Vierte Dimension
Volume III/Nr.3 Oktober 1987

VIERTE DIMENSION

METHODS>
F83 EDITOR
STETTENFELS
MOORE ABOUT FORTH
ERSTES ANS FORTH TREFFEN

FORTH MAGAZIN

5,00 DM

****

W I R B I T T E N U M B E A C H T U N G

DIESER VIERTEN DIMENSION LIEGT EINE EINLADUNG DES VDI BILDUNGSWERKES BEI,

ZU EINEM SEMINAR MIT PRAKTIKUM AM 9. UND 10. DEZEMBER 1987 IN DUESSELDORF

THEMA: F O R T H - IMPLEMENTATION UND ANWENDUNG IN AUTOMATISIERUNGSSYSTEMEN

****

.....kleinanzeige.....

suche noch software/bookware :
'News on NC 4000" No. 1+5 / F83 Handbuch von C.H.THING / FIG-FORTH Listing
fuer 8086 (ggbf.auf Disk)/ Beschreibung zu: QUICKSOFT PC-Write (drucker?)
/Laxen & Perry F83 (wie gebe ich was auf dem Drucker aus?mgl.deutsch).
Symbolischer Debugger/Tracer mit Einzelschritt Betrieb ggf. geteilten Schirm
etc fuer IBM/BIMBOMAT.
Wenn jemand was hat,bitte schreiben (Kosten werden erstattet) an:
THOMAS PRINZ ADALBERT-STIFTER-STR.2 D6930 EBERBACH a/N

.....kleinanzeige.....

euroFORML 1987 proceedings - wenige Restexemplare !!! DM 45,--

Hinweis:
In der Mitte des Heftes befinden sich die nicht gut lesbaren Seiten aus VD III / 1
Bitte entnehmen und in VD III / 1 einlegen.
FORTH - Büro Antilopenstieg 6a - 2000 Hamburg 54

(In der PDF Version nicht mehr enthalten)

- ACHTUNG NEUE ADRESSE -
FORTH GESELLSCHAFT e.V. - Antilopenstieg 6a - D 2000 HAMBURG 54

INHALT

JAHRESTREFFEN
DER
FORTH GESELLSCHAFT
vom

13. bis 15. Mai 1988

in MÜNCHEN, Kolpinghaus

EDITORIAL

Liebe Leser, zunächst einmal muß ich sie um Nachsicht bitten. Wie sie wissen, mache ich dieses Heft in meiner Freizeit für den Verein. Und Freizeit war aus privaten Gründen praktisch nicht da. Daher ist in diesem Heft der Cyclus - Lektor motivieren, diesem die Textfahnen bringen, wieder abholen, Fehler korrigieren, Textfahnen für das Layout drucken - weggefallen, da das sonst weitere Tage gekostet hätte. Und ich hatte die Termine ohnehin schon überschritten. Nun, ich hoffe, es sind nicht allzuviele schlimme Fehler in den Texten und wünsche trotzdem eine vergnügliche Lektüre.

In diesem Heft zum erstenmal dabei Terry Rayburn, ein alter Freund von Charles Moore, der zur Zeit im Forthlab Unternehmensbereich der RSO München beschäftigt ist. Sein Beitrag über Datenstrukturen ist überzeugend.

Bei den Editoren liegt was in der Luft. Die Forth Gemeinde scheint sich anzuschicken, ihr Knowhow darüber endlich mal zusammenzutragen und einen oder gar mehrere Editoren zu erschaffen. Der F83 Editor von Rudolf Mensing und die Leserbriefe regen hoffentlich dazu an, als gemeinsame Tat mit einem state-of-the-art Bildschirmeditor in Forth herauszukommen. Die frei programmierbaren Funktionstasten von Ulrich Hoffmann sind eine weitere Anregung zu diesem Thema.

Norbert Machlitt und Michael Stenzel berichten über ihren erfolgreichen Forth-Einsatz bei ihren Forschungen an Funkenstrecken. Ich würde gerne mehr solcher gelungenen Anwenderberichte hier veröffentlichen.

Chuck Moore schrieb einen ganz persönlichen Beitrag über Forth, seine Programmiersprache, in "More on NC4000". Seine feste Überzeugung "keep it simple" legt er hierin erneut dar. Klaus HH Schleisiek hat übersetzt.

Über den derzeitigen Stand der Versuche, einen offiziellen Standard des Forth zu formulieren, berichtete in der letzten FORTH DIMENSIONS Jerry Shifrin. Seine "ANS Forth Meeting Notes" haben wir abgedruckt.

Kleinkram laden. Finn Berlev griff diese Idee auf und stellt eine Lösung LOADWORD unter Verwendung des NEEDS-Ansatzes vor. Erleben wir hier die Geburt einer neuen Bewegung in der Forth public domain, die systematisch Quellen sammelt, ordnet, prüft und als Pool herausbringt? Ich drücke die Daumen, damit noch mehr mitmachen.

Die 2. euroFORML Konferenz, im November, war ein voller Erfolg. Wer vor zwei Jahre und jetzt dabei gewesen ist, wird die Entwicklung gespürt haben. Forth findet kontinuierlich weitere Verbreitung und Anwendung. Domaine sind klar die real-time Aplikationen. Großes Interesse haben in den vergangenen zwei Jahren die Forthmaschinen erfahren. Mein kurzer Bericht soll zur weiteren Lektüre der Konferenzpapiere anregen - und zu mehr Treffen dieser Art!

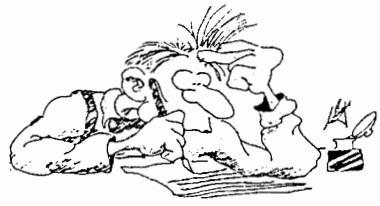
NACHRICHTEN

*** FORTH GESELLSCHAFT, JAHRESTREFFEN In diesem Jahr hat es nicht geklappt, noch neben der euroFORML ein Jahrestreffen der Forthgesellschaft zu arrangieren. Dies wird aber gleich Anfang des kommenden Jahres nachgeholt. Von Freitagabend, dem 13. Mai, bis Sonntag, den 15. Mai 1988 findet das Treffen in München statt. Die dortige Gruppe läd dazu ein. In München ist es nicht so leicht, preiswerte Unterkunft zu finden - dafür bietet der Ort mehr. Zwei Übernachtungen mit Vollpension werden im Kolpinghaus kosten: Einzelzimmer 120,-DM, Doppelzimmer 100,-DM, Mehrbettzimmer 90,-DM. Und man muß langfristig planen, da die Tagungsorte überlaufen sind. Daher ist es unbedingt erforderlich sich bis Ende Dezember verbindlich anzumelden, d.h. Formular ausfüllen und Geld überweisen! Heinz Schnitter wird das Programm arrangieren. Halten sie also in diesem und im nächsten Heft Ausschau nach den Anmeldeformularen für das Treffen.

*** FORTH TREE Viele werden es bereits gemerkt haben: Der Forth Tree, die Mailbox der FG ist offline! Die Harddisk war defekt und nicht so ohne weiteres wieder in Gang zu setzen. Über seine leidigen Erfahrungen berichtet unser Sysop Marco Pauck hoffentlich im nächsten Heft. Wann der Tree wieder funktionieren wird, war bei Redaktionsschluß noch nicht abzusehen.

*** FIG DATALINE SAN JOSE Die Forth Interest Group (FIG) hat es geschafft, sie ist in USA im landesweiten Datennetz GEnie zu erreichen. Zugriff hat man auch schon von Cannada und Japan aus. Ein Anschluß nach Europa ist geplant. Über DATEX-P gehts also noch nicht. Geboten werden drei Hauptbereiche: FIG Bulletin Board, FIG Real Time Conference, FIG Software Library. Wir werden berichten, wenns hier soweit ist. Es lohnt sich, schon mal die Modems abzustauben.

*** mpFORTH Für den Novix Chip fertig ist das Programmiersystem von Marco Pauck, das mpFORTH. Friedensalle 92, 4000 Hamburg 50, Tel: 040-3900139.



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

Die Arbeiten sollten folgendes enthalten:

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

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

IMPRESSUM

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

Redaktion: Michael Kalus, Tel: 02336-82204
Präsidentenstraße 40, 5830 Schwelm
Erscheinungsweise: Ein Heft je Quartal.
Redaktionsschluß: Der mittlere Quartalsmonat.
Auflage: 500 Stück europaweit.

Druck:

Das Forth Büro

FORTH GESELLSCHAFT e.V. - Antilopenstieg 6a - D 2000 HAMBURG 54

Tel.: 040-3904204

Jeden Dienstag von 18:00 bis 20:00 Sprechstunde.
Zu allen anderen Zeiten ist der TREE angeschlossen.

Postgiroamt Hamburg, Kto: 563211-208, BLZ 20010020

Die euroFORML Konferenz '87

Strahlend blauer Himmel und warme Abendluft machten den Aufenhalt auf 'der Burg' in diesem Jahr zur Erholung. Bei Spaziergängen in der Umgebung von Stettenfels und auf den alten Terassen der Burg, mit ihren wuchtigen Steintischen und Bänken, konnte man angenehm miteinander plaudern. Das angenehme Wetter und die Gastfreundlichkeit der Burg wogen den mangelnden modernen Komfort dort wieder auf. Kostenpunkt auch in diesem Jahr 640.-DM pro Person für die Konferenzteilnahme inclusive Unterkunft und Verpflegung über zweieinhalb Tage - Freitag, 18. bis Sonntag, 20. September.

Die Konferenz in diesem Jahr war stark beeinflußt von der Entwicklung der Forthmaschinen. Die Firmen Eurosil und Harris Semiconductor standen dabei im Mitelpunkt des allgemeinen Interesses. Ihre Vorträge wurden mit großer Aufmerksamkeit verfolgt. Anregende Konzeptionen gab es bei den Themenkomplexen 'Datenstrukturen' und 'Künstliche Intellingenz' zu erkennen, wohingegen der Block 'Die Zukunft von Forth' recht schwach ausfiel.

Im Detail nachlesen kann man die Vorträge in den Konferenzpapieren. Die Papiere füllen einen 2cm dicken Ordner. Alle Konferenzteilnehmer haben ihn zu Begin der Konferenz bekommen. Bestellen kann man die Papiere mittlerweile im Forthbüro oder man wartet auf die nächste Ausgabe der FORML PROCEEDINGS aus USA, die auch die euroFORML Papiere enthalten wird. Ob die Workshops inzwischen auch dokumentiert worden sind, weiß ich nicht.

Darüber hinaus wurden von Firmen Infos und Dokumentationen ausgelegt bzw bei ihren Vorträgen verteilt. Diese können bei den einzelnen Firmen sicher auch noch nachträglich bezogen werden. Im einzelnen waren dies:

1. Harris Semiconductor: Reprint Series "Forth Processor Core for Integrated 16-Bit Systems" und "Standard-cell CPU toolkit crafts potent processors" sowie ein 13 Seiten Info zu ihrem "F.O.R.C.E", der Forth Optimized RISC Computing Engine, Nachfolger des jungen und schon legendären Novix Forthchip. Der Beitrag über diesen Chip wurde von Dave Williams, Market Development Manager, gehalten. Er wurde bei den technischen Detailfragen unterstützt von Chris W. Malinowski, dem eigentlichen Designer des neuen Chip. Contact: Dave Williams, Harris Corporation, Melbourne, Florida, USA.
2. Eurosil: The MARC4 - A Forth Based Single Chip Micro-Computer. Dieser brandneue Einchip-Micro, basierend auf Forth, wurde in einem excellenten Beitrag von Graham Stout vorgestellt. Dazu das Datenblatt (Preliminary) des MARC4, dem 4Bit Microcomputer, inclusive seinem 'Befehlssatz', dem qFORTH. Anschrift: Eurosil electronic GmbH, Erfurter Straße 16, D-8057 Eching. Tel: 089-319060.
2. Forth,Inc.: Vortrag von Terry Rayburn: "METHODS} Objekt oriented extensions redux". Forth,Inc., Vertreten in der BRD durch RSO, Gesellschaft für technische Kybernetik, legte zudem Blätter aus zum polyFORTH II für den IBM-PC, polyFORTH/32 für VME-bus/68000-Systeme, Co-Processorboard für den IBM-PC etc. sowie für ein TMS32020 Entwicklungs und Applicationssystem. Anschrift: RSO GmbH, Am Moosfeld 85, D-8000 München 82, Tel: 089-

429188.
3. Silicon Composers machte Reklame für ein PC4000 Coprocessor IBM-PC Board mit scFORTH und scC als Sprachen.
4. Robert Bosch Gmbh.: Bericht über das Forschungsprojekt "Expertensystem in Forth", Dr. Manfred Walter stellte ihre Fortherfahrung auf diesem Gebiet vor. Begutachten konnte man das Expertensystem zur Motordiagnose auf einem Macintosh, routiniert vorgeführt und fachkundig erklärt von seinen Vätern. Anschrift: Robert Bosch GmbH, Postfach 1129, D-7310 Ploichingen, Tel: 07153-66528.

Von den Beiträgen sollen einige abgedruckt werden. Ein Anfang ist in diesem Heft gemacht: Terry Rayburn und seine METHODS}. Und hier noch weitere Beiträge, auf die ich hinweisen möchte: "A LISP Kernal for the NC4000" von Ulrich Hoffmann, (29 Screens Quelle), "LIST: A Generator for Objekt Oriented, Cyclic Linked Lists" von Karl-Dietrich Neubert (12 Seiten incl. 6 Seiten Quelle) und "cmFORTH, mpFORTH, the FORTHchip, and Optimizing Compilers" von Marco Pauck (7 Seiten). Ich hoffe, auch diese können bald abgedruckt werden; sonst muß sich der interessierte Leser die Mühe machen, und sich die Konferenzpapiere besorgen (Forthbüro oder lokale Gruppe).

Mir persönlich gut gefallen hat auch der Beitrag "Trainable Neural Nets in Forth" von John D. Carpenter - ein anderer Ansatz des KI, (22 Seiten incl. 10 Seiten Quelle. An die, die bei seinem Vortrag dabei waren: Es lohnt sich nachzulesen, wenn er auch ein lausiger Redner war!)

Nun könnte ich noch ein Weilchen weiter plaudern über Ereignisse in und am Rand der Konferenz, so zum Beispiel wie Herr Reilhofer in China war und die Zukunft entdeckte (wine and candlelight on the terrace), oder wie Herr Carpenter glücklich war, als er das Ende seines Vortrages wiedergefunden hatte, oder wie einige graue Herren von Mercedes unerkannt bleiben wollten, als sie fleißig Infos und Papiere einsammelten, oder wie wir uns ärgerten, als mitten im schönsten Workshop mit Heinz Schnitter über lokale Netze und Computer im Beschleunigerlabor der Uni München, die allesamt Forth mitteinander reden, plötzlich englisch gesprochen werden sollte, oder wie Forth Flesh dann doch neugierig wurde und 'zufällig' noch herein schaute. Doch das soll dann doch lieber in den Erinnerungen der Teilnehmer bleiben, als deren ureigenste Erlebnisse.

Zum Schluß noch einige Worte der Kritik. Die Zeit für Worshops und für den Besuch der Lobby war eindeutig zu kurz bemessen im Verhältnis zu den Vorträgen - also zu viel wurde reingepackt und manchmal auch zu breit dargestellt. Dabei hatten sich wirklich erfahrene Anwender zusammengefunden, um aus ihrer Arbeit zu berichten, sich befragen zu lassen, Knowhow auszutauschen. Und immer, wenn es gerade interessant wurde, bei Gesprächen am Rande der Konferenz oder in den Workshops, war die vorgesehene Zeit um, wurde zur Versammlung gerufen, um die nächsten Arbeitsgruppen einzuteilen oder Ergebnisse zu berichten. Mein Tip: Das nächste mal nur einen Vormittag (!) Vorträge nach vorbereiteten Papieren, den Rest Workshops und Gesprächsrunden unter kompetenten Partnern. Schließlich ist es eine FORML Konferenz - Forth

Modification Labor, mit Betonung auf Labor.

Des weitern zeigte sich auch deutlich das Problem der Vielsprachigkeit dieser europäischen Konferenz: Interessante Berichte verflachten streckenweise bis zum Lächerlichen - weil der Redner der englischen Sprache nicht mächtig war. Es ist ohnehin schon so, daß gute Fachleute nicht auch gute Redner sind. Der Druck, der auf den Vortragenden lastet - plötzlich vor einem kritischen Auditorium stehen, kurz und treffend vortragen, antworten - ist nicht jedermanns Sache. Das allein macht ja noch nichts, man merkt, da gäbe es was wichtiges zu erfahren, will sich heranfragen, aber ach... es kommt alles doch nicht rüber, weil die englischen Worte fehlen. Hier ging nicht nur die Information, sondern beim ringen um diese dann auch die kostbare Zeit verloren (siehe weiter oben).

Insgesammt hat mir diese Konferenz sehr gefallen. Ich kann mir durchaus ein Forth-Ereignis dieser Art jedes Jahr vorstellen - dafür dann nur halb so teuer! Denn schließlich kommen dann auch mindestens doppelt so viele Teilnehmer - was meint ihr?

euroFORML '87 schedule

Friday, 18 sep 87

14:00 lunch

16:00 opening remarks
organizational questions
introduction of participants

18:00 dinner

19:00 wine & candlelight on the terraces

Saturday, 19 sep 87

9:00 breakfast

10:00 Future of FORTH

- Communicating FORTH — R.Jones
- FURTHer FORTH with LEIBNIZ — A.Goppold
- The BLOCK Ghetto — K.Schleisiek\J.Carpenter

11:30 break

11:45 Data Structure

- Methods› Object-Oriented Extensions Redux — T.Rayburn
- LIST: A Generator for Object Oriented Lists — K.D.Neubert
- LISP Kernel for the NC 4000 — U.Hoffmann
- Command Interpreter for peripheral devices — K.Schleisiek

13:00 Lunch

14:00 NC 4000 and beyond

- cmFORTH, mpFORTH ... optimizing Compilers — H.Pauck
- The HARC4 - a FORTH based Single Chipper — G.Stout

16:00 Coffee break

16:30 NC 4000 and beyond

- Implementing High Speed FORTH Processors using Standard Cell Technology — C.Malinowski

18:00 Dinner

19:00 Wine & candlelight on the terraces

Sunday, 20 sep 87

9:00 Breakfast

10:00 Artifical Intelligence

- Trainable Neural Nets in FORTH — J.D.Carpenter
- Directed Nets of Rule Sets and a Hybride Search Strategy — T.Jost
- Implementation of an Expert System in FORTH — H.Walter

11:00 Workshops

- Forth Hardware
- Data Structures / Lokale Netze
- Forth in industrial control applications
- Forth in the field of AI

13:00 Lunch

14:00 Workshops

15:00 Summary of workshops, Plenary session

16:00 Coffee Break

16:30 Extemporaneous Talks

- On a first come, first talk basis

18:00 Dinner

19:00 What remained of the wine

Monday, 21 sep 87

9:00 Breakfast

the end

Aktuelle Literatur: FORTH DIMENSIONS

Volume IX, Nummer 1
Fractal Landscapes, by Phil Koopman, jr.
Forth to the Future, by Mitch Bradley
Starting Forth Inc., Interview with Elizabeth Rather
Run-Time Stack Error Checking, by Charles Shattuck
Perpetual Date Routine, by Allen Anway
Headless Compiler, by Darrel Johansen

Volume IX, Nummer 2
Flexible Test Environment, John Mullen
Forgettable Internal Names, Michael Hore
Consumerized Forth, Ken Takara
Execution Security, G.R.Jaffray, jr.
1987 Rochester Forth Conference, Jerry Shifrin
Starflight, Starbright, Interview with Tim Lee
Candidates' Statements, Fig Board Nominees

Volume IX, Nummer 3
Drill and Practice, by Richard H.Turpin
Matchpoint, by J. Brooks Breeden
The visible Forth, by Rich Franzen
ANS Forth Meeting Notes, by Jerry Shifrin
Gridplot, by Gene Thomas

FORTH - eine persönliche Sprache

Chuck Moore
Computer Cowboys

Forth ist eine Programmiersprache von außergewöhnlicher Mächtigkeit und Einfachheit. Das ist meine Ansicht, nachdem ich sie 15 Jahre benutzt habe. Ich habe das schon sehr oft gesagt. Nun wird es Zeit, daß ich es niederschreibe.

Ich möchte ein einfaches aber nicht-triviales Beispiel einer Forth Anwendung angeben und danach eine Teilmenge der Sprache, die als Einführung geeignet ist. Die Geschichte von Forth zeigt seine Mächtigkeit und Brauchbarkeit im Vergleich zum Chaos heutiger Software. Und am Ende hat die Philosophie, die Forth zugrunde liegt, zum schnellsten und am breitesten einsetzbaren Microprocessor geführt, den es heute gibt.

Geschichte

Ich habe mich 1958 am Smithonian Astrophysical Observatory (Cambridge, MA) auf die "Forth Reise" begeben. Mein Fortran Interpreter erlaubte freiformatierte Eingabe von Ausdrücken zur Bestimmung von Satellitenbahnen. Er benutzte Postfix Notation und einen Stack.

Dieser Interpreter blieb zehn Jahre lang ein Teil meines "Werkzeugkastens" und ich habe ihn nach und nach in Algol, Cobold und Assembler implementiert. Ich war ein umherschweifender Programmierer, der auf Mainframe und später Minicomputern verschiedenartige System-, Steuerungs- und Berechnungsprojekte durchführte.

1968 entdeckte ich ":", als ich für Mohasco Industries (Amsterdam NY) arbeitete. Der Interpreter konnte jetzt ein erweiterbares Vokabular verarbeiten, das auf einfachen Funktionen aufbaute, die in der darunterliegenden Sprache ausgedrückt waren. Damit habe ich auf einer IBM 1130 State-of-the-Art Grafik erzeugt und eine Univac 1108 im Timesharing genutzt. Das System hieß nun FORTH.

1970 entdeckte ich, daß Definitionen als gefädelter Code (Adressenlisten) compiliert werden können, als ich am National Radio Astronomy Observatory (Charlottesville VA) arbeitete. Forth war jetzt schneller als eine compilierte Sprache, kompakter als Assembler und es war möglich, Forth selber in Forth zu beschreiben und zu compilieren. Ich habe es auf Kitt Peak (Tucson AZ) für eine State-of-the-Art Teleskopsteuerung und Datenreduktion benutzt. Mit diesem 12 m Teleskop wurden in den 70ern viele interstellare Objekte aufgefunden.

Die ganze Zeit über war Forth meine persönliche Sprache. Niemand sonst kannte sie oder wußte sie zu schätzen. Als unstandardi sierte Sprache war sie umstritten und wurde lediglich auf Grund ihrer Leistung toleriert. Meine Produktivität vervielfachte sich; ich habe sie auf jedem erreichbaren Rechner implementiert; in Tucson wurde Elizabeth Rather der zweite Mensch, der in Forth programmierte. Forth Systeme fingen an, sich fortzupflanzen. Da die Implementation nur ungefähr einen Monat brauchte, konnte jeder eine Version haben - auf jedem Computer (DEC, DG, HP, Varian, Honeywell, IBM, CDC).

Kitt Peak hat Forth akzeptiert, NRAO jedoch nicht. 1973 entwickelte sich aus den Leuten in Tucson Forth Inc. (Manhattan Beach CA) mit einem Stammkapital von $5000. Wir haben Forth Systeme und Applikationen verkauft. Eine sehr Frühe war ein Datenverarbeitungssystem für Cybek Inc. (Clifton NJ), das noch immer weit verbreitet ist für Probleme mit hohen Anforderungen (Krankenhäuser, Lagerhaltung, Buchhaltung, Finanzierung).

Forth wird immer beliebter bei Ingenieuren und Anderen, die an Ergebnissen interessiert sind - Programmierung, Geschwindigkeit, Größe und Zuverlässigkeit. Ihm fehlen die Empfehlungen, die es attraktiv machen würde für Universitäten, Regierungsstellen und die Großindustrie. Die Forth Interest Group (San Jose CA) ist damit beschäftigt, die Akzeptanz zu erhöhen.

Forth Programmierung

Was ist eigentlich mit Forth los, daß es populär, aber nicht akzeptabel macht? Vor Forth habe ich Fortran, Assembler, Algol, Cobold, APL, LISP und so weiter benutzt, je nach den Umständen. Seitdem habe ich nur noch Forth benutzt. Es hat einige Zeit gedauert, bis ich selber überzeugt war. Heute weiß ich, daß Forth die beste Sprache ist, um mit Computern zu kommunizieren.

Diese Überzeugung gründet sich zum Teil auf meine Erfahrungen bei vielen Anwendungen; zum Teil auch auf Erfahrungen Anderer. Sie gründet sich auf die Ähnlichkeit mit natürlichen Sprachen, die über Zeiträume von tausenden von Jahren optimiert wurden. Sie speist sich aus den Aussagen von Computertheoretikern, wie eine gute Sprache beschaffen sein sollte. Und eine ganze Menge stammt aus dem Vergleich mit der schockierend schlechten Qualität konkurrierender Sprachen.

Als ich 1970 Forth entwickelte, habe ich bewußt eine ganze Menge von Problemen in Betracht gezogen, die ich selber erfahren und die ich in Diskussionen kennengelernt hatte. Probleme um Programmerstellungszeiten, Testzyklen und Kosten; Programmgröße, Geschwindigkeit und Überprüfbarkeit; das Wesen, die Kosten und die Brauchbarkeit von Dokumentation. Damals empfand ich die existenten Sprachen als total unzureichend; daraus waren meine Bemühungen um Forth entstanden.

15 Jahre später sind wir bei Mikro- und Supercomputern angekommen mit ganz neuen Sprachfamilien. Die Probleme bestehen jedoch immer noch. C, Pascal, Modula, Ada, Logo, Prolog, Occam haben die gleichen Unzulänglichkeiten wie Fortran, Algol, Cobold und PL/1. Schlimmer noch, die Sprachen sind "fett" geworden. Beurteilt nach Popularität mangels objektiver Kriterien. Noch schlimmer: Wunschdenken verläßt sich auf "Künstliche Intelligenz", "Expertensysteme" und "Neurale Netze", um magisch Antworten hervorzubringen.

Forth ist sehr verschieden davon. Es ist eine andere Art von Sprache als die oben genannten. Es ähnelt Lisp in Modularität und Interaktivität, Assembler in Geschwindigkeit und Kontrolle, keiner anderen in Kompaktheit und Vielseitigkeit. Mit einem großen Vokabular, einer einfachen Syntax und ihrer inhärenten Erweiterbarkeit ist sie analog dem Englischen, der erfolgreichsten natürlichen Sprache.

Das Schreiben von Forthprogrammen geht schnell und ist lohnend. Die Entscheidung, welche Worte definiert werden müssen, ist eine kreative Herausforderung. Dann ist ihre Definition ebenso einfach wie das Testen. Zusammengefaßt werden sie dadurch, daß hierarchisch über diesen weitere Worte definiert werden. Die Dokumentation solch kurzer Worte ist einfach. Es gibt kein "Programm", das mit den üblichen engen Vorschriften und ausführlicher Detaillierung geschrieben werden müßte.

Forth ist jedoch ein polarisierender Verstärker: es ist möglich, sehr schlechten Code zu schreiben. Der Unterschied zwischen gutem und schlechtem Forth ist größer als bei anderen Sprachen. Die Faktorisierung eines Problems in Worte kann auf viele verschiedene Arten geschehen - auf viele falsche und nur eine richtige Art.

Forth Worte

Forth ähnelt natürlichen Sprachen in Bezug auf die Größe des Vokabulars. Ein einfaches System hat 150 Worte und eine spezielle Anwendung kann mehrere hundert hinzufügen. Nur wenige davon haben irgendwelche syntaktischen Einschränkungen und kein sind in dem Sinne reserviert, daß sie nicht redefiniert werden könnten.

Hier ist ein einführendes Vokabular, das ungefähr Basic äquivalent ist (n ist eine 16bit Zahl, a ist eine Adresse):

+ - * / MOD MIN MAX	binäre arithmetische Operatoren
< > =	binäre Vergleichsoperatoren
AND OR XOR	binäre logische Operatoren
NEGATE ABS NOT	unäre Operatoren
*/	trinärer arithmetischer Operator
DUP DROP SWAP OVER	Stackoperatoren
DECIMAL HEX OCTAL	Setzen der Zahlenbasis
.	Resultat ausgeben
n .R	Rechtsbündig ausgeben
EMIT	ein Zeichen ausgeben
CR	Zeilenvorschub auslösen
KEY	hole Zeichen von der Tastatur
: ... ;	definiere ...
VARIABLE ...	definiere eine Variable
CREATE ...	definiere ein Array
n ALLOT	allokiere Platz für ein Array
n ,	initialisiere Array
a @	hole von Adresse
n a !	speichere an Adresse
(...)	ignoriere Kommentar ...
EMPTY	vergiß zusätzliche Worte

Die folgenden Worte werden nur benutzt, wenn ein neues Wort definiert wird (nach :)

n	IF	true	ELSE	false	THEN	bedingte Struktur
n	FOR	...		NEXT	Schleifenstruktur
I					Index in der Schleife

Es gibt noch eine ganze Menge von Worten, die für den ernsthaften Programmierer wichtig sind. Es fehlen in der Tat noch Worte, die nötig sind, um den Interpreter, den Compiler, die Massenspeicher- und Konsolenschnittstelle zu definieren. Au die Art ist Forth selber konstruiert. Aber die obigen Worte reichen für die meisten Anwendungen aus.

10 analoge Meßwerte können zum Beispiel so gelesen, gespeichert und gedruckt werden:

```
CREATE VOLTS   10 ALLOT
: READ      9 FOR  A/D VOLTS I + !  NEXT ;
: PRINT     9 FOR  VOLTS I + @  4 .R  NEXT ;
```

Der Gebrauch von Forth

Forth ist keine gewöhnliche Programmiersprache, obwohl es eine Sprache für Computer ist. Man schreibt keine Programme in Forth, man beschreibt ein Problem. Das macht man dadurch, daß ein Vokabular von Worten aufgebaut wird, die sowohl der Programmierer als auch der Rechner versteht. Durch die Benutzung dieser Worte kann der Computer auf Grund von Anweisungen durch den Benutzer rechnen und steuern.

Das folgende Beispiel stammt aus meinem CAD System. Der Computer berechnet Ausdrücke, die für den Schaltungsentwurf wichtig sind.

```
: RC    1000 */ ;
: öö    OVER OVER + */ ;      (öö ist ||)*
: mA    1000 SWAP */ ;
: V/    OVER + 5000 SWAP */ ;
```

Dies ist typischer Forthcode. Worte sind durch Leerzeichen voneinander getrennt und der Kommentar steht in runden Klammern. Das Wort ":" definiert das darauffolgende Wort als die sich anschließende Folge von Worten bis zum ";". Worte werden von links nach rechts ausgeführt. Operatoren folgen den zugehörigen Argumenten (Postfix Notation). Das Wort */ verbraucht 3 Argumente und multipliziert eines mit dem Verhältnis der beiden anderen.

Diese Definitionen werden compiliert, so daß sich eine Bedeutung für den Computer ergibt. Die Bedeutung für den Menschen wird durch den Namen, durch das Lesen der Definition oder durch Kommentare wie folgt erzeugt:

Benutzung			Ergebnis
n	n	RC	Zeitkonstante eines Widerstands und eines Kondensators in Picofarad.
n	n	öö	Widerstand einer Parallelschaltung zweier Widerstände.
n	n	mA	Strom, der durch eine Spannung an einem Widerstand erzeugt wird.
n	n	V/	Spannung, die durch einen Widerstandsteiler an 5 Volt erzeugt wird.

Wenn diese Worte definiert sind, so ist es unnötig, ein Programm zu schreiben, mit dem eine Eingabe geholt wird oder mit dem eine Ausgabe formatiert wird. Resultate stehen für den Fortgang der Berechnung zur Verfügung. Wenn man eingibt

```
100 200 öö 300 öö .
```

dann wird von dem Wort "." der Widerstand dreier parallel geschalteter Widerstände ausgegeben. Variable brauchen keine Namen. Werte sind als ganze Zahlen ausgedrückt in Größen, die Ingenieuren geläufig sind (ns, mA).

Es ist der mnemonische Wert von Kleinschreibung und Sonderzeichen zu beachten. Das führt zur Kompaktheit, kann aber auch einfach geändert werden.

```
: DEFINIERE    : ;
```

Das Wort DEFINIERE ist jetzt Synonym mit : und kann genauso benutzt werden

```
DEFINIERE NS    1000 */ ;
```

Forth Computer

Bis vor kurzem mußte sich Forth mit anderen Sprachen auf deren Feld messen - auf Rechnern, die für sie optimiert waren. 1984 habe ich selber einen Mikroprozessor entworfen, der für Forth optimiert ist, nachdem ich 10 Jahre lang gewartet hatte, daß jemand anderes es tut. Er ist als NC 4016 von Novix Inc. (Cupertino CA) erhältlich und demnächst von Harris im Rahmen des F.O.R.C.E. Konzepts.

Die Grundlage der Forthphilosophie ist Einfachheit. Einfache Worte, einfache Syntax, einfache Protokolle. Wird dies auf Hardware angewandt, ergeben sich ähnliche Resultate. Der NC 4016 ist ein Gate-Array mit nur 16.000 Transistoren - nur 1/10 der anderer heutiger Mikroprozessoren. Ee hat keinen Mikrocode und führt Forth in Mikroprogrammgeschwindigkeit aus (bis zu 30 MIPS). Da es ein CMOS Chip ist, braucht es wenig Energie (0,5W) und bleibt auf Zimmertemperatur. Ein vollständiger Computer paßt auf eine Postkarte, läßt sich aus Batterien speisen und braucht nur 10 externe IC's, davon 8 Speicherbausteine. Und er führt 8 MIPS aus.

Diese Art revolutionärer Leistung einer Hochsprachenmaschine ist nur auf Grund von Einfachheit möglich. Es war möglich, die wesentlichen Details zu optimieren. So brauchen zum Beispiel die Call/Returninstruktionen, die wesentlich für eine modulare Sprac sind, zusammen nur einen Taktzyklus.

Von daher können sie zu meiner Überzeugung, daß Forth eine hervorragende Sprache ist, auch noch hinzufügen, daß es auch noch zu sehr attraktiver Hardware führt. Die Sprache, die am einfachsten zu programmieren ist, ist nun auch noch diejenige, die am schnellsten ausgeführt wird. Sie werden in Zukunft noch sehr viel mehr über Forth hören.

* unterschiedliche Belegung der ASCII Zeichen im USA od. BRD-Zeichensatz. Sorry.

"Der Friede ist das Meisterstück des Verstandes"

(Inschrift am Turmzimmerkamin von Schloß Stettenfeld)

Interviews

Klaus Schleisiek über die euroFORML'87

Mein Eindruck von der Konferenz - als Veranstalter: Wir sind am Anfang durch eine unglaubliche Zitter-Periode gegangen. Denn durch eine Kette von bis heute unaufgeklärten Vorkommnissen sind manche Briefe in der Vorbereitungsphase in den USA nicht angekommen, sodas weder in der FORTH DIMENSIONS noch in dem JOURNAL OF FORTH oder den ROCESTER PROCEEDINGS rechtzeitig Ankündigungen der Konferenz erschienen sind. Da haben wir dann eine Notaktion gestartet und persönliche Einladungen versandt, an alle, die uns bekannt waren. Auf einer Beratungssitzung vor sechs Wochen konnten wir dann hochrechnen, das so 20 bis 25 Teilnehmer da sein werden, und wir haben gesehen, daß sich die Konferenz dann ja so eben tragen würde und also weitergemacht - fast hätten wir aber die ganze Sache abgeblasen.

Dann wollte es das gütige Geschick, daß Harris ihre Forth-Chip-Architektur in 'Markt&Technik' vorstellten. Daran konnten wir anknüpfen und haben unsere Ankündigung dort noch untergebracht. Dieser Aufruf hat nochmal ca. 10 Konferenzteilnehmer hinzugewonnen. So waren wir jetzt rund 35 Leute hier.

Der Ablauf der Konferenz klappte diesmal wesentlich besser als vor zwei Jahren. Wir hatten ein Programm, das ich noch ansprechender fand als damals, es lief gut ab, wenn auch heute, am Sonntagnachmittag schließlich doch manches auseinander gelaufen ist. 2 1/4 Tage Programm scheint doch einfach zu lang zu sein, 1 3/4 Tage sind wohl vernünftiger und bieten mehr Raum zur eigenen Gestalltung (Lobby). Die Konferenzteilnehmer hätten alle wohl gern noch mehr Zeit zum plaudern gehabt - anstatt zu oft von der Glocke zu Vorträgen gerufen zu werden.

Bemerkenswert finde ich, das die Firma Harris als erste von kommerziellen Forthsystem und Hardware-Anbietern gemerkt hat, daß es so etwas wie eine Forthcomunity gibt, die übergreifend über einzelne Firmen hinweg organisiert ist. Ich bin da optimistisch, daß es hier zu einer Zusammenarbeit kommen wird - ich sehe auch, das ihr Entwurf etwas taugt und der Support besser sein wird, als das, was Novix bisher bieten konnte.

Wie die Finanzlage nach der Konferenz nun ist, überblicke ich zur Zeit nicht, da die Abrechnung erst in Hamburg stattfinden wird - aber ich denke, es wird ein kleiner Überschuß für die Forth Gesellschaft bleiben.

Interview mit Graham Stout, Eurosil, über das MARC4 Projekt weiter hinten im Heft.
Im nächsten Heft: C.W.Malinowski, Harris Semiconductor; Terry Rayburn, RSO Gmbh München.

Briefe

Meine Wünsche an FORTH

Ich bin 44 Jahre alt, gelernter Bauschlosser, jetzt als Spulen- und Ankerwickler bei der Deutschen Bundesbahn beschäftigt. Plattdeutsch, Mathematik und Computer sind mein Hobby. Hinter mir liegen HP Taschenrechner, Casio FX-702, TI-99/4A. Jetzt benutze ich einen Schneider CPC 6128. Basic, Logo, Elan habe ich probiert. Nun habe ich mit Forth begonnen. RVS-Forth, F83 und Volksforth habe ich nun und bin Mitglied in der Forthgesellschaft geworden. Ich wünsche es mir so, daß man mit wenigen Worten schon nützliche Programme schreiben kann:
1. Leichte Stringverwaltung mit entsprechenden Vergleichsbefehlen für ganze und für Teilstrings.
2. Wenige Worte zum Anlegen einer sequentiellen Datei und einer Randomdatei.
3. BCD-Zahlen mit einer Genauigkeit wie sie zb. der TI-99/4A hat, also 11 stellige Mantisse und einen Exponenten bis ca. 120.
4. Alle mathematischen Funktionen wie im Basic und Comal.
Dieses wünsche ich mir als zuladbare Pakete auf den Forthkern und zwar unabhängig voneinander. Ich glaube nur so kann Forth eine allgemeine Verbreitung finden.

Ich würde gern eine Fachgruppe EDITOREN sehen bzw. eine solche koordinieren. Ich betrachte es als wünschenswert, Editoren von Forth aus zu beherrschen für alle möglichen Zwecke. Ich füge einen Editor für das F83 bei, der allerdings noch nicht so weit ausgebaut ist, wie der des Volksforth. Gegen Zusendung einer 3" Diskette (Schneider CPC mit CPM plus oder CPM 2.2) sowie des Portos gebe ich gerne das Programm weiter.
Rudolf Mensing, Flottkamp 19, 2358 Kaltenkirchen, Tel: 04191-6499
(Forth Magazin: Wir haben das Listing abgedruckt. Siehe unten. Ohne Gewähr.)

*

Forth Programm Editor

Zum Thema Editoren: Als ich 1985 einen Screeneditor für fig-Forth auf AppleII suchte (selbst eingetippt), war einfach keiner aufzutreiben. Also hab ich selbst einen Editor verfasst. Ich finde, ihr solltet mal einen Screen-Editor bringen, damit er wirklich zur Hand ist, wenn man danach sucht (es muß ja nicht unbedingt meiner sein). Mein Listing gebe ich gerne ab, auch mein fig-Forth 1.1 ist zu haben.
Wolfgang Mües, Hagenring 22, 33 Braunschweig, Tel: 0531-330869

*

Betr.: Projekt: Integriertes Programmpacket in Volksforth83

Den Einstieg in Forth bekam ich durch die Datenbank H&D-Base, eine dBase-II kompatible Datenbank auf dem Atari-520-ST+, welche das dBase-II durch Forth emuliert. Da in dieser Datenbank die Sprache Forth ebenfalls zur Verfügung steht, lag es für mich nahe, diese zu erlernen, um selbst ergänzende Routinen zu programmieren. Dabei stellte ich fest, daß Forth schnell und durch seine Erweiterbarkeit sehr flexibel ist, Ihnen alles sicherlich nichts neues.
Ich habe nun vor, unter Forth ein integriertes Programmpaket zu erstellen, welches die drei Bereiche Textverarbeitung, Datenbank und Graphik umfasst.
Mein Anteil an diesem Programmpaket wäre die Erstellung des Textprogrammes. Es soll Eigenschaften aufweisen, die ich bei vielen professionellen Programmen vermisse:
So z.B. Tastaturmakros und Textmakros, die Möglichkeit Kürzel im Text zu benutzen, welche nach Eingabe von der Tastatur auf dem Screen durch Texte ersetzt werden.
Ebenfalls hoffe ich NLQ als Graphikdruck implementieren zu können.
Dabei wäre ich natürlich über jede Hilfe und Unterstützung aus der Forth-Gemeinde dankbar.
Weiterhin wären die Teile Datenbank und Graphik - auch Teilfunktionen davon - zu übernehmen, sowie der Teil der Daten zwischen den einzelnen Programmen austauschen kann.

<u>Zum praktischen Vorgehen:</u>

Existenz eines Koordinators, welcher an der Programmerstellung nicht beteiligt zu sein braucht, jedoch bei Problemen Hilfestellung leisten könnte, bzw. weiß wer helfen kann.
Programmiersprache soll das Volksforth_83 sein.
Um den Arbeitsaufwand für den Einzelnen in Grenzen zu halten, habe ich mir gedacht das Programm in viele kleine Routinen aufzuteilen und diese an Interessierte zu vergeben, wobei eine genaue Beschreibung erfolgen müßte, was das einzelne Modul zu leisten hat.
Wen eine derartige Aufgabe reizt möge sich bei mir melden.
mit freundlichen Grüßen und Hoffnung auf regen Zuspruch und ebenso rege Unterstützung verbleibe ich

E. Teshmar

Dr. Elemer Teshmar, Danziger Baumgang 97, 6800 Mannheim 31
Telefon: 0621/781726

* * *

AUGUST 1987

ANS FORTH MEETING NOTES

JERRY SHIFRIN - MCLEAN, VIRGINIA

These notes about the first meeting of the ANS Forth Technical Committee (August 3-4, 1987 at CBEMA Headquarters in Washington, DC) are not minutes of the meeting, nor are they part of the ANS Forth Technical Committee's official documentation. They represent my personal impressions only. The reason I mention this is that during the course of the meeting, it became clear there are serious liability considerations for anyone publishing anything which someone could interpret as anything approaching a standard. Therefore, I must caution everyone that this IS NOT a standards document. With that out of the way, I'll try to give you a summary of the events.

The meeting began at 9:00 a.m. on Monday and was attended by many of the leaders of the Forth Community, including a number of folks from the Forth Standards Team (FST). Elizabeth Rather (as the "convenor") was acting chairperson, and Ray Duncan volunteered to be acting secretary for the initial meeting. Elizabeth made some opening remarks on the maturation of Forth and the need for greater acceptance. She described the scope of this effort as mainly oriented towards describing common practice, neither attempting to fix Forth nor using the standard as an instrument for change.

The agenda was approved without dissent.

There followed a discussion on international involvement. The X3 representative suggested that early involvement would lengthen the process. It turns out that there is a requirement for an international liaison, but it wasn't clear at what point that would become important. Chuck Moore felt strongly that international involvement was important. I don't recall that any decision was reached on this.

Following this was a discussion on validation suites. The original scope proposal specifically excluded them from this group's activity. The membership voted not to exclude the possibility of such development.

Cathy Kachurik of CBEMA presented a tutorial on the X3 structure and process.

Charlie Keane proposed adopting the Forth-83 Standard as a "BASIS" document. As I understood it, the basis document becomes the working document for all activity of the technical committee (TC). That is, all changes, deletions, additions, etc. are proposed as updates to this document. There was a discussion about whether to restrict this document to Chapter 12 (the Required Word Set). This was defeated 15-1. A motion to adopt chapters 1-16 carried unanimously. This excluded only the appendices (uncontrolled reference words, experimental proposals, charter, membership, and proposal/comment forms).

I put up a motion to include floating point as part of the scope of work. This was defeated by a vote of 4-12. A motion was approved (14-3) to eliminate the timeframe constraints on standard language extensions. Previously, this had indicated that language extensions could not be considered until the approval of the ANS Forth standard.

Two additions to the Scope of Work document were approved: that the TC will review existing and proposed programming language standards; and to consider the impact of the standard on current and anticipated hardware technology. Another change to this document was to change the number of users required for a Forth system in order to be considered for non-compliance with the Forth-83 standard; this number was reduced from 1000 to 200 users of a particular Forth implementation. The Scope of Work document (X3J14/87-002) was then approved as amended.

The TC Subcommittees document (X3J14/87-004) was approved with minor changes. There are four subcommittees: Documentation, Research, Logistics, and Technical.

The Plan of Work document (X3J14/87-003) was then approved with a few changes. It was agreed to remove specific topic areas from the meeting plan in order to allow the work to proceed faster if possible. Chuck Moore got a motion passed to require that at least one meeting be held in San Francisco. The membership then agreed to hold the next meeting in San Francisco, but could not find anyone willing to host that meeting. Elizabeth Rather then volunteered to host the meeting in Southern California, and that was agreed on. I put up a motion to co-schedule and co-locate the ANS Forth meetings with the FORML and Rochester conferences. This was defeated 4-10.

Next was the Call for Officers. The X3 Secretariat appoints the officers from a list of volunteers. The following people volunteered at the meeting:

Chair: Charlie Keane, Bill Dress, Larry Forsley

Vice Chair: Bill Dress, Ray Duncan

Secretary (appt'd.by chair):Martin Tracy

International Representative: Larry Forsley
Vocabulary Representative (appointed by the chair): Ted Dickens
Documentation Editor (appointed by the chair): Ron Braithwaite

Applications for these positions may be accepted until August 31st. Each requires a letter of intent and qualification, along with a letter from your employer indicating that they understand the amount of time needed for taking on these assignments.

Greg Bailey and Don Colburn then proposed that the technical subcommittee agree to mark up the standards document, indicating areas to be deleted, modified, added, and areas of deviation from accepted practice. This was agreed 13-3.

The group then agreed 9-4 to hold the next meeting on November 11-12, 1987 in Southern California at FORTH, Inc.

Elizabeth named acting chairs for each of the subcommittees: Ted Dickens, documentation; Gary Betts, logistics; Guy Kelly, research; Greg Bailey, technical.

I passed out documentation and gave a brief description of the ANSForth bulletin board on MCI Mail.

There was then a review of all action items and the TC adjourned. This was immediately followed by the convening of the Technical Subcommittee (TSC).

Martin Tracy volunteered to serve as acting TSC secretary. There was a lengthy discussion on the proper name for this subcommittee (I'm using TSC in these notes, but that may not be accurate) and its voting membership requirements. At issue was whether it was a formal subcommittee which would carry additional documentation requirements. As I recall, no conclusion was reached on this. We did get the impression that in order to be a voting member of the TSC, you had to be a voting member of the TC.

The TSC then drew together a list of goals: identify a kernel of highly compatible words, decide a strategy for layering and extensions, amass information on the TC desires and needs, and define a mechanism for handling proposals.

I ran out of steam around this point and stopped taking notes, but most of the remaining discussion was on the proposal process, voting membership, and plans for the next meeting.

Commentary

While the preceding describes events to the best of my recollection and note-taking abilities, I thought I'd add a few opinions and observations of my own:

First, I think this effort is off to a great start. The membership includes an excellent and reasonably well-balanced group of vendors, users, and other interested folks. I believe most of the early objections to this effort have been resolved by the make-up of the TC. Additionally, there was a clearly cooperative spirit among the attendees.

It was very clear to me that the TC was determined to pursue an open organization. Several discussions were concerned with how to publicize our procedures and encourage participation. This, along with CBEMA's requirements for "due process" will, I think, result in an excellent document.

Unfortunately, it seems there is still an IEEE cloud hanging over this effort. I thought a compromise had been reached, but apparently there are still a few people pursuing the IEEE Forth alternative. We'll have to wait and see what happens.

Chuck Moore, in spite of his avowed opposition to a Forth standard, was extremely cooperative. My impression is that he was mainly concerned with having wide participation and not shutting off the possibility of new Forth development. (He also mentioned that he was working on a new Forth compiler.)

Don Colburn seemed to feel that we could put out a draft document much earlier than planned and was surprised at the idea that there would be any difficulty in reaching a consensus.

I felt that Elizabeth did an excellent job of chairing the meeting, but suppose she was wise in not volunteering as the permanent Chair. This way, we avoid even the appearance of a FORTH, Inc.-dominated effort.

I have a couple of concerns about the course of this project. Most difficult for me to reconcile is the notion of a standard documenting common usage among the major Forth implementations. In some cases, this may cause a reversion of some language features back to the way they were before the 83 standard. For example, FORTH, Inc. never changed its definition of LEAVE to correspond with the 83-standard; i.e., it does not immediately leave the loop. One could, therefore, argue that the 83 standard LEAVE is not in common usage. Thus, it seems to me that the ANS standard might either leave its effect undefined or else omit it entirely. Worse, I think, would be to revert its meaning back to the 79 standard.

My other main concern is with the minimalist approach. I guess it's the only sensible way of getting this out in a reasonable amount of time, but I worry that most proposals will simply be put aside with the note that they're outside the documented scope of the ANS Forth effort.

On the positive side, I think this group has enough talent and dedication to complete a superb standard in a reasonable amount of time. I offer my personal thanks to everyone involved for providing two days of stimulating discussion. A special tip of my Forth beanie goes to Elizabeth Rather and Martin Tracy for doing the bulk of the work in pulling this activity together.

Other Notes of Interest

Don and Chris Colburn were kind enough to invite everyone over to their house Monday evening for pizza and pool (swimming, that is). It was very pleasant and provided the opportunity for people to get to know each other a bit better. Don took us down for a tour of his workshop; it looked like a Mac farm. Don demonstrated the Mac II running several animated graphics tasks under MacForth in separate windows. Very nice.

I was very happy to have Martin Tracy and Guy Kelly stay at my house, but regret the short time available for Forth talk. Guy demonstrated his new, implementation-independent Forth editor. It seemed very powerful — it could work with screen files or native blocks. In addition, he provided three ways for moving stuff around — cut and paste, a line stack (push and pop a line at a time), and a "barrel" (push stuff into the barrel, and select from it in any order). I believe Guy will be offering this for sale.

He includes drivers for several Forths, including F83, MVP, Uniforth, and LMI.

Martin, as usual, has numerous irons in the fire, including his polyFORTH implementation for Digital Signal Processing (DSP) chips, his upcoming *Dr. Dobb's* article and column, along with work on ZenForth, and his continuing involvement with the Forth Model Library. Martin mentioned that Wil Baden had started helping him on the Zen project. Martin stopped by the Potomac FIG meeting (on his way to the airport) and talked about several of his projects.

Miscellaneous Notes

X3 has raised its membership fees: $200 for one principal membership (includes one alternate), $150 for observer and each additional alternate. The X3 Secretariat may be reached at: CBEMA, 311 First St. N.W., Suite 500, Washington DC 20001, 202-737-8888.

Jerry Shifrin is employed by MCI and is the sysop for the East Coast Forth Board (703-442-8695).

MCI MAIL'S ANS FORTH BBS

MCI Telecommunications is sponsoring a bulletin board on MCI Mail in support of the ANSI Forth standards activity known as X3J14. This board will contain agendas, proposals, minutes of meetings, and related information. If you are interested in the development of the ANS Forth standard, this is the place to see what's going on.

ANS Forth Bulletin Board

ANSForth is the main heading for several types of message areas. From the main "Command:" prompt, type VIEW ANSFORTH to see all currently active areas. The following are currently available:

1. General: general information on X3J14; membership, documentation, officers, etc.
2. Agenda: agendas for X3J14 meetings.
3. Minutes: minutes of previous meetings.
4. Proposals: proposals for consideration by X3J14.
5. Comments: comments on active proposals.
6. Misc.: uncategorized messages.

In general, bulletin board messages will only remain available for 90 days (this is an MCI Mail constraint). Archived versions of older messages will be available for downloading from the East Coast Forth Board.

Viewing the Bulletin Board

To access the ANSForth Bulletin Board type

VIEW ANSFORTH <*subarea-name*>

For example, type

VIEW ANSFORTH AGENDA

to see the upcoming agenda.

If you don't know the particular subarea's name, type at least ANSFORTH, then choose the subarea you want from the list of matching names. You will then see a "View:" prompt; type one of the following commands:

SCAN	To display a scan table of items on the board, showing the date posted, subject, and the size of the item.
READ	To display all items or those selected by scan number, with page breaks and a pause between items.
PRINT	To display all items or those selected by scan number, with no page breaks or pauses between items.
LEAVE	To exit the board and return to mail mode.
EXIT	To log off MCI Mail.

Posting Messages on the Board

Unlike many other bulletin boards, MCI Mail only allows the bulletin board owner (me) to post messages. Therefore, in order to get a message posted for public view, you must send it to me to be forwarded.

The best way to send me a message is via MCI Mail. I check this mailbox daily, and forward messages to the bulletin board with just a few keystrokes. To send me a message on MCI Mail, enter the CREATE command and at the "TO:" prompt enter "Gerald A. Shifrin". Alternatively, you may address a message to my MCI Mail id: 299-4103.

The second best way to get a message posted is to leave it on the East Coast Forth Board at 703-442-8695, addressed to SYSOP with instructions for it to be posted on the ANS Forth Bulletin Board. Similarly, you may upload a file (also with appropriate instructions) to me. This will get your message posted within a day or two.

If you don't telecommunicate but want to get a message posted, send me a floppy disk in IBM PC format containing the file(s) to be posted. Mail this to: Jerry Shifrin, 6212 Loch Raven Dr., McLean, VA 22101.

I'm sorry, but I won't be able to return your disk unless you include a self-addressed, stamped mailer. Please do not send written material to be posted. I don't have a document scanner and I'm not a great typist.

Prices

The annual mailbox fee is $18, payable upon registration and billed on the anniversary date of service. This allows you to read messages sent to you. There are additional charges for sending messages and access fees; these are described below. The initial $18 registration fee will be credited against any charges you have in the first two months.

It costs $.45 to send an instant message of up to 500 characters; $1 for up to 7500 characters, and another $1 for each subsequent 7500 characters. There is no connect-time charge if you call through a local MCI Mail number. There is a $.05/minute charge if you access MCI Mail via Tymnet. —*JS*

[Editor's note: You may find the standards-related news and discussion on FIG's GEnie conference (see announcement elsewhere), and on the East Coast Forth Board, timely and complete enough for most purposes.]

Das MARC4 Projekt

Graham Stout Interview bei der euroFORML'87

Vor zwei Jahren, ganz am Anfang des Projektes, waren wir nur zu zweit, Rob Talty aus Australien und ich - wir sind auch jetzt hier zusammen auf die euroFORML gekommen. Wir haben damals die Hauptpunkte für das Projekt festgelegt. So im November '85 etwa hatten wir die wichtigsten Punkte dokumentiert: Den Entwurf des Instruktionssatzes, wie der Microcode aussehen sollte, den groben Ablaufplan und sowas. Ich sollte das Design des Core machen, ROM, RAM, PLA's, Microcodes - also die eigentliche Forthmaschine - und es war Rob's Aufgabe, die I/O-Teile auszuarbeiten, weil er die Erfahrungen mit Real-Time-Control-Anwendungen hat. Er arbeitete also die Bus-Spezifikationen aus, den Interupt-Controller, I/O-Port-Logic und solche Sachen.

Nach dem Jahreswechsel holten wir dann noch einen guten Software-Mann dazu, Gerhard Goettle, er spezifizierte das Entwicklungssystem zum Chip und andere Aspekte der Software. Zu der Zeit waren wir eigentlich gar keine Forthexperten und entwickelten auf einem IBM-PC in Turbo-Pascal mit Hilfe der Toolbox die Simulation der Forthmaschine. Mit Gerhard machte ich damit das Fein-Tuning des Befehlssatzes, also welche Instruktionen dann tatsächlich reingenommen wurden, wann Branch-Flags gesetzt werden sollten und wann besser nicht usw. Im Sommer letzten Jahres dann holten wir noch Reiner Both dazu, der seine Diplomarbeit machte. Er schrieb den Forthcompiler. Wir haben ihn dann zusammen optimiert, das p-pulling, register-tracking usw.

Heute wollte ich zur Konferenz eigentlich schon die ersten Silicons mitbringen, aber ich hab sie nicht mehr rechtzeitig bekommen. Fertig sind sie. In Zusammenarbeit mit einem Kunden fertigen wir nämlich gerade eine erste Anwendung (psst... geheim), eine Serienfertigung, die im ersten Quartal nächsten Jahres herausgeht.

Was gibt es in der Zukunft? Eurosil wird mehr Peripherie-Bausteine herausbringen: LCD's, Timer, AD-Wandler. Wir arbeiten auch an einer EEPROM-Technik mit sehr niedrigem Verbrauch. Vielleicht kommt auch der Forthchip einmal damit heraus.

(Forth Magazin: Wer daran interessiert sein sollte, soll bitte unbedingt den Mund aufmachen und sich laut und deutlich zu Wort melden. Hier im Magazin und auch bitteschön bei Eurosil. Vielleicht kriegen wir dann den Chip wirklich mit EEPROM - einzeln im Laden!)

Digitalisierung von Oszillogrammen

Norbert Machlitt / Michael Stenzel

Universität München, Sektion Physik

Am Coulombwall 1

D 8046 Garching

Zusammenfassung

Digitalisierung von Oszillogrammen

Die Digitalisierung von Oszillogrammen ermöglicht die rechnergestützte Auswertung und Speicherung oszillographierbarer Vorgänge. Ein einfaches Interface konvertiert das serielle BAS-Videosignal einer Fernsehkamera in ein DMA-kompatibles Datenformat für den ATARI 1040 und erzeugt das Protokoll für den Transfer. Einlesen und Darstellung des Bildes ist in Echtzeit möglich. Das Programm ist in FORTH geschrieben und bietet Datenreduktion, interaktive Kurvenanalyse und Fileververwaltung.

Schlüsselworte :

Digitale Oszilloskop-Kamera
ATARI 1040
DMA Transfer

Das Oszilloskop wird bei vielen physikalischen Messungen zur Darstellung der elektrischer Meßgrößen als Kurve verwendet. Die direkte Art der Wandlung elektrischer Spannungswerte in einen Kurvenzug auf der Braun'schen Röhre hat jedoch den entscheidenden Nachteil, daß dieses Schirmbild latent ist. Eine Aufnahme zur Dokumentation oder auch die Analyse der Kurvenform im Spannungs- oder Zeitbereich (skalieren, logarithmieren, Fourierzerlegung) ist meist nur unter Verwendung photographischer Methoden möglich. Es gibt wohl einige Spezialgeräte wie Digitaloszilloskope, Transientenrekorder oder ähnliche, mit welchen dieser Nachteil überwunden wird, gemeinsam ist diesen Geräten jedoch, daß sie nicht die universelle Verwendbarkeit des Analogoszilloskops haben. Eine Umsetzung der im Oszillogramm enthaltenen Information in ein für numerische Behandlung geeignete Darstellung scheint der beste Weg zu sein, die Möglichkeiten des Analog-Oszilloskops mit den Methoden der rechnerunterstützten Datenanalyse zu verbinden. Bei unserer Anwendung bestand das spezielle Problem, das Bild eines extrem schnellen Oszilloskops mit Speicherröhre, auf dem der Spannungsverlauf an einer Funkenstrecke während eines elektrischen Überschlags aufgezeichnet wird, zu dokumentieren und eine genaue Untersuchung des Kurvenverlaufs vorzunehmen. Dabei sollen aus einer Vielzahl von Messungen (n>1000) Gesetzmäßigkeiten der statistischen Prozesse beim elektrischen Durchschlag abgeleitet werden und somit einer genaueren Beschreibung des Hochspannungsverhaltens komplexer Strukturen dienen. Aus diesen Überlegungen heraus entstand das Konzept einer digitalen Oszilloskopkamera, die diesen speziellen Anforderungen angepaßt wurde. Da nicht nur die reine Digitalisierung und Speicherung von Videobildern, sondern eine integrierte Bildverarbeitung und Datenreduktion im Vordergrund standen, wurde eine weitgehende Softwarelösung angestrebt. Die Hardware besteht hauptsächlich aus einem ATARI 1040 mit Hard-Disk, einer CCD-Videokamera und einer Interface-Platine.

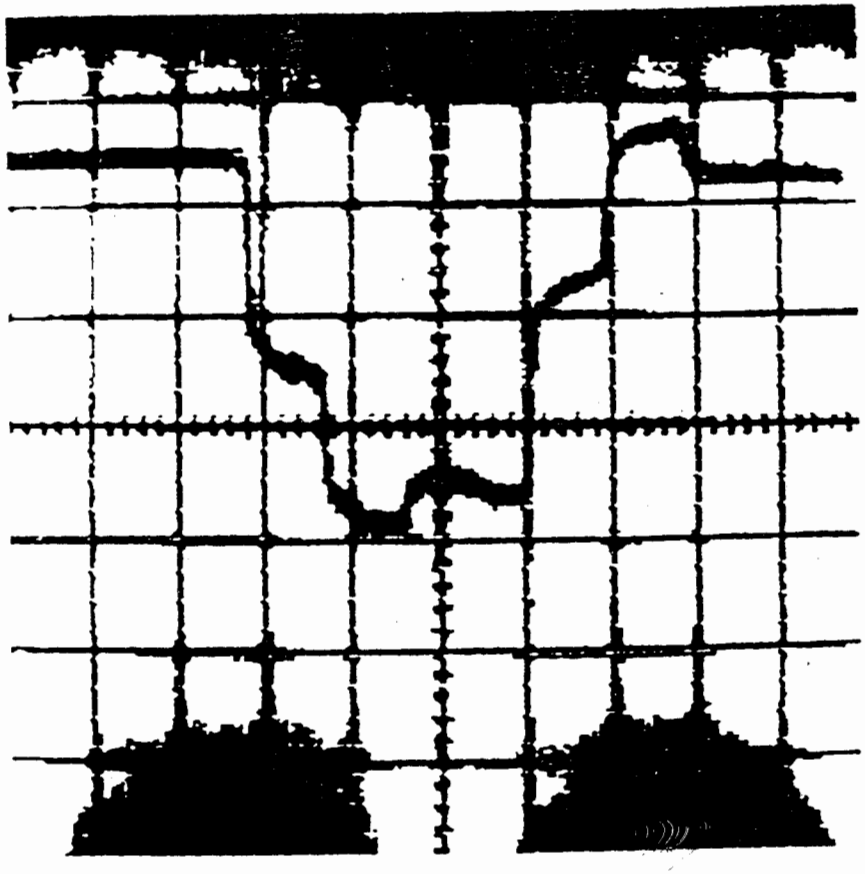
Die Oszillogramme weisen ausreichend Kontrast auf, sodaß die Konvertierung des BAS-Ausgangssignals der Videokamera in ein monochromes Digitalbild den Bildinhalt genügend genau darstellt. Mittels einer einfachen Komparatorschaltung, bei welcher der Vergleichspegel über einen 8 bit DAC gesetzt werden kann, wird der Helligkeitswert in das ortsaufgelöste binäre Muster umgesetzt. Zwei als Wechselpuffer arbeitende 8 Bit Schieberegister ermöglichen den asynchronen Byte-

Transfer über den DMA-Port in den Rechner. Ein Oszillator und zwei Teilerstufen erzeugen, von den Synchronimpulsen getriggert, das nötige Timing für die Abtastung der Bildpunkte und die Weiterschaltung der Zeilen. Die Adressdekoderlogik wurde so konzipiert, daß die am selben Port angeschlossene Hard-Disk weiterhin parallel dazu betrieben werden kann. Der gesamte Aufbau ist mit üblichen Logikbausteinen bestückt und findet auf einer Europakarte Platz. Die Speicherung des Bildes geschieht sofort im Memory des Rechners und steht dort für weitere Bearbeitung zur Verfügung.

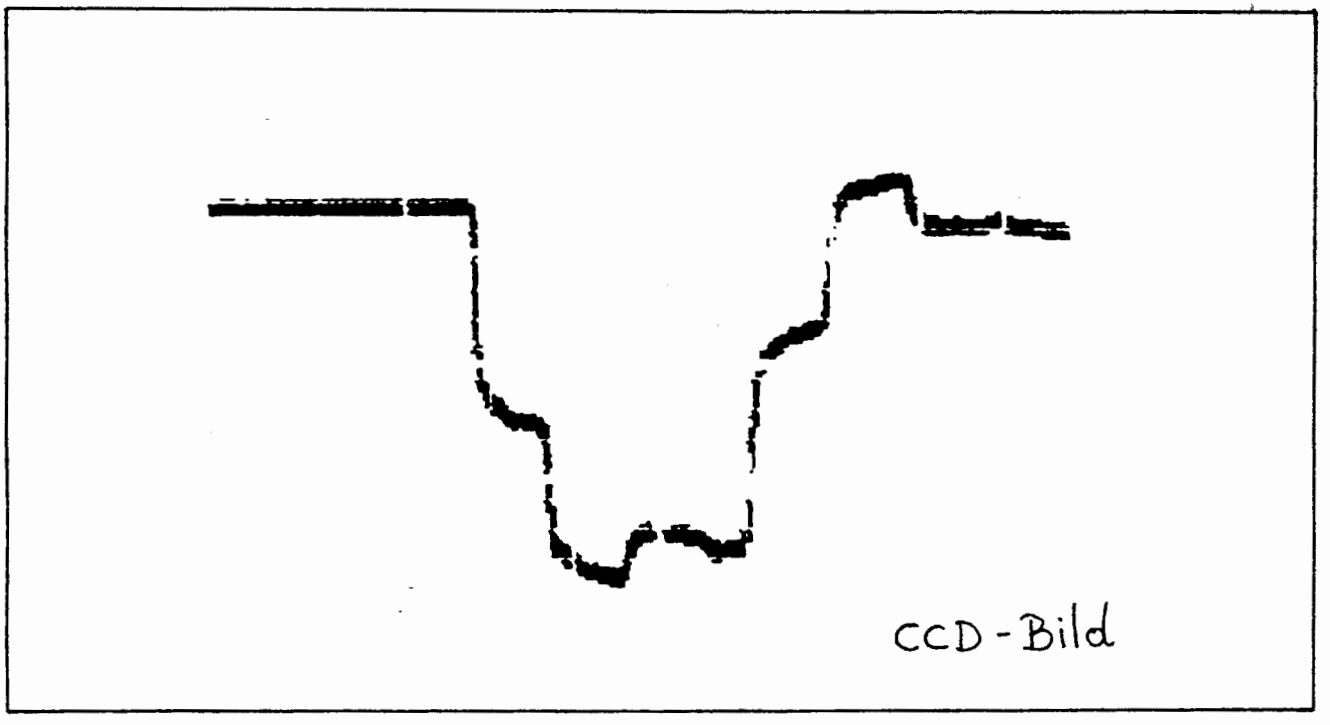
Der Kernteil des eigentlichen Programms ist eine dem Treiber der Hard-Disk ähnliche Routine in FORTH-Assembler. Darin werden auch die Daten für die Ausgabe auf dem Monitor aufbereitet, wobei die Schwelle für den Schwarz/Weiss Wert manuell gewählt werden kann. Ebenfalls ist eine geometrische Mittelwertbildung über die endliche Strahlbreite und somit eine Datenreduktion um den Faktor 16 vorgesehen. Dieses Datenformat (1/2 kByte Kurvendaten, optional noch Zeit und Kommentar ist auch die Schnittstelle zu einem "Replay-Programm", in dem die individuelle benutzerspezifische Aufbereitung der Ergebnisse stattfindet. Eine File-Verwaltung dient der Dokumentation und zur Archivierung der Ergebnisse. Zu jedem Zeitpunkt ist die Aktivierung einer "Help--Funktion" möglich, bei der alle in diesem Status verfügbaren Worte gelistet und kurz erklärt werden. Das gesamte Programm ist in volksFORTH beziehungsweise FORTH-Assembler geschrieben wobei besonders in der Testphase während des Aufbaus die guten interpretativen Eigenschaften voll ausgenutzt wurden, während nun die hohe Geschwindigkeit des kompilierten Teils zum Tragen kommt. Eine Tabelle mit einer Aufstellung der technischen Daten und eine Hardcopy eines Oszillogramms sowie eine vom Raster bereinigte Aufnahme sind im Folgenden zu sehen. Eine weitere Abbildung zeigt den reduzierten und linear interpolierten Plot dieser Kurve mit einem Fadenkreuz und mehreren markierten Punkten.

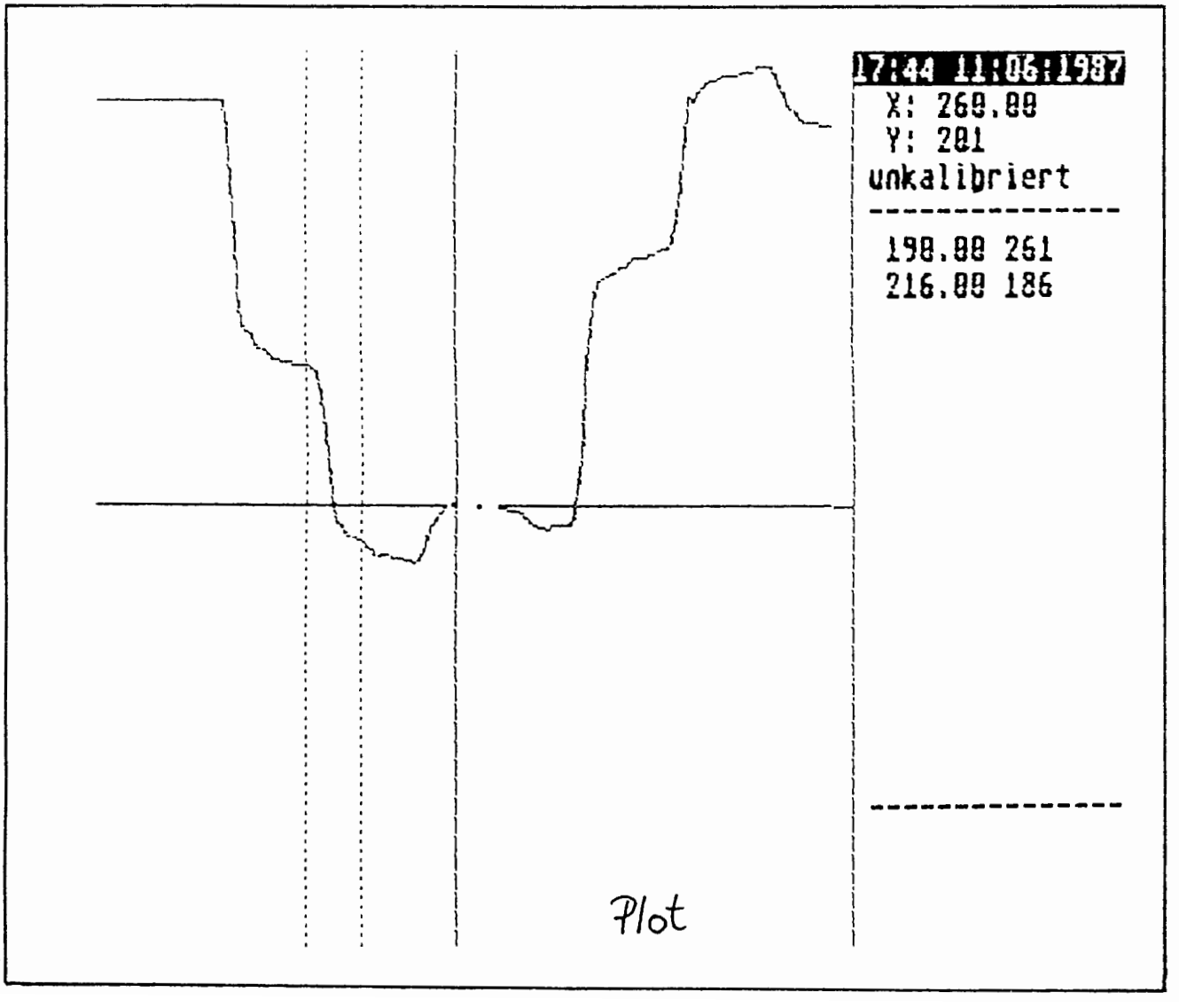
Tabelle : Technische Daten der digitalen Oszilloskopkamera

* Auflösung	: horizontal	256 Punkte
	: vertikal	240 Zeilen
* Kontrastumfang	: maximal 255 Graustufen	
* Digitalisierungszeit	: 16,6 ms (bei 60 Hz Kamera)	
* Bildfrequenz	: 30 Hz (Echtzeit)	
* Speicherbedarf	: 1/2 kByte pro Aufnahme (reduziert)	
* Optionen	: Variable Belichtungszeit	
	: Untergrundreduktion	
	: Darstellung mehrere Grauwerte	
	: Kurvenüberlagerung	
	: Kalibrierung der Zeitachse	



Oszilloskop - Bild


CCD-Bild


17:44 11:06:1987
X: 260.00
Y: 201
unkalibriert
198.00 261
216.00 186
Plot

METHODS> OBJECT-ORIENTED EXTENSIONS REDUX

Terry Rayburn
forthlab Unternehmensbereich der RSO GmbH
Postfach 82 05 23
8000 Muenchen 82
BRD

ABSTRACT

The METHODS> approach allows a layer of object-oriented extensions to be superimposed on Forth systems without modification to the underlying structure such as adding multiple code fields. In typical Forth fashion, the operators and operands are defined by the programmer to suit the application. The METHODS> kernel only provides the glue to bind them together.

Experience with the approach since the last presentation at euroFORML 1985 has resulted in simplifications to aid portability and a change of syntax that makes code more consistent and readable.

THE IMPORTANCE OF SYNTAX

One of the things I've learned after a few painful years of Forth programming is to ask early in the design phase, "What do I want this code to look like?" This seems like a simple enough idea, but I don't know how many times I've created code difficult to read, understand and re-use simply from lack of attention to this question. To be more precise, I've recently been looking at Forth code from three viewpoints and this has helped me stratify my designs into separate layers, each with an independent goal and scope.

First I've been asking, "How can I make my code look like the machine?" In the lowest layer, my goal is to create an interactive software interface to the machine that does the machine functions, and little else. What I like to do is create Forth words with the same names as chip registers, chip instructions, ports or firmware functions. Since I make a point of excluding any of the ideas of the application at this level, any programmer on the team could build from this base to start a new application. The behavior of these words is generally described in the hardware/ firmware manual in more detail than I wish to add in comments. By using the same names and structure, I make the relevant features easy to find and use.

Second I ask, "How do I want this to look to the programmer?" This is where the conventions of data processing come in handy. I use Forth's power to create classes of data structures and operators. I let the operations on the data structures hide the complexity of the chip instructions, consciously creating a layer of tools for the programmer to do things "in the large" the way the application designer might conceive them. You do the same thing when you replace

```
1C7 PORT@   3 XOR   1C7 PORT!
```

with

```
RED FILTER THROW
```

Third I ask, "How do I want this to look to the user?" The user interface is expensive in terms of complexity, memory and time resources in modern systems. It has to be, in order to be competitive. It's one of the few places where the customer can see your cleverness, and programmers live or die by their cleverness. Moreover, market forces continue to drive products to differentiate themselves when performing the same basic function. Sophisticated algorithms, which are internal, are harder to sell than flash and glitter which is visible.

Most of the confusion I've seen in my code has been the result of lumping the programmer stratum in with the end-user stratum, leaving too many complex functions doing work directly at too low a level. User interfaces are subject to trends, whims and fashions. If I can provide an intact programmer interface, the program might still attract admirers with a little consmetic surgery late in life. This paper is about the machine and programmer interfaces.

THE PROBLEM WITH OPERATORS

I've noticed over and over again how hard it is to come up with good names for the verbs or operators in my desired syntax. Apparently in computing there are really only a very few activities that we do, and mostly we twiddle the form and location of data. I found myself over and over again creating phrases like:

```
n COLOR !
n FILTER.COLOR.C!
n ACTIVE.FILTER.COLOR.PORT!
```

when what I really wanted in all three cases was:

```
PUT
```

In fact, when I really got into Forth and started creating a couple of dozen CREATE ... DOES> definitions per project I realized that I was applying only a small number of basic operations to the data but having to scratch my head to come up with two dozen different names. Those basic operations are:

```
PUT       or STORE          or TO
GET       or FETCH          or FROM
AUGMENT   or INCREMENT      or STEP
SET       or ON
RESET     or OFF
SEE       or DISPLAY
```

As a result of these and other frustrations, I found myself turning to the "TO SOLUTION" as proposed by Paul Bartholdi [BAR79]. This idea has been

the germ for a lot of work. Notably, Schleisiek explored multiple code fields and forward reading operators [SCH83]; Duff and Iverson implemented a fully object-oriented language descended from Forth and Smalltalk [DUF84]; and Pountain provided a set of object-oriented extensions to Forth rather than create a new language [POU86]. The latter was published after I had already committed to my approach. What I wanted was a set of words that would let me define a data structure and the operators that know about that specific data structure but can be referred to with general purpose operator names. I presented this structure at the last EuroForml and after a couple of years of living with it I would like to correct my more egregious errors.

A FAILURE OF SYNTAX

One of the things I did in my previous attempt was to use a forward reading syntax:

```
n TO COLOR
n TO ACTIVE.COLOR.FILTER
```

This seemed a natural thing to do. I was anxious to show that Forth could adapt to a "normal" structure and be more accessible to novice programmers. It turned out to be a big mistake, but it took me about a year to figure it out. For one thing, I'm not really interested in creating a whole new language. I have had to code in Forth from many different vendors (or non-vendors in the case of F83), sometimes 3 or 4 on a project. It turns out that mixing infix and postfix syntax makes life much harder on the novice programmer, who still has to deal with normal Forth in much of the rest of the application. The problem becomes apparent when you compare phrases for moving data to a position on an indexed array:

```
n  1 4 3  TO ACTIVE.COLOR.MATRIX
```

leaving the indices to the array too far away from the object to be clearly understood as modifiers. Fully consistent infix notation wouldn't be bad. It would look something like:

```
n TO ACTIVE.COLOR.MATRIX(1,4,3)
```

But this kind of structure is hard to do in a single pass compiler, and remains jarringly out of place unless you intend to rewrite the whole language. So, I reverted to the more Forth-like syntax:

```
n 1 4 3 ACTIVE.COLOR.MATRIX TO
```

It's no accident that Forth is an RPN language. The power became apparent when I saw how it simplified my "generic operator" extensions. The simpler code is also simpler to port between Forths, making it easier to give those multi-vendor projects the same "look" in the programmer and machine interfaces.

A SIMPLE-MINDED LOOK AT A SIMPLE-MINDED IDEA

You can call this idea what you want. You can think of it as binding data structures to their operators. You can think of it as structured programming extended beyond the flow of control to the data itself. You can call it information hiding. You can call it object-oriented programming. I always choose the name that has the most current impact to make myself seem more important. Here's the basic idea: I want each defined instance of a data structure to identify itself by type. Somewhere in the program I can find a table of operators for that type. Each position in the table is associated with a general idea of an operation I might wish to perform on that datum. Suppose we had 8 bit, 16 bit, and 32 bit integer data structures that we wished to access. We might want operators to fetch, store and display the data.

Generic operation	Specific operation tables		
	8 bit	16 bit	32 bit
GET	C@	@	2@
PUT	C!	!	2!
VUE	C?	?	2?

You can see that I've taken the liberty to propose a few non-standard definitions to display the data. You might be able to guess how they would work. Now I admit this is a trivial example. In real life, the data structures that I need are more challenging: boolean "logicals", floating point numbers, pre-scaled numbers, vectors, matrices, strings, read-only ports, write-only ports, ports with inverted logic, indexed records, fields within records and so on ad infinitum.

Forth already has defining words for binding data structures to operations: CREATE creates an instance of the data structure identified with a head, the next few words allocate the storage area and the parameters required for the data structure and then DOES> points to the run-time action of the object. In the head of new definition, the instance of the data type, is a code field. In the code field is an address that uniquely identifies the data type. All definitions with the same contents in the code field are members of the same family. The contents of the parameter field(s) make them unique. Now, the only change that I want is to be able to have multiple operations in my defining word instead of one. I could define:

```
: HALF      CREATE  0 C,     DOES> EXIT  C@  C!  C? ;
: INTEGER   CREATE  0 ,      DOES> EXIT   @   !   ? ;
: DOUBLE    CREATE  0 , 0 ,  DOES> EXIT  2@  2!  2? ;
```

Instances of these types leave the address of their parameter areas on the stack at execution time. Now, I only need to define generic operation words that can find the operations table from the parameter field address. They back up to the code field, get the address of the native code stub compiled by DOES> and index forward to the appropriate object-specific operation.

THE CONTROVERSIAL EXIT

In my newest version of this idea, the words DOES> EXIT are replaced with the definition METHODS>. It is defined:

```
: METHODS>   [COMPILE] DOES>  COMPILE EXIT ;  IMMEDIATE
```

No one really likes the EXIT being there. It looks untidy and wasteful. In fact, in my previous version I went to great lengths to build a definition of METHODS> that executed the first method in the table as a default and then returned to the caller. This provided a default behavior when the object was invoked without a preceeding operator.

There were three problems with the approach. First was the infix notation itself, as outlined above. Second, the complexity of the little code segment compiled by METHODS> was excruciatingly difficult to explain and slowed down porting the idea to other Forth implementations. Third, I found that I often needed access to the object parameter area. Strictly speaking, the idea of binding objects to operators is to restrict access to data structures to a set of well-defined and well-tested procedures. In real applications I found that there was always some object that was identical to other members of the family except for one tiny requirement in the specification. Having the object leave its address gave me a way to access these data structures in some ad hoc manner. More often, I needed to dump the parameter area to check on operation of the code. Getting the parameter field address as the default behavior of a named object encourages the interactive debugging that is such an important part of the Forth programming style.

So, the EXIT that looks lazy and crude is actually a simplification that I've reached after a lot of field trail.

LATE BINDING AND FLEXIBILITY, EARLY BINDING AND EFFICIENCY

The generic operators that I suggested earlier might be classified as late-binding. That is, the object-specific operator is found at run-time and executed. This provides a great deal of flexibility. In a data entry application, the kernel algorithm of field retrieval, display, editing, and storing could be written with the record and field selection being external to the algorithm. At compile time, we don't know which objects are going to be operated on; the user chooses them during execution. We would want operators of the sort already described. These operators also work in the Forth interpreter, so no special test code has to be defined to access the objects.

But suppose that I wanted maximum efficiency at run-time. I would like the generic operator to do some of the work at compile time, looking up the object-specific operation and compiling it directly. Since the last thing compiled was the address of the code field of the object, my operation can find the METHODS> table and extract the proper one to compile. This requires that the object and operation be named together as in the fragment:

```
INTEGER CYCLES
: xxx   ... CYCLES [GET]  ...  ;
```

Operators of this sort are called early-binding operators. I use the convention of putting square brackets around the same name as the corresponding late-binding operator to indicate compile-time action.

An obvious variation would be to make all the generic operators "state-smart," having one behavior in definitions and the other when being interpreted. If you want to do this, you will need a compiler switch to toggle between modes, since you will not want early-binding in every definition. I find it too much trouble.

I use a defining word METHOD: to define late-binding operators and a defining word [METHOD]: to define early-binding operators. They take as a parameter an index into the METHODS> table. For instance:

```
0 METHOD: GET                0 [METHOD]: [GET]
1 METHOD: PUT                1 [METHOD]: [PUT]
2 METHOD: VUE                2 [METHOD]: [VUE]
```

AN EXAMPLE

The attached listing is a long example of this approach. F83 for the IBM-PC was used to program the ubiquitous 8250 UART. This chip has single bit and multiple bit fields, multiple register addresses, special latching mechanisms, read-only registers and write-only registers; in short, sufficient to demonstrate how all the chip features can be accessed and the interface remain consistent. At the end of the declarations section is an example of "application layer" code for initializing and testing the chip. This is only an example. This technique has actually been used on much more ambitious projects including real-time control of custom hardware; a Hitachi HD3484 Advanced CRT Controller graphics package; and multi-processing objects for a VMEbus project with multiple 68000's.

POSSIBLE EXTENSIONS

This package as presented does no error checking. I frequently build a version of the operators that perform range and validity checking and use them in the early phases of the project, substituting simpler and faster ones when the application code has been checked out.

A more pertinent question is: how do I bind generic operators to specific classes of objects? In the example, I make the read-only register have a write operation that is "not defined". But registers that are read/write and registers that are read-only are in the same general class. What we might want to prohibit is applying an INCREMENT operation designed for an INTEGER data type to a STRING data type, where it would have little meaning. The structure as defined provides no protection mechanism of this sort.

A common feature of object-oriented programming is the ability to define simply new classes of objects. In one idea, the programming environment

infers operations and structure from a model or example of the data known as a template. Another fertile field for study is inheritability: the declaration of a class connects it to a super-class. It inherits the data structure and/or operational characteristics of the family. Unique characteristics are then declared which become part of the inheritance this class would pass to future sub-classes as they are defined. For full treatments of this subject, see [POU86] and [DUF84].

Finally, these words lack an "message passing" structure for communicating between independent objects with parameters and selectors. The overhead of message passing systems would have been too much for my real-time applications, so I am content with the weaknesses in my implementation. Perhaps without message-passing and hierarchical classes this is too primitive to be called "object-oriented." If it is, you may exercise your Forth-like option to change the name.

The strength of these METHODS> definitions as refined in the last two years is their simplicity and portability. Extensions of the sort described above probably fit better in new or experimental Forth-like languages rather than this package whose proper home is just under the application layer in a standard Forth, but if you discover anything interesting, let me know!

REFERENCES

[BAR79] P. Bartholdi, "The TO Solution", Forth Dimensions, vol 1, no.4.

[SCH83] K. Schleisiek, "Multiple Code Field Data Types and Prefix Operations", Journal of Forth Application and Research, vol 1, no.2.

[DUF84] C. Duff and N. Iverson, "Forth Meets Smalltalk", Journal of Forth Application and Research, vol 2, no.3.

[POU86] D. Pountain, "Object Oriented Extensions to Forth", Journal of Forth Application and Research, vol 3, no.3.

FORTSETZUNG S. 34

217 — F83 SOURCE $20/21/30 _____
Henry Laxen & Michael Perry
A complete listing of F83 including source and shadow screens. Includes introduction on getting started.

218 — FOOTSTEPS IN AN EMPTY VALLEY
(NC4000 Single Chip Forth Engine) $25/26/35 _____
Dr. C. H. Ting
A thorough examination and explanation of the NC4000 Forth chip including the complete source to cmForth from Charles Moore.

219 — FORTH: A TEXT AND REFERENCE $22/23/33 _____
Mahlon G. Kelly & Nicholas Spies
A textbook approach to Forth with comprehensive references to MMS-FORTH and the 79 and 83 Forth Standards.

220 — FORTH ENCYCLOPEDIA $25/26/35 _____
Mitch Derick & Linda Baker
A detailed look at each fig-FORTH instruction.

225 — FORTH FUNDAMENTALS, V.1 $16/17/20 _____
Kevin McCabe
A textbook approach to 79-Standard Forth

230 — FORTH FUNDAMENTALS, V.2 $13/14/18 _____
Kevin McCabe
A glossary.

232 — FORTH NOTEBOOK $25/26/35 _____
Dr. C. H. Ting
Good examples and applications. Great learning aid. PolyFORTH is the dialect used. Some conversion advice is included. Code is well documented.

233 — FORTH TOOLS $22/23/32 _____
Gary Feierbach & Paul Thomas
The standard tools required to create and debug Forth-based applications.

235 — INSIDE F-83 $25/26/35 _____
Dr. C. H. Ting
Invaluable for those using F-83.

NEW
237 — LIBRARY OF FORTH ROUTINES AND UTILITIES
James D. Terry $23/25/35 _____
Comprehensive collection of professional quality computer code for Forth; offers routines that can be put to use in almost any Forth application, including expert systems and natural language interfaces.

240 — MASTERING FORTH $18/19/22 _____
Anita Anderson & Martin Tracy
A step-by-step tutorial including each of the commands of the Forth-83 International Standard; with utilities, extensions and numerous examples.

245 — STARTING FORTH, 2nd Edition (soft cover)
Leo Brodie $20/21/30 _____
NEW
In this new edition of Starting Forth, the most popular and complete introduction to Forth, syntax has been expanded to include the new Forth '83 Standard.

246 — STARTING FORTH (hard cover) $20/21/30 _____
Leo Brodie

255 — THINKING FORTH (soft cover) $16/17/20 _____
Leo Brodie
The sequel to "Starting Forth". An intermediate text on style and form.

265 — THREADED INTERPRETIVE LANGUAGES
R. G. Loelinger $25/26/35 _____
Step-by-step development of a non-standard Z-80 Forth.

267 — TOOLBOOK OF FORTH $23/25/35 _____
NEW
(Dr. Dobb's)
Edited by Marlin Ouverson
Expanded and revised versions of the best Forth articles collected in the pages of Dr. Dobb's Journal.

270 — UNDERSTANDING FORTH $3.50/5/6 _____
Joseph Reymann
A brief introduction to Forth and overview of its structure.

■ *ROCHESTER PROCEEDINGS*

The Institute for Applied Forth Research, Inc. is a non-profit organization which supports and promotes the application of Forth. It sponsors the annual Rochester Forth Conference.

321 — ROCHESTER 1981 $25/28/35 _____
(Standards Conference)
79-Standard, implementing Forth, data structures, vocabularies, applications and working group reports.

322 — ROCHESTER 1982 $25/28/35 _____
(Data bases & Process Control)
Machine independence, project management, data structures, mathematics and working group reports.

323 — ROCHESTER 1983 $25/28/35 _____
(Forth Applications)
Forth in robotics, graphics, high-speed data acquisition, real-time problems, file management, Forth-like languages, new techniques for implementing Forth and working group reports.

324— ROCHESTER 1984 $25/28/35 _____
(Forth Applications)
Forth in image analysis, operating systems, Forth chips, func-tional programming, real-time applications, cross-compilation, multi-tasking, new techniques and working group reports.

325 — ROCHESTER 1985 $20/21/30 _____
(Software Management & Engineering)
Improving software productivity, using Forth in a space shuttle experiment, automation of an airport, development of MAGIC/L, and a Forth-based business applications language; includes working group reports.

■ *THE JOURNAL OF FORTH APPLICATION & RESEARCH*

A refereed technical journal published by the Institute for Applied Forth Research, Inc.

401 — JOURNAL OF FORTH RESEARCH V.1
Robotics/Data Structures $30/33/38 _____

403 — JOURNAL OF FORTH RESEARCH V.2 #1
Forth Machines $15/16/18 _____

404 — JOURNAL OF FORTH RESEARCH V.2 #2
Real-Time Systems $15/16/18 _____

405 — JOURNAL OF FORTH RESEARCH V.2 #3
Enhancing Forth $15/16/18 _____

406 — JOURNAL OF FORTH RESEARCH V.2 #4
Extended Addressing $15/16/18 _____

407 — JOURNAL OF FORTH RESEARCH V.3 #1
Forth-based laboratory systems and data structures.
$15/16/18 _____

409 — JOURNAL OF FORTH RESEARCH V.3 #3
Application Languages $15/16/18 _____

410 — JOURNAL OF FORTH RESEARCH V.3 #4
Applications, Arthmatic extensions $15/16/18 _____

■ *DR. DOBB'S JOURNAL*

This magazine produces an annual special Forth issue which includes source-code listing for various Forth applications.

422 — DR. DOBB'S 9/82 $5/6/7 _____
423 — DR. DOBB'S 9/83 $5/6/7 _____
424 — DR. DOBB'S 9/84 $5/6/7 _____
425 — DR. DOBB'S 10/85 $5/6/7 _____

0

```
0 \ methods                                              tgr12aug87
1 ************************************************************
2 methods by terry rayburn
3
4
5
6
7
8
9
10
11
12
13
14
15
```

1

```
0 \ methods                                                  tgr
1 : METHODS>   [COMPILE] DOES>   COMPILE EXIT ;   IMMEDIATE
2
3 \ early binding
4 : [METHOD]:   CREATE  2* ,  IMMEDIATE DOES>   HERE 2- @  @
5    5 +  SWAP @ +  @ , ;
6
7 \ late binding
8 : METHOD:   CREATE 2* ,  DOES>  @ OVER 2- @  5 + +  PERFORM ;
9
10 0 METHOD: GET          1 METHOD: PUT          2 METHOD: VUE
11 0 [METHOD]: [GET]      1 [METHOD]: [PUT]      2 [METHOD]: [VUE]
12
13
14
15
```

METHODS> is used in the form CREATE ... METHODS> to create an object whose run-time operators follow in a table. The basic behavior of a methods definition is to return the address of its parameter area whenever the object is invoked.
[METHOD]: creates generic early-binding (compile-time) operators that know how to find the object-specific behavior of the word compiled just before them. The CFA of an METHODS> object points to a code stub just before the table of operations. These words compile the specific-behavior, so cannot be used interactively.
METHOD: creates generic late-binding (run-time) operators that find their object-specific behavior from the PFA of the object on the top of stack. These are less efficient operators but may be used interactively or compiled in a definition where the programmer wishes to leave the identification of the object until run-time.

2

```
0 \ 8250 serial port
1 HEX
2 VARIABLE PORT.ADDRESS
3 : COM1   3F8 PORT.ADDRESS ! ;
4 : COM2   2F8 PORT.ADDRESS ! ;
5
6 : +P ( o -a)   PORT.ADDRESS @ + ;
7
8
9
10
11
12
13
14
15
```

PORT.ADDRESS holds the address of the current serial port.
COM1 and COM2 declare the current serial port by storing the address in PORT.ADDRESS.

+P takes a register offset and adds it to PORT.ADDRESS.

3

```
0 \ shifts                                              tgr
1 HEX
2 CODE <SH ( n m - n')
3   CX POP   AX POP   BEGIN   CX ROR   8000 # CX TEST
4   0= WHILE   AX SHL   REPEAT   1PUSH   C;
5 CODE >SH ( n m - n')
6   CX POP   AX POP   BEGIN   CX ROR   8000 # CX TEST
7   0= WHILE   AX SHR   REPEAT   1PUSH   C;
8 DECIMAL
9
10
11
12
13
14
15
```

These shift words convert data between the bit positions in a register and the right-justified form on the stack that a human finds more convenient. A tip of the hat to Dave Johnson for this idea.
These words are implemented in code in the example as they would be in a typical project. The complexity of the data access is completely hidden in well tested data structures.
<SH takes a mask and a value. The value is shifted left until it fits right-justified in the space indicated by the 1 bits of the mask. The extraneous bits of the value are masked out, just in case.
>SH takes a mask and a value. The value is masked off so that only the field of interest marked by 1 bits in the mask remain. Then the value is shifted until it is right justified on the stack.

4

```
0 \ DATA TYPE - BITS:
1 : @BITS ( a - n)   2@ +P PC@  OVER AND  SWAP >SH ;
2 : !BITS ( n a)   2@  2DUP +P PC@  SWAP  -1 XOR AND >R
3   >R  DUP >R  <SH R> AND   R> SWAP R>  OR  SWAP +P PC! ;
4 : .BITS ( a)   @BITS . ;
5
6
7 : BITS: ( m n)   CREATE , ,   METHODS> @BITS  !BITS  .BITS ;
8
9
10
11
12
13
14
15
```

The BITS: data type is for bit fields up to 8 bits wide in the registers.
All of these 'object-specific' operators expect the address of the parameter field of the object on the stack. At that address is the offset from the port base address in PORT.ADDRESS of this register. The next parameter field holds the mask that defines the field location and width with 1 bits.
@BITS fetches the register contents, masks off the defined field and right justifies it on the stack.
!BITS masks a field into a copy of the register value and writes it into the register.
.BITS displays the register contents.

m n BITS: xxx creates an instance xxx of data type BITS: with mask m to be applied to port offset n.

5

```
0 \ interrupt enable register                           tgr
1 1 1 BITS: EDAI
2 2 1 BITS: ETXHREI
3 4 1 BITS: ERLSI
4 8 1 BITS: EMSI
5
6 \ interrupt identification register
7 1 2 BITS: -IP
8 6 2 BITS: IID
9
10
11
12
13
14
15
```

We choose names for the substructures of the interrupt enable register and interrupt identication register that suggest the 8250 chip documentation labels. The interrupt pending bit is named -IP to recall its negative logic.

6

```
0 \ line control register
1   3 3 BITS: WLS
2   4 3 BITS: STB
3   8 3 BITS: PEN
4  16 3 BITS: EPS
5  32 3 BITS: STICK.PARITY
6  64 3 BITS: SET.BREAK
7 128 3 BITS: DLAB
8 \ line status register
9   1 5 BITS: DR
10  2 5 BITS: ORE
11  4 5 BITS: PE
12  8 5 BITS: FE
13 16 5 BITS: BI
14 32 5 BITS: THRE
15 64 5 BITS: TSRE
```

tgr

The names of the substructures of the line control register and the line status register keep their 8250 chip names except ORE is used for OverRunError instead of "OR".

7

```
0 \ modem control register
1   1 4 BITS: DTR
2   2 4 BITS: RTS
3   4 4 BITS: OUT1
4   8 4 BITS: OUT2
5  16 4 BITS: LOOPBACK
6 \ modem status register
7   1 6 BITS: DCTS
8   2 6 BITS: DDSR
9   4 6 BITS: TERI
10  8 6 BITS: DRLSD
11 16 6 BITS: CTS
12 32 6 BITS: DSR
13 64 6 BITS: RI
14 128 6 BITS: RLSD
15
```

tgr

The names of the substructures of the modem status register and the modem control register keep the 8250 chip names except we use LOOPBACK for what the call "LOOP".

8

```
0 \ DATA TYPE - READ.ONLY:
1 : @ROR ( a -n)   @ +P  PC@ ;
2 : -defined    ABORT" method not defined for this data type " ;
3 : .ROR ( a)   @ROR EMIT ;
4
5 : READ.ONLY: ( n)   CREATE ,  METHODS>  @ROR  -defined  .ROR ;
6
7
8
9
10
11
12
13
14
15
```

This is the READ.ONLY: data type for 8 bit registers.
All of these "object-specific" operators expect the address of the parameter field of the ooject on the stack. At that address is the offset from the port base address in PORT.ADDRESS of this register.

@ROR put on the stack the contents of the register.
.ROR emits the last recieved character.
n READ.ONLY: xxx creates an instance xxx of data type READ.ONLY: with port offset n. Note that the PUT method is undefined.

9

```
0 \ DATA TYPE - WRITE.ONLY:
1 : !WOR ( n a)   2DUP !  2+ @ +P PC! ;
2 : .WOR ( a)   @ EMIT ;
3
4 : WRITE.ONLY: ( n)   CREATE 0 , ,  METHODS>   @  !WOR  .WOR ;
5
6
7
8
9
10
11
12
13
14
15
```

This is the WRITE.ONLY: data type for 8 bit registers. In case we want to find out what was in that register, we keep a copy of it local to the declared instance.
All of these 'object-specific' operators expect the address of the parameter field of the object on the stack. At that address is the offset from the port base address in PORT.ADDRESS of this register.

!WOR writes a copy of the 8 bit data on the stack to the register and makes a local copy.
.WOR emits a copy of the last transmitted character.
n WRITE.ONLY: xxx creates an instance xxx of data type WRITE.ONLY: with port offset n. Note that the GET method is simply @.

10

```
0 \ DATA TYPE - SPECIAL:                                  tgr
1 : @SR ( a -n)   1 DLAB [PUT]  @ +P  P@  0 DLAB [PUT] ;
2 : !SR ( n a)   1 DLAB [PUT]  @ +P  P!  0 DLAB [PUT] ;
3 : .SR ( a)   @SR  . ;
4
5 : SPECIAL: ( n)   CREATE ,  METHODS> @SR  !SR  .SR ;
6
7
8
9
10
11
12
13
14
15
```

The SPECIAL: data type is for registers that require that DLAB be set before access. In the 8250, there is only one such register. It is also the only one that can be thought of as being 16 bits wide.
All of these 'object-specific' operators expect the address of the parameter field of the object on the stack. At that address is the offset from the port base address in PORT.ADDRESS of this register.
@SR fetches the register contents 16 bits wide.
!SR stores the 16 bit value on the stack into the register.
.SR displays the register contents.

n SPECIAL: xxx creates an instance xxx of data type SPECIAL: with port offset n.

11

```
0 \ Remaining registers                                   tgr
1 0 READ.ONLY: RX
2 0 WRITE.ONLY: TX
3 0 SPECIAL: DIVISOR.LATCH
4
5
6
7
8
9
10
11
12
13
14
15
```

We define the remaining 8250 registers here. Note that these are all 'single' instances of the data type, and might have been handled just as well outside the scope of METHODS>. In a real application, more instances of these types might appear. But even when they don't, it is sometimes useful to have the same unifying operators anyway.
RX is the 8250 read buffer for incoming bytes.
TX is the transmit buffer for outgoing bytes.
DIVISOR.LATCH holds the divisor that affects the baud rate.

12

```
0 \ 8250 initialization                                  tgr
1 : BAUD ( n)   >R  1843200. 16 MU/MOD  R> M/MOD
2   DIVISOR.LATCH [PUT]   2DROP ;
3
4 3 CONSTANT EVEN
5 0 CONSTANT NO
6 1 CONSTANT ODD
7 : PARITY ( n)   2 /MOD  EPS [PUT]   PEN [PUT]  ;
8
9 : BITS ( n)   5 - WLS [PUT] ;
10
11 : SET ( n)   TRUE SWAP  PUT ;
12 : RESET ( n)   FALSE SWAP  PUT ;
13
14
15
```

On this screen is defined what I call a 'programmer interface.' It consists of the words that support what the application programmer is likely to want to do easily.

BAUD sets the baud rate for the current port.
EVEN, NO and ODD are used with PARITY to set the appropriate bits.

BITS sets the port for 5,6,7 or 8 bits. Range checking could be added.

SET and RESET are probably superfluous but demonstrate 'late binding' (run-time). Given a data structure address, SET sets the register bit and RESET clears it to zero.

13

```
0 \ 8250 self test                                       tgr
1 : wait ( reg - f)   32000 0 DO  DUP GET IF  0= LEAVE  THEN  LOOP
2   DUP IF  LOOPBACK RESET  THEN ;
3 : SELF.TEST
4   DIVISOR.LATCH [GET]  DUP   1+ DUP  DIVISOR.LATCH [PUT]
5   DIVISOR.LATCH [GET] = NOT ABORT" port not available "
6   DIVISOR.LATCH [PUT]   LOOPBACK SET   0
7   1024 0 DO  I C@  DUP  TX [PUT]
8      THRE WAIT  ABORT" transmitter holding register stuck "
9      TSRE WAIT  ABORT" transmitter shift register stuck "
10     DR WAIT  ABORT" data never received "
11     RX [GET] = NOT IF  1+  THEN
12  LOOP  LOOPBACK RESET  . ." /1024 transmission errors " ;
13
14
15
```

wait takes a named object - a single BIT: data type register - and waits for that bit to be set. If it does not become set, the test is over and loopback is cleared. A TRUE condition on output is a failure.

SELF-TEST operates on the current COMx port. It tries to change the baud rate just to see if the port is alive. It then watches the progress of bytes through the transmit and recieve registers

14

```
0 \ initialization examples                              tgr
1 COM1   19200 BAUD   8 BITS  ODD PARITY   DTR SET  RTS RESET
2   CR   .( COM1 self test in progress )  SELF.TEST
3 COM2   50 BAUD  5 BITS  NO PARITY   DTR RESET   RTS RESET
4   CR   .( COM2 self test in progress )  SELF.TEST
5
6
7
8
9
10
11
12
13
14
15
```

Remember, this is only an example.

F83 Editor für C128 CPM+ und CPC464/6128

Rudolf Mensing, Kaltenkirchen

Der hier vorgestellte Editor basiert auf einem kleinen Editor von A.Goppold und R.Bouteiller (1), der wiederum an einen Vorschlag aus FORTH DIMENSIONS angelehnt ist (2). Ich habe diesen Editor weiter bearbeitet und für das F83 von Laxen und Perry zurecht gemacht, welches wir auf einem Commodore 128 mit CPM+, Schneider CPC464 und CPC6128 CPM Plus haben. Die Quelle habe ich kommentiert, sodas sich hier weitere Erklärungen erübrigen. Für weitere Anregungen wäre ich dankbar.

(1) A.Goppold, Forth - ein Programmiersystem ohne Grenzen.
(2) S.H.Daniel, The Forth Inc. Line Editor, Forth Dimensions III/3 (1981), S.80-88.

PS:

Wenn ich in FORTH ßrogrammiere fallen mir oft als erstes plattdeutsche (neuniederdeutsche) Worte ein, oft wohl durch das €nglische, da sie der englischen Lautung ähnlicher ist als die #ochdeutsche.

Das Plattdeutsche besitzt keine verbindliche Rechtschreibung.

So schreibe ich einige Besonderheiten, die nicht allgemein üblich sind.

z.B. folgende:

ei	wie das a in englisch age
ou	ähnlich ow wie in englisch bowl
oy oi	wie eu in heute
eu	wie ö in König, langer Vokal
oe	wie ö in können, offener Laut
y	wie ü Hülle und wie üh in fühlen
c	wie k in komm
k	wie ch in ich oder in ach
sk	wie sch in Schule
w	wie w in wer
v	oft wie b oft wie w
z	am Wort- oder Silbenende wie s in so, stimmhafter S-Laut
s	vor Vokal stimmhaft, sonst stimmlos

die anderen Buchstaben entsprechen dem Hochdeutschen.

ein Beispiel:

Cinners, ji meut tou skoul, dei cloc is al akt.
Kinder, ihr müßt zur Schule, es ist schon 8 Uhr.

Sou, nun mooc ic slus,	So, ich mache nun Schluß
ic groyt Jou	ich grüße Sie
Joun	Ihr

Rudolf Mensing

Page No 1 volksFORTH83 der FORTH-Gesellschaft eV EDITOR.BLK

0

```
0 \ EINLEITUNG                                        Rudolf Mensing
1
2 Dieser Editor basiert auf den kleinen Editor veroeffentlicht in
3              andreas goppold / roger bouteiller
4          forth :  ein programmiersystem ohne grenzen
5           von A. Goppold nach FORTH DIM II/3 ,  p83
6
7
8 Dieser Editor kommt mit wenigen Befehlen des Betriebssystem aus.
9
10 Notwendig sind :            Bildschirmloeschen
11                   Cursor an jede Bildschirmposition bringen
12
13
14
15
```

1

```
0 \ lodskerm                                    13jun87msg  Ladescreen
1
2 \  2    LOAD    \ CPC 464   CPC 664   CPM 2.2
3    3    LOAD    \ CPC 6128            CPM Plus
4 \  4    LOAD    \ Commodore 128       CPM+
5   22    LOAD    \ COPY und -COPY
6   23    LOAD    \ PRT WLIST
7   24 25 THRU    \ printer
8    5 14 THRU    \ editer
9
10
11
12
13
14
15
```

2

```
0 \ CPC464        CPM 2.2                       14jun87msg  CPC 464        CPM 2.2
1
2 EDITOR CPC-464 FORTH CAPS ON
3 ' EPSON     IS INIT-PR
4 ' (WHERE)   IS WHERE
5 : DEMIT ( c1 c2 --)  SWAP EMIT EMIT   ;
6 : XY    ( col row --)  31 EMIT SWAP 1+ EMIT 1+ EMIT  ;
7 : CLS   12 EMIT ( --)  ;
8 ' CLS IS DARK
9 : S.C   03 EMIT ( --)  ;
10 : R.C   02 EMIT ( --)  ;
11 : >DISC ( --)  ;
12
13
14
15
```

26

Anwenderfaehig gemacht fuer F83 von Laxen & Perry
auf Commodore 128 mit CPM+ und Schneider CPC
sowie veraendert ergaenzt und erweitert.

Meine Ideen stehen allen zum Anwenden und zum Benutzen zur freien Verfuegung.
Aenderungen, Verbesserungen und Erweiterungen sind erwuenscht.

Kontaktadresse: Rudolf Mensing
Flottkamp 19
2358 Kaltenkirchen
Tel. 04191 / 6499

27

Die Kommentare belasse ich hier im Ladescreen,
da mit L <ret> der Screen # 1 auf dem Bildschirm kommt,
dessen Inhalt zur Orientierung oft genug ausreicht.

Mit ^W kann man zwischen Screen und Schadow-Screen hin und her
wechseln. Der CurSoR verbleibt beim wechseln in der gleichen
Zeile, er ist danach aber immer in der ersten Spalte zu finden

28

F83 Editor initialisieren
PRINTER Anpassung
Fehler in Scr# wird angezeigt, der F83 Editor wird aktiviert
DoppelEMIT, der 1. Wert wird zuerst emittiert.
AT oder GOTOXY -Funktion
Bildschirmloeschbefehl des Schneider CPC
F83 -Bildschirmloeschbefehl an Schneider angepasst
Set.Cursor Cursor sichtbar
Reset.Cursor Cursor nicht sichtbar
Funktionslos, wird nur gebraucht fuer COPY und -COPY

Page No 2 volksFORTH83 der FORTH-Gesellschaft eV EDITOR.BLK

3 29

```
0 \ CPC6128 CPM Plus                                  15aug87msg  CPC 6128       CPM Plus
1
2 EDITOR CPC-6128 FORTH CAPS ON                                   F83 Editor initialisieren
3 ' EPSON     IS  INIT-PR                                         PRINTER Anpassung
4 ' (WHERE)   IS  WHERE                                           Fehler in Scr# wird angezeigt
5 27 CONSTANT ESC                                                 Escape-Konstante
6 : DEMIT ( c1 c2 --)    SWAP EMIT EMIT ;                         DoppelEMIT
7 : XY    ( col row --)  ESC 89 DEMIT                             AT oder GOTOXY -Funktion
8         0 MAX 24 MIN 32 + EMIT       0 MAX 79 MIN 32 + EMIT ;
9 : CLS   ( --) ESC 69 DEMIT    ESC 72 DEMIT ;                    Bildschirmloeschbefehl des Schneider CPC
10 ' CLS IS DARK                                                  F83 -Bildschirmloeschbefehl an Schneider angepasst
11 : S.C  ( --) ESC ASCII e DEMIT  ;                              Set.Cursor      Cursor sichtbar
12 : R.C  ( --) ESC ASCII f DEMIT  ;                              Reset.Cursor    Cursor nicht sichtbar
13 : >DISC ( --) 255 48 BDOS DROP  ;                              Schreibt die CPM Plus Puffer auf Diskette
14 : FLUSH ( --) FLUSH >DISC  ;                                   CPM Plus taugliche Version
15 : DONE  ( --) DONE >DISC ." ready now"  ;                      CPM Plus taugliche Version mit zusaetzlicher Fertigmeldung
```

4 30

```
0 \ Commodore CPM+                                    15aug87msg  COMMODORE      CPM+
1
2 EDITOR TELEVIDEO FORTH CAPS ON                                  F83 Editor initialisieren
3 ' EPSON     IS  INIT-PR                                         PRINTER Anpassung
4 ' (WHERE)   IS  WHERE                                           Fehler in Scr# wird angezeigt
5 27 CONSTANT ESC                                                 Escape-Konstante
6 : DEMIT ( c1 c2 --) SWAP EMIT EMIT ;                            DoppelEMIT, der erstgenannte Wert wird zuerst emittiert
7 : XY    ( col row --)  ESC 61 DEMIT                             AT oder GOTOXY -Funktion
8         0 MAX 24 MIN 32 + EMIT       0 MAX 79 MIN 32 + EMIT ;
9 : CLS   ( --) 26 EMIT  ;
10 ' CLS IS DARK                                                  F83 -Bildschirmloeschbefehl an Commodore angepasst
11 : S.C  ( --) ( ??? )  ;                                        Set.Cursor      Code mir nicht bekannt
12 : R.C  ( --) ( ??? )  ;                                        Reset.Cursor    Code mir nicht bekannt
13 : >DISC ( --) 255 48 BDOS DROP  ;
14 : FLUSH ( --) FLUSH >DISC  ;
15 : DONE  ( --) DONE >DISC ." ready now"  ;
```

5 31

```
0 \ lyt editer   definiskoun weur  variablen          27jul87msg  Definitionsworte und Variablen
1
2 : ZUORDNE  ( u --)                                              Definitionswort  definiert ein Wort in dem Tastencodes
3         CONSTANT                                                eincompiliert sind, und welches dann bei Aufruf einen
4         DOES>         ( c -- idx )                              Indexwert liefert
5            DUP @
6            -ROT DUP @ 0
7            DO   2+ 2DUP @ =
8               IF 2DROP DROP   I 0 0   LEAVE   THEN
9            LOOP   2DROP ;
10 : TUE-FALL: ( u --)                                            Definitionswort  HIDE versteckt das zu definierende Wort
11         CONSTANT HIDE ]                                        ]    schaltet den Compiler ein
12         DOES> 2+ SWAP 2* + @ EXECUTE ;                         die Gegenstuecke REVEAL und [ befinden sich im Wort ;
13 VARIABLE HILF                                                  deshalb muessen die Definitionen mit ; abgeschlossen werden.
14 VARIABLE PRUEF
15 VARIABLE SCR-DIFF                                              Variablen-Deklaration
```

Page No 3 volksFORTH83 der FORTH-Gesellschaft eV EDITOR.BLK

6 32

```
0 \ lyt editer    wiest de editerbefeels              30jul87msg  Zeigt Editorbefehle
1
2 : EDITORBEFEHLE ( --)                                           zeigt Editor-Befehle an
3                   0 2 XY      ." COPY = Leerzeichen einfuegen   Diese Angaben gelten fuer den Schneider CPC ohne Umdefinierung
4 ^E    = Cursor rauf     CR  = Neue Zeile"                       der Tastencodes, daher ESC-Taste und ^[ mit gleicher Wirkung
5                   0 3 XY      ." ^S   = Cursor links            aber unterschiedlichen Tastencodes.
6 ^X    = Cursor runter   ^D  = Cursor rechts"                    Bei anderen Rechnern enfaellt die Unterscheidung bei den Codes,
7                   0 5 XY      ." CLR  = Zeichen loeschen        daher kann die Anzeige verkuerzt werden.
8 ^UP   = Cursor home     ^L  = Schirm loeschen"                  Eine Aenderung sollte leicht moeglich sein, wenn man den Text
9                   0 6 XY      ." ^LI  = Tab links               einfach ueberschreibt und den ueberflussigen Rest loescht.
10 ^DOWN = Cursor Scr end  ^RE = Tab rechts"
11                  0 8 XY      ." DEL  = Vorh.Zeichen loeschen
12 TAB   = horizontal Tab  ESC = Editor verlassen"
13                  0 9 XY      ." ^W   = Shadow-Screen
14 ^I    = horizontal Tab  ^[  = Editor verlassen"
15 15 13 XY ." Zurueck zum Screen mit beliebiger Taste"   CR  ;
```

7 33

```
0 \ lyt editer    HILF, PRUEF un bloc verskuven       30jul87msg  HILF PRUEF und im Block verschieben
1
2 : RESETPRUEF  ( --)  PRUEF OFF  ;                               Setzt d. Variable PRUEF auf den Wert 0
3 : SETPRUEF    ( --)  PRUEF ON   UPDATE   ;                      Setzt d. Variable PRUEF auf den Wert -1, d. Screen wird updated
4 : ?PRUEF    ( c --)  PRUEF @ 0= IF DUP DUP                      Prueft und setzt bei bedarf die Variable PRUEF, wenn der Screen
5                                 31 > SWAP 128 < *  SWAP DUP     geaendert worden ist.
6                                 16 = SWAP 224 = + + DUP
7                                 IF SETPRUEF THEN
8                               THEN DROP ;                       addr1 = absolute CSR-Posit.   bl = Code f. Leerzeichen
9 : BL-POS   ( -- 1 addr1 bl 1 addr1 addr2)                       addr2 = letzte Blockadresse    1 = Stringlaenge hinter d. CSR
10                R# @ DUP C/L / 1+ C/L * 1- OVER - SWAP           Berechnet einige Werte im Block
11                SCR @ BLOCK + 2DUP BL -ROT 2DUP +  ;
12 : LMOVE    ( 1 addr1 bl 1 addr1 addr2 -- 1 addr1)              String um eine Position
13                -ROT DUP 1+ SWAP ROT CMOVE C! ;                         im Block nach links verschieben, fuegt Leerzeichen an.
14 : RMOVE    ( 1 addr1 bl 1 addr1 -- 1 addr1)                    String um eine Position
15                SWAP OVER DUP 1+ ROT CMOVE> C! ;                        im Block nach rechts verschieben, fuegt Leerzeichen ein
```

8 34

```
0 \ lyt editer    bloc ferskuven  cursor bewegen      07jun87msg  Verschiebung im Block  Cursor Bewegungen
1
2 : ?RMOVE ( 1 addr1 bl 1 addr1 addr2 -- 1 addr1)                 Prueft ob in der Zeile nach rechts verschoben werden kann.
3         C@ BL = IF   RMOVE
4                 ELSE 7 EMIT 2DROP DROP
5                 THEN   ;
6 : BL-MOVE   ( -- 1 addr) BL-POS LMOVE   ;                       Verschiebt einen String im Block
7 : BL-MOVE>  ( -- 1 addr) BL-POS ?RMOVE  ;                       Verschiebt einen String im Block, wenn noch moeglich
8 : CSRPOS    ( --)        R# @ C/L /MOD 2 + SWAP 4 + SWAP XY ;   Positioniert den CSR auf den Bildschirm
9 : CSR!      ( posit --)  0 MAX 1024 /MOD DROP R# ! ;            speichert die CSR-Posit. in die Variable R#
10 : CSR+     ( n --)      R# @ + CSR! ;                          R# + TOS werden auf den Stack gelegt
11 : CSR+POS  ( n --)      CSR+ CSRPOS ;                          Der CSR wird relativ um n Verschoben, R# wird aktualisiert
12 : CSR!POS  ( u --)      CSR! CSRPOS ;                          Der CSR wird gesetzt, R# wird aktualisiert
13 : LF+CR    ( n --)      R# @ C/L / + C/L * CSR! CSRPOS ;       Setzt den CSR um n Zeilen tiefer an den Zeilenanfang
14 : LINKS-OBEN   ( --)    0 CSR!POS ;                            Der CSR wird in die erste Position gebracht
15 : RECHTS-UNTEN ( --)  1023 CSR!POS ;                           Der CSR wird in die letzte Position gebracht
```

Page No 4 volksFORTH83 der FORTH-Gesellschaft eV EDITOR.BLK

9 35

```
0 \ lyt editer    cursor bewegen  bloc ennern          30jul87msg  Cursor Bewegung  Block Aenderung
1
2 : RESTAURIERE   ( 1 addr --)    SWAP 1+ TYPE CSRPOS   ;          Bring den im Block verschobenen String auf den Bildschirm
3 : EINFUEGE      ( --)   BL-MOVE> RESTAURIERE   ;                 Fuegt ein Leerzeichen ein, im Block und auf dem Bildschirm
4 : SCR+          ( -- scr#)   SCR @ 1+  ;                         legt die um 1 inkrementierte Screennummer auf den Stack
5 : SCR-          ( -- scr#)   SCR @ 1-  ;                         legt die um 1 dekrementierte Screennummer auf den Stack
6 : CORR-SCR      ( -- scr#)   SCR @ SCR-DIFF @                    Berechnet den korrespondierenden Screen ( SCR -- Schadow SCR )
7                 2DUP < IF + ELSE - THEN  ;
8 : WAEHL-SCR     ( -- scr#)   38 1 XY ."   SCREEN: "              Eingabe fuer eine Screennummer
9                 QUERY 32 WORD NUMBER DROP  ;
10 : BLOCK!   ( c --) SCR @ BLOCK R# @ + C! 1 CSR+POS ;            Schreibt ein Zeichen in den Block
11 : LOESCHEN ( --)   LINKS-OBEN 1025 0 DO BL DUP EMIT BLOCK! LOOP  Loescht den Screeninhalt im Block und auf dem Bildschirm
12                    SETPRUEF LINKS-OBEN ;                        diese Funktion ist verbesserungsbedueftig, zu langsam
13 : TABULATOR  ( --) R# @ 8 / 8 * 8 + CSR!POS ;                   Setzt den CSR auf die naechste Tabulatorposition
14 : -TABULATOR ( --) R# @ 7 + 8 / 8 * 8 - CSR!POS ;               Setzt den CSR auf die vorhergehende Tabulatorposition
15
```

10 36

```
0 \ lyt editer    editer acskoun                      15aug87msg  Editor Aktionen
1
2 : ZEICHEN       ( c -- c) DUP 20 <                              Wenn es ein Zeichen ist, wird es dargestellt und in den Block
3                 IF 7 EMIT ELSE DUP DUP EMIT BLOCK! THEN ;          geschrieben
4 : SAVE?         ( --) PRUEF @ IF FLUSH THEN  ;                  Wenn PRUEF gesetzt, werden die Bloecke auf Diskette geschrieben
5 : ENDE          ( --) CLS SAVE? SCR @ . PRUEF @ NOT             Ausstieg aus dem Editor, bei Bedarf Datensicherung
6                 IF ." un" THEN ." modified READY" DROP QUIT  ;
7 : CSR-RUNTER    ( --) C/L CSR+POS ;                             Setzt den CSR eine Zeile tiefer
8 : CSR-LINKS     ( --) -1 CSR+POS ;                              Setzt den CSR um eine Position nach links
9 : CSR-HOCH      ( --) C/L NEGATE CSR+POS ;                      Setzt den CSR eine Zeile hoeher
10 : CSR-RECHTS   ( --) 1 CSR+POS ;                               Setzt den CSR um eine Position nach rechts
11 : NAECHST-ZEIL ( --) 1 LF+CR  ;                                Setzt den CSR eine Zeile tiefer an den Zeilenanfang
12 : REIN         ( --) BL-MOVE RESTAURIERE  ;                    Loescht das Zeichen unter dem CSR
13 : RADIERE      ( --) CSR-LINKS REIN ;                          Loescht das Zeichen links vom CSR
14 : SCR-DIFF!    ( --) CAPACITY 2 / SCR-DIFF !  ;                Berechnet die Differenz zwischen Screen uns Shadow-Screen
15
```

11 37

```
0 \ lyt editer    skerm foerwisen hoelp wesseln       15aug87msg  Screen zeigen  Hilfe Wechsel
1
2 : CSR>ZA      ( --)  R# @ C/L / C/L * HILF ! ;                  CSR zum Zeilenanfang Position wird zwischengespeichert
3 : HILFSZEILE  ( --)  38 1 XY ." ^H = Hilfe "  ;                 Bringt den Vermerk zur Hilfe auf den Bildschirm
4 : ZEIG-SCR  ( scr# --)                                          Bringt den den Screen zum edieren auf den Bildschirm
5             RESETPRUEF CLS LIST LINKS-OBEN HILFSZEILE CSRPOS ;
6 : ZURUECK   ( --) SAVE? SCR- ZEIG-SCR   ;                       Bringt den vorhergehenden Screen    ;       zum edieren
7 : VOR       ( --) SAVE? SCR+ ZEIG-SCR   ;                       Bringt den nachfolgenden Screen     ;          auf
8 : WAHL      ( --) SAVE? WAEHL-SCR  ZEIG-SCR  ;                  Bringt den gewaehlten Screen        ;     den Bildschirm
9 : SCR-WCHL  ( --) CSR>ZA CORR-SCR ZEIG-SCR HILF @ CSR!POS  ;     Wechselt zwischen Screen und Shadow-Screen
10 : HELF     ( --) R# @ HILF ! CLS EDITORBEFEHLE R.C KEY 2DROP   Bringt die Editor-Befehle auf den Bildschirm und speichert
11            S.C CLS SCR @ LIST HILFSZEILE HILF @ CSRPOS  ;      die CSR-Position fuer die Wiederdarstellung des Screens
12
13
14
15
```

```
Page No  5            volksFORTH83 der FORTH-Gesellschaft eV     EDITOR.BLK

                  12                                                  38

0 \ lyt editer    behanneln fun ynnerskeidlik ingoven   30jul87msg  Behandlung unterschiedlicher Eingaben
1
2 26 TUE-FALL:   FALL  ( idx -- addr)
3
4 ( 0 )  CSR-HOCH      ( 1 ) CSR-RUNTER     ( 2 ) CSR-RECHTS       Die Adressen dieser Woerter werden hier eincompilert,
5 ( 3 )  CSR-LINKS     ( 4 ) LOESCHEN       ( 5 ) NAECHST-ZEIL     mit WLIST FALL gut nachpruefbar
6 ( 6 )  ENDE          ( 7 ) TABULATOR      ( 8 ) RADIERE          Das Wort FALL bringt diese eincompilierten Worte zur
7 ( 9 )  CSR-HOCH      ( 10 ) CSR-RUNTER    ( 11 ) CSR-LINKS       Ausfuehrung
8 ( 12 ) CSR-RECHTS    ( 13 ) REIN          ( 14 ) ENDE
9 ( 15 ) EINFUEGE      ( 16 ) ZURUECK       ( 17 ) VOR
10 ( 18 ) WAHL         ( 19 ) TABULATOR     ( 20 ) -TABULATOR      Beim Commodore 128 kann hier gekuerzt werden
11 ( 21 ) LINKS-OBEN   ( 22 ) RECHTS-UNTEN  ( 23 ) HELF
12 ( 24 ) SCR-WCHL     ( 25 ) ZEICHEN
13 ;                \ Abschluss mit Semikolon
14
15

                  13                                                  39

 0 \ lyt editer    tasten touwies                      27jul87msg  Tasten Zuordnungen
 1
 2 25 ZUORDNE  TASTE    ( c -- idx)
 3
 4 ( 0 ^E )     05 ,    ( 1 ^X )     24 ,    ( 2 ^D )    04 ,      Die Tastencodes werden hier eingetragen mit ,
 5 ( 3 ^S )     19 ,    ( 4 ^L )     12 ,    ( 5 CR )    13 ,      das Wort TASTE liefert eine Inderzahl fuer das Wort FALL
 6 ( 6 ^I )     27 ,    ( 7 ^1 )     09 ,    ( 8 DEL )  127 ,
 7 ( 9 OP )    240 ,    ( 10 DO )    241 ,   ( 11 LI )  242 ,
 8 ( 12 RE )   243 ,    ( 13 CLR )    16 ,   ( 14 ESC ) 252 ,
 9 ( 15 COPY ) 224 ,    ( 16 ^B )     02 ,   ( 17 ^N )   14 ,
10 ( 18 ^J )    10 ,    ( 19 ^RE )   251 ,   ( 20 ^LI ) 250 ,     Beim Commodore 128 kann hier gekuerzt werden
11 ( 21 ^OP )  248 ,    ( 22 ^DO )   249 ,   ( 23 ^H )   08 ,     ^C und ESC liefern z.B. beide den Code 27
12 ( 24 ^W )    23 ;                                              nicht jedoch beim CPC, hier liefert die ESC-Taste
13                                                                wenn sie nicht undefiniert wird den Code 252
14
15

                  14                                                  40

 0 \ lyt editer    boverst flak                        30jul87msg  Oberstes Ende , Letzter Screen des eigentlichen Editors
 1
 2 : LADE          ( --) SCR @ LOAD  ;                             Laedt den zuletzt gerufenen Screen
 3 : EDIER-SCR     ( --) BEGIN                                     Eingabeschleife
 4                       KEY DUP TASTE FALL ?PRUEF
 5                     AGAIN ;
 6 : EDIER         ( scr# --) SCR-DIFF! ZEIG-SCR EDIER-SCR ;       Ruft den Editor auf
 7 : SCR#?         ( -- 1 | scr# -- scr#) DEPTH 0= IF 1 THEN ;     Legt eine 1 auf den Stack, wenn er leer ist
 8 : L             ( scr# -- | --) SCR#? EDIER ;                   Screen 1 , wenn keine Screennummer angegeben wird
 9 CLS                                                             Loescht den Bildschirm fuer die nachfolgenden Bemerkungen,
10 .(              zum editieren schreibe: n EDIER <return> ) CR   die anschliessend auf dem Bildschirm erscheinen
11 .(                                  oder (n) L <return> ) CR CR
12
13
14
15
```

Page No 6 volksFORTH83 der FORTH-Gesellschaft eV EDITOR.BLK

22

```
0 \ -copy   copy neidefiniskoun                        29mai87msg
1
2 : -COPY  ( 1.scr# letzscr# zielscr# --) DEPTH 3 =
3    IF   ROT 2DUP <
4          IF    >R >R 1 SWAP 1+ R> SWAP R>
5          ELSE  ROT 2DUP SWAP - -ROT >R >R + -1 SWAP R> R>
6          THEN
7          DO   2DUP SWAP + SWAP I SWAP COPY >DISC OVER   +LOOP
8          2DROP
9    ELSE ." nicht exakt 3 Parameter " ABORT
10   THEN ;
11 : COPY  ( quellscr# zielscr# --) DEPTH 2 =
12   IF    COPY >DISC
13   ELSE  ." nicht exakt 2 Parameter " ABORT
14   THEN  ;
15
```

48

-COPY COPY Neudefinition

copiert,auch ueberlappend abwaerts und aufwaerts, es muessen exakt 3 Parameter auf dem STACK liegen. Diese Version kopiert sicher auf die Diskette auch mit CPM Plus, die leider beim F83 bei CONVEY nicht gewaehrleistet ist. Exakte Parameteranzahl vermeidet Verwechslung mit COPY
>DISC kann bei CPM 2.2 auch gestrichen werden

exakt 2 Parameter muessen auf dem Stack liegen
>DISC wird bei CPM 2.2 nicht benoetigt
10 20 15 COPY koennte fatal sein, wenn 10 20 15 -COPY gemeint war, denn der urspruengliche Screen 15 waere dann zerstoert
Voll CPM Plus tauglich, wie -COPY. Siehe auch dort

23

```
0 \ WDUMP WLIST                                        10mai87msg
1 \ : .ID ( addr --)  .NAME  ;
2
3 : PRT ( u --) BASE @ HEX SWAP 5 U.R BASE ! ;     HEX
4 : WDUMP ( addr --) CR
5    BEGIN DUP PRT ." :" DUP 2+ SWAP @ DUP
6      PRT DUP >NAME DUP HERE
7      U< IF DUP DUP C@ 1F AND + 1+ ROT
8          = IF 2 SPACES .ID
9            ELSE DROP
10           THEN
11         ELSE 2DROP
12         THEN
13     CR KEY? UNTIL DROP ;                        DECIMAL
14
15 : WLIST ( --)  ' WDUMP ;
```

49

PRT WDUMP WLIST (.ID = .NAME)
wenn geladen, WLIST fuer Volks-FORTH geeignet

rechtsbuendige Anzeige in HEX
WordDUMP zeigt Adressen an und zugehoerige Namen
Aufruf: ' <name> WDUMP oder ' <name> 1+ WDUMP

WDUMP und WLIST zeigen an :
Speicheradresse Speicherinhalt (Name des Forthwortes)
der Name des Forthwortes wird nur angezeigt wenn vorhanden
die Werte werden als 16 bit - Werte angegeben
ein Wortende wird nicht erkannt
Sprungweiten kann man erkennen, entweder die relative Sprungweite oder die Sprungadresse, je nach Forth-Version

WordLIST Aufruf: WLIST <name>

24

```
0 \ Printerstyren                                      13jun87msg
1
2 : PRINTER  ['] (PRINT) IS EMIT ;
3 : CONSOLE  ['] (EMIT) IS EMIT ;
4 : NLQ      ESC ASCII x DEMIT 1 EMIT ;
5 : DRAFT    ESC ASCII x DEMIT 0 EMIT ;
6 : DEUTSCH  ESC ASCII 6 DEMIT ;  : AMERIKA  ESC ASCII 7 DEMIT ;
7 : ENG      15 EMIT ;            : -ENG     18 EMIT ;
8 : HERVOR   ESC ASCII E DEMIT ;  : -HERVOR  ESC ASCII F DEMIT ;
9 : DOPPEL   ESC ASCII G DEMIT ;  : -DOPPEL  ESC ASCII H DEMIT ;
10    DEFER EIN                      DEFER AUS
11 : (EIN  AMERIKA ENG ;          : (AUS  DEUTSCH -ENG ;
12 : ((EIN AMERIKA DOPPEL  ;      : ((AUS DEUTSCH -DOPPEL -ENG ;
13 : ()EIN AMERIKA HERVOR  ;      : ()AUS DEUTSCH -HERVOR  ;
14
15  ' (EIN IS EIN                  ' (AUS IS AUS
```

50

Druckeransteuerung

Ausgabe zum Printer
Ausgabe zum Bildschirm
umschalten auf amerikanischen Zeichensatz
umschalten auf deutschen Zeichensatz

Engschrift ein Engschrift aus
" —— + amerikanischen " —— + deutschen
Zeichensatz

```
Page No 7            volksFORTH83 der FORTH-Gesellschaft eV     EDITOR.BLK

                    25                                                  51

 0 \ PLIST   PRINT DISPLAY   2LIST+P                  13jun87msg  PLIST   PRINT DISPLAY   2LIST+P
 1
 2 : PLIST    PRINTER EIN LIST AUS CONSOLE ;                     listet einen Screen zum Printer auf
 3
 4 : 2PLIST   DUP 1+ SWAP PRINTER EIN LIST CR CR LIST AUS CONSOLE ;  listet zwei Screens zum Printer auf
 5
 6
 7 : PRINT 1 PRINTING ! ;          : DISPLAY 0 PRINTING ! ;        set oder reset der Variablen printing
 8
 9 : 2LIST+P   DUP 1+ SWAP PRINT EIN LIST CR CR LIST AUS DISPLAY ;  listet zwei Screens entweder nur zum Bildschirm
10                                                                      oder zum Bildschirm und Printer auf
11
12
13
14
15

                         o                                                   o

 0 \ EINLEITUNG                                Rudolf Mensing \ EINLEITUNG                                Rudolf Mensing
 1
 2 Dieser Editor basiert auf den kleinen Editor veroeffentlicht in  Dieser Editor basiert auf den kleinen Editor veroeffentlicht in
 3              andreas goppold / roger bouteiller                                  andreas goppold / roger bouteiller
 4          forth :  ein programmiersystem ohne grenzen                         forth :  ein programmiersystem ohne grenzen
 5           von A. Goppold nach FORTH DIM II/3 ,  p83                           von A. Goppold nach FORTH DIM II/3 ,  p83
 6
 7
 8 Dieser Editor kommt mit wenigen Befehlen des Betriebssystem aus. Dieser Editor kommt mit wenigen Befehlen des Betriebssystem aus.
 9
10 Notwendig sind :           Bildschirmloeschen                    Notwendig sind :           Bildschirmloeschen
11                  Cursor an jede Bildschirmposition bringen                     Cursor an jede Bildschirmposition bringen
12
13
14
15

                         o                                                   o

 0 \ EINLEITUNG                                Rudolf Mensing \ EINLEITUNG                                Rudolf Mensing
 1
 2 Dieser Editor basiert auf den kleinen Editor veroeffentlicht in  Dieser Editor basiert auf den kleinen Editor veroeffentlicht in
 3              andreas goppold / roger bouteiller                                  andreas goppold / roger bouteiller
 4          forth :  ein programmiersystem ohne grenzen                         forth :  ein programmiersystem ohne grenzen
 5           von A. Goppold nach FORTH DIM II/3 ,  p83                           von A. Goppold nach FORTH DIM II/3 ,  p83
 6
 7
 8 Dieser Editor kommt mit wenigen Befehlen des Betriebssystem aus. Dieser Editor kommt mit wenigen Befehlen des Betriebssystem aus.
 9
10 Notwendig sind :           Bildschirmloeschen                    Notwendig sind :           Bildschirmloeschen
11                  Cursor an jede Bildschirmoosition bringen                     Cursor an jede Bildschirmposition bringen
12
13
14
15
```

"Frei programmierbare Funktionstasten", U. Hoffmann, Kiel, August 1987

In diesem Artikel wird eine einfache Methode vorgestellt, um in Forth-Systemen, die Blöcke als Massenspeicher unterstützen, frei programmierbare Funktionstasten einzubinden. Eine Implementation für volksFORTH wird angegeben.

Stichworte: Funktionstaste, Queue, Screen, Block, Key

Es sind schon sehr schöne Artikel über Queues in der Vierten Dimension erschienen ([1], [2]).
Hier soll nun eine Beispielanwendung vorgestellt werden:

Frei programmierbare Funktionstasten.

Was soll geschehen?

Die meisten Computer haben auf der Tastatur neben dem normalen Schreibmaschinenfeld noch zusätzlich Funktionstasten, die meist einen bisher unbenutzten Funktionscode liefern. (Etwa einen >$80).
Wünschenswert wäre es nun, könnte man einzelne Phrasen auf diese Tasten legen, und sie mit einem Tastendruck wieder hervorrufen.

Wie soll das geschehen?

Tastatureingaben laufen in Forth über das Wort KEY, das in modernen Systemen vektorisiert ist, und so durch eine eigene Routine ersetzt werden kann. Hier soll KEY so verändert werden, daß es zusätzlich auch die Funktionstasten bearbeiten kann.
Die Zeichen, die das neue KEY liefert, kommen aus einer Queue.
Ist diese leer, wenn KEY aufgerufen wird, so soll die Tastatur abgefragt werden (altes KEY).
Ist dabei eine Funktionstaste gedrückt, so wird der entsprechende Text in die Queue gesteckt, ansonsten wird direkt der Wert der gedrückten Taste von KEY abgeliefert.
Ist beim Aufruf des neuen KEYs die Queue nicht leer, so wird das vorderste Zeichen aus der Queue genommen und von KEY abgeliefert. Die Tastatur wird dann nicht angesehen.

Wo liegen die Texte der Funktionstasten?

Die zu den Funktionstasten gehörigen Texte sollen einfach programmiert werden können. Ein Werkzeug, das Texte gut bearbeiten kann, ist in FORTH aber schon vorhanden: der Editor.
Die Texte sollen daher ab Screen 0 des speziellen Files KEYFILE liegen, jeweils eine Zeile pro Funktionstaste. Dort können sie auch bequem mit dem Editor geändert werden. Damit alle Zeichen auf Funktionstasten darstellbar sind, werden die Texte in Trennzeichen eingefasst. Das Zeichen in Spalte null ist das Trennzeichen. Der Text enthält alle Zeichen von Spalte eins bis zum nächsten Auftreten des Trennzeichens (ausschließlich dieses Zeichens).

Und das Programm?

Das kann jetzt aufgestellt werden. Die angegebene Version ist für volksFORTH geschrieben und benutzt dessen spezielle Möglichkeit der Eingabeumleitung mittels INPUT:.

[1] Bernd Pennemann, Queues in Forth,
Forth Magazin 'Vierte Dimension' II/3 (1986), S.27-30

[2] Klaus Schleisiek, Das Wort des Monats #3,
Forth Magazin 'Vierte Dimension' III/1 (1987), S.15

Page No 1 volksFORTH83 der FORTH-Gesellschaft eV FUNCKEY.SCR

1

```
0 \ Functionkeys LOAD-Screen                          UH 24Aug87
1
2 1 +load     \ Klaus' Queue primitives
3
4 2 3 +thru   \ Functionkeys
5
6 keyfile Assign funckey.scr
7
8 install   forget install
9
A fkeyboard save
B
C
D
E
F
```

0

```
0 \ Functionkeys                                      UH 26Aug87
"Hier steht der Text fuer Funktionstaste 1"
'Auch Anfuehrungszeichen sind moeglich "'
"Das Zeichen der ersten Spalte ist das Symbol"
#Welches auch das Ende des Textes markiert. #
!So koennen alle Zeichen im Text erscheinen.!
!            (auch Leerzeichen)           !
=Hier der Text fuer Taste #7 =
"und der von Taste acht."

 Dieses File enthaelt Defintionen, die auf einem Forth System
 mit block-orientiertem Massenspeicher die Einbindung von frei
 programmierbaren Funktionstasten ermoeglicht.
```

2

```
0 \ Klaus' Queue primitves  VD III/1                  UH 24Aug87
1
2 : more? ( addr -- n )  c@ ;
3
4 : ?string ( n -- n )  dup &255 u> abort" overflow" ;
5
6 : append ( char addr -- )
7    dup c@ 1+ ?string  over c!  dup c@ +  c! ;
8
9 : detract ( addr -- char )
A    dup c@ 1- dup 0> and  over c!
B    count >r  dup count -rot  swap r> cmove ;
C
D
E
F
```

7

```
0 \ Klaus' Queue primitves  VD III/1                  UH 24Aug87
Liefert die Anzahl der Zeichen, die in der Queue <addr> sind.

Testet, ob die Laenge fuer "counted Strings" zulaessig ist.

Haengt das Zeichen <char> hinten an den String <addr> an.

Nimmt das vorderste Zeichen <char> aus dem String <addr>.
```

3

```
0 \ Functionkeys                                      UH 26Aug87
1
2 : 8 Constant #fkeys
3
4 : Create fkeys  #fkeys 2* allot   fkeys #fkeys 2* erase
5
6 : fkey# ( key -- # )  fkeys  #fkeys 0 DO  2dup @ =
7      IF 2drop I 1+ endloop exit THEN 2+ LOOP 2drop false ;
8
9 : delimited ( addr maxlen -- addr' len )
A      swap count >r   dup  rot  r> scan  drop over - ;
B
C File keyfile
D
E : ftext ( fkey# -- addr )  isfile push   keyfile
F      l/s /mod block  swap c/l *  + ;
```

8

```
0 \ Functionkeys                                      UH 26Aug87
So viele Funktionstasten werden unterstuetzt

Diese Tabelle haelt die Werte der einzelnen Funktionstasten

Liefert die Nummer <#> der Funktionstaste mit dem Wert <key>.
 <#> ist FALSE, falls <key> nicht Wert einer Funktionstaste ist.

<addr> zeigt auf einen String der von einem deliater einge-
  schlossen ist und hoechstens die Laenge <maxlen> hat.
  liefert den "netto" String ohne delimiter.
Das File aus dem die Funktionstastentexte genommen werden sollen

Berechnet zu der Nummer einer Funktionstaste die Anfangsadresse
  des entsprechenden Textes. Er liegt im keyfile ab Screen 0.
```

```
   Page No  2              volksFORTH83 der FORTH-Gesellschaft eV      FUNCKEY.SCR

                    4                                                  9

0 \ keybuffer fkey fkey?                          UH 27Aug87 \ keybuffer fkey fkey?                          UH 27Aug87
1
2 : Create keybuffer  0 c,  c/l allot                        Die Tastatur - Queue, die die Funktionstastentexte aufniamt
3
4 : : fkey ( -- c )                                          Das neue key, das auch die Behandlung von Funktionstasten
5      BEGIN pause keybuffer more? IF keybuffer detract exit THEN   enthaelt.
6      (key  dup   fkey# ?dup  0= ?exit  nip
7      ftext  c/l 1- delimited  keybuffer place  REPEAT ;
8
9 : : fkey? ( -- f )  pause  keybuffer more? ?dup ?exit (key? ;   Auch key? muss den neuen Anforderungen gerecht werden.
A
B Input: fkeyboard     fkey fkey? (decode (expect ;              Der neue Input-Vektor.
C
D : install ( -- )   fkeys  #fkeys 0 DO                      Ermittelt die Werte der Funktionstasten und traegt sie in die
E    cr ." Bitte Funktionstaste " 1 1+ . ." druecken:"          Tabelle ein.
F    key over !   2+  LOOP  drop ;
```

Kleinkram laden, Antwort

Finn Berlev, Farun, Dänemark

Sie suchten einen Weg, den "Kleinkram" des Forth zu laden (1). Ich habe darüber nachgedacht und probiere gerade aus, solch eine Sammlung von Forth-Definitionen als File zusammenzustellen - denn ich denke, das kann sehr nützlich sein. Ich entdeckte, das die Worte aus "A Forth Dictionary" (2) abgewandelt werden können, um diese Aufgabe zu erfüllen. Dabei verwende ich das NEEDS des Volksforth zusammen mit meinem LOADWORD. Falls ihnen meine Lösung nützlich erscheint, machen sie freien Gebrauch davon.

(1) M.Kalus, Kleinkram laden, Forth Magazin, 1/87, S.34
(2) F.Berlev, A Forth Dictionary, Forth Magazin, 2/87, S.35-37

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

0

```
0                Kleinkram laden                      FB 1/8 87
1        In VD87 III/1 you ask for a word to load Kleinkram.
2 Thinking on this, i realized that the words used in " A FORTH
3 dictionary"  III/2 could be modified to do the job.
4
5 Loadfile <word>    should load <word>  and if necessary a few
6 other words. For the scheme to work :
7 1. Collect the definitions in a file ( extras.scr ).
8 2. The word - in capital letters to speed up the search -
9    should be found in the first line to be loaded.
A 3. An exit or \\ should interrupt the loading.
B 4. (load  ( block offset -- )  and search
C    should be available - as they are in VolksFORTH rev 3.80.
D
E
F
```

1

```
0 \ Loadscreen for loadword
1
2 : (explain    \ unchanged from " A FORTH dictionary "
3       swap count rot block b/blk search ;
4 : loadword    \ loadword <word>
5       scr push caps push extras.scr caps off
6       bl word capitalize capacity 2
7       DO  dup i (explain
8          IF  dup c/l mod - i swap (load leave
9          ELSE  drop
A          THEN
B       LOOP  drop ;
C
D
E
F
```

2

```
0 \ cs ltype leap  pi
1
2 ' clearstack  alias  CS                                  \\
3
4 : LTYPE  ( ladr n -- )  \ like type for longadr
5       0 ?do 2dup lc@ emit 1. d+ loop 2drop ;             \\
6
7 : LEAP   rdrop rdrop rdrop rdrop ;
8  \\  as leave, but jumps out of the word. From Thinking Forth
9
A &355 &113  2constant PI                                  \\
B
C
D
E
F
```

3

```
\ random choose  vonbis vonbisan

variable RANDOM  here random !     \ CHOOSE
: choose  ( ul -- u2 )
     random @ &31421 * &6927 + dup random ! um* nip ;
 \\ u2 a random number,  0 <= u2 < ul. From Starting Forth

: VONBIS  ( n von bis -- f )   >r over > swap r> > or not ;
 \\ true if  von <= n <= bis

!needs vonbis  loadword vonbis     \ VONBISAN
: vonbisan ( n von bis -- f )  1- vonbis ;
 \\ true if  von <= n < bis
```

4

```
\\ Some examples

loadword cs        loads cs (laziness)

loadword choose    loads random and choose
loadword random    also loads random and choose

loadword vonbis    loads vonbis
loadword vonbisan  loads if necessary vonbis and then vonbisan
```

5

Maybe this can be of some help to you. If so you can use it freely. I have already started to build such a file, which may be very helpfull.

Med venlig hilsen

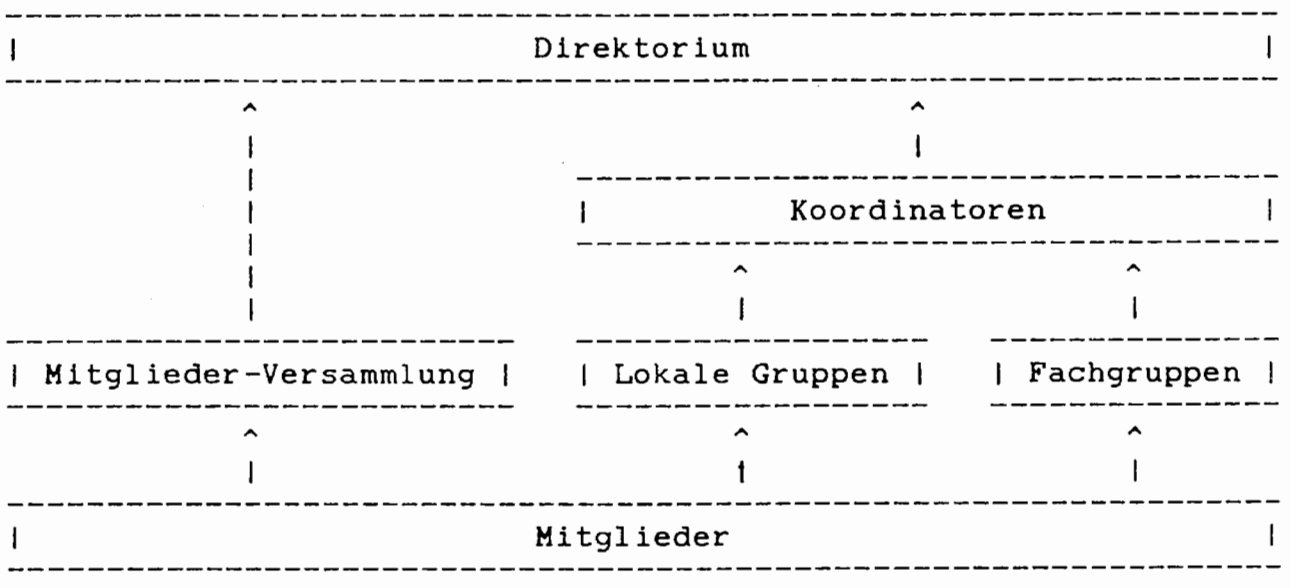
Finn Berlev
Lillevangsvej 92, 3520 Farum DK

Finn Berlev

Die Forth Gesellschaft eV - ein Überblick

Struktur

Die Forth Gesellschaft hat ihren Verwaltungssitz in Form des Forth Büros in Hamburg. Von hier aus werden alle bundesweiten Aktivitäten koordiniert, die VIERTE DIMENSION versandt und allgemeine Anfragen bearbeitet. Hier trifft sich auch das Direktorium, um die auf der jährlichen Mitgliederversammlung getroffenen Beschlüsse durchzuführen und über sonstige aktuelle Ergeignisse zu beraten. Die "wahre" Arbeit der Forth Gesellschaft aber findet in den Lokalen Gruppen und Fachgruppen statt. Diese haben jeweils einen Gruppenkoordinator, der die Aktivitäten verantwortlich leitet. Die Realisation der Lokalen Gruppen ermöglicht es Kontaktsuchenden, in ihrem jeweiligen Wohnbezirk Ansprechpartner zu finden.



Geschichte

Anfang 1984 wurde die Forth Gesellschaft (BGB) in Norddeutschland von 24 Forthlern gegründet. Ende 1984 wurde aus ihr ein e.V., dem schließlich Ende 1985 die Gemeinnützigkeit zuerkannt wurde. Zum Jahresende 1986 zählte die Forth Gesellschaft eV ca. 200 Mitglieder in ganz Europa.

Ziele

Das Hauptziel der Forth Gesellschaft ist es, die Verbreitung der Programmiersprache Forth und der mit ihr verknüpften Prinzipien des Programmierens zu fördern. Dieses erfolgt durch Lehrveranstaltungen, Informationstreffen der Mitglieder, Herausgabe von Veröffentlichungen und gemeinsame Arbeit an Projekten, die dem Vereinszweck dienen und dem Grundsatz der Gemeinnützigkeit entsprechen. Das Wirken des Vereins ist nicht auf einen wirtschaftlichen Geschäftsbetrieb gerichtet.

Die Forth Gesellschaft eV unterstützt und koordiniert Aktivitäten, die in der Verantwortung von einzelnen oder von Arbeitsgruppen liegen. Im Folgenden ein kurzer Überblick über die wichtigsten gegenwärtigen Aktivitäten. Details erfahren Sie bei den jeweils angegebenen Adressen. Initiativen für weitere Aktivitäten sind erwünscht!

VIERTE DIMENSION Forth Magazin
Die einzige deutschsprachige Forth Publikation für Fachartikel, Anzeigen und Kontaktgesuche. Die Qualität der Fachartikel wird durch das Forth Review Board gesichert.
Redaktion: Michael Kalus
Präsidentenstr.40, 5830 Schwelm, Tel.: 02336-82204

volksFORTH83 Arbeitsgemeinschaft
Forth für C64, Atari ST und Schneider CP/M (MSDOS in Vorb.).
Kontakt: Bernd Pennemann
Steilshooper Str.46, 2000 Hamburg 60, Tel.: 040-690O539

ForthTREE
Das Informations- und Kommunikationssystem der FG (Mailbox).
Zu erreichen unter Tel.: 040-3904204 (300 + 1200bd).
Weitere lokale Systeme in Planung.
SysOp: Marco Pauck
Friedensallee 92, 2000 Hamburg 50, Tel.: 040-3900139

Hotline im Forth Büro
Zentrale Ansprechstelle für allgemeine Informationen. Kann bei Bedarf an die entsprechenden Experten weitervermitteln.
Dienstags von 18-20 Uhr unter der ForthTREE Telefonnummer.

Bibliothek
Eine im Forth Büro befindliche Bibliothek zum Thema Forth, die allen Mitgliedern zur Verfügung steht.

Kopierservice
Preiswerte Möglichkeit für alle Mitglieder, um Artikel und Programme aus der Forth-Bibliothek zu erhalten.
Kontakt: Thomas Prinz
Adalbert-Stifter-Str.2, 6930 Eberbach a/N, Tel.: 06271-2830

Jahrestagung
Bundesweit organisiertes Treffen von Forthlern mit anschließender Jahreshauptversammlung der Forth Gesellschaft eV.

euroFORML
Regelmäßige Veranstaltung der europäischen Forth Modification Laboratory Konferenz mit Forthexperten aus Europa und den USA.
Kontakt: Klaus Schleisiek
Steinberg 8a, 2000 Wedel, Tel.: 04103-13255

(Stand: Dezember 1986)

FORTH GESELLSCHAFT e.V.
Antilopenstieg 6a
D 2000 HAMBURG 54

Antrag auf Mitgliedschaft in der Forth Gesellschaft eV

Name: _____

Strasse: _____

Ort: _____

Tel.: _____

Freiwillige Angaben für die interne Mitgliederliste:
(um Kontakte zu knüpfen und den Erfahrungsaustausch zu fördern)

Alter: _____ Beruf: _____ Forth: _____

Computer: _____ Betriebssystem: _____

sonstiges: _____

jährlicher Beitrag:

[] DM 32,- (Studenten + Arbeitslose, nur mit Ausweis)
[] DM 64,- (ordentliches Mitglied / Auslandsadresse)
[] DM 128,- (förderndes Mitglied / Firmen + Institutionen)

Ausserdem unterstütze ich die Forth Gesellschaft eV mit einer Spende von DM _____

[] Ich möchte eine Gebietsgruppe (Lokale Gruppe) gründen. Daher bin ich mit einer Veröffentlichung meiner Adresse in der VIERTEn DIMENSION einverstanden.

[] Ich möchte eine Fachgruppe gründen. Daher bin ich mit einer Veröffentlichung meiner Adresse in der VIERTEn DIMENSION einverstanden. Das Thema der Fachgruppe soll lauten:

_____

[] Ich habe außerdem noch folgende Anregungen / Kritiken:

_____

Den Gesamtbetrag von DM _____ habe ich

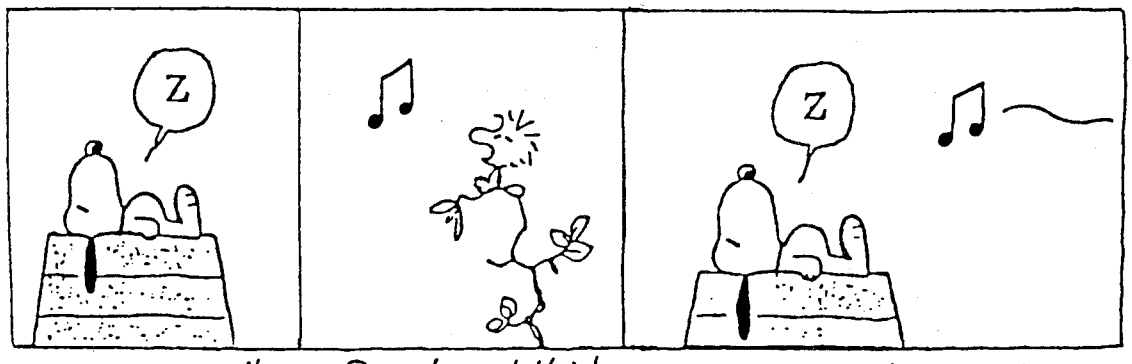
[] am _____ auf Ihr Konto überwiesen
[] als Verrechnungscheck beigelegt

Datum: _____ Unterschrift: _____

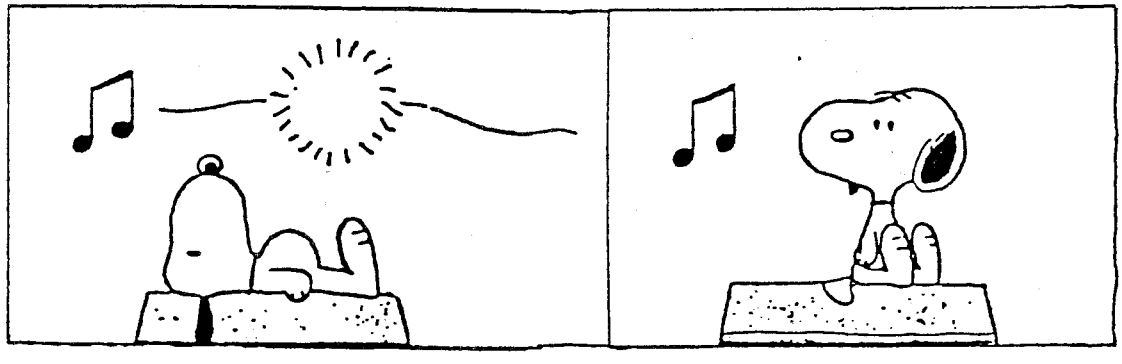
FORTH GESELLSCHAFT e.V. - Antilopenstieg 6a - D 2000 HAMBURG 54

Postgiroamt Hmb., Forth Gesellschaft, Konto: 56 32 11 - 208
BLZ: 200 100 20

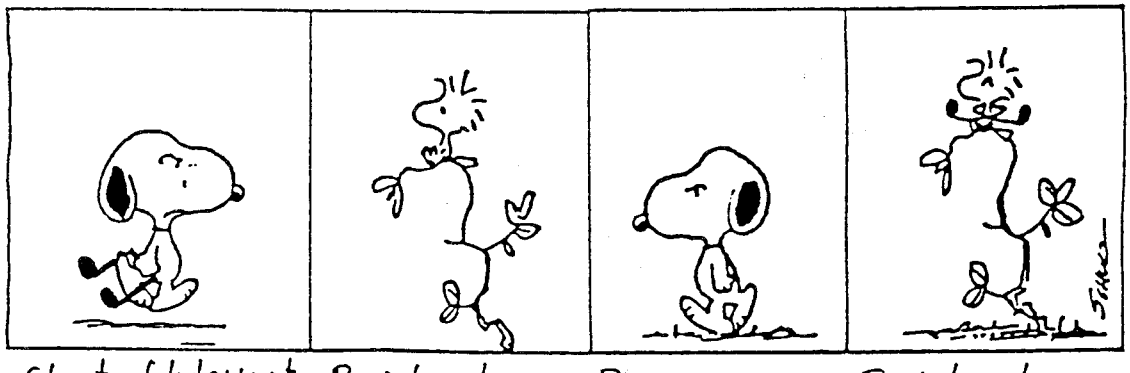
SIMPLE INTERRUPTSYSTEM



Processor idling. Peripheral thinks, he needs service. Interrupt pending.



INTERRUPT ! Restart, INTERRUPT.
Processor leaves idle state.



Start of Interrupt service routine. Peripheral anxiously awaiting service. Processor returning from Interrupt service. Peripheral knowing it's interrupt was acknowledged.

AKTIVE FORTH GRUPPEN

6100 Darmstadt: Andreas Soeder, 06257-2744
Treffen an der VHS an einem Mittwoch in der Mitte des Monats.
2000 Hamburg: Klaus Schleisiek, 040-4905195
Treffen jeden vierten Samstag im Monat ab 16:00Uhr in der Berufsfachschule für Radio- und Fernsehtechnik, Eimsbüttelerstr.64-66.
8000 München: Heinz Schnitter, 089-3103385
und Christoph Krinninger 089-7259382
Treffen jeden vierten Mittwoch im Monat 19:30Uhr im Vereinsraum 1 im Bürgerhaus Unterschleißheim am Rathausplatz (S-Bahnhaltepunkt S1 Unterschleißheim)
5600 Wuppertal: Michael Kalus, 02336-82204
Treffen jeden vierten Freitag im Monat ab 20:00Uhr im Bahnhof Ottenbruch, Funckstrasse, W'tal-Elberfeld.

Hier könnten aktive Gruppen entstehen:
Berlin: Bernd Pennemann, 030-7923923
Düsseldorf: Ralf Hohmann, 0211-289929
Rhein/Neckar Raum: Thomas Prinz, 06271-2830

INTERESSEN

Volksforth und Ultraforth: Bernd Pennemann, 030-7923923
Graphik und Animation: Marco Pauck, 040-3900139
32Bit Systeme: Robert Jones, 02434-4579
Forthchips, -maschinen und RISC: Roland Steck, 06151-661192
Künstliche Intelligenz: Ulrich Hoffmann, 04307-6869
Globale Daten Netzwerke: Klaus Schleisiek, 04103-13255
Realtime, relationale Netze: Wigand Gawenda, 040-446941

DAS FORTH BÜRO
der
Forth Gesellschaft eV

FORTH GESELLSCHAFT e.V. - Antilopenstieg 6a - D 2000 HAMBURG 54

Postgiroamt Hamburg, Kto: 563211-208, BLZ 20010020

(Leer gelassen)