VIERTE DIMENSION

7. Jahrgang Nr. 4 Dezember 1991

euroFORML '91

Eingabeeditor

Infix-Konverter

Swapping Data

FORTH MAGAZIN

7,50 DM

FORTH-Magazin
Vierte Dimension

Nr. 4 Dezember 1991

Inhalt

EDITORIAL

"Wenn Du ein Forth kennst – kennst Du ein Forth"

Dieses geflügelte Wort der Forth-Gemeinde spielt auf die zum Teil erheblichen Unterschiede zwischen verschiedenen Forth-Systemen an. Diese Unterschiede sind einer der Hauptgründe dafür, daß "das Rad immer wieder neu erfunden werden muß". Mit dem kommenden ANSI-Standard soll etwas an dieser Situation geändert werden. Der "Draft Proposed Standard" dpANS-2 wurde der Öffentlichkeit vorgestellt und befindet sich gegenwärtig in der öffentlichen Begutachtung, die noch bis Ende Februar 1992 läuft. Näheres zu dpANS-2 findet sich im Beitrag von Michael Kalus, der dankenswerterweise den euroFORML-Konferenz-Vortrag von Mitch Bradley über ansFORTH übersetzt hat. Was es dort sonst noch zu erleben gab, ist meinem Bericht über die euroFORML'91 Konferenz zu entnehmen.

Solange (dp)ANS-Systeme noch nicht weit verbreitet sind, müssen wir uns damit begnügen, Ideen auszutauschen und "das Rad nachzuempfinden", wenn wir nicht gerade das System benutzen, das im jeweiligen Artikel angesprochen wird. Damit das Nachempfinden leichter wird, haben die Autoren dieser Ausgabe wieder einmal in ihre Werkzeugkästen gegriffen und dabei manch wirklich nützliches Werkzeug herausgezogen.

Heinrich Hohl berichtet über seinen "intelligenten Editor für Eingabefelder", dessen Ursprungsversion schon einige Jahre alt ist. Die hier vorgestellte Fassung ist eine Überarbeitung, in der die Erfahrungen beim Einsatz dieses Tools eingeflossen sind.

Für alle, denen der Speicher zu knapp wird, stellt Frank Stüß vor, wie eine dynamische Pufferverwaltung in Forth zu realisieren ist.

Eines der Hauptvorteile von Forth ist die Verwendung der Postfix-Notation, die zu den Schlüsseln der Erweiterbarkeit von Forth gehört. Zuweilen ist Postfix jedoch aber auch hinderlich, etwa wenn Formeln umgesetzt werden sollen. Das Werkzeug um auch dieser Notwendigkeit nachzukommen, findet im Beitrag von Bernd Paysan seine Realisierung.

Jörg Staben verrät wie man volksFORTH dazu überredet, den MSDOS-Kommando- Interpreter COMMAND.COM zu vergessen und statt dessen mit dem vornehmeren 4DOS Vorlieb zu nehmen.

Die Implementierung des F-PC-Systems wird von Arndt Klingelnberg beleuchtet. Gegenüber klassischen Forth-Systemen (FIG, F83, volks) setzt F-PC spezielle Threading-Techniken ein, um möglichst viel des unter MS-DOS zur Verfügung stehenden Speichers für Definitionen nutzen zu können. Dank Intel findet dadurch die Segmentiererei nun auch bei Forth Einzug.

Allen Lesern wünsche ich viel Spaß beim Studieren der Artikel und ein erfolgreiches Jahr 1992. Ulrich Hoffmann

IMPRESSUM

FORTH-*Magazin "Vierte Dimension"*

Herausgeber:
FORTH-Gesellschaft e.V.

Editor, Satz und Layout:
Peter Dinies, Metzstraße 38, W-2300 Kiel 1, Tel. 0431/1 32 39

Beirat:
Ulrich Hoffmann, Jens Storjohann, Michael Kalus

Illustrationen
Rolf Kretzschmar

Kontaktadresse:
FORTH-Büro, Postfach 1110, W-8044 Unterschleißheim, Tel. 0 89/3 17 37 84 oder FORTH-Mailbox, München, Tel. 0 89/871 45 48 8N1 "Konferenz Vierte Dimension".

Quelltextservice:
Der Quelltext von Beiträgen, die mit dem Diskettensymbol gekennzeichnet sind, ist auf der Leserservice-Diskette zur jeweiligen Ausgabe oder in der FORTH-Mailbox zu finden.

Redaktionsschluß:
Erste Woche im mittleren Quartalsmonat
Erscheinungsweise vierteljährlich

Auflage:
Ca. 1.000

Druck:
Buch- und Offsetdruckerei Bickel & Söhne, Frankfurter Ring 243, W-8000 München 40

Preis:
Einzelheft DM 7,50, Abonnementpreis DM 40,-, bei Auslandsadresse DM 45,- inklusive Versandkosten

*Die euroFORML vom 11.-13. Oktober 1991 in Marienbad (Mariánsé Láznê) / CSFR bestätigte ein weiteres Mal, daß sie **das** internationale europäische FORTH-Ereignis geworden ist.*

euroFORML'91

Ulrich Hoffmann

Frühere euroFORML-Konferenzen waren von starker Beiteiligung aus Amerika (85) oder Deutschland bzw. England selbst geprägt (86/87). Die Konferenzen der letzten Jahre (89-91) zeichnen sich aber durch wirklich internationale Beteiligung aus. So gab es 1991 Teilnehmer aus Belgien, Bulgarien, China, Deutschland, Estland, Frankreich, Großbritanien, Norwegen, Schweden, Schweiz, UdSSR und aus den USA.

Dieses Jahr fand die Konferenz in Marienbad statt. Die Konferenzleitung wählte diesen Ort und verzichtete auf die normalerweise zu erzielenden Gewinne, um den Osteuropäischen FORTHlern die Teilnahme finanzierbar zu machen. Marienbad ist ein 14000 Einwohner großer Kurort, eine historische Stätte für Trink- und Badekuren mit zahlreichen Mineralquellen. Es ist vergleichbar mir dem nur wenige Kilometer entfernten und bekannteren Karlsbad, nur um vieles schöner. Die euroFORML-Konferenzräume lagen im Casino etwa fünf Minuten Spaziergang entlang des wunderschönen Kurparks vom Hotel Bohemia entfernt, indem die Teilnehmer und Teilnehmerinnen untergebracht waren. Ein Besuch in Marienbad erinnert ungemein an eine Reise in das alte Europa der Kaiserzeit, dies kombiniert mit einer High-Tech-Computerkonferenz wird zu einer extrem faszinierenden Mischung.

Die Veränderungen in den osteuropäischen Ländern haben es möglich gemacht, daß immer mehr Forth-ProgrammiererInnen von dort zur euroFORML kommen konnten. Dies gab den anderen Teilnehmern die Gelegenheit sich hautnah über die jüngsten dortigen Geschehnisse zu informieren. Der erste Abend war dann auch ein erster Höhepunkt. Mit eindrucksvollen Berichten über die wirtschaftliche und politische Situation in Osteuropa, sowie drastischen Schilderungen über die Anreise beeindruckten Sergei Baranoff (UdSSR), Chavdar Levkov (Bulgarien) und Januus Pöial (Estland). Die dort aufgeworfenen Fragestellungen blieben bis zum Rest der Konferenz brennende Diskussionsthemen. Ein weiteres wichtiges Ereignis des ersten Abends war die "Intronisation des Zeremonienmeisters". Roy Goddard wurde, durch seine außergewöhnlich komischen Darbietungen während der vergangenen Konferenzen, die Ehre zuteil auf Lebenszeit den euroFORML-Wettbewerb ausrichten zu dürfen. Für dieses Jahr regte er an, Forth-Lieder zu dichten, die dann zum Ende der Konferenz hin prämiert werden würden. Während der Krönungszeremonie rezitierte Marina Kern Forth-Gedichte gemäß des neuesten von ihr entwickelten Pooh-Forth-Pronounciation-Standards.

Während der einzelnen Sitzungen wurden wieder wie gewohnt Vorträge über die neuesten Entwicklungen auf dem Forth-Sektor gehalten. Gibt es in den USA eine Konferenz für die kommerziellen Anbieter (Rochester) und eine für den Club der Forth-Gurus (FORML), so spricht die euroFORML auf einer viel breiteren Basis an. Berichte über Forth-Applikationen gehören genauso zum Programm wie das Vorstellen von neuesten Implementierungstechniken.

So stellte N. J. Nelson beispielsweise seine Forth-Anwendung vor, bei der ganze Groß-Wäschereien via Forth gesteuert werden, wohingegen Graham Dolding zeigte, was zu tun ist, um Forth unter MS-DOS zu einem TSR-Programm zu machen. Eines der zentralen Forth-Themen der Konferenz war der kommende ANSI-Standard. Mitch Bradley sprach dazu die jüngsten Entwicklungen an (vgl. Übersetzung von M. Kalus auf Seite 32 in diesem Heft). Spektakulär war es, zu erleben, wie Mitch Bradley zur Beendigung seines Open-Boot-Firmware Vortrags den Open-Boot-Song nach der Titelmelodie der Zeichentrickserie "Die Feuersteins" zusammen mit dem Auditoriums anstimmte. Bengt Grahn stellte SMAN vor, ein sprachenunabhängiges System, das zur modularen Verwaltung von Dokumenten, z.B. auch von Quellcode, eingesetzt werden

Fortsetzung auf Seite 18

Jahrestreffen

und

Ordentliche Mitgliederversammlung

der

Forth Gesellschaft e. V.

Wann:

Freitag ab 19 Uhr bis Sonntag
27. bis 29. März 1991

Wo:

Congress-Hotel
O-2550 Rostock 22
Leningrader Str. 45

Tagesordnungspunkte

1. Rechenschaftsbericht des Direktoriums
2. Kassenbericht und Bilanz 1991
3. Aussprache, Entlastung, Wahl des Direktoriums
4. Neue Ziele der Gesellschaft
5. Verschiedenes

Ergänzende Tagesordnungspunkte sind bis Ende Februar über das FORTH Büro dem Direktorium einzureichen.

Das alljährliche Treffen beginnt am Freitag, dem 27. März mit Vorträgen zu Forth in Wissenschaft und Industrie. Veranstalter des Treffens ist die Uni Rostock gemeinsam mit FORTecH Software GmbH.

Die Tagung endet am 29. März nach der ordentlichen Mitgliederversammlung der Forth Gesellschaft e.V. Die Teilnahme daran ist kostenlos. Bitte kommen sie recht zahlreich und nehmen Sie sich auch Zeit für unsere Mitgliederversammlung, damit diese beschlußfähig wird Es stehen wichtige Themen auf der Tagesordnung, die die nächsten Ziele der Gesellschaft festlegen sollen.

Zur Anmeldung verwenden Sie bitte das dieser Ausgabe beiliegende Formular oder wenden Sie sich an das Forth Büro in München.

Leserbriefe

Ich würde mich gerne als Koordinator für ein kleines und ein größeres Betriebssystem, geschrieben in Forth zur Verfügung stellen.
Wer nicht weiß, welche Programme man in Forth schreiben kann, der soll mich kontaktieren. Ich weiß soviele Programmideen, das sie für 1000 Programmierer langen. Meine Forth Mitgliedsnummer ist 2700, Meine Adresse Ludwig Braun junior PF 1236, W-8425 Neustadt-Donau.

Wieso gibt es kein Betriebssystem, das in Forth geschrieben ist?

Fig Forth, Forth 79 und Forth 83 haben nur ein kleines unkomfortables Betriebssystemverhalten. Wieso gibt es noch kein größeres Public Domain Betriebssystem, das in Forth geschrieben ist.

1. Ebene: Ein Forth 83 Kern, der um Dos Routinen erweitert wurde, mit einen Forth als Komandointerpreter.
2. Ebene: Eine grafische Benutzeroberfläche, die ebenfalls in Forth geschrieben wurde.
3. Ebene: Verschiedene Sprachen, die alle in Forth geschrieben sind.
4. Ebene: Anwendungen

Ausgehend von diesen 4 Ebenen, kann man diese Strukur erweitern um Multitasking, Multiuser, Netzwerkfähigkeit, Coprozessorfähigkeit, Multiprozessorfähigkeit und anderes.

Sehr gut dafür geeignete Prozessoren wären nach meiner Ansicht: RTX 2000, FRP 1600, eventuell 680x0.

Die 1. Ebene muß man noch unterteilen in Assembler Teil, Debugger, Tracer, Forth Interpreter, Dateiverwaltungssystem, Hardwareroutinen, und eventuell obengenannte Fähigkeiten. Selbstverständlich soll ein solches System seine Unkosten auch wieder hereinbringen. Deswegen ist an einen Unkostenbeitrag für Handbücher und Lehrbücher gedacht. Der Quellcode kann ebenfalls zur Verfügung gestellt werden. Und Firmen die dieses in Forth geschriebene Betriebssystem samt Zubehör an ihre Rechner anpassen wollen, können dies gegen Geld tun.

Noch etwas zu den verwendbaren Forth Interpretern, Es sollte ein Standard Forth sein. Ansi Forth oder Forth 83 wären die richtigen dafür.

Ich würde mich als Koordinator für ein einfaches Forthbetriebssystem zur Verfügung stellen. Und zwar für 680x0 (ATARI, AMIGA) und 80386 Systeme. Meine Adresse Ludwig Braun junior PF 1236 8425 Neustadt

Mach's Fensterl auf, MSDOS!

Portierungen von Atari-ST-Programmen auf MSDOS-Rechner scheitern leider an der inkompatiblen Graphikumgebung.

Des öfteren habe ich in der VD von dem Wunsch gelesen, es mögen Programme mit einem Vokabular erstellt werden, das sowohl MSDOS- als auch Atari-Rechner verstehen.

Wie ebenso erkannt, hat die Portabilität aber ihre Grenzen in der Graphikeinbindung. Der Atari-Anwender ist hier durch sein im Rom vorliegendes Graphik-Paket GEM recht verwöhnt.

Als Lehrer, der einen Atari ST besitzt und der für seine Schüler ab und zu eine kleine Applikation schreiben möchte, stehe ich aber vor dem Problem, daß in der MSDOS-Welt die Fenster-Umgebung keineswegs Industriestandard ist, weder bei den väterlichen Rechnern meiner Schüler noch gar in den Schulen selbst.

Leider hat sich der Atari ST nicht als Schul-Standard durchgesetzt, jedenfalls nicht in Bayern. In den höheren Etagen der Entscheidungsträger setzte man auf den Industrie-Standard. In unserem Gymnasium wurden z.B. AT-Rechner angeschafft ohne Maus und ohne Festplatte, mit krümeligem Bernstein-Monitor und genau einem 5 1/4 Laufwerk (dem Steuerzahler würden Tränen kommen, wüßte er auch noch, zu welchem Preis...). Die nackte MSDOS-Bedieneroberfläche zudem frustriert die sowieso nicht besonders computerinteressierten Schüler zusätzlich. In der Regel ist eine Schulung durch den Kollegen Informatiklehrer nötig.

Hätte sich GEM auf den MSDOS-Rechnern durchgesetzt, dann wäre die Portierung von Programmen des GEM-Computers Atari kein Problem. Doch wie es scheint, entwickelt sich MS-Windows zum neuen Standard.

Doch ob Windows auf Schulebene Sinn macht, kann ich nicht glauben. So macht es laut c't erst Sinn, mit Windows zu arbeiten, wenn man einen 386-er Rechner mit mindestens 25 Mhz, 4MB Ram und einer 80-MB-Festplatte besitzt. Und ich bin mir ziemlich sicher, daß das bayerische Kultusministerium bei den derzeitigen Finanzproblemen nicht einmal das Geld für die nötigen Windows-Lizenzen ausgeben würde.

Ja und da frage ich mich, ob unter den VD-Leser jemand einen Vorschlag hätte, wie das Problem zu lösen wäre. In der Ausgabe vom März 1990 der VD berichtet etwa Frank Stüss von einem Treiber für die Hercules-Grafik-Karte im volksFORTH für den PC, den Programmierer auf "andere Karten und Rechner" ausdehnen sollen. Es wäre nun sicherlich kein Problem, die von ihm vorgeschlagenen Wörter auf den ST zu übertragen, doch fände ich es sinnvoller, daß die PC-User sich GEM nähern. Ich weiß nicht, wie intensiv und ausgefeilt das F-PC-Forth Menüs, Fenster und Dialogboxen benützt und ob ein event-dispatcher vorhanden ist, der sich im Hintergrund um die Verwaltung all dieser schönen Dinge kümmert, doch vielleicht sind da Routinen, bzw. Wörter vorhanden, die sich in das GEM des Atari übersetzen lassen. Dann könnte man eine gemeinsame Sprache sprechen.

Ich würde mich freuen, wenn sich schon jemand einmal Gedanken darüber gemacht hätte und will gerne zugeben, daß eine Fensterumgebung in der Echtzeitprogrammierung eine zweitrangige Rolle spielt. Aus der Sicht des reinen Anwenders aber ist allein entscheidend, was er sieht, wenn er den Rechner einschaltet. Und wenn ich ehrlich bin, habe ich auf dem Atari am liebsten mit dem 32Forth von R.Aumiller und D.Luda gearbeitet, weil es, völlig in GEM eingebunden, ein Höchstmaß an Komfort und Transparenz bietet, auch wenn es zugegebenermaßen dadurch etwas langsam ist.

Zbigniew Diaczyszyn

Leserbriefe

Bericht aus der lokalen Gruppe Rhein/Ruhr

Als ich in diesem Jahr auf dem Rückweg von der Jahrestagung in München war, war ich innerlich nicht voll befriedigt. Heute weiß ich den Grund: anders als in den Vorjahren hatten die lokalen Koordinatoren nicht ihren Bericht zur Lage der Nation abgegeben.
Das ist aber nicht so schlimm, denn schließlich verfügt die Forth-Gesellschaft ja noch über das Organ 'Vierte Dimension', so daß der festgestellte Mangel hier und jetzt abgestellt werden kann. Das wichtigste zunächst in Stichpunkten:

- es gibt uns noch
- wir treffen uns regelmäig

Es gibt einiges Erfreuliche zu vermelden. Neben dem lokalen Imperator hat sich im Ruhrgebiet ein fester Stamm von 3-4 Unentwegten gebildet, der mit großer Regelmäßigkeit bei den Treffen erscheint. Damit ist der Fortbestand der Gruppe auch für die nächste Zeit gesichert. Zusammen mit dem einen oder anderen immer-mal-wieder-Besucher ist bei unseren Treffen ein farbenfroher Nachmittag inzwischen garantiert. Meistens sind zwei bis drei Rechner unterschiedlicher Couleur vorhanden, so daß es für jeden etwas zu sehen gibt. Manchmal reicht das Engagement einzelner sogar zu lehrreichen Kurzvorträgen über ANSI-Forth, Spiele-Strategien, Listenverarbeitung oder universelle Kontrollstrukturen aus. Probleme wie Altglas-Farberkennung wurden auch schon gemeinsam diskutiert.
Immer wieder Diskussionsstoff bietet in letzter Zeit das Layout und der Inhalt unserer VD. Neben dem reinen Kaffetrinken, das natürlich immer noch zentraler Punkt der Tagesordnung ist, kommt es also nie zu Leerlauf.
Sogar Kleinprojekte werden gemeinsam durchgeführt. So konnte sich die lokale Gruppe im Oktober wie bereits im letzten Jahr auf einem Computer-Flohmarkt der wenig interessierten Öffentlichkeit stellen. Der Stand wurde von Albert Schaefer, Jörg Staben und mir gesponsort. Im letzten Jahr fiel dabei ein Mitglied für unsere lokale Gruppe und für die FG ab, das heute zum festen Stamm der Gruppe gehört. Der Erfolg dieses Jahres bleibt abzuwarten (Stand: Oktober 91).
Alles in allem ist aus dem Ruhrgebiet Frohsinn zu vermelden. Leider haben uns noch nicht alle Forth-Anwender aus unserer Region besucht. Beiträge wie diese sollen das aber ändern. Rufen Sie mich an!!

Jörg Plewe
Tel.: 0208/423514

Die vierte Dimension aus Autorensicht

In letzter Zeit ist öffentlich viel Kritik an der VD, sowohl bzgl. Layout als auch Inhalt, laut geworden. Da ich noch wenig Grund sehe, hier einmal ein Lob auszusprechen, habe ich mir einen neuen Kritikpunkt gesucht. Ich habe damit einige Zeit gewartet, weil ich an einem Ausrutscher gar nicht herummäkeln würde.
In einem Artikel vermittelt ein Autor nicht nur Fachwissen. In den Köpfen der Leser baut sich auch ein Bild des Autors auf. Er stellt indirekt auch sich selbst dar. Der Autor wird durch die Artikel vielen Leuten bekannt, die z.T. für ihn selbst auch einmal wichtig sein können. Der Artikel repräsentiert das öffentliche Bild des Autors. Deshalb finde ich es wichtig, daß die Artikel weder in Redaktion noch in Layout und Satz ohne Zustimmung des Autors verfälscht werden.
Die folgenden Kritikpunkte ergeben sich aus dem Vergleich meiner Artikel, die ich in der VD staunend gelesen habe, und den Originalen, die ich irgendwann einmal auf einer Diskette dem redaktionellen Beirat geschickt habe. Die Diskrepanzen sind erstaunlich. Ich kann hier nicht für andere Autoren sprechen, da ich ihre Originale nicht kenne. Es drängt sich mir aber die Vermutung auf, daß es ihnen ähnlich ergangen ist.
Es begann mit dem 'Katastrophenheft' VD Vol.7 Nr.1 in diesem Jahr. Artikel waren allesamt reichhaltig mit "" verziert worden, was so mancher Formulierung ein ganz anderes aussehen verlieh. Aber auch komplette Phrasen wurden verändert und Absatzreihenfolgen verstellt. Daß dabei noch auf leider nicht vorhandene Listings verwiesen wurde kann man gemessen an solchen Verfehlungen schon fast verzeihen. Die Orthografie war an vielen Stellen erschreckend eigenwillig.
Im letzten Heft (Vol.7 Nr.3) habe ich es dann wieder mit einem Kurzbeitrag (F68K - letzte Neuigkeiten) versucht. Das Ergebnis zwingt mich zu diesem Leserbrief. Jeder Leser muß zu der Erkenntnis kommen, daß ich Forth schon in frühen Jahren gelernt haben muß, und zwar in der Zeit, in der andere Kinder die Grundschule besuchten. In der einen Spalte traf ich auf 5 (in Worten 'fünf') Rechtschreib-/Tippfehler. Das erschreckende dabei: in meinem sofort kontrollierten Original waren diese Fehler noch nicht drin!
Kein Mensch ist perfekt. Jeder Autor wird solche Fehler mit seinen Artikeln abgeben und sich freuen, wenn diese im Verlauf von Redaktion und Layout ausgemerzt werden. Das umgekehrte aber darf nicht passieren! Das ist sehr ärgerlich und man fragt sich, warum man jedesmal eine Diskette opfert, um die Artikel in maschinenlesbarer Form abzuliefern.
Artikel in der VD werden nicht honoriert. Die Autoren opfern ihre Freizeit für den Verein. Die Motive können dabei von reinem Idealismus bis zu purer Geltungssucht reichen. Auf jeden Fall will sich jeder Autor mit dem Artikel, in den er einige Mühe investiert hat, einen Namen machen. Was er auf keinen Fall will, ist sich durch die Veröffentlichung als Analphabet oder Technotrottel zu disqualifizieren.
Die VD ist nicht die c't. Will sagen, Autoren sind Mangelware. Man sollte alles versuchen, diese nicht zu verärgern. Meine ich.

Jörg Plewe

Ansonsten: weiter so!

Ich denke, daß manche Leser hier einfach zu hohe Anforderungen stellen. Für den geringen personellen Aufwand (eine einzige Person) darf das Layout als gut gelungen betrachtet werden. Ich persönlich wünsche mir nur die Listings etwas kleiner abgedruckt als oft üblich - das würde die Listings meiner Meinung etwas übersichtlicher machen. Ansonsten: weiter so!

Heinrich Hohl

Nachrichten

ANS Forth Update

Der Entwurf des Standards liegt nun der Öffentlichkeit vor. Die Dauer der ersten Lesung (public review) ist auf vier Monate angesetzt (18.10.91-25.2.92). Die danach folgenden Lesungen werden immer zwei Monate dauern, solange bis der Standard schließlich angenommen ist. Sie können sich beteiligen.

Alle Kommentare bitte an:
X3 Secetariat/CBEMA,
Attn: Lynn Barra
311 First St. NW, Suite 500
Washington,
DC 20001-2178. USA.

Eine Kopie jeweils an:
American National
Standards Institute,
Attn: BSR Center
11 West 42nd St
New York, NY 10036. USA.

Bezugsquelle in Deutschland:
Information Handling
Tel: 089-85599041
Fax: 089-8596687

Bezugsquelle in USA:
Global Engineering
Documents, Inc.
2805 McGaw Ave.
Irvine, CA 92714
(800) 854-7179
(714) 261-1455 (outside USA)
Domestic orders: $50.00
International orders: $65.00

Quellen:
M.Bradley, U.Hoffmann
Angaben ohne Gewähr.

P.S. Einige Arbeitskopien des dpANS habe ich noch übrig! Englisch 200 Seiten gebunden. 32,- DM zuzügl. Versand.
Michael Kalus, Tel. 04523-4177

F68K - letzte Neuigkeiten

Auch in diesem Quartal gibt es wieder einige kleine Verbesserungen/Erweiterungen/Änderungen des Freeware-Forthsystems F68K für alle M68000-Computer.
Von Wolfgang Schemmert stammt der erste funktionstüchtige Laderfür Commodore Amiga. Er basiert auf dem in C geschriebenen Lader für Atari ST. Er ist ihm deshalb auch funktional ähnlich. Damit läuft F68K endlich auch auf dem System, das der ursprüngliche Anlaß für die Arbeit an F68K war.
Reinhard Scharnagl war so nett, einen Disassembler zu verfassen, der neben den 'normalen' 68000-Befehlen auch die höheren Prozessoren und Coprozessoren 'kann'. Damit ist eine weitere wichtige Komponente geschaffen worden, die die Qualität des Gesamtsystems deutlich erhöht.
*Die in diesem Heft (hoffentlich) *** vorgestellten UCS (Universal Control Structures) von Kevin Haddock habe ich bereits auf F68K portiert, wobei es erstaunlich wenig Gegenwehr gab, obwohl dort doch relativ systemnah agiert wurde. Die 2000-Byte Blöcke des F68K unterstützen den interpretierenden Modus der UCS besonders gut. Es hat sich gezeigt, da sich F68K gerade fürs Experimentieren durch seine saubere Implementation sehr gut eignet. Es gibt nur wenige F68K-typische Riffe, die umschifft werden müssen.*
Dirk Kutscher, dem Verfasser des Sinclair QL Laders, ist es gelungen, durch die Veröffentlichung eines Artikels über F68K in der QI-User-Zeitschrift QUASAR in diesen Kreisen eine richtige kleine Forth-Diskussion auszulösen.

Das alles soll zeigen: F68K lebt und bleibt in Bewegung!

F68K ist derzeit auf folgenden Systemen unmittelbar verfügbar (Lieferumfang):

Atari ST
Commodore Amiga
Sinclair QL

Weitere Systeme auf Anfrage.
Informationen gibt es beim Autor.

Dipl. Phys. Jörg Plewe
Großenbaumer Str. 27
4330 Mülheim an der Ruhr
Tel.: 0 208/42 35 14

**** Anmerk. d. Redaktion:*
Der Artikel UCS mußte leider verschoben werden.

Kleinanzeige

Super8 FORTH-Prozessor
(siehe VD Nr. 3/91)
- Platine unbestückt
- Bauteile
- Super8 mit FORTH im ROM (Zilog 0887520PSC)
- fertig aufgebaute Platine

von: Hans-Günther Willers
Gartenstraße 11
8047 Karlsfeld
Tel.: 08131 - 96375

Kleinanzeige

Super8™ -FORTH in Silicon

— FORTH im 8Kbyte ON-Chip ROM
— Transparente Entwicklungsumgebung kompiliert ROM-fähigen Code
— Zugrifff auf alle ON-Chip Resourcen
— ausführliches Handbuch
— Quellcode von S8-Assembler und S8-FORTH

Muster 0887520PSC bei:
FORTHWORKS
Ulrike Schnitter
Tel: 089-3103385

Kleinanzeige

68040 Sammelbestellung Rechner hat 25 Mhz, 4 Mbyte Ram, 1 Mbyte Video Ram, 110 Mhz Pixelfrequenz, 256 Farben, 1280 X 1024 Auflös. RGB Analog Ausgang, 256 Kbyte stat. Dual Port Ram, 1,4 Mb Floppy, 80 Mbyte Harddisk, Gehäuse mit VME Bus, 8 Steckplätze, ca. 15000 DM, zusätzlich 4 Mbyte Ram 1.500 DM, zusätzlich 8 Mbyte Ram + 3.000 DM.

Anfragen bitte nur schriftlich an Ludwig Braun, PF 1236, 8425 Neustadt/Donau

Swapping Data

Handhabung großer lokaler Datenmengen

Frank Stüss

Stichworte :

Caching / Multitasking / lokale Daten FORTH: F-PC und andere Rechner: PC und andere

1. Einführung

Müssen Sie große Datenmengen schnell verarbeiten? Am besten eignet sich dafür natürlich ein großer RAM-Bereich.

Steht nicht genügend freies RAM zur Verfügung, kann man so Vorgehen:

a) mehr RAM kaufen (außer bei DOS), oder

b) schnell auf einen Massen speicher auslagern, oder

c) aufgeben.

Ich zeige hier meinen Versuch b) zu realisieren mit der sogenannten *Bufferverwaltung*.

Solange man DOS und andere Betriebssysteme einsetzt, welche kein automatisches Swapping/Caching besitzen und außerdem mit FORTH arbeitet, kann sich dieses Werkzeug als recht nützlich erweisen. Für diejenigen, welche unter UNIX, VMS oder anderen Mehrplatz-Betriebssystemen arbeiten, ist dieses Paket nicht vonnöten, da das jeweilige Betriebssystem schon ein Swapping besitzt. Auch unter Windows 3.0 ist ein Swapping eingebaut, sodaß man dort höchstens mit der manchmal lästigen Segmentierung zu kämpfen hat, falls man seine Anwendungen Windows-konform schreibt (SDK, Borland oder andere).

Aber allen anderen, welche außerdem in die Verlegenheit kommen, große Datenbereiche einigermaßen bequem und schnell zu handhaben, sei die Verwendung der Bufferverwaltung empfohlen.

2.. Anforderungen an das Caching/Swapping-Modul

- Transparenz / information hiding

Der Programmierer als Anwender der Bufferverwaltung darf nicht mit den verwendeten Algorithmen bzw. Mechanismen konfrontiert werden. Die Verwendung muß ähnlich einer File-Anbindung sein. Dies soll außerdem spätere Änderungen des Algorithmus ohne Änderungen der Anwendung an sich zulassen.

- Portierbarkeit

Da wir dieses Modul auf mehreren Rechnern zum laufen bringen wollen, war die erste Pflicht, möglichst kompatibel zu programmieren. Daher wurde das komplette Modul bis auf die spezifischen Zugriffe auf das RAM und die Festplatte in High-Level geschrieben. Leider geschah dies auf einer 16-Bit-Plattform (F-PC von Tom Zimmer und anderen). In einem 32-Bit FORTH lassen sich weitere Optimierungen durchführen.

An Stellen, wo hohe Geschwindigkeit erforderlich ist, wurde Assembler eingesetzt, jedoch kann man diesen durch einen compilerswitch auf FORTH-Worte umschalten (Beispiel 1, Seite 11)

- Multitasking-Fähigkeit

Es soll möglich sein, von mehreren Tasks auf die lokalen Daten zugreifen zu können, daher mußte auf die Verwendung von globalen Variablen verzichtet werden. Es wird *ein non-preemptive-multitasker* verwendet (also der, der mit PAUSE den Taskwechsel vollzieht).

Es ist möglich, von mehreren Tasks aus auf einen Buffer zuzugreifen. Dieser muss jedoch global angelegt und geöffnet werden, falls er für mehrere Tasks bestimmt ist.

- Einstellbarer Speicherbedarf

Die Größe des Cache-Speichers ist einstellbar.

Unter F-PC ist die Benutzung von EMS oder linearem DOS-Speicher möglich (siehe auch Beispiel1).

Alle diese Punkte wurden nach meinem Vermögen verwirklicht.

3. Mechanismus

Es gilt einen Mittelweg zwischen beiden Extremen zu finden:

Ich habe sehr viele Daten, mehr als ins RAM passen -

Ich muß sehr schnell auf diese Daten zugreifen, am besten mit RAM-Geschwindigkeit.

Dazu kommt die Notwendigkeit, mehrere unterschiedliche Datenbe-

stände mit verschiedener Länge verwalten zu können. Folgende wichtige Punkte sind für den effektiven Ablauf des später vorgestellten Algorithmus Voraussetzung:

- Der Zugriff auf die Datenbestände erfolgt nicht rein sequentiell. D.h. man liest nicht immer die gesamten Daten von Anfang bis Ende.

- Der Zugriff darf nicht 'echt oder unecht' zufällig sein, bzw. muß ungleich verteilt sein.

Erfolgt der Zugriff hingegen rein sequentiell oder ist er zufällig verteilt, dann sind dies Extremfälle für den verwendeten Algorithmus, in denen er nicht besonders effektiv arbeitet. Gut ist es aber, wenn bestimmte Bereiche eines Datenbestands öfter die Ehre eines Zugriffs erteilt bekommen, andere weniger oder gar nicht.

Ein Beispiel soll das erläutern (einer der Gründe, wieso dieses Paket gebaut wurde): Man hat drei sehr große, bis (für DOS-Verhältnisse) riesige Datenbanken, deren Indices jeweils mehrere Mb groß sind. Die Datenbanken arbeiten relational, also eine Datenstruktur in der einen Datenbank steht in Relation mit der entsprechenden in einer anderen Datenbank.

Wenn man dieses System bei der Arbeit beobachtet, stellt man fest, daß die Zugriffe auf die Indices kreuz und quer erfolgen. Normalerweise häufen sich die Zugriffe an bestimmten immer wieder benötigten Daten. Hier ist der Einstieg zu finden: Es müssen sich soviele dieser Bereiche mit häufigem Zugriff im RAM befinden, bzw. unter schnellem Zugriff befinden, *wie nur irgend möglich.*

Man muß also feststellen, wo diese Datenbereiche liegen, um sie dann im RAM zu halten. Das birgt natürlich einen Nachteil der in der Natur der Sache liegt: um herauszufinden, welche Datenbereiche am nötigsten gebraucht werden, muß man sie Brauchen, bzw. man kann Daten nur ins RAM laden, wenn auf sie schon oft (und langsam) zugegriffen wurde.

Dieses Paket ist also nicht an Stellen zu verwenden, an welchen

Beispiel 1

```
 entry@ , entry!  - Zugriffe auf einen Blockentry
{

maschine not
#IF
: entry@ ( absblock -- u )  \ holt Eintrag aus entry-Map
    ( #b/blockentry * ) 2* tableseg swap @l
 ;

: entry! ( u absblock -- )  \ speichert Eintrag
    ( #b/blockentry * ) 2* tableseg swap !l
 ;
 \ in beiden Faellen wird keine Ruecksicht auf das Access-bit genommen
 \ Diese Zugriffe stellen die unterste Ebene dar.
#ELSE
 Code entry@ ( absblock -- u )
   pop BX
   mov DS, ' tableseg body
   shl BX, 1
   mov AX, 0 [BX]
   mov BX, CS
   mov DS, BX
   lpush
 end-code

 Code entry! ( u absblock -- )
   pop BX
   pop AX
   mov DS, ' tableseg body
   shl BX, 1
   mov 0 [BX], AX
   mov BX, CS
   mov DS, BX
   Next
 end-code
#THEN
```

Glosssar

```
Buffer: compile-time ( -- )  child-run-time
        ( -- haendell )
```
erstellt ein Buffer-control-Block ähnlich den FCB's. Jedoch interessiert den Anwender nur das Buffer-handle, welches als Referenz für jegliche Aktion eines Buffers dient.

```
buf>buf ( hdle1 daddr1 hdle2 daddr2 dlen -- )
```
kopiert dlen byte vom Buffer hdle1 ab Position daddr1 zum Buffer hdle2 nach Position daddr2. Im Fall von hdle1=hdle2 wird auf die richtige Verschiebung geachtet. Von byteweisem Verschieben sollte nach Möglichkeit abgesehen werden.

```
buf>system ( hdle daddr seg:addr dlen -- )
```
überträgt (dlen) byte vom Buffer (hdle) an die Adresse (seg:addr) im linearen DOS-Speicherbereich.

```
system>buf ( seg:addr hdle daddr dlen -- )
```
überträgt (dlen) byte von der Adresse (seg:addr) im linearen DOS-Speicherbereich in den Buffer (hdle) an die Adresse (daddr).

```
b@ , b!, bc@, bc!, b2@, b2!
    ( c | n | d hdle daddr -- ) !
    ( hdle daddr -- c | n | d ) @
```
Sind die optimierten Zugriffe auf ein File.

eine Zugriffszeit kleiner als Massenspeicher *unbedingt*, d.h. notwendig zur Funktion, erforderlich ist.

4.. Verwaltung der Datenblöcke in Tabellen

Die Datenbereiche werden willkürlich in Blöcken gleicher Länge dargestellt.

Da wir in einer binären Umgebung arbeiten, werden die Blöcke die Länge von 2^n Byte haben, um eventuelle Adressberechnungen möglichst schnell und maschinennah ausführen zu können.

Im F-PC-System haben die Blöcke die (willkürliche) Größe von 1 kbyte.

Um die Buchhaltung zu bewältigen, sind einige Tabellen nötig. Als erstes sei da die *Entry-map* zu nennen. Dies ist die Tabelle aller möglichen Blöcke, welche in die Bufferverwaltung passen. Im ursprünglichen System wurden 20 k Blöcke verwaltet, waren also 20 Mbyte an aktivem Datenbestand möglich: diese Zahl war rein willkürlich festgelegt.

In der Entry-map stehen folgende Daten:

1. Occupied-Flag (1Bit)

   - zeigt an, daß auf diesen Block gerade zugegriffen wird (locking).

2. Updated-Flag (1Bit)

   - zeigt an, daß auf diesen Block ein Schreibzugriff getätigt wurde.

3. RAM-entry# (14Bit)

   - entweder 0 (Block liegt nicht im RAM), oder die Nummer des entsprechenden RAM-0entrys +1.

Aus diesen Daten geht hervor, daß man mit dieser Mimik auf maximal 16 Mbyte RAM zugreifen kann.

Die zweite Tabelle ist die sog. *RAM-map*. Sie enthält die Daten für alle verfügbaren RAM-Blöcke und deren Gewichtung (siehe später), desweiteren noch die Rückverbindung zur entry-map.

Die dritte Tabelle beinhaltet die Referenzen der RAM-Blöcke, welche keine besonders hohe Zugriffsdichte aufweisen, der sogenannte *Low-prior-queue*.

5. Der Ablauf

So, jetzt können wir loslegen.

Man öffnet also so einen Buffer. Es geschieht folgendes:

Zunächst wird ausgerechnet, wie viele Blöcke dieser Buffer beansprucht.

Diese werden dann in der entry-map eingetragen.

Dann wird nachgeschaut, ob noch RAM-Blöcke verfügbar sind. Falls dem so ist, werden die entsprechenden Datenblöcke vom File ins RAM eingelesen.

Ein Zugriff kann auf verschiedene Weisen vonstatten gehen.

Bei allen Zugriffen wird zuerst der relative Block innerhalb des Datenbestands und die relative Adresse ermittelt. Aus erstem dann der absolute Block in der entry-map errechnet. In diesem wird nachgeschaut, ob er in einem RAM-Block liegt (RAM-entry# 0). Ist dies so (Fall 1), wird die Block-Adresse berechnet, die Relativadresse addiert und der Zugriff getätigt.

Ist es ein schreibender Zugriff, wird das Updated-Bit in der Entry-map gesetzt.

Falls kein RAM-Block reserviert ist, wird zunächst nachgeschaut, ob noch welche unbelegt sind. Wenn ja (Fall 2), wird ein unbelegter benutzt, der Datenbereich von Disk gelesen und auf den RAM-Block zugegriffen.

Ist jedoch kein Block mehr frei, wird nachgesehen, ob ein Block im low-prior-queue ist. Ist dies so (Fall 3), wird dieser Block zurückgeschrieben und so ein Block freigemacht.

Geht auch das nicht (Fall 4), wird auf den Massenspeicher zugegriffen.

Um die Zugriffe gewichten zu können, wird jedem RAM-Block eine Priorität zugewiesen (ein Tabellenfeld in der RAM-map). Bei jedem Zugriff auf einen Block, der im RAM liegt, wird die Priorität des Blocks um einen bestimmten Betrag erhöht. Der Maximalwert der Priorität ist 255 (8Bit-Feld).

Wenn ein Zugriff auf einen Block erfolgt, der nicht im RAM liegt, wird die Priorität aller Blöcke, die im RAM liegen um eins erniedrigt. Stößt die Verwaltung dabei auf einen Block, der die Priorität null hat, wird dieser in der low-prior-queue vermerkt, also nicht direkt freigegeben.

Wenn ein Block neu ins RAM eingelesen wird, erhält er nicht die Priorität null, damit er nicht gleich wieder entfernt wird. Der Grundwert läßt sich über das Value PRPR (prior-preset) festlegen. Der default-Wert liegt (wieder willkürlich) bei 10.

Der Wert der Prioritätserhöhung bei RAM-Zugriff (PRINC - prior increment) läßt sich ebenfalls einstellen. Optimale Werte sind Problem-, Disk- und RAM-abhängig, also nicht trivial zu finden. Den Experimentierfreudigen sei hier ein weites Betätigungsfeld gegeben.

Es wurde besonderen Wert darauf gelegt, daß zeitintensive Teile des Algorithmus nur dann ausgeführt werden, wenn ein Diskzugriff erfolgt, man sich also ohnehin in einem kleinen Teil eines zeitlich gesehen großen Abschnitts befindet.

So werden die Prioritäten *aller* Blöcke nur bei *Diskzugriff* herunter gezählt, jedoch nur die Priorität *eines* Blocks bei einem *RAM-Zugriff* erhöht. Wenn man sich für eine bestimmte Plattform entschieden hat, kann man noch weitere Optimierungen durch den Einsatz von Assembler durchführen, was hier sicherlich sinnvoll erscheint.

6. Die Datensicherheit

Was passiert, wenn der Rechner abstürzt und es befinden sich noch Blöcke im RAM? Nun, dann wären die Daten normalerweise verloren. Es gibt verschiedene Methoden, dies zu verhindern. Die sicherste ist, jeden schreibenden Zugriff auf einen RAM-Block auch auf den Massenspeicher weiterzugeben, damit die Daten von RAM und Disk immer gleich sind (sog. *writethru*), dies kostet jedoch sehr viel Zeit. Für die Problematik, des in etwa gleich häufigen Lesen und Schreibens, ist dies nicht optimal. Jedoch hat diese Methode einen weiteren wichtigen und eventuell entscheidenden Vorteil:

In einem Netzwerk ist die immer gewährte Aktualität der Daten auf Disk streng notwendig, da in kommerziellen Netzwerken der High-Level-Zugriff übers Netz immer auf ein File auf Disk geht.

Einen nicht so sicheren Weg habe ich hier beschritten:

Der Rechner soll die Daten auf Disk immer dann retten, wenn er Zeit dazu hat.

Dazu wurde ein Background-Task gestartet, welcher in einer Schleife jeden Eintrag in der entry-map durchläuft. Ist der aktuelle Block dann im RAM und außerdem noch 'updated', dann wird er auf Disk geschrieben.

Diese Vorgehensweise erfordert natürlich einige Sicherungen, damit sich Zugriff und Backup nicht in die Quere kommen.

```
(Beispiel 2)


]       FNK 10/18/90 17:42:52.82

   Multitasking Backup

[

: backup-buf ( -- )
   BEGIN organizing @ 0=
         IF   buffers 0
              ?DO   organizing @ IF leave THEN          \ globale Aktionen ?
                    i buf[i]
                    dup bufopened? not
                    IF drop leave THEN
                    dup startblock                     \ 1. Block in Buffer
                    over blocks over + swap
                    ?DO   organizing @ IF leave THEN
                          i setaccess                  \ Zugriff setzen
                          dup i ?write-block
                          i clearaccess
                          10 0 DO pause LOOP           \ anderen Tasks
                                                       \ Zeit lassen
                    LOOP  ( alle Blocks )
                    drop
              LOOP  ( alle Buffer )
         THEN
         10 0 do pause loop
   AGAIN  ( endlos )
;
```

Glossar - Fortsetzung:

b@ , b! , b2@ , b2! sind nur an alignierten Adressen möglich !

bufassoc (hdle string --)
benennt das file eines buffers auf (string). Dieses Wort darf nur auf geschlossene Buffer angewendet werden.

bufrename (hdle string --)
benennt das file eines Buffers um auf (string). Dieses Wort darf nur auf geöffnete Buffer angewendet werden.

bufopen (hdle --)
Ein Buffer wird geöffnet, d.h. zur Benutzung freigegeben. Ist dem Buffer zuvor ein Filename zugeordnet worden, so wird ein nicht-temporärer Buffer geoffnet, sonst ein temporärer. Falls noch Ram-Speicher unbelegt ist, wird dieser von diesem File in algorithmischer Weise belegt.

bufclose (hdle --)
Ein Buffer wird geschlossen. Der okupierte Speicher wieder freigegeben. Der filebereich eines temp-Buffers wird gelöscht.

bufsetlen (dlen hdle --)
Setzt die Länge des Buffers hdle auf dlen. Falls der Speicherbereich (ram oder Massenspeicher) erschöpft ist, wird ein Fehlerstatus erzeugt. Im Falle einer Verkürzung gehen die Daten des Restes verloren, im Falle einer Verlängerung stehen zufällige Daten im angehängten Teil.

bufappend (hdle dlen --)
Verlängerung eines Buffers um dlen

bufshorten (hdle dlen --)
kürzt einen Buffer logisch, d.h. die File-Länge bleibt erhalten und wird gegebenenfalls beim runterfahren der Bufferverwaltung korrigiert (syscom: FNK noch nicht realisiert, d.h. wenn ein File eine bestimmte Länge hat, kann diese nur noch vergrössert werden)

bufdelete (hdle --)
schliesst einen Buffer und löscht dann den zugehörigen File-Bereich.

bufopened? (hdle -- f)
f ist wahr, wenn Buffer (hdle) geöffnet.

bufgetlen (hdle -- dlen)
ermittelt die logische Länge (dlen) des Buffers (hdle) in byte

Homepath (-- string)
ist ein sowohl gecounteter als auch zero-ended String

gethomepath (--)
ist ein deffered-Wort, welches den gecounteten und zeroended String des Pfades, auf welchen sich die Buffer-Verwaltung bezieht, in den String homepath kopiert. Default-wort: myhomepath

?buferror (#err f --)
Es wird eine Fehlernummer übernommen, welche im Fall von f=true einen adäquaten Fehler auslöst.

Die Backup-Routine dazu sehen Sie in (Beispiel 2, Seite 13).

Wenn der Rechner bei einer Anwendung gerade im Leerlauf steht, wird man feststellen, daß die LED der Festplatte flackert, obwohl er im Moment dort doch gar nichts zu suchen hat...

Das ist eben der Backup, der - keine Angst - nicht etwa die Platte formatiert oder löscht, sondern zivile Daten zurückschreibt.

7. Diskussion

Mittlerweile setze ich die Bufferverwaltung fast überall dort ein, wo der Datenbereich größer als 1 kbyte ist. Ich kann beliebig viel 'nichtganz'-RAM benutzen, soweit es die Festplatte zulässt. Z.B. erreichen Hilfsspeicher für Bildschirmausschnitte schnell die Grenze von 100 kbyte bzw. 1 Mbyte im Grafikmodus bei größeren Anwendungen. Diese Datenmengen sind speziell in einem 16-Bit-System ziemlich unhandlich. Bei Benutzung der Bufferverwaltung ergeben sich schnelle Zugriffszeiten für Bildschirmausschnitte, hoher Zugriffsfrequenz, während Fenster, welche nicht so häufig benötigt werden, ab und zu etwas langsamer (flippflapp) dargestellt werden.

Da wir unter DOS arbeiten, ergibt sich das größte Problem mit der Speicherbeschränkung (die berühmt-berüchtigte Schallgrenze 640k/1Mbyte).

Unter DOS kann man mit friedlichen Mitteln nur an mehr Speicher kommen, wenn man EMS oder XMS ausnutzt.

Dies wurde getan. Dabei hat man aber Schwierigkeiten mit den sehr langen Zugriffszeiten über Interrupts. Das heißt, ohne EMS sind die RAM-Zugriffszeiten um ein vielfaches höher.

In Systemen ohne diese lästige Begrenzung gilt dies natürlich nicht.

Da das relativ neue Windows 3.0 (zur Zeit und soweit mir bekannt ist), noch kein FORTH kann, welches auch noch alle Speichervorteile der Umgebung ausnutzt, ist auch hier zunächst nur EMS oder XMS gefragt.

Leider hat die Bufferverwaltung noch einen kleinen Schönheitsfehler:

Die resultierenden File-bereiche werden Block- und nicht byteweise angelegt. Daraus ergibt sich, daß wenn man das File liest, am Endes des Files einige Byte Müll stehen (der Rest eines Blocks). Dieses Problem will ich jedoch in späteren Versionen beheben.

Ein weiterer Nachteil, der in der Natur der Sache liegt, ist die Tatsache, daß alle Daten, die von der Bufferverwaltung bedient werden (noch) nicht netzwerkfähig sind. Das liegt daran, daß eventuell im RAM gebufferte aktuelle Daten nicht auf Disk liegen, sodaß diese von einem anderen Netzteilnehmer nicht gelesen werden können, bzw. falsch gelesen werden. Dem kann man durch writethru abhelfen, was in der nächsten Version einschaltbar sein wird.

Es soll einen Schalter geben, der zwischen writethru und dem Multitasking-Backup auswählen kann. Dies wird man dann sogar Bufferbezogen wählen können, um sowohl lokale Daten, als auch netzwerkfähige Datenbereiche zu erhalten.

Literaturhinweise:

spezifisch - keine

Nachrichten

F-PC 'Zimmer'Forth 'ak'-UpDate

Nicht zuletzt über Anregungen auf der Euroformel ist die ák'Version mittlerweile enorm fort(h)geschritten. Zur Vorbereitung der UpDates bitte (!) ich um Korrekturen, Ergänzungen usw., natürlich kommentiert und maschinenlesbar. (Adresse im Anhang)
Arndt Klingelnberg

Infix nach Postfix

Bernd Paysan

Infix-Postfixwandler mit Prioritätensteuerung

Stichworte:

**** Infixinterpreter***

Böse Zungen behaupten, FORTH-Programmierer müßten deshalb in Postfix programmieren, weil die FORTH-Entwickler nicht fähig sind, eine "normale" Infix-Notation zu implementieren (was zugegebenermaßen auch schwieriger ist).

Natürlich ist das nicht richtig. Auch daß die Postfix-Notation nach kurzer Zeit genauso locker von der Hand geht, wie das, was man in der Schule gelernt hat, kann man nicht von der Hand weisen. Trotzdem gibt es für einen Infix-Interpreter genügend Anwendungsmöglichkeiten, sei es in einer Applikation, die ihre FORTH-Herkunft schamhaft verstecken will.

Gottseidank ist FORTH ja eine syntaxfreie Sprache; es hindert uns also niemand daran, ihr eine neue Syntax aufzuprägen. Dabei soll so wenig wie möglich an den bisherigen Eigenschaften des FORTH-Systems herumgedreht werden. So ist es ganz praktisch, daß Zahlen einfach gewandelt und auf den Stack gelegt werden, man kann das beibehalten. Auch die Operationen sollen weiterhin vom Interpreter ausgeführt werden, nur dürfen sie nicht dieselbe Aktion haben wie bisher.

Nun kommen die Operatoren in Infixnotation grundsätzlich zu früh, als daß man sie schon richtig ausführen könnte. Eines der beiden zu bearbeitenden Werte steht noch aus. Man muß die Ausführung mindestens bis zur nächsten Zahl verschieben. Auch dort kann dann eine Operation mit höherer Priorität stehen, dann wird man weiter verschieben müssen.

Eine recht einfache und dennoch mächtige Lösung ist es, die Operation auf einen Stack zu schieben. Frühestens die nächste Operation (oder eine "Klammer-zu") wird diese Operation ausführen - und auch das nur, wenn ihre Priorität größer oder gleich der nächsten Operation ist. Ansonsten wird weiter gestackt.

Nun zum Programm selbst: Es wird ein 16 Zellen langer Operatorstack angelegt. Die Variable OP enthält die Zahl der belegten Zellen, gleich dahinter beginnt der Stack selbst. 16 Zellen bedeutet eine Menge Klammerebenen und vorrangige Ausführungen. Es dürfte normalerweise reichen. >O schiebt einen Operator auf den Stack, O> holt ihn wieder herunter. Ein Operator ist eine Struktur, die Priorität und Adresse des Befehls (in dieser Reihenfolge) enthält.

PRIO@ liest die Priorität des obersten Elements des Operatorstacks aus. OP, führt den obersten Operator aus bzw. compiliert ihn.

Die eigentliche Aktion erfolgt im Defining Word IN. Es erzeugt einen Infixoperator mit angegebener Priorität. Er muß einen gleichnahmigen Postfixoperator besitzen, der dann die eigentliche Operation übernimmt. Erzeugt wird ein immediate Word. Bei der Ausführung werden alle bereits auf dem Stack liegenden Operatoren gleicher oder größerer Priorität ausgeführt. Negative Prioritäten bilden eine Ausnahme (das ist für die Klammern wichtig): Sie werden nicht ausgeführt. Anschließend wird der gerade bearbeitete Operator auf den Stack gelegt.

Im zweiten Screnn werden nun zuerst verschiedene Infixoperatoren definiert. Die Zuweisungen := und =: (zugewiesene Variable am Ende des Ausdrucks) werden definiert. Es gibt drei Ebenen von Prioritäten: Die höchste sind arithmetische Operationen. Dazwischen liegen Vergleichsoperationen. In der niedrigsten Ebene sind logische Befehle und Zuweisungen. Hier hat AND eine höherere Priorität wie OR, damit können Ausdrücke in disjunktiver Form klammerfrei ausgedrückt werden. XOR wird wie ein OR aufgefaßt, das entspricht seiner Expansion in disjunktive Form:

```
(NOT a AND b) OR (a AND NOT b)
```

Für die Klammern wird ein eigener Operator ((mit negativer Priorität angelegt. (legt diesen Operator einfach auf den Stack, ohne einen Operator aufzulösen.) dagegen löst alle innerhalb der Klammer aufgetretenen Operatoren auf und löscht

dann beide ((-Operatoren vom Stack.

$ ist ein Umschalter, der vom normalen Verhalten auf das INFIX-Verhalten umschaltet und zurück. Ich habe mich für diesen Namen entschieden, weil TeX dieses Zeichen für den mathematischen Modus verwendet. Natürlich kann man auch längere und offensichtlichere Namen wie INFIX(und) INFIX wählen oder was auch immer. $ umschließt die ganze Formel mit einer unsichtbaren Klammer, die dazu dient, am Ende der Formel alle Operationen aufzulösen.

Da bigFORTH ein TO-Konzept bietet, habe ich auch das implementiert. Es muß nur die implizite Klammer von $ vorher geschlossen und nachher wieder geöffnet werden, da TO seine Infix-Charakteristik bereits mitbringt. Zusammen mit lokalen Variablen (die allerdings noch nicht im bigFORTH drin sind) hat man fast schon ein Gefühl wie in einer "normalen" Sprache. Man sollte allerdings nicht vergessen, zwischen jedem Befehl und auch um die Klammern Leerzeichen einzugeben, es wird nämlich der normale Scanner von FORTH (WORD) verwendet.

Auf alle Fälle ist FORTH nun gerüstet, auch solche Eingaben wie:

```
$ ( 1 + 2 * 3 ) *
( 4 - 5 + 6 - 7 + 8 ) $ . 42  ok
```

richtig auszuführen.

Übrigens: Dieser Infix-Interpreter behandelt Fehleingaben ziemlich lax. Soweit die Anzahl öffnender und schließender Klammern gleich ist, stimmt für ihn die Rechnung. Man kann auch Ausdrücke in der Lisp-üblichen Lambda-Notation eingeben. Beispiel:

```
$ ( + 1 ( * 2 3 ) ) $.
```

Da die Ausführung bis zum nächsten Operator bzw. zur nächsten Klammer verschoben wird, klappt auch das.

Source für volksFORTH:

```
\ Infixinterpreter                                        05aug91py
\needs cells    ' 2* alias cells
\needs cell+    ' 2+ alias cell+
Vocabulary infix   infix also definitions
| Variable op $10 cells allot
| : op@ ( -- addr ) op dup @ cells + ;
| : >o ( o -- ) 1 op +!  op@ ! ;
| : o> ( -- o ) op@ @ -1 op +! ;
| : prio@ ( -- prio ) op@ @ @ ;
| : op, ( -- ) o> cell+ @ state @ IF , ELSE execute THEN ;

: in ( cfa priority -- )
 >in @ ' -rot >in ! create , , immediate
 DOES> dup >r @ 0 max >r  BEGIN prio@ r@ < 0= WHILE op, REPEAT
       rdrop r> >o ;
                                                    -->


\ Infixinterpreter                                        05aug91py
' ! alias =:     | :  := swap ! ;       | ' noop alias ((
20 in +          20 in -          21 in *          21 in /
21 in mod        21 in max        21 in min
10 in =          10 in >          10 in <
02 in or         03 in and        04 in not        02 in xor
01 in :=         01 in =:         | -1 in ((

| : unbalanced  abort" unbalanced Expression!" ;
: (    [ ' (( >body ] Literal >o ; immediate
: )    op @ 0= unbalanced  [compile] (( o> o> 2drop ; immediate
\ : to  [compile] ) [compile] to [compile] ( ; immediate
: $    toss [compile] ) op @ unbalanced ; immediate
forth definitions
: $    op off also infix [compile] ( ; immediate
Onlyforth
```

Source für bigFORTH 1.10:

```
\ Infixinterpreter                                      05aug91py

Module infix

| Variable op $10 cells allot
| : op@ ( -- addr ) op dup @ cells + ;
| : >o ( o -- ) 1 op +!  op@ ! ;
| : o> ( -- o ) op@ @ -1 op +! ;
| : prio@ ( -- prio ) op@ @ @ ;
| : op, ( -- ) o> cell+ @ state @ IF cfa, ELSE execute THEN ;

: in ( cfa priority -- )
 >in @ ' -rot >in ! create , A, immediate
 DOES> dup >r @ 0 max >r  BEGIN prio@ r@ < 0= WHILE op, REPEAT
       rdrop r> >o ;
                                                  -->



\ Infixinterpreter                                      05aug91py
' ! alias =:     |: := swap ! ;       | ' noop alias ((
20 in +          20 in -          21 in *          21 in /
21 in mod        21 in max        21 in min
10 in =          10 in >          10 in <
02 in or         03 in and        04 in not        02 in xor
01 in :=         01 in =:         | -1 in ((

| : unbalanced  abort" unbalanced Expression!" ;
: (    [ ' (( >body ] ALiteral >o ; immediate
: )    op @ 0= unbalanced  compile (( o> o> 2drop ; immediate
: to   compile ) compile to compile ( ; immediate
: $    op off also infix compile ( ; immediate
export $ ;
: $    toss compile ) op @ unbalanced ; immediate
Module;
```

Fortsetzung von Seite 5

EUROFORML

soll. SMAN ist eine Entwicklung, die auf der Linie von Wolf Wejgaards HOLON und von DFF (Detlev Heid, Frank Raschke, Frank Stüß) liegt. Paul Kopff zeigte mit dem Paket TGV, wie man der VGA-Karte IBM-kompatibler PCs direkt auf Register-Ebene zu Leibe rückt und damit den Durchsatz gegenüber BIOS-Routinen etwa verzehnfachen kann. Darauf aufbauend realisierte er 3D-Graphik-Animation per Rot-Grün-Brille. Der von Gordon Charlton entwickelte String-Matcher übertrifft an Universalität alle bisher auf diesem Gebiet vorgestellten Forth-Lösungen. Da er trotzdem nur etwa zehn Screens umfaßt, wurde er vom euroFORML-Wettbewerbskomittee mit dem Preis der eindruckvollsten vorgestellten Technik ausgezeichnet.

Während des Samstag-Nachmittags fanden die Arbeitsgruppen statt. Themen wie "ANS-Forth", "Object-Oriented-Forth" und "Formal Forth" wurden diskutiert, dann in der Abschlußsitzung am Sonntag dem gesamten Auditorium vorgetragen.

Der Samstag Abend wurde gekrönt durch ein Jazz Konzert von Alex v. Schippenbach (Piano) und Paul Lovens (Percussion), das auf Anregung von Marina Kern im Casino dargeboten wurde. Es mündete in eine heitere und interessante Nacht, in der nicht nur Forth-Themen diskutiert wurden.

Der Programmpunkt "impromptu talks" am Sonntag Morgen ist erfahrungsgemäß einer der spannensten Teile. Hier werden in einem engen Zeitrahmen geballt Ideen und Meinungen, aber auch zuweilen Komik dargeboten. Nicht selten wird die wesentliche Forth-Idee während der "impromptu talks" geäußert. Den feierlichen Abschluß der Konferenz bildete die Vorstellung der Gewinner des Konferenz-Wettbewerbs:

```
: song ( x x -- )
   BEGIN  over and over
          and over  AGAIN ;

: author ( -- )
   DOES song interpret ;
```

(Weiter auf Seite 34)

* Problem *

Keine zweite Shell unter 4DOS/volks4TH

Jörg Staben

FORTH-Kategorie:
FORTH-Splitter

FORTH-Standard:
FORTH 83

FORTH-System:
volksFORTH
systemspezifisch

Schlüsselwörter:

4DOS
command.com
shell

Zur Zeit erfreut sich der Kommando-Interpreter 4DOS v.3 großer Beliebtheit.

Dieses 4DOS ersetzt den altvertrauten COMMAND.COM und beglückt den Nutzer mit vielen neuen oder - von anderen Programmen her - altvertrauten Funktionen. Beispielhaft seien hier die editierbare Kommandozeile oder der Stapelspeicher für Eingabezeilen genannt.

Dieses Programm 4DOS besteht aus einem Lader namens 4DOS.COM und den eigentlichen Kommandoprozessoren: Für den XT das 4dos88.EXE und für stolze AT- und höher-Besitzer das 4dos286.EXE .

Die Installation dieses neuen Kommandoprozessors wird über den SHELL-Befehl in der CONFIG.SYS durchgeführt. Dabei befinden sich der Lader 4DOS.COM und der Kommando-interpreter 4DOSx86.EXE üblicherweise in einem SubDirectory namens \4DOS , so daß beim Booten der Befehl SHELL=C:\4DOS\4DOS.COM dafür sorgt, daß alles seine Wege geht.

Aber leider, leider wird - zumindest bei mir - die Environment Variable COMSPEC nicht korrekt gesetzt; sie wird auf
COMSPEC=C:\4DOS.COM eingestellt - der Pfad wird also unterschlagen!

Dies hat zur Folge, daß im volks4TH 3.81.41 für den PC die schönen Worte
CALL , DOS: , MSDOS und vielleicht andere ähnliche auch nicht funktionieren.

Diese Worte fieseln aus dem Environment den Namen und den Wohnort des Kommandointerpreters heraus und starten ihn ein zweites Mal.

Wird dieser Kommandointerpreter dann nicht gefunden, weil er sich nicht im angegebenen Pfad befindet, so kommt volksFORTH mit der lapidaren Meldung FILE NOT FOUND oder irgendsowas.

Eine solche Meldung ist natürlich uneinsichtig, wo der arglose Anwender doch genau weiß, daß die gewünschte Datei gefunden werden müßte. Und schon wird heftig auf's volks4TH geschimpft...

Sollte dieser Fall auch bei Ihnen auftreten, so kann man sich mit dem SET-Befehl SET (ohne Parameter) das Environment anzeigen lassen und kontrollieren. Sollte der Pfad für den Kommandointerpreter auch bei Ihnen nicht stimmen, korrigieren Sie ihn bitte innerhalb der Datei AUTOEXEC.BAT mit dem Befehl SET COMSPEC=C:\4DOS\4DOS.COM oder auf 4DOS-Ebene mit dem ESET-Befehl ESET COMSPEC.

Danach dürften keine Probleme mehr im Zusammenspiel zwischen volks4TH und dem neuen Kommandointerpreter auftreten.

Ein intelligenter Editor für Eingabefelder

Heinrich Hohl

Mit Hilfe des hier beschriebenen Zeileneditors lassen sich auf einfache Weise komfortable Eingaberoutinen für Daten verschiedener Art erstellen. Da der Editor nur für die jeweilige Eingaberoutine gültige Zeichen akzeptiert, lassen sich Fehleingaben weitgehend vermeiden.

Schlüsselworte

Eingaberoutinen, Bit-Tabellen

Einleitung

Die Bedienbarkeit eines Programms wird in nicht zu unterschätzendem Maße von der Qualität der Routinen geprägt, welche für die Eingabe von Daten zuständig sind. Im Gegensatz zu anderen Programmiersprachen bietet FORTH von Haus aus nur sehr grundlegende Eingaberoutinen an, was vor allem für Einsteiger ein großes Handikap darstellt. Es erweist sich letztlich aber als sinnvoller, den Entwurf einer komfortablen Eingaberoutine dem Programmierer zu überlassen. Nur dieser kennt die genauen Umstände, unter denen seine Routine arbeiten soll, und er kann sie hierfür optimal gestalten. Jede Eingaberoutine sollte aber zumindest den einen Grundsatz befolgen, Eingabefehler des Benutzers nicht erst nach Beendigung der Eingabe zu reklamieren, sondern nach Möglichkeit bereits während der Eingabe.

Ich möchte hier einen kleinen Zeileneditor namens LEDIT vorstellen, welcher diese Anforderung erfüllt. Bei diesem durchläuft jedes eingegebene Zeichen sofort eine Routine namens FIXUP, welche als falsch erkannte Zeichen sofort ausbessert. Anschließend findet durch die Routine VALID? eine Kontrolle der Zulässigkeit des Zeichens in der jeweiligen Eingaberoutine statt. Von dieser wird das Zeichen letztlich akzeptiert oder abgelehnt. Mit Hilfe des Zeileneditors ist es einfach möglich, komfortable Eingaberoutinen für Daten verschiedener Art zu erstellen. Um das Verhalten des Zeileneditors an die jeweilige Eingaberoutine anpassen zu können, wurden die beiden obengenannten Korrekturroutinen vektoriell ausgelegt [1]. Geschrieben wurde der Zeileneditor mit PC/FORTH 3.2 von LMI. Für die Freunde von F-PC werden aber Änderungshinweise gegeben, um den Zeileneditor auch mit diesem Compiler benutzen zu können.

Der Zeileneditor

Der Zeileneditor selbst wird in den Screens 2-8 definiert. Auf dem Stack erwartet die Routine LEDIT Adresse und Länge eines zu editierenden Speicherbereichs, und außerdem die Anzahl von Zeichen, welche der String hinterher maximal enthalten darf. So ergeben sich für den Aufruf des Zeileneditors (oder einer darauf basierenden Eingaberoutine) drei verschiedene Möglichkeiten:

1) der Zeileneditor wird ohne vorgegebene Daten aufgerufen. Dies erweist sich bei Zahleneingaben häufig als sinnvoll. Beispiel: *0 0 5 LEDIT.*

2) um dem Benutzer einen naheliegenden Vorschlag für die Eingabe zu machen, kann der Zeileneditor mit einem Default-Wert aufgerufen werden. Beispiel: *"STANDARD.DFV" COUNT 12 LEDIT* (bei F-PC ist das COUNT nicht nötig).

3) dem Benutzer wird eine evtl. bereits früher gemachte Eingabe zur Änderung vorgelegt. Falls es sich hierbei um eine Zahl handelt, muß diese von einer geeigneten Routine in einen String umgewandelt werden. Beispiel: *Filename COUNT 12 LEDIT* für die Eingabe eines in der Stringvariablen 'Filename' abgelegten Dateinamens, oder *Index @ (U.) 4 LEDIT* für die Eingabe einer in der Variablen 'Index' gespeicherten vorzeichenlosen Zahl einfacher Länge.

Editiert wird übrigens nie der vorgegebene Originalstring selbst, sondern nur eine in den dynamischen Stringbuffer gebrachte Kopie von diesem. Es ist deshalb nicht nötig, von den zu editierenden Strings Sicherheitskopien anzufertigen.

Als Resultat legt der Zeileneditor Adresse und Länge des Speicherbereichs auf den Stack, an dem sich die eingegebenen Daten befinden. Sollte der String die Länge Null

besitzen, so ist die Adresse belanglos und wird daher gar nicht erst übergeben.

Zum Editieren stehen dem Benutzer die Tasten BACKSPACE, CURSOR LEFT, CURSOR RIGHT, HOME, END und DEL zur Verfügung, welche ihre üblichen Funktionen ausführen. Die Taste INS dient als Umschalter zwischen Einfüge- und Überschreibmodus, wobei auch die Größe des Cursors geändert wird.

Zum Abbrechen einer Eingabe dient wie üblich die ESC-Taste. Wie im Fall einer leeren Eingabe liegt danach auf dem Stack lediglich eine Null, deren Herkunft sich aber durch Abfragen der globalen Variable 'Esc' sofort klären läßt. Diese Variable enthält nämlich nur dann ein wahres Flag, wenn die letzte Eingabe per ESC abgebrochen wurde.

Außerdem möchte ich noch auf eine besondere Eigenart des Zeileneditors hinweisen, welche sich als überaus praktisch erwiesen hat: wenn die erste vom Benutzer betätigte Taste ein gültiges Zeichen liefert, so geht der Zeileneditor davon aus, daß der vorgegebene String gar nicht editiert werden soll. Ansonsten hätte der Benutzer nämlich zuerst die Cursortasten benützt, um den Cursor an die richtige Position zu bringen. Er löscht deshalb den vorgegebenen String, um Platz für eine Neueingabe zu schaffen.

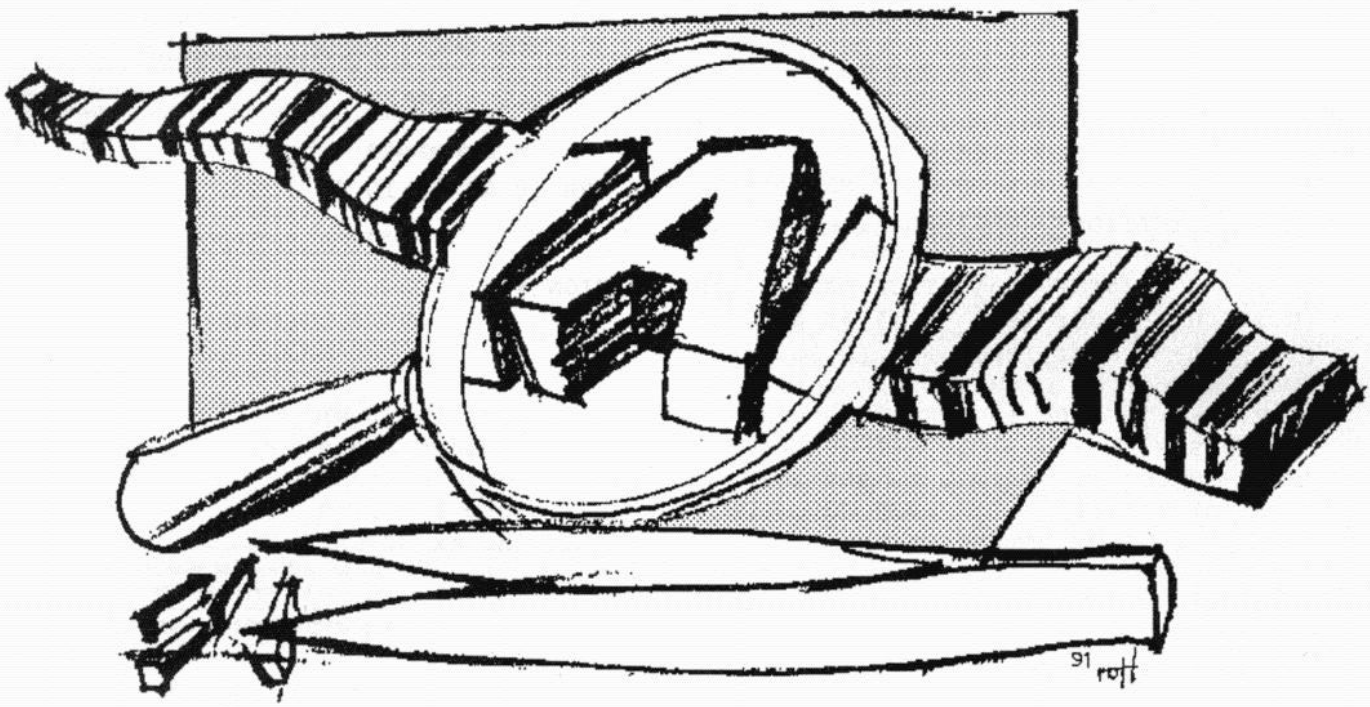
Korrektur und Überprüfung von Zeichen

Die beiden vektorisierten Routinen FIXUP und VALID? wurden unmittelbar nach ihrer Definition in Screen 4 per NOOP zum Nichtstun bestimmt. Das bedeutet, daß der Zeileneditor zunächst jedes beliebige Zeichen akzeptieren wird, ohne es vorher zu korrigieren oder auf Zulässigkeit hin zu überprüfen.

In Screen 9 werden nun etwas sinnvollere Routinen definiert, die sich für den Einsatz in FIXUP eignen. Die Routine DECIMAL-NUMBER z.B. ist für die Eingabe von Zahlen vorgesehen und wird jedes eingegebene Komma sofort in einen Punkt umwandeln. Außerdem muß bei Eingabe von Zahlen im Exponentialformat ein eventuell klein geschrieben vorgefundenes 'e' per UPC in den entsprechenden Großbuchstaben umgewandelt werden.

Zur Überprüfung der Zulässigkeit von Zeichen durch das Wort VALID? wird hier eine Routine namens ALLOWED? eingesetzt, welche in den Screens 10-11 zu finden ist. Die Routine basiert auf der Idee, alle in einer bestimmten Eingaberoutine erlaubten Zeichen in Form einer Tabelle zu speichern [2]. Dadurch können sämtliche Tasten gesperrt werden, welche mit Sicherheit zu einer fehlerhaften Eingabe führen würden. Für Zahleneingaben z.B. dürfen nur Ziffern vorwendet werden, eventuell auch noch das Minuszeichen, der Dezimalpunkt, und ein großes 'E' für Zahlen im Exponentialformat. Bekanntlich darf ein für MS-DOS bestimmter Dateiname ebenfalls nur ganz bestimmte Zeichen enthalten, die Regelung hierfür ist jedoch wesentlich komplizierter. Bei der Eingabe eines Dateinamens sollte sich der Anwender darüber nicht den Kopf zerbrechen müssen, sondern im Zweifelsfall auf gut Glück Zeichen eingeben dürfen. Von der Vergleichstabelle als ungültig erkannte Zeichen werden schließlich nicht akzeptiert und können daher keinen weiteren Schaden anrichten.

Um den Platzbedarf von Vergleichstabellen gering zu halten, werden die einzelnen *Bits* einer Zahl als Flag dafür verwendet, ob der entsprechende Buchstabe erlaubt ist oder nicht. Dann sind für die 256 möglichen Zeichen lediglich 32 Byte Speicherplatz nötig; für viele



Anwendungen reichen sogar noch kürzere Tabellen von nur 16 Byte Länge aus, welche den ASCII-Zeichen von 0 bis 127 entsprechen. In diesem Fall müssen allerdings sämtliche Zeichen oberhalb von 127 auf das als ungültig markierte Zeichen 127 abgelenkt werden, was von der Routine SHORT besorgt wird.

Im Gegensatz zur Lösung des Problems in [2] ist bei der vorliegenden Implementation kein besonderer Generator zum Erstellen eigener Tabellen nötig - die nötigen Werte können direkt an einer in Achtereinheiten unterteilten ASCII-Tabelle abgelesen werden. Dazu ein Beispiel: laut Screen 10 repräsentiert Byte 8 jeder Vergleichstabelle die Zeichen @ABCDEFG. Wird dieses Byte z.B. auf den hexadezimalen Wert 72 gesetzt (binär 01110010), so werden die Zeichen ABCF als erlaubt markiert, die Zeichen @DE dagegen nicht. Beachtet werden muß hierbei stets, daß die Bytes mit Null beginnend durchnumeriert sind.

Auf diese Weise erstellte Vergleichstabellen und darauf basierende Eingaberoutinen sind in den Screens 12-15 enthalten. Diese Routinen sind bereits in der Lage, einen großen Teil an Eingabefehlern von vornherein abzufangen. Natürlich wurden hier noch nicht alle Feinheiten berücksichtigt: Zahlen mit mehreren Dezimalpunkten, mit dem Vorzeichen mitten in der Zahl etc. führen noch immer zu Fehlern, die nur durch zusätzliche Routinen in VALID? verhindert werden könnten. Die Bewältigung der verbleibenden Probleme soll aber programmierfreudigen Lesern überlassen werden.

Hinweise für Benutzer von F-PC

Um das Listing auch mit F-PC verwenden zu können, sind einige Modifikationen nötig. Vor dem Compilieren muß unbedingt CAPS OFF eingegeben werden, sonst sorgen die Worte 'State' und 'Limit' für Verwirrung. Außerdem ist es nötig, einen dynamischen Stringbuffer [3] anzulegen.

Die Cursorgröße kann mit den im F-PC bereits vorhandenen Worten BIG-CURSOR und NORM-CURSOR geändert werden, so daß die entsprechenden Definitionen entfallen. Auch enthält F-PC erfreulicherweise bereits alle nötigen Routinen zum Umwandeln von Zahlen in formatierte Strings, wie (.), (D.), (U.) etc.

Ersetzt man jetzt noch GOTOXY durch AT, ?XY durch AT?, 2000 10 BEEP durch 2000 1 TONE, ENDCASE durch DROP ENDCASE, und definiert ferner

```
: BINARY ( -- )  2 BASE ! ;

: PCKEY ( -- n | n 0)

 BIOSKEY SPLIT SWAP
  DUP IF  NIP THEN ;
```

so steht einem Einsatz des Zeileneditors unter F-PC nichts mehr im Wege. An dieser Stelle möchte ich noch Herrn Jörg Staben danken, der mir den F-PC Compiler zur Verfügung gestellt hat.

Quellen

[1] Leo Brodie:
Starting FORTH
Second Edition 1987
Prentice Hall, New Jersey

[2] Michael Ham:
'Factoring in Forth'
Dr. Dobb's Toolbook
of Forth II
First Edition 1987
M&T Publishing, California

[3] Heinrich Hohl:
'Der dynamische
Stringbuffer'
Vierte Dimension 7/2,
Juni 1991


```
Screen #01
\ Load screen                                    HH 18:20 03.10.91
 FORTH DEFINITIONS  DECIMAL

: (D.) ( d -- addr len)  TUCK DABS <# #S ROT SIGN #> ;
: (U.) ( u -- addr len)  0 <# #S #> ;
: (.)  ( n -- addr len)  S>D (D.) ;

: BIPP ( -- )  2000 10 BEEP ;           \ a short and high beep
: BIG-CURSOR ( -- )  1 12 SET-CUR ;     \ used for insert state
: SMALL-CURSOR ( -- )  11 12 SET-CUR ;  \ used in replace state
: S! ( ^str1 ^str2 -- )  OVER C@ 1+ CMOVE ;    \ store a string
  HERE BL C, 1 2CONSTANT Blank          \ used as 'Blank STRCAT'

  2 16 THRU
  EXCISE Chars RESULT
  EXCISE Masks Masks
```

Screen #02

```
\ Line editor /1/                                HH 18:20 03.10.91
\ Global variables:
 VARIABLE Esc              \ data input finished using ESC key?
 VARIABLE 'FIXUP           \ vectored execution of FIXUP
 VARIABLE 'VALID?          \ vectored execution of VALID?

\ Local variables:
 VARIABLE Chars            \ number of characters already input
 VARIABLE First            \ first key pressed by user?
 VARIABLE Limit            \ maximum number of chars to input
 VARIABLE Pos              \ current input position in string
 VARIABLE State            \ editor state: -1=insert, 0=replace
```

Screen #03

```
\ Line editor /2/                                HH 18:20 03.10.91
: SHOW-CURSOR ( -- )
  State @  IF BIG-CURSOR ELSE SMALL-CURSOR THEN ;

: ALONG ( n -- )  DUP Pos +!  ?XY >R + R> GOTOXY ;
: BACK  ( n -- )  NEGATE ALONG ;
\ move cursor n chars along or back

: ADDRESS ( -- addr)  STRBUF  Pos @  + ;
\ address corresponding to current input position in string

: REST ( -- n)  Chars @  Pos @  - ;
\ length of string to the right of current input position

: ROOM? ( -- ?)  Chars @  Limit @  U< ;
\ true if there is room left to insert one more character
```

Screen #04

```
\ Line editor /3/                                HH 18:20 03.10.91
: FIXUP  ( char -- char')  'FIXUP  PERFORM ;  ' NOOP 'FIXUP  !
: VALID? ( char -- ?)      'VALID? PERFORM ;  ' NOOP 'VALID? !

: STORE ( char -- )  DUP ADDRESS C!  EMIT  1 Pos +! ;
\ store character at current input position and emit it

: INSERT ( char -- )
  ROOM?
  IF  REST IF ADDRESS DUP 1+ REST CMOVE> THEN
      STORE  1 Chars +!
      ?XY  ADDRESS REST TYPE  GOTOXY
  ELSE DROP BIPP THEN ;

: REPLACE ( char -- )  REST IF STORE ELSE INSERT THEN ;
```

```
Screen #05
\ Line editor /4/                                  HH 18:20 03.10.91
: BACKSPACE ( -- )
  Pos @ 0>
  IF  ADDRESS DUP 1- REST CMOVE  -1 Chars +!  1 BACK
      ?XY  ADDRESS REST TYPE SPACE  GOTOXY
  ELSE BIPP THEN ;

: ESCAPE ( -- )  Chars OFF  Esc ON ;

: CHARACTER ( char -- )
  FIXUP  DUP VALID?
  IF  First @
      IF  ?XY REST SPACES GOTOXY  Chars OFF  THEN
      State @
      IF INSERT ELSE REPLACE THEN
  ELSE DROP BIPP THEN ;


Screen #06
\ Line editor /5/                                  HH 18:20 03.10.91
: LEFT ( -- )
  Pos @  0>
  IF 1 BACK ELSE BIPP THEN ;

: RIGHT ( -- )
  Pos @  Chars @  U<
  IF 1 ALONG ELSE BIPP THEN ;

: HOME ( -- )  Pos @ BACK ;
: END  ( -- )  REST ALONG ;

: INS ( -- )  State @  0=  State !  SHOW-CURSOR ;
: DEL ( -- )  REST  IF 1 ALONG BACKSPACE ELSE BIPP THEN ;



Screen #07
\ Line editor /6/                                  HH 18:20 03.10.91
: NORMAL ( char -- ?)
  FALSE SWAP
  CASE  08 OF BACKSPACE ENDOF
        13 OF         0= ENDOF
        27 OF ESCAPE 0= ENDOF
        DUP CHARACTER
  ENDCASE ;

: EXTENDED ( char -- ?)
  FALSE SWAP
  CASE  75 OF LEFT ENDOF    77 OF RIGHT ENDOF
        71 OF HOME ENDOF    79 OF END   ENDOF
        82 OF INS  ENDOF    83 OF DEL   ENDOF
       BIPP
 ENDCASE ;
```

Screen #08

```
\ Line editor /7/                                HH 18:20 03.10.91
: INIT ( addr len #chars -- )
  DUP +STRBUF Limit !  DUP Chars !
  ?XY 2OVER TYPE GOTOXY  STRBUF SWAP CMOVE
  Esc OFF  First ON  Pos OFF  State OFF ;

: RESULT ( -- addr' len' |0)
  Chars @  DUP IF STRBUF SWAP THEN ;

: LEDIT ( addr len #chars -- addr' len' |0)
  INIT SHOW-CURSOR
  BEGIN  PCKEY ?DUP
         IF NORMAL ELSE EXTENDED THEN  First OFF
  UNTIL  RESULT SMALL-CURSOR ;
\ Edit a copy of the given string and accept up to #chars
\ characters for this string.
```

Screen #09

```
\ Fix possible input mistakes                    HH 18:20 03.10.91
: UPC ( char -- char')
  DUP  ASCII a <  OVER  ASCII z >  OR
  NOT  IF 32 XOR THEN ;
\ force lower case characters to upper case

: SHORT  ( char -- char')  127 MIN ;
\ neccessary if short comparison table is used (chars 0-127)

: CAPITAL ( char -- char')  SHORT UPC ;
\ input has to be made in capital characters

: DECIMAL-NUMBER ( char -- char')
  DUP ASCII , =  IF 2+ THEN  CAPITAL ;
\ a komma was meant to be a decimal point
```

Screen #10

```
\ Character comparison tables /1/                HH 18:20 03.10.91
\\ Comparison tables contain up to 32 bytes, numbered 0-31.
The most important characters are listed here (to allow the '§'
character set byte 2 to 04). Byte 4 starts with the blank char.
Byte  4 represents:   ! " # $ % & '
Byte  5 represents: ( ) * + , - . /
Byte  6 represents: 0 1 2 3 4 5 6 7
Byte  7 represents: 8 9 : ; < = > ?
Byte  8 represents: @ A B C D E F G
Byte  9 represents: H I J K L M N O
Byte 10 represents: P Q R S T U V W
Byte 11 represents: X Y Z [ \ ] ^ _
Byte 12 represents: `a b c d e f g
Byte 13 represents: h i j k l m n o
Byte 14 represents: p q r s t u v w
Byte 15 represents: x y z { | } ~
```

```
Screen #11
\ Character comparison tables /2/              HH 18:20 03.10.91
  VARIABLE Allowed \ stores address of comparison table to use

  CREATE Masks \ bit masks to test state of single bits
  BINARY 10000000 C, 01000000 C, 00100000 C, 00010000 C,
         00001000 C, 00000100 C, 00000010 C, 00000001 C,
  DECIMAL

: ALLOWED? ( char -- ?)
  8 /MOD                        \ calculate mask# and byte#
  Allowed @ + C@                \ fetch corresponding byte
  SWAP Masks + C@               \ fetch corresponding bit mask
  AND 0<> ;
\ true if given character is allowed by comparison table

  ' ALLOWED? 'VALID? !          \ default checking method



Screen #12
\ Character comparison tables /3/              HH 18:20 03.10.91
CREATE {Single} \ single length number
  HEX 00 C, 00 C, 00 C, 00 C, 00 C, 04 C, FF C, C0 C,
      00 C, 00 C, 00 C, 00 C, 00 C, 00 C, 00 C, 00 C,
CREATE {uSingle} \ unsigned single length number
      00 C, 00 C, 00 C, 00 C, 00 C, 00 C, FF C, C0 C,
      00 C, 00 C, 00 C, 00 C, 00 C, 00 C, 00 C, 00 C,
CREATE {Double} \ double length number
      00 C, 00 C, 00 C, 00 C, 00 C, 06 C, FF C, C0 C,
      00 C, 00 C, 00 C, 00 C, 00 C, 00 C, 00 C, 00 C,
CREATE {uDouble} \ unsigned double length number
      00 C, 00 C, 00 C, 00 C, 00 C, 02 C, FF C, C0 C,
      00 C, 00 C, 00 C, 00 C, 00 C, 00 C, 00 C, 00 C,
CREATE {Float} \ floating point number
      00 C, 00 C, 00 C, 00 C, 00 C, 06 C, FF C, C0 C,
      04 C, 00 C, 00 C, 00 C, 00 C, 00 C, 00 C, 00 C,



Screen #13
\ Character comparison tables /4/              HH 18:20 03.10.91
CREATE {File} \ MS-DOS file name with extension
      00 C, 00 C, 00 C, 00 C, 5F C, C6 C, FF C, C0 C,
      FF C, FF C, FF C, E3 C, FF C, FF C, FF C, F6 C,
CREATE {Wild} \ as {File}, but also allow '*' and '?'
      00 C, 00 C, 00 C, 00 C, 5F C, E6 C, FF C, C1 C,
      FF C, FF C, FF C, E3 C, FF C, FF C, FF C, F6 C,
CREATE {Path} \ as {File}, but also allow ':' and '\'
      00 C, 00 C, 00 C, 00 C, 5F C, C6 C, FF C, E0 C,
      FF C, FF C, FF C, EB C, FF C, FF C, FF C, F6 C,
CREATE {Text} \ long table for text with any characters
      00 C, 00 C, 04 C, 00 C, FF C, FF C, FF C, FF C,
      FF C, FF C, FF C, FF C, FF C, FF C, FF C, FE C,
      FF C, FF C, FF C, FF C, FF C, FF C, FF C, FF C,
      FF C, FF C, FF C, FF C, FF C, FF C, FF C, FF C,
DECIMAL
```

```
Screen #14
\ Specific input routines /1/                    HH 18:20 03.10.91
: STR>D ( addr len -- d -1 |0)
  Blank STRCAT STRPCK  NUMBER?  IF -1 ELSE 2DROP 0 THEN ;
\ convert string to double number or return false flag

: #INPUT ( addr len #chars -- n -1 |0)
  {Single} Allowed !  ['] SHORT 'FIXUP !
  LEDIT  DUP IF STR>D THEN  DUP IF NIP THEN ;

: U#INPUT ( addr len #chars -- u -1 |0)
  {uSingle} Allowed !  ['] SHORT 'FIXUP !
  LEDIT  DUP IF STR>D THEN  DUP IF NIP THEN ;

: D#INPUT ( addr len #chars -- d -1 |0)
  {Double} Allowed !  ['] DECIMAL-NUMBER 'FIXUP !
  LEDIT  DUP IF STR>D THEN ;


Screen #15
\ Specific input routines /2/                    HH 18:20 03.10.91
: UD#INPUT ( addr len #chars -- ud -1 |0)
  {uDouble} Allowed !  ['] DECIMAL-NUMBER 'FIXUP !
  LEDIT  DUP IF STR>D THEN ;

: FILE-INPUT ( addr len #chars -- ^str -1 |0)
  {File} Allowed !  ['] CAPITAL 'FIXUP !
  LEDIT  DUP IF STRPCK -1 THEN ;

: PATH-INPUT ( addr len #chars -- ^str -1 |0)
  {Path} Allowed !  ['] CAPITAL 'FIXUP !
  LEDIT  DUP IF STRPCK -1 THEN ;

: TEXT-INPUT ( addr len #chars -- ^str -1 |0)
  {Text} Allowed !  ['] NOOP 'FIXUP !
  LEDIT  DUP IF STRPCK -1 THEN ;


Screen #16
\ Demos                                          HH 18:20 03.10.91
  VARIABLE Index
: DEMO1 ( -- )
  CR ." Input an index: " Index @ (U.) 4 U#INPUT CR
  IF    Index !
  ELSE  Esc @ IF EXIT THEN
        ." Bad input, not accepted." CR
  THEN  ." Index now contents: " Index @ U. CR ;

  CREATE Name 0 C, 12 ALLOT
: DEMO2 ( -- )
  CR ." Input a filename: " Name COUNT 12 FILE-INPUT CR
  IF    Name S!
  ELSE  Esc @ IF EXIT THEN
        ." Bad input, not accepted." CR
  THEN  ." Filename now contents: " Name COUNT TYPE CR ;
```

Direct Threaded Code am Beispiel F-PC

Create Does en Detail

arndt klingelnberg

Ein Einblick in den Aufbau von F-PC Code, PC-FORTH (fast) ohne seg:offset Probleme.

F-PC v. 3.50 ist ein interessantes Forth System für den PC, da es all den Speicher, den DOS überhaupt zur Verfügung stellt, auch nutzen kann. F-PC muß natürlich mit den Vor-//Nachteilen des intel//microsoft//ibm Triumvirats leben und versucht das beste daraus zu machen. F-PC verteilt sich auf mindestens 3 Segmente, 2 davon sind max. 64kByte groß, das 3. könnte theoretisch 1 MB betragen, wenn dies denn DOS genehm wäre. Für Editor, Buffer usw. können natürlich noch beliebig weitere Segmente 'ALLOCiert' werden.

Spezielle Erweiterungen außer acht lassend, möchte ich hier die drei Segmente des F-PC kurz erklären und die Aktionen anhand des CREATE DOES Beispiels erläutern. Das Schaubild ist übrigens über HyperHelp zugänglich (F1 Forth-Help INTERNALS CREATE-DOESs), allerdings wurde es erweitert (sagte ich korrigiert?!?).

Im HauptSegment (Code CS:) befinden sich sämtliche Maschinensprache-Sequenzen inklusive 'schnellen' Daten. Unter schnelle Daten fällt z.B. eine char-Umwandlungstabelle in 7bit GROSSbuchstaben oder ein Feld von 80 (!) spaces zum schnellen TYPE derselben (KEIN DO BL EMIT LOOP).

Header oder CodeFieldAddress-Listen wie auch die speicherplatzfressenden ." strings liegen im HeadSegment (Y) bzw. ListSegment (X). Alle Addressen sind über lange Speicheropteratoren (L@ L! ...) ansprechbar und zwar in der auch bei i86 ASM üblichen seg:offset Art. Sie benötigen also auf dem Stack ZWEI 16bit Werte und zwar die Paragraphen-Adresse (um 4bit verschobene Adresse) und den Offset relativ hierzu. Das ergibt 20bit AdressRaum wie bei allen üblichen Programmen unter DOS. Dazu gibt es vereinfachte Operatoren die direkt auf spezielle Segmente greifen, wie z.B. Ydump Xdump . (EMS wird unterstützt, entweder zur Auslagerung der Header oder durch dynamisches ALLOCieren von 4*16 kB aus weiteren 15 MB.)

Da Forth für den Anwender fast immer in CS abläuft, genügen die üblichen @ ! ... Operatoren fast immer, um die platzsparende Arbeitsweise von ." oder die Auslagerung der CFA-Listen muß er sich ja nicht kümmern, es sei denn, er lege selbst Hand an das System.

Durch HEADERLESS kann der Headerbereich zusätzlich stark entlastet werden. Auch große Programme können so leicht wartbar d.h. modular sein. Zudem kann das Head-Segment eben auch aus den 640 kB heraus in den Extended Speicher (EMS) gelegt werden.
(Eine Bemerkung am Rande: verwenden Sie NUR NEAT-boards, und zwar auch für 386er- Systeme, der Speicher oberhalb 640 kB ist flexibler nutzbar. Mittlerweile gibt es auch andere Hardware-Lösungen. Weiterhin notwendig: ein Memory Manager, möglichst besser als MSDOS 5.)

Das XSegment bzw. ListSegment verdankt seine einfache Adressierung der Tatsache, daß dort sehr oft nur Paragraph Boundaries adressiert werden, 16b-Adressen reichen also für 1MB aus. Als Nachteil wird über PARAGRAPH (ein 16b-align) nach jeder Colon-Defintion einige Bytes sinnlos weggeworfen, und zwar durchschnittlich knapp 8 Byte je Definition.

siehe Zeichnung Seite 29:

Das Beispiel: CREATE DOES

Anhand des Aufbaues von CREATE DOES Definitionen kann die Aufteilung in CodeSegment und ListSegment gut aufgezeigt werden. Zudem ist auch erkennbar, wie ein DIRECT THREADED CODE Forth aufgebaut ist.

GREMIT erzeugt Forth-Wörtchen, die gerade mal einen 8bit (graphic) char ausgeben, das wird u.a. für die intelligente Editor Umrahmung genutzt, das Wörtchen HAPPY ist wirklich nur für SHOW-zwecke implementiert.

In F-PC enthält das CodeSegment durchweg MaschinenCode und Pointer (displacements == relative Sprünge im CodeSegment oder Pointer in das LIST-Segmet == absolute offsets im XSeg). Colon-

Defintions enthalten in der CFA einen E9-JMP zum nachfolgenden 16bit-NEST-displacement. NEST wiederum arbeitet mit dem darauffolgenden Pointer (angeordent im BODY) und springt unter Berücksichtigung von XSeg an den zugehörigen Paragraphen im List Segment (X) und arbeitet dort die jeweiligen (16bit) CFAs ab. UNNEST schließt die CFA Liste einer jeden Colon-Defintion ab und ermöglicht es, dann wieder zurückzuspringen.

So arbeitet GREMIT die CFAs von CREATE C, (;CODE) ab.

Nach (;Code) wird wieder ins CODE Segment zurückgesprungen und dort weiterer Code abgelegt, und zwar die DOES Aktionen kodiert: ein E8-CALL mit nachfolgendem Displacement zum DOdoes Label. Dazu gehört wiederum ein Pointer in den ListSpace und dort die von dem 'Abkömmling' von GREMIT abzuarbeitetenden CFAs. Der Anfang liegt auf einer 16bit-PARAGRAPH boundary. Es wird '1' auf den Stack gelegt (LIT) und mit TYPE wird EIN char ab der Body-Adresse von Happy ausgegeben, in diesem Fall eben eine ASCII 2. Über DOES wurde ja bereits die BODY-Adresse auf den Stack gelegt.

Happy kostet nur noch 4 Byte im Code Space und bei einem 5char langen Namen 12 Byte Header Space (7 Byte Header Overhead). List Space wird für die einzelnen Kinder eines Create-Wortes nicht benötigt.

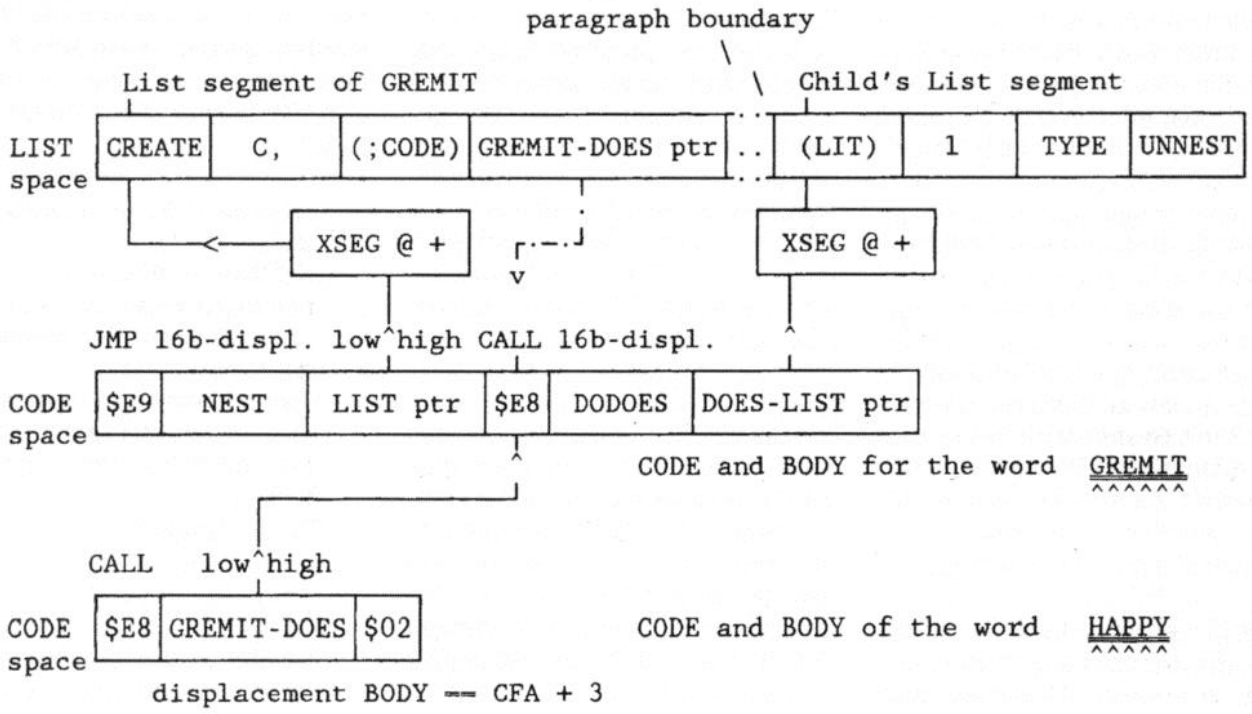
GREMIT selbst benötigt 10 Byte Codespace , 32 Byte entsprechend 2 Paragraphen List space (davon 16 Byte ungenutzt) und 13 Byte Header space.

Zur Vertiefung wird Vack 'Programmieren mit FORTH' empfohlen. Direkter Fadencode wie auch andere Implementationen sind dort gut erklärt (auch mit Skizzen).

```
: GREMIT                    \ creates words, that will type  8b-chars
    CREATE                  ( compiletime: nl <name> -- )
                c,
    DOES>                   ( child runtime:  -- )
                1 type ;

$02   GREMIT happy          \  2 is the hex value for "●" a HAPPY face.
```

F-PC Internal Structure



Die Stacks, TIB, PAD und HERE

Ein System-Einstieg nicht nur in F-PC

Happy RETURN Stack Loading wünscht

arndt klingelnberg

Die Anordnung der Stacks und HilfsBuffer ist in vielen FORTH-Systemen sehr ähnlich. Als Beispiel ziehe ich daher ein mittlerweile von mir in das F-PC HyperHelpSystem integrierte HyperSkizze heran. Hyper deshalb, weil mit Mouse oder (Shift) TAB zwischen den signifikanten Forth Wörtchen herumgesprungen werden kann (auf dem Bildschirm in REVERSE-Feldern, in der Skizze unterstrichen) und sofort ShadowHelpText und SourceText zur Verfügung stehen (das 'sofort' muß natürlich bei DiskettenSystemen relativiert werden) .

Bei dieser Betrachtung spielt es keine Rolle, daß F-PC im Code Segment, dh. im unteren Teil, dem Dictionary, keinerlei Platz verbraucht für HEADER und für CFA Listen , da diese platzsparend in andere Segmente ausgelagert sind. Bei allen diskussionswürdigen Gedanken zur Segmentierung und intels REAL mode schafft das Aufteilen in Segmente ein gewisse Aufgeräumtheit und gleichzeitig die Möglichkeit im 16bit Bereich zu bleiben. (Gegenüber 32bit Systemen entfernt man sich nicht so meilenweit von 8bit embedded controllern, was ich als Vorteil empfinde. embedded == 'to fix firmly in a surrounding mass'.)

Beginnen wir unten in CodeSegment: Hier befindet sich MaschinenCode, in anderen Systemen eben auch noch die Header und die CFA Listen von Colon-Definitions. HERE wächst als obere Grenze des Dictionarys mit jeder neuen Defintion. PAD wird benutzt für ALLE Zahlenausgaben. < # beginnt einen string von PAD aus nach unten zu bilden, die hinteren DIGITs, CHARs zuerst. Über HLD und HOLD kann man übrigens leicht einen string aus beliebigen chars CATENATEn , es müssen nicht immer Ziffern sein. PAD ist 'floating' , d.h. er wandert (hier in 80 Byte Abstand) abhängig von HERE . PAD ist ein vielseitiger temporärer Buffer, eben auch zum Verbinden von strings (Ende zuerst), die Betonung liegt auf temporär.

Springen wir ganz nach oben. 'Oben' kann je nach vorhandenem Speicher bzw. genutzter Addressierungsart bzw. gewünschter Systemgröße vorgegeben sein bzw. gewählt werden (bei intel86 begrenzt auf 64 kB). Hier befindet sich der ReturnStack, das Element im Forth, was sich so über Rekursionen freut und das FORTH für Anfänger bei EXITs aus DO-LOOPs heraus so interessant macht.

Die (DOS) commando line beim Aufruf von F-PC wird in den CMDBUF übernommen und dort interpretiert. (TIB wird dorthin temporär umgelegt). Hier klemmte einiges, der Return Stack überschrieb die CommandoLine. Aus diesem Grund entstand dieses Schaubild zur bequemen Analyse und als Konsequenz letztendlich wurde mehr Platz für den ReturnStack geschaffen (nun $100==256 Byte==128 Einträge). Nested LOADs (laden eines Quelltextes von einem anderen aus) laden den ReturnStack jedesmal mit gut 30 Werten (60 Bytes), da sollte Platz vorhanden sein.

Unterhalb von ReturnStack und CMDBUF ist der TerminalInputBuffer angesiedelt. Dort liegt alldas, was im FORTH-direct modus eingetippt wird. Die Variable < IN gibt an, bis wohin der EingabeString bereits schon interpretiert wurde. Einige FORTH-Worte nutzen < IN , um WORDs mehrmals zu bearbeiten.

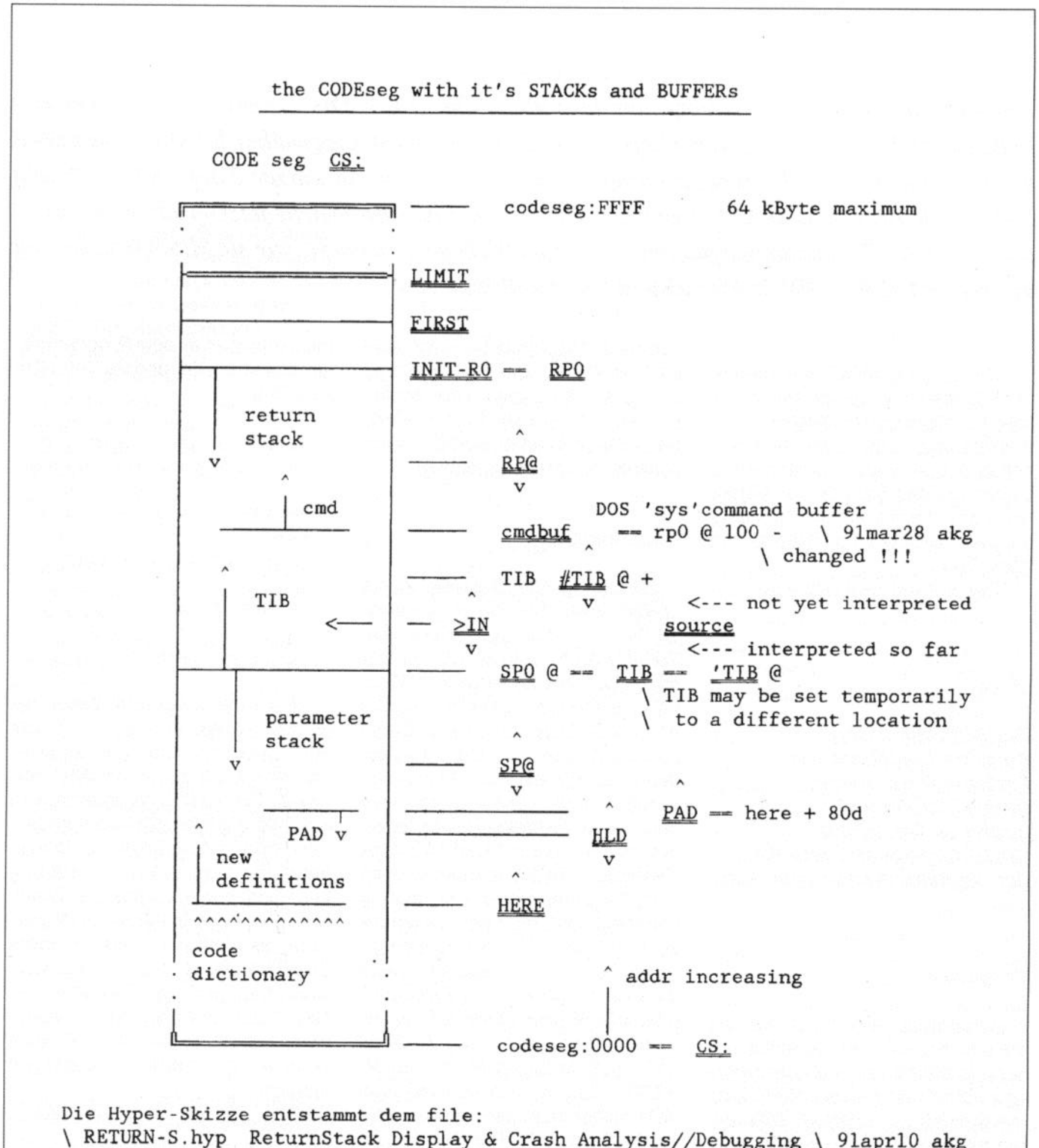
Wiederum darunter befindet sich der ParameterStack, der Stack also, dem die üblichen StackKommentare gelten und in dem herumgeSWAPpt wird. Hier entzünden sich die Geister, wenn es um UPN geht:

> UPN == RPN == reversed polish notation
> Bill Hewlett drehte die polnische Schreibweise um, und gab die Zahlen immer zuerst ein.
> Jan Lukasiewics stellte die Rechenzeichen (auch mehrfach gruppiert, ohne Nutzung von Klammern) *vor* die Zahlen.

Bei 8bit Controllern werden alle Stacks oft unten im Speicherbereich angeordnet. Es ergeben sich Vorteile einer einfachen schnellen PAGE ZERO Adressierung oder Nutzung

interner µP-Register. Ferner muß die Speicherteilung in ROM und RAM beachtet werden. Stacks, Variablen und DEFERred Worte laufen nun mal recht schlecht im ROM. Ansonsten sind die Verflechtungen der Stacks und Buffer aber auch dort recht ähnlich.

Bei besonderen Anforderungen an das System bzw. 'unerklärlichen' Fehlern sollte ruhig mal eine Analyse der gegenseitigen Interaktionen erfolgen. Für Debugging Zwecke habe ich ein .SR geschaffen (display of return stack content) wie auch ein Crash-Info , das mir die Reserven zwischen den verschiedenen Buffern//Stacks anzeigt, für ímmediate' Anwendungen innerhalb von COLON-Definitions (zum Debuggen) als [.SR] oder [crash-info] anwenden. Auch hierdurch wurde mir wiederum ein wenig neue Erkenntnis zuteil, unter anderem eben, wie doch so etwas wie ein .SR ein ganz anderes Verständnis für FORTH bzw. maschinennahe Programmierung geben kann, und so entbrannte auch die Liebe zu n FOR...NEXT Schleifen.



Die Hyper-Skizze entstammt dem file:
\ RETURN-S.hyp ReturnStack Display & Crash Analysis//Debugging \ 91apr10 akg

dpANS

Mitch Bradley

Übersetzung von Michael Kalus

Der Entwurf des Standards liegt nun der Öffentlichkeit vor. Die Lesungen (public review) laufen jetzt, bis der Standard schließlich angenommen wird. Gegenüber BASIS15 sind doch noch einmal Einwände berücksichtigt worden. Es kamen in kurzer Folge BASIS16 und BASIS17 heraus. Das dpANS-2 vom 3.August ist die Fassung, welche jetzt wirklich öffentlich ausliegt. Hier die Zusammenfasung der nun amtlichen Vorschläge, wie sie Mitch Bradley auf der EuroFORML 1991 in Marienbad vorgestellt hat.

Der Vorgang ANS Forth brauchte 18 Treffen in 4 Jahren. Das Kommitee genügte den förmlichen ANSI Anforderungen. Mehr als 1000 Vorschläge von vielen verschieden Leuten wurden beraten. 28 Personen waren Mitglieder des Kommitees. Jetzt arbeiten 19 Personen im Kommitee.

Der Entwurf liegt nun aus.

Ziele

Breite Zustimmung.
Portable Applikationen.
Portable Programme.
Unterstützung vieler moderner Computer.
Unterschiedliche Techniken der Implementation zulassen.

Vorgehen

Grundlage war Forth 83. Es wurden Funktionen spezifiziert, nicht Implementationen. Entwicklung soll erlaubt werden, ohne Code von Kunden zu verletzen. Differenzen zwischen Forth 79 und Forth 83 wurden beigelegt - kontroverse Worte nahm man heraus und bietet nun beide Möglichkeiten unter neuen Namen an. Die Beschreibung der Einschränkungen einer echten Portabilität wurde formalisiert. Neue Möglichkeiten werden in optionalen Wordsets angeboten.

Einzelheiten

Grundlegender Datentyp ist die "Zelle" (cell). Sie hat mindestens 16-Bits kann aber auch größer sein. Portable Arithmetik der Adressen ist mit diesen Worten möglich: CELL+ CELLS CHAR+ CHARS ALIGN ALIGNED. Definitionen und Daten können im gleichen oder verschiedenen Bereichen liegen. Annahmen darüber sind nicht erlaubt. Man kann nur in Parameterfelder hinein adressieren, die mit CREATE oder CREATE ... DOES erzeugt worden sind. Die Zahl, welche von ' oder [']
hinterlegt wird, muß keine Adresse sein. Sie wird "Tätigkeitszeichen" (execution token) genannt. Diese Zahl kann man nur von EXECUTE oder BODY oder COMPILE, verarbeiten lassen. Das wort NOT ist nicht mehr im Standard. 0= und INVERT sind drin, und so kann man NOT selber erzeugen, entweder für Forth-79 oder -83. Die Richtung, in die gerundet wird, bei / oder MOD ist nicht festgelegt. Neue Primitives dazu erlauben es dem Programmierer, das eine oder andere Verhalten zu wählen.

```
UM/MOD ( ud u -- r q )
Division ohne Vorzeichen.

SM/REM ( d n -- r q )
symmetrische Division
- gerundet nach Null

FM/MOD ( d n -- r q )
'floored' Division -
gerundet nach -Unendlich.
```

Ein Standardsystem bietet alle nötigen Worte. Alle weitern Worte mit Namen aus dem Standard müssen sich auch standardmäßig verhalten. Ein Standardprogramm wird auf jedem Standardsystem laufen. Ein Standardprogramm mit Voraussetzungen an seine Umgebung kann bestimmte zusätzliche Eigenschaften des Systems benötigen. Aber man braucht natürlich keine Standardprogramme zu schreiben, wenn man nicht mag. Die "Pärchen-Regel" der Kontrollstrukturen wurde verallgemeinert. Nun sind z.B. auch Schleifen mit mehreren Ausgängen möglich:
BEGIN .. WHILE .. WHILE .. REPEAT THEN

Der Standard ist in mehrere "Wordsets" aufgeteilt. Nur der Kern

(core) ist erforderlich, andere Wordsets werden wahlweise hinzugenommen. Jedes Wordset hat einen Basisteil und einen Teil mit Erweiterungen (extensions). Wenn ein System ein Wordset anbietet, muß es davon alle Basisworte bieten. Von den Erweiterungen brauchen nur einzelne hinzugenommen zu werden. Die Worte brauchen nicht im Wörterbuch (dictionary) vorhanden zu sein. Eine Implementation kann auch verschiedene Worte als Quelltext liefern. Ein Standardprogram kann nur den ASCII-Zeichensatz benutzen.

Standardsysteme müssen ASCII Zeichen und können Erweiterungen bieten. Ein Standardprogramm mit Voraussetzungen an seine Umgebung kann auch erweiterte Zeichensätze benutzen.

Neue Worte

Einige Worte sind neu gegenüber dem Forth 83 (Wie berichtet in VD 2, 3/91). Hier noch einmal eine kurze Übersicht über den neuesten Stand.

ENVIRONMENT? gibt Eigenarten des Systems an. VALUE definiert eine Variable, die ihren Wert selbst auf den Stack legt. Ihr Wert ist unbestimmt, bis ein Wert mit
X TO VARIABLEN-NAME zugewiesen wird.

UNUSED liefert den verbliebenen Speicherplatz. POSTPONE verallgemeinert COMPILE und [COMPILE]. PARSE ist eine verbesserte Version von WORD. Es sucht ebenfalls nach einem Delimiter, schließt dabei aber vorangehende Delimiter ein und hinterläßt auf dem Stack die Adresse der Zeichenkette und deren Länge. Das umstrittene FORGET hat seine Lösung durch den MARKER gefunden. MARKER NAME ist die Stelle bis zu der nun FORGET wirkt. ACCEPT ersetzt EXPECT und löst als einfachere Form einige Probleme auf. ACCEPT wird erst mit dem Zeichen CR abgeschlossen, keine anderen Bedingungen. Die Kontrollstrukturen kennen jetzt CASE OF ENDOF ENDCASE. Eaker's CASE hat damit (erstmal) gewonnen. :NONAME erzeugt eine namenlose Colondefinition und hinterläßt dabei deren Ausführungszeichen auf dem Stack. TRUE und FALSE sind Schreibweisen für Flags geblieben.

Die Bedingungen wurden erweitert auf <> 0> und 0<> neben den schon bekannten. Auch ?DO wurde anerkannt, denn es erlaubt die Konstruktion einer Zähl-Schleife, die auch keinmal ausgeführt werden kann.

Die Double-Number-Operatoren bekamen mit 2>R 2R@ und 2R> noch Zuwachs. CHAR [CHAR] und BL dienen dem Umgang mit einzelnen Zeichen (character literals). Mit UNLOOP kann man eine DO LOOP nun auch vorzeitig über ein EXIT verlassen. Das Schieben von Bits gelingt jetzt mit SHIFT recht gut. Auch RECURSE wurde aufgenommen. RECURSE führt die derzeitige Definition nochmal aus. >NUMBER soll ein verbessertes CONVERT darstellen und wandelt eine Ziffernfolge aus der Eingabe in die interne Form um. COMPILE, arbeitet ähnlich wie das alte , (komma) wird aber für Tätigkeitszeichen (execution token) gebraucht. Mit BYE wird Forth verlassen.

Die weiteren Wordsets enthalten Worte für Block Verwaltung, Double-Number, Error-Handling, Facility, File-Access, Floting-Point, Locals, Memory-Allocation, Programming-Tools, Search-Order, und Strings.

Der üblichen Befehlssatz zum Umgang mit Fließkommazahlen in Arithmetik, Trigonometrie und transzendenten Funktionen wird jetzt angeboten. Für lokale Variablen wurde die Funktionsweise festgelegt, nicht aber die Syntax. Locale Variablen sind "self fetching" und werden nach der VALUE Semantik benutzt. Die Suchfolge hat als Basisoperatoren wenige primitive Worte geschaffen und bietet in der Extentsion die alten Bekannten wie VOCABULARY ONLY ALSO usw. Zur Speicherverwaltung aufgenommen wurden noch ALLOCATE und FREE. Damit wird Speicher vom System angefordert bzw. dahin zurückgegeben. RESIZE ändert die Größe eines bereitgestellten Stücks Speicher und AVAILABLE erfragt das größte bereitstehende Stück.

Die Eingabe (input stream) kann von der Tastatur, vom Block, einem Block in einem File, einer Zeichenkette im Speicher (string) oder einem Textfile kommen. Wenn Textfiles unterstützt werden, müssen Blocks auch unterstützt werden. Textfiles werden Zeile für Zeile abgearbeitet, WORD kann eine Zeilengrenze nicht überschreiten.

EVALUATE interpretiert einen String. REFILL lädt den nächsten Teil der Eingabe. SAVE-INPUT markiert eine Stelle in der Eingabe und RESTORE-INPUT kehrt dorthin zurück. INCLUDE-FILE interpretiert ein bereits geöfnetes File und INCLUDED interpretiert ein File, das mit seinem Namen genannt wird. INCLUDE-FILE verarbeitet ein File-ID und INCLUDED die Adresse und Länge des Filenamen. SOURCE-FILE liefert die Kennung der Quelle, also Tastatur oder String oder File-ID. Und der "Backslash" \ sorgt dafür das der Rest einer Zeile nicht mehr interpretiert wird. Schließlich wird noch die Möglichkeit zu bedingter Compilation geboten mit [IF] [ELSE] [THEN].

Für den Umgang mit Datenfiles gibt es jetzt ebenfals klare Ausdrükke. OPEN-FILE CLOSE-FILE öffnen und schließen Files. READ-FILE WRITE-FILE lesen und schreiben je eine Anzahl Bytes im File und READ-LINE WRITE-LINE tun dies mit einer Zeile. Die Position dabei wird mit FILE-POSITION geholt und mit REPOSITION-FILE gesetzt. Die Größe eines Files liefert FILE-SIZE, RESIZE-FILE setzt diese. FILE-BLOCK FILE-BUFFER usw. meinen Blöcke in Files.

Die Fehlerbehandlung wird nach dem Prinzip CATCH und THROW erfolgen.

KEY wird von der Tastatur nur Standardzeichen holen und keine erweiterten Codes. Dafür gibt es EKEY als Extension. KEY? und EMIT? stellen fest, ob ein Zeichen ansteht bzw. gesendet werden kann. AT-XY setzt den Cursor und PAGE wirft eine Seite aus.

MS liefert eine in Millisekunden definierbare Verzögerung. TI-

Änderungen des Core Wordset gegenüber BASIS15

Aus den Stack-Kommentaren sind die Kürzel w und mask verschwunden.
Sie wurden durch x ersetzt. Das x hat jetzt die Bedeutung "Wort auf dem Stack".

2* und 2/ sind nun doch ausdrücklich Bit-Schiebe-Operation.

BLK wurde auch noch in das Block-Wordset verbannt.

Der Schleifenindex J wurde aufgegeben.

S>D kehrte aus dem Doublenumber- in das Core-Wordset zurück.

SM/MOD wurde umbenannt nach SM/REM und hat damit seinen richtigen Namen wieder.

TRUE wurde wie FALSE verbannt in die Core-Extensions.

WER WILL ?

ein Thema des ANS für unser Heft näher erklären?

* Error-Handling
* File-Access
* Floating-Point
* Locals
* Search-Order
* Strings

Bitte an den Editor wenden!

Euroforml-
Fortsetzung v. S. 18

war mein Beitrag, der wider Erwarten dem Komittee besser zusagte als die übrigen Vorschläge. Die zweite Definition verpflichtete mich dann auch, das Lied vor dem Auditorium anzustimmen...

Obwohl sich viele der Konferenz-Teilnehmer von früheren euroFORMLs her kannten, war die ZuME&DATE gibt die Tageszeit in sechs Items auf den Stack (-- sec min hour day month year).

Bei den Programmierwerkzeugen finden sich die lieben alten Bekannten alle wieder wie
SEE WORD .S ASSEMBLER usw.

Zu den Kontrollstrukturen ist zu sagen, daß BRANCH ?BRANCH MARK usw. verlassen worden sind. Kontrollstrukturen werden gebaut, indem IF THEN BEGIN UNTIL u.s.w. mit Hilfe von POSTPONE eingearbeitet werden.

Die Kontrolle geschieht über Kontrollzeichen (flow token) auf dem Stack. Diese können mit STILL und SO manipuliert werden.

Hinzugekommen ist AHEAD als Vorwärtsverzweigung.

Ausgelassen hat der Standard diesesmal Probleme wie Multitasking und die Unterschiede ROM gegen RAM Programmierung. Diese wurden allerdings im Anhang diskutiert. Ebenfals nicht behandelt wurde DEFER und Worte, um Hardware anzusprechen. Und auch Schwierigkeiten außerhalb der USA mit ANS (internationalization) wurden übergangen.

Euroforml
Fortsetzung

sammensetzung wieder so bunt, daß es für jeden die Möglichkeit gab, neue Kontakte zu knüpfen. So ist es nicht unwahrscheinlich, daß viele Teilnehmer Marienbad mit dem festen Vorsatz verließen, auf alle Fälle bei der euroFORML'92 in Southampton mit dabei sein zu wollen.

Für alle, die dieses Jahr die Chance verpaßt haben, an der euroFORML teilzunehmen, gibt es noch die Möglichkeit Konferenzbände beim

euroFORML Konferenzbüro
c/o Marina Kern
Uhlenhorster Weg 3
2000 Hamburg 76

zu bekommen.

-- FORTH-Gruppen --

FORTH-Gruppen:

W-1000 Berlin
Claus Vogt
Tel. 030/2 16 89 38
Treffen nach Absprache

W-4130 Moers 1
Friederich Prinz
Tel. 02841/5 83 98
Treffen jeden Samstag
14 Uhr, Moerser Arbeitslosenzentrum, Donaustr.1

Gruppe Rhein-Ruhr:
Jörg Plewe
Tel. 0208/42 35 14
W-4000 Düsseldorf
Gebäude des S-Bahnhof Derendorf,
Münsterstraße 199
Treffen jeden ersten Sonnabend im Monat

W-6800 Mannheim
Thomas Prinz
Tel. 06271/28 30
Ewald Rieger
Tel. 06239/86 32
Treffen jeden ersten Mittwoch im Monat im Vereinslokal desSegelvereins Mannheim e.V.
Flugplatz, Mannheim-Neuostheim

W-7000 Stuttgart
Wolf-Helge Neumann
Tel. 0711/88 26 38
Treffen nach Abspache

O-Leipzig
FORTH-Gruppe Leipzig:
Michael Balig, Lützner Plan 17
O-7033 Leipzig
Dr. Jürgen Hesse
Tel. 041/69 56 02
Liselotte-Herrman-Str. 40
O-7050 Leipzig

W-5100 Aachen
Arndt Klingelnberg
Tel. 02404-61648
Treffen jeden ersten Montag als Gruppe des Computer-Club der RWTH, Seminargebäude Raum 214

FORTH für Ratsuchende:

Jörg Staben
Tel. 02103/5 56 09
dienstags und freitags von 20.00-22.00 Uhr

Frank Stüss
Tel. 06187/9 15 03

Karl Schroer
Tel. 02845/2 89 51

Andreas Findewirth
Tel. 05221/2 35 04

Andreas Jennen
W-1000 Berlin, UUCP

Jörg Plewe
Tel. 0208/42 35 14

FORTH Fachgruppen:

W-6800 Mannheim
FIS (FORTH Integriertes System) - Datenbank, Textverarbeitung, Kalkulation
Postadresse:
Dr. med.
Elemer Teshmar
Danziger Baumgang 97
W-6800 Mannheim 31

FORTH Interessengebiete:

volksFORTH/ultraFORTH
Klaus Kohl
Tel. 08233/3 05 24
Klaus Schleisiek-Kern

Künstliche Intelligenz
Ulrich Hoffmann
Tel. 0431/67 88 50

NC4000 Novix Chip
Klaus Schleisiek-Kern
Tel. 040/2 20 25 39

Relationale Netze
HS/Forth
Künstliche Intelligenz
Realtime
Wigand Gawenda
Tel. 040/44 69 41

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

32FORTH
Rainer Aumiller
Tel. 089/6 70 83 55

PostScript/FORTHscript
Christoph Krinninger
Tel. 089/ 7 25 93 82

FORTH im Unterricht
Rolf Kretschmar
Tel. 02401/43 90

Objekt-orientiertes FORTH
Christoph Krinninger
Tel. 089/7 25 93 82
Ulrich Hoffmann
Tel. 0431/67 88 50

F-PC Zimmer FORTH
ASYST (Meßtechnik)
embedded controller
(H8/5xx == TDS2020 fig)
(8051 ... e FORTH ...)
(6511-MiniBee RSCforth)
Arndt Klingelnberg
Tel. 02404/6 16 48

FORTH Fachgruppengründung:

Grafik, Arithmetik
W-7000 Stuttgart 80
Jörg Tomes
Tel. 07 11/7 80 22 93
nur am Wochenende

FORTH Gruppengründung:

W-3300 Braunschweig
Martin Holzapfel
Tel. 05 31/35 12 62
32-Bit Systeme

W-2000 Hamburg
Wigand Gawenda
Tel. 040/44 69 41
Treffen jeden ersten Dienstag im Monat
Themen und Treffen nach Absprache

FORTH-MAILBOX

SYSOP

Jens Wilke

Tel. 089/8 71 45 48
300-2400 baud
Parameter 8N1

(Leere Seite)