated that a count-down register would work better and faster. Now the higher-level **DO LOOP** function becomes a part of the functional language, if it is going to be used. So the language changes.

With any Forth kernel, in hardware or emulated, it is an easy job to implement any desired dialect of functional Forth. Each vendor has his own idea of what should be included and what should be excluded. Each vendor provides a slightly different dialect of Forth. Most vendors make their kernel and the basic part of their functional Forth proprietary.

Let us review the public-domain versions of the primitive Forth functions. I started with the first public-domain version readily available — the fig-FORTH Model. The installation manual provided a verbal definition, and the several implementations clarified any possible misunderstandings. The system worked well. I did a moderate amount of programming with it.

Then came the 79-Standard. This was the result of about twenty Forth programmers who addressed some of the "problems" of the fig-FORTH Model. They did several things.

First, they changed the functional definitions for forty words previously defined in the fig-FORTH Model. Some of the changes were simply the use of an alias for the same function. Other changes were of a minor nature. The improvement to the compiler directive **CREATE DOES>** was perhaps the most significant. The ability to write special compiler directives as part of an application program is unique to Forth among computer languages.

Second, the 79-Standard went beyond these functional changes. It included a list of additional "Requirements" for any program adhering to the 79-Standard. In the Standard publication under Section 8, "Use":

"A Forth Standard program may reference only the definitions of the Required Word Set, and definitions which are subsequently defined in terms of these words . . ."

This is patently ridiculous. At the November 1981 FORML Conference, I had an implementation of Forth which contained only the 148 words in the required word set. None of the members of the Standards Team who were there could do anything with the program. No vendor I know of has built a product in complete conformity with the restrictions imposed by the 79-Standard document.

About this same time, Robert L. Smith released and copyrighted a Forth-79 Standard Conversion. This publication consisted of a series of screens which could be loaded on a fig-FORTH Model. They would redefine the necessary forty words in the required word set. He admonishes the user to meet the other requirements of the 79-Standard.

Instead of conversion screens, I modified the compiler source code for the fig-FORTH Model to conform with the 79-Standard Required Word Set and made the additional functions required for a headerless operating system. This was a simple matter of changing a flag for the cross-compiler. I must acknowledge the efforts of Jerry Boutelle, who adapted his cross-compiler for the job and added many of the features. In a period of months two revisions were made. The resulting MVP-FORTH has remained stable for four years! The glossary *All About Forth* provides a reference to the common functions in public-domain implementations of FORTH up to that time.

Added to the MVP-FORTH kernel are a number of utilities and some supplemental definitions that will make this functional Forth almost completely compatible with Leo Brodie's *Starting Forth*. The differences are related to his use of a proprietary product (poly-

FORTH) which was supposed to be 79-Standard. Alan Winfield's *The Complete Forth* provides an excellent alternative tutorial.

Copyright protection of software is a continuing problem. The spirit of fig-FORTH was to put all of the source code and documentation in the public domain, asking only for appropriate acknowledgment. MVP-FORTH adopted the same spirit and placed all of the basic source code and documentation in the public domain. The contents of Volume I in the MVP-FORTH Series, *All About Forth*, are released without restrictions. Each entry includes a functional definition, indicates the source, an implementation, the usage in the MVP-FORTH kernel, an example with a note and a general comment. The general comment includes known differences in function among dialects.

As an interesting aside concerning the significance of copyrights, we had some correspondence with the publisher of *Starting Forth*. They claimed they had a copyright on all of the functional definitions included in their book. They claimed we could not include any of their functional definitions in *All About Forth*. I made an exhaustive study of prior functional definitions of the same words and was able to cite at least one prior definition for each word. Some of those prior definitions were also copyrighted and the publisher had failed to secure a proper release. So much for copyrights.

Other vendors approached the 79-Standard in various ways. Generally, their documentation has been excellent. I have always felt that the more implementations of Forth there are available, the more Forth will be used. By the time these products were on the market, the Standards Team was at it again and came out with the 83-Standard. In my opinion, this was a great disservice to the advancement of Forth.

When the 83-Standard was first available, I made a very careful comparison of the new functional definitions of the Required Word Set with those in the 79-Standard. The number of required words was reduced from 148 to 132. All but five had some change in the functional definitions. No implementations were included as in the original fig-FORTH Model. In fact, some of the adopted functions had never been tested by the team.

In fairness to the members of the Standards Team, they are a dedicated group whose sole objective has been to improve and advance Forth. Many of the changes I found were simply attempts to clarify the wording of the previous standard.

However, they saw fit to change the functional definition of some words without changing the names. **PICK** and **ROLL** are examples. They required that the value on the stack be decreased by one from the value according to the 79-Standard. Thus:

```
: ROT 3 ROLL ; ( 79-Standard )
: ROT 2 ROLL ; ( 83-Standard )
```

When you know of this incompatibility, it is easy to go through your code and change all the values to make it function. But I can see no improvement. Once a convention is adopted, stay with it.

I have no inclination to go through such a careful comparison again. Most of the changes made little difference. However, as has been observed by members of the Standards Team, most people don't do floored division. Forth has enough problems as it is. Why add to them with obscure changes? Forth needs stability.

In addition to the changes in the Required Word Set, similiar requirements to those cited above in the 79-Standard are included in the 1983 document. There is no way to verify the compliance of the many systems purporting now to be 83-Standard.

In the best spirit of Forth, Laxen and Perry have done an implementation of Forth which has become known as F83. It is unfortunate that this has been assumed to be the 83-Standard. It goes far beyond the 83-Standard. It includes nearly 1200 words, and contains many excellent examples of problem solving with Forth. They provide full source code and shadow screens to assist the user. Unfortunately, there is no tutorial such as *Starting Forth* to go along with it. Every Forth programmer should be familiar with the many techniques these master Forth programmers have used.

Among the vendors, Laboratory Microsystems, Inc. has a version which is supposed to comply with the 83-Standard. After finishing his implementation, Ray Duncan wrote a most interesting commentary on the 83-Standard which was published in *Dr. Dobb's Journal*. Other vendors have also implemented what they call 83-Standard Forth. Each of the vendors has excellent documentation for its particular implementation. A variety of other books on Forth are gradually appearing. Each is based on a specific Forth dialect, many of which are proprietary and copyrighted. However, many of the examples and ideas are portable to other Forth dialects with minimal effort. These books are a great help to the intermediate Forth programmer.

Already, some members of the Standards Team are soliciting suggestions for an 87-Standard. It is hoped that the FORML Conference this year will be able to address some of these recommendations.

I would humbly urge those interested in promoting the careful evolution of Forth to take a lesson from the pharmaceutical industry. Only after years in the chemical laboratory and more years of animal testing, are new drugs released for clinical trials. Only after all of the testing and trials have proven satisfactory are drugs finally released for general clinical use.

The Forth Modification Laboratory, FORML, is a fitting place for the laboratory development of modifications. The modifications should first be tried in the laboratory. Favorable results from such work should be submitted to clinical trial in the hands of vendors. Only by acceptance on the part of vendors should changes to a standard be adopted. But then it will not really be necessary: the modifications will have evolved into the common base of the functional Forth language. The standard will be established by common usage.

There is a recurring question of standard libraries. If people would publish their techniques, they could be adapted into most Forth dialects. But there is a reservation on the part of many authors. They want to have some return from all of their efforts. It is only reasonable that they be rewarded for their efforts.

Mountain View Press has found a partial answer to the problem. Namely, though some of their nine volumes are copyrighted, the contents are released for non-commercial use. At least the user can learn from the examples. It is highly likely that he will want to redo any algorithm in his dialect for his own application. Certainly it is not reasonable to let others reprint a book for profit as has been done with Volume 1 of the MVP-FORTH Series.

The current edition of Volume 3 in the MVP-FORTH Series is an example of the evolution of such thinking. The original text was written more than four years ago, and has been actively used since then. In 1985, author Phil Koopman agreed to a restricted copyright releasing it for non-commercial use. Each entry is modeled on *All About Forth* and includes a functional definition, a high-level Forth implementation, an example with a note and a comment.

The local fig-FORTH community still objected: they could not use it because of the copyright, as open as it was. Some in the community have copyrighted their work and made no concessions to non-commercial use. This year, Phil Koopman released his work from copyright, with no restrictions. I hope more Forth authors will see fit to follow his example.

To argue about Forth standards is for those who have nothing better to do. Let Forth evolve like any natural language. Unlike other programming languages, it is easy to start over and meta-compile a new kernel. It is easy to build a new functional system.

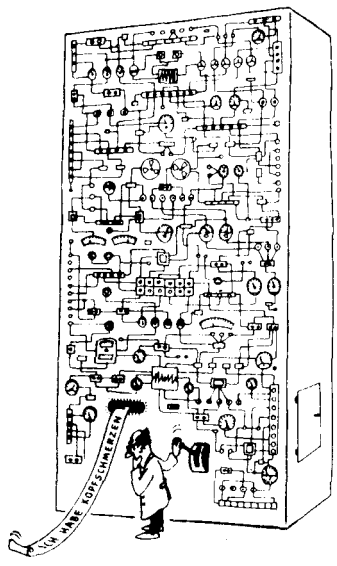
Keep the FORML work active in the background. Encourage regional FORML workshops. As modern microcomputers are becoming more powerful, something more than sixteen-bit address space is needed. How to incorporate this new hardware into the language presents several alternatives. None of the existing public-domain implementations address this problem. The existing standards are simply not ompatible with thirty-two-bit stacks.

Don't let the existing standards be an albatross to the language.

We have an urgent need for a stable language for beginners, for the management team entering new projects and for administrators new to the language. Let common usage provide a dynamic standard to meet the evolving needs. Let everyone participate.

In diesem Zusammenhang ist ein Beitrag von Charles Curley aus dem Jahre 1984 eine vergnügliche Lektüre. Darin versucht er zu zeigen, daß alle "Standards" bisher entgegen ihrem erklärten Ziel die Konfusion nur erhöht hätten. Es seien Produkte von Komitees und das sähe man ihnen auch an. Ähnliche Komitees hätten uns COBOL, ADA und die Inquisition gebracht...

Der Beitrag hat acht Seiten. Gibts beim Prinz Literatursevice. (mk)



Aus FORTH DIMENSIONS, Volume 8, Number 5 Januar/Februar 1987

Screenless Forth

Carl A. Wenrich
Tampa, Florida

Don't get me wrong: I love my Laxen & Perry F83 package. It is the most elegant piece of code I've seen since the last thing I wrote myself. But, somehow, I've never been able to get to the point where I actually enjoy screen editing. Even with everything that's done to help, I still find it tedious.

On the other hand, editing with my **SEE** editor (C Ware Corporation, P.O. Box C, Sunnyvale, CA 94087) is a pure joy. So to have my cake and eat it too, I wrote this little piece for my IBM PC to escape the tyranny of the silent screen. It allows you to create source modules using any ASCII text file editor (even DOS's EDLIN, if you're desperate).

Here's how it works. F83 is set up with four disk buffers of 1024 bytes each at the top of memory. I just redefined that space as a 4K source file buffer. Any programs larger than 4K can be broken down into 4K modules and chained together easily.

Let's take a look at the commands required to implement this screenless Forth system. As you can see by glancing at the listing, there really isn't very much to it. What we have is yet another indication of the power of Forth: you can do quite a lot with very little.

Since some of the new words are duplicates of existing commands, we begin by defining a new vocabulary named **UNSCREEN** to keep them separate. **B/FILE** is the variable that will hold the number of bytes in whatever source file we load. **MOD-BUF** is the address of the 4K buffer at hex F000 where the file will go.

REC-SIZE and **FILE-SIZE** serve as offsets into the file control block; they leave the record-size and file-size addresses, respectively. **OPEN-FILE** is similar to the existing **OPEN-FILE** command, except this one checks to see that the source file is no larger than 4K. If it is, we abort with an appropriate error message; if it isn't, we store the number of bytes in **B/FILE**.

READ-CHAR reads one character from the source file. **READ-SEQ** is the command that reads a sequential source file into the 4K buffer at **MOD-BUF**. The record size is set to one so that the file you need is the file you get. The DTA (data transfer address) is set up at **PAD**. Each time a character is brought in, it

```
     1
 0 \ LOAD BLOCK                                          05APR86CW
 1
 2   ONLY FORTH ALSO DEFINITIONS
 3
 4   WARNING OFF
 5
 6 : NLOAD .S (LOAD) ;   ' NLOAD IS LOAD
 7
 8   2 4 THRU
 9
10
11


     2
 0 \ UNSCREEN REC-SIZE FILE-SIZE OPEN-FILE READ-CHAR     05APR86CW
 1
 2   VOCABULARY UNSCREEN
 3   ONLY FORTH ALSO DOS ALSO UNSCREEN DEFINITIONS
 4   VARIABLE B/FILE   61440 CONSTANT MOD-BUF
 5
 6 : REC-SIZE (S adr -- adr' ) 14 + ;
 7 : FILE-SIZE (S adr -- adr' ) 16 + ;
 8
 9 : OPEN-FILE (S -- ) IN-FILE @ DUP 15 BDOS DOS-ERR?
10   ABORT" Open error"  FILE-SIZE @ DUP 4096 >
11   ABORT" File over 4k"  B/FILE ! ;
12
13 : READ-CHAR (S -- ) IN-FILE @ 20 BDOS DOS-ERR?
14   ABORT" Read error"  ;
15
```

```
     3
 \ READ-SEQ (LOAD) (SOURCE)                              05APR86CW

: READ-SEQ (S -- ) IN-FILE @ DUP REC-SIZE 1 SWAP !
  FILE-SIZE @ 0 DO
    READ-CHAR PAD C@ BL MAX MOD-BUF >IN @ + C! 1 >IN +!
  LOOP ;

: (LOAD) (S -- ) ?DEFINE !FILES OPEN-FILE >IN OFF
  PAD SET-DMA READ-SEQ >IN OFF BLK ON RUN ;

: (SOURCE) (S -- adr len ) BLK @ IF
    MOD-BUF B/FILE @ ELSE TIB #TIB @
  THEN ;


     4
\ (?ERROR)                                               05APR86CW

: (?ERROR) (S adr len f -- ) IF
    TYPE CR SP0 @ SP! PRINTING OFF BLK @ IF
      CR MOD-BUF >IN @ BOUNDS DO I C@ EMIT LOOP
    THEN QUIT
  ELSE
    2DROP
  THEN ;

  (LOAD) IS LOAD   ' (?ERROR) IS ?ERROR   ' (SOURCE) IS SOURCE
```

is compared to **BL**. Printable characters are transferred to **MOD-BUF** and control characters are converted to blank spaces.

(LOAD) fires up the interpreter after the file has been read into memory. It combines the functions of the normal **OPEN** and **(LOAD)** commands. After **LOAD** is revectored to the **UNSCREEN** version of **(LOAD)**, all you have to do is type "**LOAD** filename.ext" and the file will be opened, read into memory and interpreted.

If there are no detectable errors in the source file, you will receive the all-familiar "ok" from the interpreter. Of course, you will have to revector **LOAD**, **SOURCE** and **?ERROR** back to **FORTH** vocabulary versions if you want to play with screens for any reason.

Any detectable source file error will trigger a memory dump from the first byte of the source file buffer **MOD-BUF** to the end of the offending word. This will let you know exactly where the error was found. If a standard message is associated with the error, it will be displayed as well.

(SOURCE) is a slightly modified version of same. **BLK** is now used as a flag which indicates whether the input stream is coming from the keyboard or from the module buffer. **MOD-BUF** supplies the address, and **B/FILE** supplies the number of bytes to be interpreted.

?ERROR is again a modification of the **FORTH** vocabulary's version. But instead of leaving parameters for the **WHERE** command, it dumps the module buffer up to and including the word that triggered the abort. Of course, if you happen to be interpreting from the keyboard, it just flags the error as before.

The only thing left to do now is revector **LOAD**, **SOURCE** and **?ERROR**. Once this is done, you had better not try any screen manipulations unless you first revector back to the **FORTH** versions, because you will probably crash.

But now you are free to load one or more ASCII text files and they will be interpreted just as though they were screen files. To demonstrate how this is done, and how easily files can be chained, here's a little sample session. It assumes that three files of Forth code have already been created. It also assumes that the last two lines of code in **FILEA.BLK** look like this:

```
CR .( LOAD FILEB.BLK )
LOAD FILEB.BLK
```

and that the last two lines of code in **FILEB.BLK** look like this:

```
CR .( LOAD FILEC.BLK )
LOAD FILEC.BLK
```

Now, assuming that the **UNSCREEN** definitions have been loaded, all you have to do is type **LOAD FILEA.BLK** and wait. If the files are large (near 4K), it will go down something like this:

The selected drive will come on and **FILEA.BLK** will be read into memory. After the drive goes off, it will seem as though nothing is happening. Actually, the file is now being interpreted. As soon as the interpreter gets to the end of **FILEA.BLK** you will see **LOAD FILEB.BLK** appear on the screen and the drive will come on again. **FILEB.BLK** will now be read in and interpreted. **LOAD FILEC.BLK** will then appear, and **FILEC.BLK** will be read in and interpreted.

At this point, you are ready to run your application. You may leave your image by entering "**SAVE-SYSTEM** filename.com" and boot right into it by entering "' program **IS BOOT**".

In any case, I think you will find that editing source modules will become a bit more enjoyable. And as an added bonus, you will find they take up a great deal less disk space — screens are notorious disk hogs because of all the white space they require. As a result, you will probably be more likely to structure (indent) your Forth source code the way it was intended, instead of squeezing it into that 16x64 box like most of us.

Die Fachgruppen der Forth Gesellschaft eV - eine Übersicht

Seit der Gründung der Forth Gesellschaft eV gab es neben der regionalen Konzentration von Aktivitäten in Form der Lokalen Gruppen, auch eine Förderung von inhaltlichen Aktivitätskonzentrationen in Form der sog. Fachgruppen. Mit der im letzten Jahr erfolgten Umstrukturierung der Forth Gesellschaft eV hat sich auch die Arbeitsweise der Fachgruppen geändert. Außerdem sind einige neue entstanden, so daß es nicht nur für frischgebackene Mitglieder wichtig ist, noch einmal an dieser Stelle Sinn und Arbeitsweise der Fachgruppen dargelegt zu bekommen.

In den Fachgruppen sollen spezielle Themen, seien es Teilaspekte von Forth oder Anwendungsgebiete, in denen Forth eine Rolle spielt, konzentriert bearbeitet werden. Das kann in vielfältiger Form geschehen, z.B. durch konkrete Projekt-Arbeit an einer Implementation oder auch duch Zusammenstellen und Analyse der bisherigen Ergebnisse in einem Arbeitsgebiet.

Eine Gruppe setzt sich zwanglos aus den an einem Fachgebiet interessierten Forth Gesellschafts Mitgliedern zusammen. Dabei funktioniert die Kommunikation innerhalb der Gruppe nach folgendem Grundprinzip. Jeder, der einer Fachgruppe beitreten möchte, sei es weil er aktiv mitarbeiten will oder weil er nur die innerhalb der Gruppe in Umlauf befindlichen Informationen erhalten möchte, schickt dem Fachkoordinator eine Anzahl von frankierten Rückumschlägen. Die Anzahl hängt von der Aktivität und Arbeitsstruktur der Gruppe ab. Aufgabe des Fachkoordinators ist es nun, in mehr oder weniger regelmäßigen Abständen, die so bei ihm registrierten Gruppenmitglieder mit Informationen zu versorgen. Diese können, müssen aber nicht aus eigener Feder stammen. Im Idealfall ist der Koordinator 'nur' verantwortlicher Verteiler (und eben "Koordinator") für Informationen, die von den Mitgliedern der Gruppe erstellt, gesammelt und bearbeitet werden.

Abhängig von der jeweiligen Struktur und Arbeitsweise einer Fachgruppe koennen natürlich neben der 'gelben Post' auch weitere Arten der Kommunikation möglich sein. Das Einrichten eines öffentlichen oder privaten Zweiges innerhalb des forthTREE's wäre z.B. denkbar.

Wer sich für eines der anschließend aufgeführten Gebiete interessiert, ist also aufgerufen, sich in aktiver oder passiver Form an einer Fachgruppe zu beteiligen. Zuständiger Ansprechpartner ist der jeweils aufgeführte Fachkoordinator.

Initiativen für die Gründung neuer Fachgruppen sind erwünscht! Bei Interesse bitte an das Forth Büro wenden.

Ich hoffe, daß sich durch die Neugestaltung der Fachgruppen für die Mitgliedern mehr Möglichkeiten für Aktivitäten ergeben, als das bisher durch den im wesentlichen auf die VIERTE DIMENSION und den forthTREE beschränkten Informationsfluß möglich war.

Marco Pauck

Fachgruppen der FG

VOLKS/ULTRA-FORTH
Das Public Domain Forth der Forth Gesellschaft eV
Bernd Pennemann, Steilh. Str.46, 2000 Hamburg 60, Tel.: 040-6900539

GRAPHIK
Graphik, Animation und Bildverarbeitung
Marco Pauck, Friedensallee 92, 2000 Hamburg 50, Tel.: 040-3900139

32-BIT-FORTH
32-bit Forth, Implementationen und Applikationen
Robert Jones, Venloer Str.14, 5144 Wegberg, Tel.: 02434-4579

FORTH-MASCHINEN
Forth-Chips, RISC-Architekturen und Forth-Maschinen
Roland Steck, Ohylstr.33, 6100 Darmstadt, Tel.: 06151-661192

KÜNSTLICHE INTELLIGENZ
Künstliche Intelligenz, Experten-Systeme, LisP und ProLog in Forth
Ulrich Hoffmann, Harmsstr.71, 2300 Kiel 1, Tel.: 04307-6869

DATENKOMMUNIKATION
Datenfernübertragung, internationale Netzwerke
Klaus Schleisiek, Steinberg 81a, 2000 Wedel, Tel.: 04103-13255

NUMERIK
Numerik und Floating Point für Forth
Jens Storjohann, Parkstr.23, 2800 Bremen 1, Tel.: 0421-3499375

KLEINKRAM LADEN /

```
: VONBIS  ( n von bis -- f )  >r over > swap r> or not ;
( richtig falls von <= n <= bis, sonst falsch)

: VONBISAN  ( n von bis -- f ) 1- vonbis ;
( richig falls von <= n < bis, sonst falsch)
```

Hieran ist interessant, daß aus einer Phrase zu einem Grundproblem die ähnlichen Fälle ganz einfach hervorzubringen sind.

Wurde grundsätzliches klassisch gelöst, gehört es in eine Wortsammlung. Beispiele über Abwandlungen gehören dazu. So entsteht nach und nach ein Wörterbuch des Forth. Als Textfile. Im Forthsystem selbst haben diese Worte nichts zu suchen, sie blähen es nur unnötig auf. So etwas wird bei Bedarf dazugeladen. Das Volksforth kennt den Befehl NEEDS um ganze Files dazuzuladen. Ich wünsche mir ein Wort, daß sich einzelne Definitionen aus einer Sammlung holt. Wer schreibt es? (mk)

FORTH GRUPPEN

- Darmstadt: Andreas Soeder, 06257-2744
  Treffen an der VHS an einem Mittwoch in der Mitte des Monats.

- Hamburg: Bernd Pennemann, 040-6900539
  Treffen jeden vierten Samstag im Monat ab 16:00Uhr in der Berufsfachschule für Radio- und Fernsehtechnik, Eimsbüttelerstr.64-66.

- Karsruhe: Michael Weiss, 0721-854994
  Treffen jeden dritten Mittwoch im Monat ab 19:00Uhr im Jugend- und Begegnungszentrum, Krohnenplatz.

- München: Heinz Schnitter, 089-3103385
  und Christoph Krinninger 089-7259382
  Treffen jeden vierten Mittwoch im Monat 19:30Uhr im Vereinsraum 1 im Bürgerhaus Unterschleißheim am Rathausplatz (S-Bahnhaltepunkt S1 Unterschleißheim)

- Wuppertal: Michael Kalus, 02336-82204
  Treffen jeden vierten Freitag im Monat ab 20:00Uhr im Bahnhof Ottenbruch, Funckstrasse, W'tal-Elberfeld.
  (Winterpause noch bis April)

FORTH KONTAKTE

Braunschweiger Raum: Eckhard Heyne, 05352-58087
Freiburger Raum: Markus Gimbel, 07641-42819
Frankfurt: Dr. Uwe Gerz, 06103-6072
Hannover: Eckhard Heyne, 05352-58087
Mainz: Thomas Jung, 06131-689608
Moers: Hans Chrapia, 0203-3793274
Paderborn: Thomas Asche, 05251-26496
Rhein/Neckar Raum: Thomas Prinz, 06271-2830
Stuttgart: Hans-Peter Diettrich, 0711-6407419

Villach österreich: Heinz Klambauer, 04242-33566

FORTH GRUPPEN EUROPA

- Belgien FIG Chapter: Luk van Loock, Tel: 03-658-6343
  Lariksdreff 20, B-2120 Schoten
  Und: Jean-Marc Bertinchamps, Tel: 071-213858
  Rue N. Monnom, 2, B-6290 Nalinnes
- Englisches FIG Chapter: Keith Goldie-Morrison
  Bradden Old Rectory, Towchester, Northhamptonshire, NN128ED
- Frankreich FIG Chapter: Jean-Daniel Dodin, Tel: (16-61)44-03
  77 Rue du Cagire, F-31100 Toulouse
  Und: Petremann, Association Jedi,
  8, Rue Pourier de Narcay, 75014 Paris
- Holland FIG Chapter: Adriaan van Roosmalen, Tel: 31-76-713104
  Heusden Houtsestraat 134, NL-4817 We Breda,
- Irland FIG Chapter: Hugh Doggs, Tel: 051-75757 od. 051-74124
  Newton School, Waterford
- Italien FIG Chapter: Marco Tausel, 02-645-8688
  Via Gerolamo Forni 48, I-20161 Milano
- Schweiz FIG Chapter: Max Hugelshofer, Tel: 01-833-3333
  Stationsstrasse, CH-8306 Bruttisellen,
  Und: Renato Mauerhofer, Cassinelle 17, CH-6982 Agno

Hallo, liebe Freunde des Forth! Das Jahr 1987 hat begonnen und es ist nun wieder an euch, dafür zu sorgen, das der Vorhang aufgeht für noch mehr WORDS über FORTH. Was ist zu tun? Anrufen, schreiben, vorbeikommen in der FORTH GESELLSCHAFT, im FORTH BÜRO, in der LOKALEN GRUPPE und FORTHschritte diskutieren. FORTH Kerne sind in der public domaine, viel kann hier schon ausprobiert werden. F83 und VOLKSFORTH und die FORTHMASCHINEN und FIG FORTH und und und... Hier im FORTH MAGAZIN kann veröffentlicht werden und mitgeteilt werden. Hier wird gezeigt und gelernt. Hier ist FORTH transparent. Hier können alle Fragen gestellt werden und hier werden alle Fragen beantwortet. Hier kann man Mitglied werden. Schnell den Beitrag bezahlt, damit der Vorhang 1987 ganz aufgeht für noch mehr FORTH WORDS ok