VIERTE DIMENSION

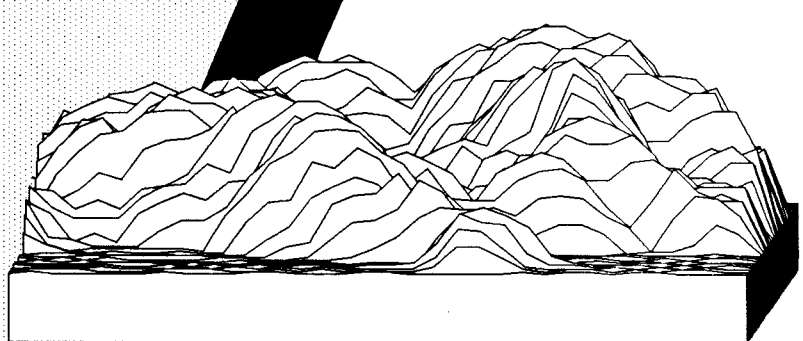
Volume V/Nr.2 - Juni 1989

Themen

FORTH-Tagung 1989
FORTH-Mailbox München
Textinterpreter
Das Wesen der UPN
Die Pals der RTX-Mini-BEE
Fraktale Berge

FORTH MAGAZIN

7,50 DM



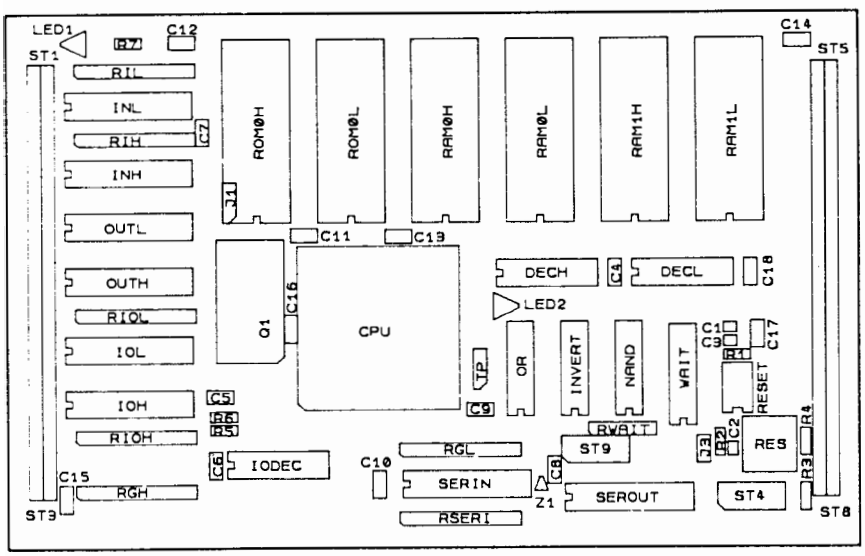
Real Time Express Real Time Express Real Time Express

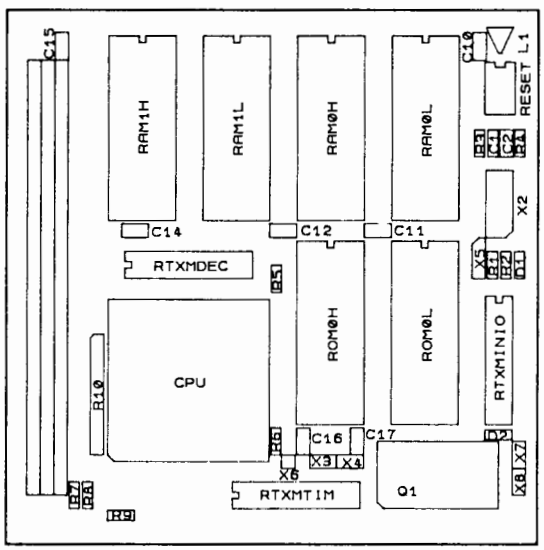
10.000.000 FORTH-Operationen pro Sekunde!

Der REALTIME-EXPRESS läuft weiter !

Die "Emulatorkarte": sämtliche Signale des RTX2000 auf einer 96-poligen VG-Leiste. Serielle Softwareschnittstelle, Systemtakt 6,7 MHz, 64 kByte RAM bestückt, erweiterbar on Board auf 128 kByte, FG-FORTH, kompaktes System 100mm x 100mm durch PALs

Die mc-RISC-Karte: vorgestellt in mc 6/89, 16 Bit Input-Port, 16 Bit-Output-Port, 16 Bit I/O-Bus, serielle Softwareschnittstelle, 8 MHz Systemtakt, 64 kByte RAM bestückt, on Board auf 128 kByte erweiterbar, FG-FORTH, Format 100mm x 160mm.





SONDERPREIS für Mitglieder der FORTH-Gesellschaft:

DM 1.777,--

(incl. Versand, incl. MWSt)

Hochschulrabatt auf Anfrage!

FORTH

REAL TIME EXPRES und RTX ist TradeMark der HARRIS CORPORATION, Palm Bay, Florida

RTX2000

Karten zum Einsteigen erhalten Sie ab sofort bei:
BRÜHL ELEKTRONIK ENTWICKLUNGS-GESELLSCHAFT mbH, Hegelstraße 10, 8500 Nürnberg 10, Tel. 0911/359088
RTX2000, 64 KB PROM, 64 KB RAM, FG-FORTH-Compiler: DM 2.000,–

DELTA t

Die Firma mit dem FORTH - KNOW - HOW

Seitdem es in Echtzeit geht wird der Zufall immer greifbarer

Viele zeichnen Daten schnell auf.
Wir verarbeiten bis zu einer Million Samples/Sekunde mit unserem Multiprozessorsystem auf der Basis von Forth-RISC-Prozessoren.

Ulrich Hoffmann Marina Kern Klaus Schleisiek

DELTA t Entwicklungsgesellschaft für computergesteuerte Systeme mbH
Telefon 040 / 644 57 82 · Roter Hahn 42 · D - 2000 Hamburg 72

EDITORIAL

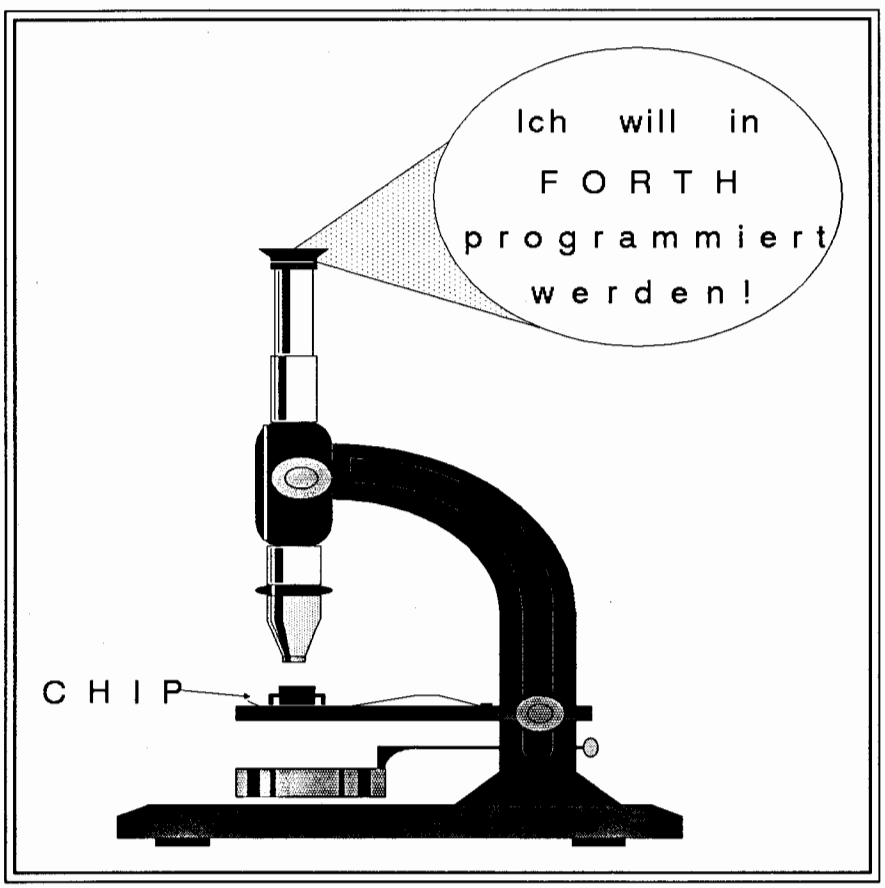
Wie Sie sicherlich sofort bemerkt haben, präsentiert sich die 'Vierte Dimension' im neuen (alten) Kleid. Auf der Jahreshauptversammlung in Aachen wurde auf Antrag von Herrn Schleisiek-Kern beschlossen, die 'Vierte Dimension' wieder im alten Outfit erscheinen zu lassen. Wir möchten nun Sie liebe Leser/innen fragen, welches Titelbild Ihnen besser gefällt. Schließlich wird die Zeitschrift für Sie gemacht und es ist uns wichtig Ihre Meinung zu diesem Thema zu erfahren.

Am Innenleben der 'Vierten Dimension' wurde nichts verändert und so finden Sie auch in dieser Ausgabe wieder viel Interessantes, Anregendes und Aktuelles. Auch wenn der Sommerurlaub vor der Tür steht, möchten wir Sie wieder um rege Mitarbeit bitten. Vielleicht haben Sie ja in einer Musestunde am Strand einen guten Einfall?!

Schöne Urlaubstage und gute Erholung wünscht Ihnen Ihre Redaktion!

Denise Luda

Rainer Aumiller



IMPRESSUM

Titel:
FORTH MAGAZIN 'Vierte Dimension'
Zeitschrift der Mitglieder der FORTH-Gesellschaft e.V. © 1989

Herausgeber:
FORTH-Gesellschaft e.V.

Redaktion:
D. LUDA Software, Staudingerstr. 65, 8000 München 83, Tel. 089/670 83 55

Kontaktadresse:
Entweder direkt die Redaktion anrufen bzw. anschreiben, das FORTH-Büro in München, Postfach 1110, 8044 Unterschleißheim, Tel.: 089/3173784 kontaktieren oder die FORTH-Mailbox München (s.u.) 'Konferenz Vierte Dimension' benutzen.

Quelltextservice:
Der Quelltext von Beiträgen, die mit diesem Symbol gekennzeichnet sind, ist auf der Leserservice-Diskette zur jeweiligen Ausgabe oder in der FORTH-Mailbox in München Tel. 089/7259625 8N1 zu finden.

Quelltext Service

Autoren dieser Ausgabe:
Jörg Staben, Friederich Prinz, Claus Kühnel, Michael Kalus, Peter Saupe, Jörg Plewe, Christoph Krinninger, Frank Stüss, Andreas Findewirth, Ulrich Paul.

Erscheinungsweise:
Vierteljährlich

Redaktionsschluß:
Die zweite Woche im mittleren Quartalsmonat

Auflage:
ca. 1000 Stück

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

Bezugspreis:
Einzelheft DM 7,50, Abonnement 4 Hefte DM 40,- inklusive Versand.

Vierte Dimension Inhalt

LESERBRIEFE und Zuschriften

GEM und volksFORTH-83 auf dem Atari ST

Sehr geehrte Damen und Herren, als Mitglied der FORTH-Gesellschaft (Mitgliedsnr. 2443) schreibe ich Ihnen diesen Leserbrief, der sich mit Problemen der beiden GEM-Wörter **gtext** und **justified** in volksFORTH-83 bei längeren Strings befaßt.

Ich arbeite mit einem Atari ST, den ich in volksFORTH-83 programmiere. Bei der Ausgabe von längeren Strings mit den Worten **gtext** oder **justified** erntete ich in der Vergangenheit regelmäßig einen Adreß-Error. Konkret wollte ich in einem Programm Strings mit max. 40 Zeichen ausgeben lassen.

Ganz verzweifelt habe ich den Fehler in meinem Programm gesucht - ohne den geringsten Erfolg! Dem Wahnsinn nahe, habe ich in meiner Not Herrn Bernd Pennemann angerufen, der mir dann auch gleich den entscheidenden Hinweis gab:

Im File GEM\BASICS.SCR sind auf Screen# 1 die Arrays, die das VDI benötigt, definiert. Wichtig ist dabei das Array **intin**. In diesem Array werden u.a. auch Strings an das VDI übergeben. Und eben dieses Array ist für Strings dieser Länge zu klein!!

Nun könnte man meinen, dadurch daß das Array mit der Sequenz &60 allot 60 Bytes breit ist, könnte man 60 Zeichen lange Strings übergeben - Pustekuchen, es sind nur 30 Zeichen! Das VDI benutzt dieses Array nämlich **nicht** als Byte-Feld, sondern liest aus dem untersten Byte eines (bekanntlich 16 Bit, also 2 Byte breiten) Wortes das Zeichen heraus.

Übergibt man Strings, die länger als 30 Zeichen sind, wird das nachfolgende Feld mit seinen Daten überschrieben. Daher die Fehlermeldung: Adreß-Error.

Man braucht sich allerdings nur das Feld entsprechend den jeweiligen Erfordernissen zu dimensionieren, und schon ist das Problem gelöst.

An dieser Stelle noch einmal vielen Dank an Bernd Pennemann für seine Hilfe!

Ich habe bei der Mitgliederversammlung der FORTH-Gesellschaft in Aachen kurz mit Herrn Klaus Kohl (der ja jetzt das volksFORTH-83 betreut) über dieses Problem gesprochen. Man kann wohl davon ausgehen, daß das überarbeitete FORTH entsprechend geändert ist.

Ich kann mir gut vorstellen, daß noch mehr FORTH Programmier(innen) mit diesem Problem zu kämpfen haben.

Martin Josef Warken
Potsdamer Allee 12
6690 Sankt Wedel

Anm. der Red.: In GEM-Programmen wird das Wort-Feld intin üblicherweise 128 Elemente also 256 Bytes groß dimensioniert. Eine obere Grenze für dieses Feld kann mit der VDI-Funktion vq_extnd bestimmt werden. Das Wort-Feld intout enthält nach dem Aufruf dieser Funktion im 15.ten Element die maximale Länge des intin-Feldes, die benutzt werden kann (-1 = kein Maximum). Mit einem ähnlichen Problem hatte übrigens auch Christoph Krinniger bei seinem Programm 'Fraktale Berge' zu kämpfen.

TURBO FORTH GRATIS

Liebe FORTH-Redaktion, ich bin von der Französischen FORTH-Gesellschaft JEDI ermächtigt worden, das von mir ins Deutsche übertragene Grundsystem von Turbo-FORTH an private Interessenten kostenlos abzugeben.

(Kostenlos schließt ein gelegentliches Dankeschön und vielleicht sogar eine Diskette mit brauchbaren Utilities (5 1/4") nicht aus. Alles andere ist Pfennigkram und eines FORTHianers unwürdig. Diese Dinge machen mir großen Spaß. Ohne jegliche Empfangsbestätigung, wie schon vorgekommen, ist der Spaß allerdings weniger groß.)

Es handelt sich um folgende Dateien:

- TURBO.COM
- FORTH.VOC
- ROOT.VOC
- LIES.DAS
- ERATOS.FTH
- SOUNDS.FTH

TURBO.COM

Vollständiges Turbo-FORTH-System mit eingedeutschten Meldungen.

FORTH.VOC

Deutsche Erklärung sämtlicher Worte des (üblichen) Vokabulars FORTH (etwa 600). Diese Liste kann mit einem Editor (ich verwende den von Turbo-PASCAL) oder dem Public-Domain-LIST oder TYPE usw. ausgelesen werden (ASCII-Datei) und natürlich auch über den Drucker (ich verwende einen NEC-P6) ausgegeben werden. Eigentlich wichtig ist aber, daß bei ihrem Vorhandensein aus dem System heraus die Kurzerklärung eines jeden FORTH-Wortes über HELP Wort abgerufen werden kann. (Ich verwende aus naheliegenden Gründen eine RAM-Disk.) Zusammen mit der Möglichkeit, über SEE 'Wort' die einzelnen (Nichtmaschinen-)Worte decompiliert abzurufen, ist das ein wirklich wunderbares Arbeiten.

ROOT.VOC

Wie bei FORTH.VOC, jedoch auf die 9 Worte des für die Vokabularverwaltung zuständigen Vokabulares ROOT angewandt.

Zuschriften

LIES.DAS

Ins Deutsche übersetzte Beschreibung des bisher geschaffenen Turbo-FORTH-Gesamtsystems (9 DIN-A4-Seiten).

ERATOS.FTH

Primzahlen nach dem Sieb des Eratosthenes. Beispiel für ein mit beliebigem ASCII-Editor erstellbares FORTH-Quellprogramm, das über INCLUDE ERATOS ins Grundsystem gerufen wird. (Grundsätzlich lassen sich alle einmal compilierten Anwendungsprogramme per SAVE-SYSTEM ANWEND.COM abspeichern, wodurch Zeit beim Einlesen gespart wird. Mehr siehe unten.)

SOUNDS.FTH

Beispielprogramm zur Tonerzeugung. (Armer PC! Ich war von den Soundmöglichkeiten meines ATs nicht begeistert und habe zwei AY-3-8910 drangehängt. Ungeahnte Möglichkeiten, aber kein Mensch, der sich in Zeitschriftenartikeln mit der Programmierung des AY-3-8910 beschäftigt.)

Ausdrücklich betonen möchte ich, daß es zur Zeit etwa 6 Turbo-FORTH-Disketten (mit Metacompiler und vielen interessanten Utilities) gibt und sich die volle Kraft von Turbo-FORTH erst mit diesen entfaltet.

Beispielsweise habe ich mal kurz ein cursorgesteuertes Menüsystem mit 162 Positionen entwickelt (beleuchteter Balken, wie bei PC TOOLS DELUXE), das als COM-Datei in einer Batch-Datei eine Abfrage über die IF-ERROR-LEVEL-GOTO-Technik erlaubt. SAVE-SYSTEM MENUBAT.COM liefert ein Programm von 26754 Bytes Länge. Die 'Kompaktifizierung' über COMPACT1, die nur die absolut benötigten FORTH-Worte übrig läßt, und zwar ohne Knöpfe, benötigt nur 1359 Bytes für das COM-Programm. Die Disketten können (etwa DM 10,- pro Stück) bezogen werden von:

Association JEDI
17, rue de la Lancette
F-75012 Paris

oder

Herrn Marc Petremann
17 allée de la Noiseraie
F-93160 Noisy le Grand

Marc Petremann ist "secrétaire" von JEDI. In unseren Wörterbüchern steht "Schriftführer". Ich habe den Eindruck, daß er mehr von einem "Generalsekretär" an sich hat. Er versteht unsere teutonische Sprache sehr gut.

Fred Behringer
Straßbergerstr. 9c/519
8000 München 40

Treffen der lokalen Gruppen Rhein-Ruhr

Die lokale Gruppe Rhein-Ruhr trifft sich nicht mehr wie bisher jeden vierten Freitag im Monat im Ottenbrucher Bahnhof in Wuppertal, sondern nach Absprache. Ort und Zeit des nächsten Treffens sind dann jeweils bei mir zu erfahren.

Meine Telefonnummer stimmt allerdings nicht mehr mit der aus der letzten VD überein. Sie lautet jetzt: 0208/423514

Jörg Plewe
Großenbaumerstr. 27
4330 Mühlheim a.d Ruhr

Betrifft: Floating-Point-Paket für den IBM

Es wurde das Floating-Point-Paket für den 80(2)87 und das IBM-volksFORTH vorgestellt. Wer es schon benutzt und schon einige Routinen geschrieben hat oder wer noch gar zu dringende Fragen hat kann ja bei einem von uns (Frank Raschke oder mir) anrufen. Natürlich sind auch Aufklärungen von Bugs gern gesehen. Wir sind beide Physik-Studenten und von daher sehr numerisch. Wer also schon eine FFT oder eine Matrixinversion nach Gauß-Jordan oder die Quadratur des Kreises programmiert hat (Nobelpreisverdächtig), soll sich bitte melden. Er bekommt dafür das Integral nach Simpson oder DGL's nach Runge-Kutta...

Frank Stüss
An der Turnhalle 6
6369 Schöneck 2
Tel.: 06187-5019

Fraktale Graphik

Sehr geehrte 'VD' - Redaktion, beiliegend mein Versuch mit fraktalen Grafiken.

Das Programm zeichnet fraktale Gebirge. Der Grundalgorithmus stammt von : A.K.Dewdney aus 'Spektrum der Wissenschaft' März '87 (S.7). Ich habe ihn so abgeändert, daß ein Array von 1000 Bytes für die Berechnung mehr als ausreicht. Weitere Erläuterungen finden sich in 'Berg.scr'. Vielleicht gefällt es?

Mit freundlichem Gruß Rainer Saric.

P.S. : Die 'Vierte Dimension' ist prima !! Wenn die FORTH-Workstation mein Budget nicht übersteigt, würde ich sie gern 'nachbauen'. Ich bin auf weitere Artikel sehr gespannt!

Rainer Saric
Bahnhofstr.10
6258 Runkel,Lahn

Anm. der Red.: Der angesprochene Quelltext ist auf der zu jeder (neueren) VD-Ausgabe erhältlichen Diskette enthalten (oder über die FORTH-Mailbox in München Tel.: 089/7259625.) Die Bezugsquelle ist Klaus Kohl. Näheres findet man in der volksFORTH-Preisliste, die diesem Heft beiliegt.

Nachrichten aus der Provinz

von Friederich Prinz, Hombergerstr.335,
4130 Moers 1, Tel.: 02841/58398

Liebe FORTH-Freunde, zunächst wieder ein paar 'Nachrichten' aus der Provinz. In unserer FORTH-Gruppe hatte sich zunächst ein 'harter Kern' herauskristallisiert, die Gruppe der 'Aktiven', der mehr oder weniger erfahrenen FORTHler und Programmierer. Daß 'unsere' Anfänger zunächst mit starken Berührungsängsten zu kämpfen hatten, ist eigentlich kaum verwunderlich. Ohne Ausnahme neigen FORTHler, wie andere 'Experten' auch, dazu, gelegentlich in den Wolken zu schweben und auf das Unverständnis etwaiger weniger firmer Anwesenden wenig Rücksicht zu nchmen. Das war zu Anfang auch bei uns kaum anders.

Heute wissen unsere Anfänger, daß auch der 'harte Kern' der Gruppe ihre Fragen und Sorgen ernst nimmt und gerne bereit ist, auch die Wolkenschlösser zu verlassen und gelegentlich auf den harten Boden der Eleven zurückzukehren. Dazu haben Beispiele beigetragen, wie das unseres Kollegen, der im Augenblick gerade einen Rechner der Marke Eigenbau zusammen'nagelt' und trotzdem bereit ist, seine für viele nicht nachvollziehbare Arbeit zu unterbrechen und jederzeit mit Rat und Tat bei der Lösung der verschiedensten kleinen Hardware-Problemchen hilft. Das geht vom Speicherausbau über den Einbau von Grafikkarten bis zu kleinen Lötarbeiten die dem jeweiligen Rechner einen etwas höheren Systemtakt spendieren. Selbstverständlich steht gerade dieser Kamerad bei allen Anschaffungsfragen zur Verfügung. Ein solches Beispiel wirkt gegenüber den Neulingen beinahe Wunder. Sie sehen sich akzeptiert und erfahren glaubwürdig, daß jeder einmal klein angefangen und niemand von uns das bereits vergessen hat.

FORTH quasi als ein Vehikel auf dem Wege in die Welt der Computer

Eine weitere große Hilfe bei der Integration der Anfänger ist der Unterricht, den die Eleven bei uns erhalten. Wir haben hierüber bereits am Telefon gesprochen (mit Herrn Schnitter, die Red.). Ganz im Sinne 'guter Missionare' benutzen wir FORTH quasi als ein Vehikel auf dem Wege in die Welt der Computer. So bestand die erste Einheit des Unterrichts in einer Vorstellung aller gängigen Sprachen und Programm-Entwicklungssysteme. Dabei haben wir auch reine anwendungsorientierte Systeme und Systeme aus dem CAD-Bereich diskutiert. Daß FORTH im Grunde über allen diesen Sprachen und Systemen steht, war zu Anfang bloße Behauptung, die von den Eleven nicht überprüft werden konnte. Heute sieht das bereits ganz anders aus.

Im Verlauf unseres Kurses haben wir immer wieder versucht zu erklären, wie bestimmte Dinge sich dem Rechner darstellen, bzw. wie bestimmte Anweisungen von der Maschine abgearbeitet werden. So haben wir zum Beispiel bis heute, zur ungefähren Mitte unseres Kurses gemeinsam mit unseren 'Schülern' eigene Routinen entwickelt, die in vereinfachter Form Dinge wie das Einlesen von Zahlen bewerkstelligen. Dabei erwacht ganz von allein die Erkenntnis, daß solche Sachen mit anderen Sprachen gar nicht oder nur sehr schwer zu realisieren wären.

Interessanter als der eigentliche Kurs ist oft die anschließende Diskussion. Dabei schneiden wir die verschiedensten Themen an, was immer gerade anliegt. Bei diesen Diskussionen geht es meistens um irgendwelche Grundsatzfragen zum Geschehen innerhalb des Prozessors oder im BIOS oder BDOS oder zu dem, was sich eigentlich auf der Festplatte alles so tut. Wo immer das möglich ist, beantworten wir diese Fragen natürlich mit Beispielen in FORTH !

Noch erfreulicher als das gute Vorankommen unserer Einsteiger ist die Tatsache, daß wir nicht nur begreiflich machen konnten, daß niemand irgendwelche Ängste gegenüber der eigentlichen Gruppe zu haben braucht, sondern darüber hinaus auch Werbung für FORTH selbst und für die FORTH-Gesellschaft machen konnten. Unsere Einsteiger erkennen, daß FORTH ebenso ein ungewöhnlich mächtiges Werkzeug bei der Programmierarbeit ist wie auch eine großartige und unterstützenswerte Idee. Ich persönlich bin davon überzeugt, daß von den sieben 'ständigen' Zauberlehrlingen zumindest fünf noch vor dem Ende des Kurses der FORTH-Gesellschaft als Mitglieder beitreten werden.

Leider sind nicht alle Anfänger gleichmäßig interessiert und engagiert. Einige nehmen das ganze Thema recht locker auf und kommen nur dann und wann. Natürlich entstehen diesen immer größere Lücken, aber wir wollen und werden den jetzt laufenden Kurs ohnehin wiederholen und hoffen, daß die

Erfolge der etwas 'ernsthafteren' die anderen motivieren werden. Insgesamt haben wir im Augenblick maximal 13 'Schüler'.

Der relativ schnell fortschreitende Erfolg der ständigen Kursteilnehmer beruht ganz wesentlich auf einer 'Neuerung' die wir uns hier genehmigt haben. Wir benutzen inzwischen Tom Zimmer's F83. Offenbar war gerade der 'alte' blockorientierte Editor des F83 von Perry und Laxen ein großes Hindernis beim Erlernen von FORTH. Das leuchtet in gewisser Weise auch ein. Wer sich ohnehin zunächst an meist vollkommen ungewohnte Strukturen wie z.B. die UPN gewöhnen muß, ist froh, wenn er nicht noch zusätzlich relativ viele Kommandos eines unbequemen und auch unzeitgemäßen Editors auswendig lernen muß. Ich meine, daß gerade die Bequemlichkeit bei der Auswahl der 'persönlichen' Sprache oder des Systems eine große Rolle spielt. Der Forderung nach Bequemlichkeit kommt Tom Zimmer's sequentielles FORTH sehr weit entgegen.

(Übrigens ist dieser gesamte Text mit dem Editor des Zimmer-FORTH geschrieben.)

So viel zu dem was sich zur Zeit so bei uns tut. In knapp zwei Wochen findet das große Treffen in Aachen statt. Zwei unserer Mitglieder, Hans Chrapia und Karl Schroer, wollen an zumindest einem Tag daran teilnehmen. Wenn Sie es einrichten können, werden Sie am frühen Morgen in Aachen eintreffen und am Abend wieder nach Moers zurück fahren. Vorbereitungen irgendwelcher Art brauchen also für diese zwei nicht getroffen zu werden.

Sicher habt Ihr mit den Vorbereitungen für das Treffen noch alle Hände voll zu tun. Trotzdem möchte ich Euch noch gerne etwas vorstellen.

Ich habe mich als >Nicht Purist< natürlich mit einigen anderen Sprachen und Systemen als mit FORTH beschäftigt und bin dabei vor einiger Zeit auf ein Programm gestoßen, daß ich von Anfang an als ein typisches FORTH-Problem gesehen habe, als ein Programm, das unbedingt in FORTH geschrieben sein sollte.

Dabei handelt es sich um eine frei programmierbare Turing-Maschine, um einen 'fleißigen Biber' nach Tibor Rado. Ich bin auf dieses Programm in einer Sonderausgabe der CHIP-Special Sonderheft Reihe zu Turbo Pascal gestoßen. Dort war diese Turing-Maschine von H.J. Elschenbroich in Pascal realisiert. Der 'Biber' ist eines der großen Beispiele, die zeigen, daß Mathematik auch Spaß machen kann und durchaus einiges mit Experimentieren zu tun hat. Da ich, wie gesagt, der Meinung war, daß dieses Programm ein typisches FORTH-Programm ist, habe ich es nicht nur in FORTH 'übersetzt', sondern zugleich in einigen ganz wesentlichen Teilen so verändert, daß es den natürlichen Fähigkeiten von F83 angepaßt ist. Mein Biber arbeitet ungleich schneller und effizienter als der von H.J. Elschenbroich vorgestellte und ist auch vom Aufbau des Source her für jedermann nachvollziehbar.

Wenn Ihr mein Programm für gut genug haltet und Euch nicht über die Länge von 8 DIN-A4 Seiten erschreckt, gebe ich es gerne zur allgemeinen Verfügung frei. Ich meine, daß das Experimentieren mit den diversen Bibern sehr viel Spaß bereitet, und daß der beiliegende Quelltext Anfänger wie Insider einlädt auch mit dem Programm selbst zu experimentieren.

(Anm. der Red.: Das Programm BIBER wird in einer der nächsten 'Vierten Dimensionen' veröffentlicht.)

Allerdings weise ich vorsorglich darauf hin, daß der Biber einige Worte benutzt, die dem 'normalen' F83 nicht bekannt sind. Das sind Worte zur Bildschirmsteuerung, die in verschiedenen Video-Modi schalten. Aber das dürfte kein Problem darstellen.

In diesem Sinne verabschiede ich mich für heute und verbleibe mit den besten und herzlichsten Grüßen aus der Provinz am linken Niederrhein

Euer FORTH-Freund

Friederich Prinz

P.S. Ich hoffe Ihr habt alle ein frohes Ostern gefeiert. F.

Folgende 'Nachrichten' erhielten wir kurz vor Redaktionsschluß

Liebe FORTH-Freunde, ich schreibe Euch an dieser Stelle wieder ein paar Zeilen aus unserer lokalen FORTH-Gruppe hier in Moers.

Unser Kurs für Anfänger ist bald vorüber, dann gibt es 8 FORTH-Novizen mehr in der Republik. Von den anfänglich 12 Anmeldungen zu dem Kurs den unsere Gruppe in der lokalen Presse angeboten hatte, sind 8 Unentwegte übrig geblieben. Diese 8 Mann, leider ist die einzige Frau die unter den Anmeldungen war, 'abgesprungen', sind allerdings mit großem Eifer und viel Spaß bei der Sache. Da das Interesse an FORTH größer ist, als das Angebot diese Sprache zu erlernen, zeigt sich u.a. daran, daß zwei der Teilnehmer aus 70 km Entfernung zu jedem Termin angereist kommen.

Das Interesse an FORTH ist größer, als das Angebot diese Sprache zu erlernen

Wir benutzen Tom Zimmer's FORTH im Unterricht. Der bequeme sequentielle Editor vereinfacht das Lernen schon deshalb bemerkenswert, weil die Lernenden zumindest die doch unbequemen Worte des alten Blockeditors nicht mehr zu lernen brauchen. Natürlich kennen wir die Argumente der 'Puristen' pro Blockeditor, aber uns ist es zunächst wichtiger 'unsere' Sprache zu vermitteln und zu lehren.

Im Sommer werden wir unsere Aktivitäten vorübergehend in den rein privaten Bereich auslagern müssen. Das Moerser Arbeitslosenzentrum, in dem wir unsere Räume haben, wird dann von der Stadt umgebaut und renoviert. Aber schon im Spätsommer geht es hier mit gleichem Elan weiter. Wir werden dann den Kurs fortführen. Die Gruppe unserer Anfänger, die dann weit genug sein wird, um endlich 'richtig' programmieren zu lernen,

wird dann versuchen ein eigenes kleines Programm zu schreiben. Wir wissen heute noch nicht, wie dieses Programm aussehen wird. Das soll die Gruppe, zu der dann wahrscheinlich noch Interessenten hinzukommen werden, selbst entscheiden. Wir beabsichtigen lediglich Aufgabenstellungen anzubieten und den ganzen Ablauf von Planung, Konzeption, etc, quasi beratend zu begleiten. Das allernotwendigste Grundwissen wird dann vorhanden sein, alles Weitere soll selbst erarbeitet werden.

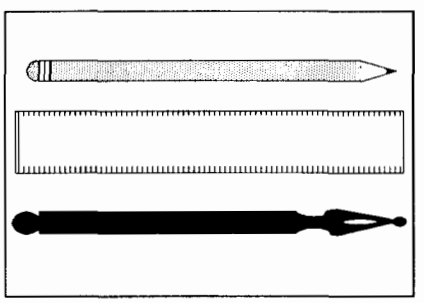
Unsere Gruppe macht inzwischen DFÜ! Seit dem 01. Mai haben wir in der MHB, einer Mailbox des Zerberus Netzes, zwei eigene Bretter zum Thema FORTH. Diese Bretter werden von uns selbst verwaltet. Hierfür sein an dieser Stelle dem SysOp unserer Box herzlich gedankt.

Das erste FORTH-Brett enthält den Kurs, den wir in dieser Art auch unseren Anfängern angeboten und hier abgehalten haben. Dieses Brett kann nur gelesen werden. Das zweite FORTH-Brett kann gelesen und beschrieben werden und dient zur Ablage von Fragen zum FORTH-Kurs, aber auch zu FORTH allgemein. Wer immer etwas zu FORTH sagen oder über FORTH wissen will, kann sich darin melden.

Wir werden abwarten müssen, wie sich das Interesse an diesem Angebot entwickelt. Wenn die FORTH Bretter hier so stark frequentiert werden, wie wir uns erhoffen, wird die MHB unsere FORTH Bretter im Zerberus Netz bundesweit anbieten!

Erreichen kann man die MHB täglich 24 Stunden mit 8N1 unter 02841/57325.

Das wär's für heute wieder einmal. Ich verbleibe mit besten Grüßen vom Niederrhein.



FORTH-Grundseminar

Folgende Mitteilung erreichte uns von der Firma RSO :

In München findet am 01./02. Juni 89 und am 29./30. Juni ein **FORTH-GRUNDSEMINAR** statt. Es wird um eine schnellstmögliche schriftliche Anmeldung gebeten.

Nach der Sommerpause finden im Herbst-Winter 1989/90 weiterhin folgende Seminare (Dauer jeweils 2 Tage) u.a. zu den folgenden Themen statt:

- DSP-Anwendung mit FORTH am 04./05. September 1989

und desweiteren:

- RTX 2000 Anwendung mit FORTH
- Datenbankanwendung unter FORTH
- Prozeßvisualisierung unter FORTH
- Netzwerk-Kommunikation mit FORTH

Gebühren sind bei der Firma:

RSO Gesellschaft für technische Kybernetik mbH
Kirchtruderingerstr. 22
8000 München 82

zu erfragen.

Hinweise für Autoren

Auch in Zukunft möchten wir Beiträge veröffentlichen, die Sie uns hoffentlich in großer Zahl liefern werden. Schicken Sie Ihre Manuskripte bitte an die Redaktion der 'Vierten Dimension' D.LUDA Software Staudingerstr. 65, 8000 München 83, Tel. 089/6708355 oder legen Sie sie in der FORTH-Mailbox München 'Konferenz Vierte Dimension' ab (8N1 Tel. 089/7259625).

Am liebsten hätten wir die Manuskripte auf einer Diskette 5 1/4" (360 Kbyte oder 1,2 Mbyte) im IBM-Format oder einer 3 1/2" Diskette (Atari-Format oder 720 Kbyte IBM-Format). Ist Ihnen das nicht möglich, können Sie auch normale Texte auf Papier einsenden. Bei Bildern sollte allerdings darauf geachtet werden, daß ein möglich guter Kontrast vorliegt. Die Arbeiten sollten in dieser Reihenfolge enthalten:

- Kurzer Titel,
- Autor,
- Zusammenfassung (ca. 50 Worte),
- Schlüsselworte (ca. 5), Text,
- Quellenangaben,
- Illustrationen,
- Tabellen,
- Quellcode.

Die Beiträge werden überarbeitet. Falls ein ausführliches Lektorieren erforderlich ist, erhalten Sie vor der Wiedergabe den Beitrag zur Korrektur und Zustimmung zurück. Layouts werden nicht mehr zur Prüfung durch die Autoren vorgelegt. Autoren erhalten auf Wunsch ein kostenloses Exemplar der 'Vierten Dimension' mit ihrem Artikel.

Invitation & call for papers

euroFORML conference
on the FORTH programming
language
and FORTH processors

from October 13th through 15th
1989
at Hotel Selau next to Forth near
Nuremberg,
Federal Republic of Germany

sponsored by
"FORTH-Gesellschaft e.V., FRG"
and "Forth Interest Group Inc,
USA"

euroFORML will be an international meeting of computer practitioners using FORTH as a problem solving tool. Lectures, workshops and presentations are planned to demonstrate techniques and problem solving strategies that have proved useful.

This years conference will specifically focus on FORTH in real time applications and its potential in the area of robotics and industrial control applications.

The conference will be held at Hotel Selau in Neunkirchen am Brand next to the village Forth 20km to the West of Nuremberg. Hotel Selau has all the usual facilities of a modern Hotel including a swimming pool and a sauna. The Hotel can accomodate 72 people in 33 double and 6 single rooms - additional single rooms will be available in the immediate vicinity.

The conference language will be English and FORTH, of course. The conference is supposed to be self organizing ie. there will not be a strict agenda prior to the conference. If you want to present your ideas you may choose one of the following formats:

Paper presentation

You will present a paper in a 10 minute talk in front of the whole group with the possibility to get immediate feedback.

Poster presentation

You will be assigned a "poster space" where you can present your ideas to a small group of people in a separate room. This is especially useful for demonstrations of hard- and software.

Workshops

You may organize or participate in workshops which will be organized at the beginning of the conference, depending on demand.

An English language proceedings will be published after the conference; papers to be included in the proceedings will be handed out at the beginning of the conference.

Registrations should be mailed by September 1st. A DM 250,- deposit per person is required. (Money order in German funds, eurocheques in foreign funds or transfer to Postgiro account# 5226 46 - 203, bank code 200 100 20). The full amount is due at the beginning of the conference. Space is limited and you are assigned on a first-come, first-serve basis.

We reserve 1/3 of the available space for students/low income at a reduced rate. Accomodation in a single room is DM 40,- extra.

Author instructions

Papers to be presented at the conference (to be included in the proceedings) will have to be mailed to FORTH-Gesellschaft eV no later then October 1st 1989 in camera ready form. The format is DIN A 4 (letter size) with a margin of 2,5 cm (1.5") on all sides. Every page should have a page number and the authors name. Papers should not exceed 15 pages. Code should be accompanied by shadow screens.

For reservations and conference papers write or call

M.Kern / c/o DELTA t GmbH / Roter Hahn 42 / D-2000 Hamburg 72 / W-Germany / (+49) (40) 644 5782

Direct your questions which are not related to the conference to

FORTH-Gesellschaft eV / Postfach 1110 / D-8044 Unterschleißheim / W-Germany / (+49) (89) 317 3784

		FORTH- Gesellschaft e.V. member rate
Participant (accomodation, three meals per day and conference proceedings)	**DM 600,-**	**DM 560,-**
Guest (accomodation and meals)	**DM 400,-**	**DM 400,-**
Student (limited openings) (accomodation, meals, conference proceedings)	**DM 300,-**	**DM 280,-**
"external" Participant (meals, conference proceedings)	**DM 450,-**	**DM 420,-**

Accomodation in a single room DM 40,- extra.

Nutzung der Firmware des CPC 6128 unter F83 und CP/M+

von Claus Kühnel

Um unter F83 die Grafikmöglichkeiten und andere Firmwareroutinen des CPC 6128 nutzen zu können, habe ich Routinen der Firmware nutzbar gemacht. Die Art und Weise der Einbindung soll das Prinzip verdeutlichen und ist nicht auf die hier vorgestellten, wenigen Grafikroutinen und die Timerfunktion beschränkt. Die gewünschten Erweiterungen lassen sich dann applikationsbezogen mühelos ergänzen.

Laxen und Perry's F83 stellt für den CPC 6128 eine komfortable Programmierumgebung unter CP/M+ (CP/M 3.0) dar. Die recht leistungsfähige Grafik des CPC 6128 kann unter Nutzung der Firmwareroutinen ebenso leicht im F83 verfügbar gemacht werden, wie in BASIC:

Zum Aufruf der Firmwareroutinen existiert im CP/M+ die BIOS-Funktion USERF (BIOS-Funktion Nr. 30), die beispielsweise über die BDOS-Funktion "Direct Bios Call" (BDOS-Funktion Nr. 50) aufrufbar ist. Aus der BIOS-Sprungleiste kann die Adresse der BIOS-Funktion USERF aber auch direkt zu FC5A(H) ermittelt werden. Mit einem normalen CALL kann die BIOS-Funktion USERF nun aufgerufen werden, wobei alle Register der Firmwareroutine übergeben werden. Die Adresse der gewünschten Firmwareroutine muß dem CALL folgen.

Im File FIRMWARE.BLK wurde eine nachladbare Erweiterung zusammengestellt, die die Grafik- und Timerfunktionen der Firmware des CPC 6128 aus F83 heraus verfügbar macht.

Quelltext Service

Die Adresse der Funktion USERF ist als Konstante **'userf** vereinbart. Mit **addr** wurde ein Definitionswort geschaffen, welches den Code für den indirekten Aufruf der Firmwareroutinen zusammenstellt. Dadurch kann die erforderliche Bytefolge bei der Definition der Firmware-Primitives einfach durch den Aufruf des Namens eingebaut werden. Der Aufruf einer Firmwareroutine beeinflußt im allgemeinen alle Register, weshalb der im Doppelregister BC liegende Instruktionpointer in diesen Fällen zu retten ist.

Die Definition des Wortes **hardcopy** erfolgte durch Vorgabe des Maschinencodes in hexadezimal codierten Bytes. Vorteilhaft ist die platzsparende Notierung, die allerdings wenig zur Lesbarkeit beiträgt.

In die Erweiterung aufgenommen wurden nur die wichtigsten Firmwareroutinen. Die Definitionen weiterer Primitives ist in der vorgestellten Weise leicht möglich, so daß bei Bedarf der gesamte Bestand an Firmwareroutinen von F83 aus nutzbar ist.

Glossarium

'userf (-- addr)
Konstante, die die Adresse der BIOS-Funktion USERF übergibt.

addr (addr --)
Definitionswort zum Aufruf von Firmwareroutinen.
Anwendung:
bbba addr gra_initialise
Der Aufruf von gra_initialise hinterläßt den Maschinencode C3 5A FC BA BB

time (-- d)
Abfrage des Timers. d repräsentiert den Timerinhalt in 1/300 sec.

!time (d --)
Setzen des Timers auf den mit d vorgegebenen Wert.

lt (--)
Bestimmung der Laufzeit des nachfolgenden Wortes.
Anwendung: 1000 lt ms (Test der Genauigkeit der Zeitverzögerung)

ms (n+ --)
Zeitverzögerung um n+ Millisekunden.

gra-init (--)
Initialisierung der Grafik. Hintergrundfarbe wird auf INK 0 gesetzt. Grafikzeichenfarbe wird auf INK 1 gesetzt. Der Koordinatenursprung liegt in der linken, unteren Ecke. Das Grafikfenster umfaßt den gesamten Bildschirm.

gra-reset (--)
Rücksetzen der Grafik auf die Standardvereinbarungen.

clg (--)
Löschen des Grafikfensters. Grafikcursor auf Koordinatenursprung setzen.

plot (x y --)
Punkt an der angegebenen, absoluten Position setzen. Der Grafikcursor befindet sich anschließend an dieser Position.

move (x y --)
Grafikcursor auf die angegebene, absolute Position setzen.

mover (x y --)
Grafikcursor relativ zur momentanen Position verschieben.

Stichworte:

» **CPC 6128,**
» **Graphikroutinen.**

Nutzung der Firmware des CPC 6128

draw (x y --)
Linie von der momentanen Position des Grafikcursors zur angegebenen, absoluten Position ziehen.

.tag (c --)
Zeichen an der momentanen Position des Grafikcursors ausgeben. Die Position des Grafikcursors ist die linke, obere Ecke des Zeichens. Die Position des Grafikcursors befindet sich anschließend um acht Punkte weiter rechts.

deltax (-- addr)
Variable, deren Inhalt die horizontale Schrittweise bei Textausgabe kennzeichnet (Default: 8).

deltay (-- addr)
Variable, deren Inhalt die vertikale Schrittweise der Textausgabe kennzeichnet (Default: 16).

.text (addr count --)
Ausgabe eines Strings im TYPE-Format beginnend an der momentanen Position des Grafikcursors. Schrittweise durch deltax und deltay vorgegeben.

hardcopy (--)
Ausgabe einer Grafik-Hardcopy auf EPSON-kompatiblen Drucker (DMP 2000). Ausgegeben wird der gesamte Bildschirminhalt. Eingestellter Grafikmodus wird anschließend wieder hergestellt.

Dr. Claus Kühnel
Zerschnitzer Str. 52
DDR-8020 Dresden

Screen# 0

```
 0 \ Firmware CPC unter F83 (CP/M+)
 1
 2 \s
 3
 4 Die Einbindung der Firmware des CPC erlaubt die komfortable
 5 Nutzung der Grafikroutinen und anderer Firmwarebestandteile
 6 von FORTH aus.
 7
 8
 9
10
11
12
13
14
15
```

Screen #1

```
 0 \ Firmware-Einsprungadressen
 1
 2 hex
 3
 4 fc5a constant 'userf
 5
 6 : addr
 7         create [ assembler ] 'userf call [ forth ] ,
 8         does> dup c@ c, 1+ dup @ , 2+ @ , ;
 9
10
11
12 bd0d addr kl_time_please
13 bd10 addr kl_time_set
14
15 -->
```

Screen #2

```
 0 \ Firmware-Einsprungadressen Grafik
 1
 2 bbba addr gra_initialise
 3 bbbd addr gra_reset
 4 bbc0 addr gra_move_absolute
 5 bbc3 addr gra_move_relative
 6 bbc9 addr gra_set_origin
 7 bbcc addr gra_set_origin
 8 bbcf addr gra_win_width
 9 bbd2 addr gra_win_heigth
10 bbd5 addr gra_get_wwidth
11 bbd8 addr gra_get_wheight
12 bbdb addr gra_clear_window
13 bbe1 addr gra_get_pen
14 bbe4 addr gra_set_paper
15 bbe7 addr gra_get_paper     -->
```

Screen #3

```
 0 \ Firmware-Einsprungadressen Grafik
 1
 2 bbea addr gra_plot_absolute
 3 bbed addr gra_plot_relative
 4 bbf0 addr gra_test_absolute
 5 bbf3 addr gra_test_relative
 6 bbf6 addr gra_line_absolute
 7 bbf9 addr gra_line_relative
 8 bbfc addr gra_wr_char
 9
10 decimal
11
12 -->
13
14
15
```

Screen #4

```
 0 \ Timer (@time !time)
 1
 2 code @time \ -- d
 3         kl_time_please
 4         h push
 5         d push
 6         next
 7 end-code
 8
 9 code !time \ d --
10         d pop
11         h pop
12         kl_time_set
13         next
14 end-code
15 -->
```

Screen #5

```
 0 \ Laufzeitbestimmung (lt ms)
 1
 2 : lt \ -
 3         0. !time
 4         ' execute
 5         @time
 6         ." Laufzeit = " d. ." /300 sec"
 7         cr
 8 ;
 9
10
11 : ms \ n+ --
12         0 ?do 3 0 do loop loop
13 ;
14
15 -->
```

Screen #6

```
 0 \ Grafikroutinen (gra_init gra_reset)
 1
 2 code gra_init \ --
 3         b push
 4         gra_initialise
 5         b pop
 6         next
 7 end-code
 8
 9 code gra_reset \ --
10         b push
11         gra_reset
12         b pop
13         next
14 end-code
15 -->
```

Screen #7

```
 0 \ Grafikroutinen (clg)
 1
 2 code clg \ --
 3         b push
 4         gra_clear_window
 5         b pop
 6         next
 7 end-code
 8
 9 -->
10
11
12
13
14
15
```

Screen #8

```
 0 \ Grafikroutinen (plot)
 1
 2 code plot \ x y --
 3         h pop
 4         d pop
 5         b push
 6         gra_plot_absolute
 7         b pop
 8         next
 9 end-code
10
11 -->
12
13
14
15
```

Screen #9

```
 0 \ Grafikroutinen (move)
 1
 2 code move \ x y --
 3         h pop
 4         d pop
 5         b push
 6         gra_move_absolute
 7         b pop
 8         next
 9 end-code
10
11 -->
12
13
14
15
```

Screen #10

```
 0 \ Grafikroutinen (mover)
 1
 2 code mover \ x y --
 3         h pop
 4         d pop
 5         b push
 6         gra_move_relative
 7         b pop
 8         next
10 end-code
11
12 -->
13
14
15
```

Screen #11

```
 0 \ Grafikroutinen (draw)
 1
 2 code draw \ x y --
 3         h pop
 4         d pop
 5         b push
 6         gra_line_absolute
 7         b pop
 8         next
 9 end-code
10
11 -->
12
13
14
15
```

Screen #12

```
 0 \ Grafikroutinen (.tag)
 1
 2 code .tag \ c --
 3         d pop
 4         e a mov
 5         b push
 6         gra_wr_char
 7         b pop
 8         next
 9 end-code
10
11 -->
12
13
14
15
```

Screen #13

```
 0 \ Grafikroutinen (.text)
 1
 2 variable deltax       8 deltax !
 3 variable deltay      16 deltay !
 4
 5 : .text \ addr count --
 6         2 ?enough
 7         over + swap
 8         do
 9                   i @ .tag
10                   deltax @ 8 - deltay @ mover
11         loop
12 ;
13
14 -->
15
```

Screen #14

```
 0 \ Grafik-Hardcopy (,loop c,loop)
 1
 2 hex
 3
 4 : ,loop
 5         rot 2dup * allot
 6         0 ?do rot here i 1+ 3 pick * - 3 pick execute loop
 7         2drop
 8 ;
 9
10 : c,loop
11         ['] c! 1 ,loop
12 ;
13
14 -->
15
```

Screen #15

```
 0 \ Grafik-Hardcopy
 1
 2 create hardcopy
 3 c5 cd 5a fc 28 bd 3e 1b cd 5a fc 2b bd 3e 40 cd 5a fc 2b bd
 4 3e 1b cd 5a fc 2b bd 3e 6c cd 5a fc 2b bd 3e 0c cd 5a fc 2b
 5 bd 3e 1b cd 5a fc 2b bd 3e 33 cd 5a fc 2b bd 3e 12 cd 5a fc
 6 2b bd 3e 0d cd 5a fc 2b bd 3e 0a cd 5a fc 2b bd cd 5a fc 0cc
 7 bb d5 e5 cd 5a fc d5 bb d5 e5 cd 5a fc d8 bb d5 e5 cd 5a fc
 8 e7 bb 47 cd 5a fc e1 bb 4f c5 cd 5a fc ba bb cd 5a fc 28 bd
 9 11 7f 02 21 fa ff cd 5a fc c9 bb 21 98 01 3e 1b cd 5a fc 2b
10 bd 3e 4c cd 5a fc 2b bd 3e 00 cd 5a fc 2b bd 3e 03 cd 5a fc
11 2b bd 11 81 fd e5 0e 00 06 06 23 23 2b 2b c5 d5 e5 cd 5a fc
12 f0 bb e1 d1 c1 fe 00 28 02 cb c1 cb 21 05 78 fe 00 20 e5 cb
13 29 79 cd 5a fc 2b bd 30 f8 e1 13 7a b3 fe 00 20 0cc 18 02 18
14
15 -->
```

Screen #16

```
 0 \ Grafik-Hardcopy (Maschinencode2)
 1
 2 a9 06 96 3e 00 cd 5a fc 2b bd cd 5a fc 09 bb 30 04 fe 20 28
 3 49 05 78 fe 00 20 e8 3e 0a cd 5a fc 2b bd 01 0c 00 3f 0ed 42
 4 7c b5 fe 00 28 02 18 cf c1 78 cd 5a fc e4 bb 79 cd 5a fc de
 5 bb e1 d1 cd 5a fc d2 bb e1 d1 cd 5a fc cf bb e1 d1 cd 5a fc
 6 c9 bb cd 5a fc 28 bd c1 c3 0a 01 18 14 00 01 96 00 3e 00 cd
 7 5a fc 2b bd 0b 78 b1 fe 00 28 c1 18 f0 00
 8 14e c,loop
 9
10 ' copy dup >body swap !
11
12 decimal
13
14 forth definitions
15 -->
```

Screen #17

```
 0 \ Grafikdemo
 1
 2 : text1 " Das ist im Grafikmodus ausgegebener Text." ;
 3 : text2 " Grafik-Primitives mit MOVE und DRAW definierbar." ;
 4
 5 : .text1 \ deltax deltay --
 6         deltay ! deltax ! 100 200 deltay @ 4 * + move text1 .text ;
 7
 8 : demo \ --
 9         gra_init clg
10         0 16 do 11 i .text1 -2 +loop
11         8 deltax ! 120 108 move text2 .text
12         100 150 move 539 150 draw 539 50 draw 100 50 draw
13         100 150 draw
14         0 0 move 638 0 draw 638 399 draw 0 399 draw 0 0 draw
14 ;
15
```

Behandlung einer CASE-Situation, Teil I

von Jörg Staben

Strukturierung mit IF ELSE THEN/ ENDIF

Ziel dieses Beitrages ist es, die vielfältigen Möglichkeiten zu zeigen, mit denen eine **Fallunterscheidung** in FORTH getroffen werden kann. Kennzeichnend für eine solche Programmsituation ist, daß von verschiedenen Möglichkeiten des Programm-Flusses genau eine ausgesucht werden soll. Ausgehend von einer übersichtlichen Problemstellung, einem Spiel, werden die notwendigen Grundlagendefinitionen und die Entwicklung der oben beschriebenen Kontrollstruktur beschrieben.

Zurückgreifend auf [1] scheint mir ein Würfelspiel geeignet, um die Notwendigkeit einer Fallunterscheidung zu zeigen. Da das Würfelspiel CRAPS hier in Deutschland kaum bekannt ist, habe ich ein Trinkspiel aus meiner Schulzeit mit ebenso einfachen Regeln aufgegriffen: Bei diesem Spiel, das ich nach [1] auch **CRAPS** nenne, ging es darum, einen Vorrat von gefüllten Gläsern unter den Mitspielern mit Hilfe des Würfels zu verteilen und leer zu trinken:

- Bei einer EINS wurde ein Glas aus dem Vorrat in der Tischmitte genommen und vor sich gestellt.
- Bei einer ZWEI oder einer DREI bekam der Nachbar/ die Nachbarin links ein Glas des eigenen Vorrates zugeschoben.
- Bei einer VIER oder einer FÜNF wurde dem Nachbarn/ der Nachbarin rechts ein Glas des eigenen Vorrates vorgesetzt.
- Bei einer SECHS wurden alle Gläser, die der Spieler/ die Spielerin vor sich stehen hatte, leer getrunken.

Damit ist so ungefähr klar, wie das Spiel ablief und welcher Spielsinn dahintersteckte...

Zuordnung ist also: 1 = nehmen, 2/3 = links, 4/5 = rechts, 6 = trinken und entsprechend der Augenzahl des Würfels soll eine der 6 möglichen Aktionen ausgeführt werden. Weil das Würfeln mit einem richtigen Würfel mehr Spaß macht, als den Zufallsgenerator des Rechners zu benutzen, soll sich unser Programm darauf beschränken, das Ergebnis dieses Würfelns einzulesen und auszuwerten. Daraufhin bekommt der Mitspieler/ die Mitspielerin mitgeteilt, welche der sechs Handlungen auszuführen ist.

Damit stellt sich schon das erste Problem, die **Zahleneingabe** in FORTH. FORTH besitzt keine Standardeingaberoutine für eine Zahl, entsprechend READ oder INPUT in anderen Sprachen. Zugleich stoßen wir auf eine Besonderheit im volksFORTH: Zu dem Wort NUMBER? gibt es ein gleichnamiges Wort NUMBER? im L&P F83 Model, das anders (besser!) mit den Parametern umgeht. Dies führt zu der Definition dieses neuen Wortes, das entsprechend dem F83-NUMBER? arbeitet:

```
: F83-number?  ( string -- d f )
   number? ?dup IF
         0< IF extend ENDIF
         true exit
   THEN drop 0 0 false ;
```

Sollte jemandem das ENDIF ungewohnt vorkommen - weil es deutlich macht, warum FORTH ohne eine Verbundanweisung wie PASCAL auskommt, wurde es mit Hilfe der Synonymdeklaration ALIAS im volksFORTH definiert und kann so parallel zum THEN benutzt werden:

```
' THEN Alias ENDIF
 immediate restrict
```

Damit steht der Definition von INPUT# und dem Einlesen von Zeichen in einen bestimmten Speicherbereich, hier nach PAD, und dem Umwandeln dieser Zeichen in eine Zahl nichts mehr im Wege. Das Wort >EXPECT ist bequemer einzusetzen als EXPECT, weil es die erwarteten Zeichen schon als counted String (Zeichenkette mit der Längenangabe im ersten Byte) an der angegebenen Adresse ablegt. Deswegen werden auch nur C/L -1, also 63 Zeichen maximal nach PAD eingelesen. Und weil wir von einem wohlwollenden Benutzer unseres Programms ausgehen, wird das nach der Konvertierung von F83-NUMBER? übergebene Flag unbeachtet weggeworfen. Ebenso unbekümmert wird die doppeltgenaue Zahl zur einfachgenaue Zahl gemacht, indem die oberen 16 Bit dieser 32-Bit-Zahl einfach gelöscht werden. Dank der fehlenden Typüberprüfung erledigt ein 2DROP beides gleichzeitig:

Quelltext Service

```
: input#  ( <string> -- n )
      pad c/l 1- >expect
      pad F83-number?  2drop ;
```

Nun definieren wir die Wörter, welche die sechs oben genannten Aktionen symbolisch ausführen sol-

Stichworte:

- **CASE,**
- **volksFORTH,**
- **Würfelspiel**

Behandlung einer CASE-Situation

len. Dabei richten wir uns nach den Spielregeln, die für jedes Würfelergebnis eine Handlung vorschreiben (siehe Bild 1):

SCHIEBEN ist eine Dummyprozedur dessen Notwendigkeit sich erst sehr spät ergibt. Trotzdem verbessert SCHIEBEN die Lesbarkeit, die zwar nicht Gegenstand dieses Beitrags ist, aber auch immer beachtet werden sollte. Ein freundliches Programm sagt dem Anwender auch, was erwartet wird und gibt Informationen aus, somit definieren wir (siehe Bild 2):

Jetzt können wir uns auf das Wort konzentrieren, das die bereits eingelesene Augenzahl auswertet. Das Wort AUSWERTUNG soll entsprechend einem Selektor genau eine von 6 möglichen Prozeduren ausführen. Also wird man prüfen, ob diese oder diese oder ... der Möglichkeiten in Frage kommt. Bei 6 Möglichkeiten wird die Lösung mit IF..ELSE..THEN schon ziemlich unübersichtlich ! Dazu kommt noch die Prüfung, ob der übergebene Parameter eine Zahl zwischen (between) 1 und 6 war. Die Definition von BETWEEN ist volksFORTH gemäß recht kurz (siehe Bild 3):

Ein Verzicht auf den ELSE-Teil führt schon zu einer übersichtlicheren Form (siehe Bild 4):

Da eine solche Prüfung auf Gleichheit in der Programmierpraxis oft vorkommt, stellt das volksFORTH dafür das Wort case? zur Verfügung. case? vergleicht die obersten beiden Stackwerte miteinander. Bei Ungleichheit bleibt der Testwert (Selektor) erhalten, so daß die Worte DUP und = dadurch ersetzt werden (siehe Bild 5).

Bei dieser Auswertung wird aus dem Quelltext zu wenig deutlich, daß bei ZWEI und DREI dieselbe Handlung ausgeführt wird, wie auch VIER und FÜNF die gleichen Aktionen zur Folge haben. =OR prüft deshalb einen Testwert n2 auf Gleichheit mit einer unter einem Flag f1 liegenden Zahl n2. Das Ergebnis dieses Tests wird mit dem bereits vorliegenden Flag ORverknüpft. Dieses neue Flag f2 und der Testwert n1 werden übergeben:

```
\ nehmen  trinken  links  rechts  schieben

: nehmen    bright ." ein Glas nehmen"          normal 2 spaces ;
: trinken   bright ." alle Gläser austrinken"   normal 2 spaces ;
: links     bright ." ein Glas nach LINKS"      normal 2 spaces ;
: rechts    bright ." ein Glas nach RECHTS"     normal 2 spaces ;

: schieben ;
```

Bild 1

```
: Anfrage     cr ." Sollen Sie nehmen, trinken oder schieben? "
   cr ." Bitte Ihre Augenzahl und <cr> : " ;
Glückwunsch  cr ." Viel Glück beim nächsten Wurf ... " ;
```

Bild 2

```
: between
( Wert Untergrenze Obergrenze
 -- ff | tf, wenn Untergrenze < = Wert < = Obergrenze )
   1+ uwithin ;

: Auswertung.1 ( Wurfergebnis --)
   dup 1 = IF nehmen ELSE
      dup 2 = IF links schieben ELSE
      dup 3 = IF links schieben ELSE
         dup 4 = IF rechts schieben ELSE
         dup 5 = IF rechts schieben ELSE
            dup 6 = IF trinken THEN
            THEN
            THEN
         THEN
         THEN
      THEN
   1 6 between not IF invers ."Betrug!" normal ENDIF ;
```

Bild 3

```
: Auswertung.2   ( Wurfergebnis --)
   dup 1 = IF nehmen   ENDIF
   dup 2 = IF links schieben   ENDIF
   dup 3 = IF links schieben   ENDIF
   dup 4 = IF rechts schieben  ENDIF
   dup 5 = IF rechts schieben  ENDIF
   dup 6 = IF trinken    ENDIF

   1 6 between not IF invers ." Betrug!" normal ENDIF ;
```

Bild 4

```
: Auswertung.3   ( Wurfergebnis --)

   1 case? IF nehmen           exit ENDIF
   2 case? IF links schieben   exit ENDIF
   3 case? IF links schieben   exit ENDIF
   4 case? IF rechts schieben  exit ENDIF
   5 case? IF rechts schieben  exit ENDIF
   6 case? IF trinken          exit ENDIF

   1 6 between not IF
      invers ." Betrug!" normal
   ENDIF ;
```

Bild 5

Behandlung einer CASE-Situation

```
\ =or

code =or ( n1 f1 n2 -- n1 f2 )
   A D xchg D pop
   S W mov
      W ) A cmp
      0= ?[ -1 # D mov ]?
   next
end-code

\ : =or ( n1 f1 n2 -- n1 f2 )
   2 pick = or ;
```

sicherlich der bekanntere ist, der auch in der Literatur und in Quelltexten häufig Erwähnung und Verwendung findet.

Herr Heinz Schnitter hat diesen Eaker-CASE im volksFORTH implementiert und dabei Veränderungen in der Struktur und Verbesserungen in der Anwendung vorgenommen. Diese CASE-Kontrollstruktur besticht durch ihre Lesbarkeit und ist leicht zu verstehen.

```
: Auswertung.4   ( Wurfergebnis --)
   dup
   1 6 between IF
      dup 1 =          IF nehmen          ENDIF
      dup 2 = 3 =or    IF links schieben  ENDIF
      dup 4 = 5 =or    IF rechts schieben ENDIF
      dup 6 =          IF trinken         ENDIF
   ELSE
      invers ." Betrug!" normal
   ENDIF
drop ;
```

Bild 6

Die Funktion dieses Wortes scheint zuerst kompliziert, bringt aber im Quelltext eine deutliche Verbesserung (siehe Bild 6):

Damit haben wir ohne eine CASE-Anweisung eine sehr übersichtliche Steuerung des Programm-Flusses geschaffen. Die Plausibilitätsprüfung, ob die eingegebene Zahl zwischen 1 und 6 lag, ist hier an den Anfang gerückt und wird in einem einzigen ELSE-Zweig abgearbeitet.

Strukturelles CASE

Allerdings stellen viele Programmiersprachen eine CASE-Anweisung zur Verfügung, die wie in PASCAL mit Hilfe eines Fall-Indizes eine Liste von Fall-Konstanten auswertet und eine entsprechende Anweisung ausführt. Obwohl ein solches CASE-Konstrukt - wie oben gezeigt - nicht notwendig ist, macht es Programme besser lesbar und liegt bei Problemstellungen wie der Auswertung eines gegebenen Index eigentlich näher. Dies ist in [1] ausführlich diskutiert worden, wobei aber der ältere Eaker-CASE [2] von Dr. Charles Eaker, Gewinner des CASE Wettbewerbes in der FORTH Dimension Vol.II Nr. 3,

```
\ caselist initlist
\ >marklist >resolvelist

| variable caselist

| : initlist   ( list -- addr )
   dup @ swap off
;

| : >marklist   ( list -- )
   here over @ , swap !
;

| : >resolvelist   ( addr list -- )
   BEGIN dup @
   WHILE dup dup @ dup @ rot !
         >resolve
   REPEAT !
;

\ case    elsecase    endcase

: CASE   caselist initlist 4
; immediate restrict

: ELSECASE
   4 ?pairs
   compile drop 6
; immediate restrict

: ENDCASE
   dup 4 =
   IF drop compile drop
   ELSE 6 ?pairs
   THEN caselist >resolvelist
; immediate restrict

\ of    endof

: OF
   4 ?pairs
   compile over
   compile =
   compile ?branch
   >mark compile drop 5
; immediate restrict

: ENDOF
   5 ?pairs
   compile branch
   caselist >marklist
   >resolve 4
; immediate restrict
```

Diese Implementierung des Eaker-CASE stellt eine Verbesserung gegenüber dem Original dar, indem Herr Schnitter die Kontrollstruktur um ELSECASE erweitert hat. Selbstverständlich ist die neue Version vollkommen aufwärtskompatibel mit der Originalversion.

Verbesserung:

In der Originalversion der CASE Kontrollstruktur ist es nicht möglich, zwischen dem letzten ENDOF und ENDCASE einen Wert oder ein Flag auf den Stapel zu legen, da ENDCASE grundsätzlich den "Top of Stack" entfernte. In der verbesserten Version bereinigt ELSECASE den Stapel. ELSECASE muß jedoch nicht aufgerufen werden ; in diesem Fall kompiliert ENDCASE wie bisher ein DROP. Es ist jetzt möglich, zwischen den Worten ELSECASE und ENDCASE - wie auch zwischen OF und ENDOF - einen Wert auf den Stapel zu legen und diesen außerhalb der CASE Kontrollstruktur zu verwenden.

Änderung:

Die Vorwärtsreferenzen werden nicht über den Stack aufgelöst, sondern über eine verkettete Liste. Die Variable caselist enthält die Startadresse für noch nicht bekannte Sprungadressen. Die Schachtelungstiefe mehrerer CASE Konstruktionen ist beliebig und wird durch initlist gelöst. >marklist füllt zur Kompilierzeit die Liste der Vorwärtsreferenzen und >resolvelist löst sie wieder auf.

Behandlung einer CASE-Situation

```
: Control    bl word 1+ c@ $BF and state @
   IF [compile] Literal THEN
; immediate

: Tastaturabfrage
   ." exit mit ctrl x" cr
   BEGIN key
      CASE control A OF ." action ^a " cr false ENDOF
           control B OF ." action ^b " cr false ENDOF
           control C OF ." action ^c " cr false ENDOF
           control D OF ." action ^d " cr false ENDOF
           control X OF ." exit " true ENDOF
      ELSECASE
           ." befehl unbekannt " cr false
      ENDCASE
   UNTIL ;
```

Bild 7

```
: Auswertung.5   ( Augenzahl -- )
   CASE
      1 OF nehmen   ENDOF
      2 OF links schieben   ENDOF
      3 OF links schieben   ENDOF
      4 OF rechts schieben ENDOF
      5 OF rechts schieben ENDOF
      6 OF trinken ENDOF
   ELSECASE
      invers ." Betrug!" normal
   ENDCASE
;
```

Bild 8

Anwendungshinweis:

Wenn jemand diese Definitionen außerhalb der Zusammenstellung seines Arbeitssystems hinzulädt, sollten nach dem Kompilieren die Namen der mit | als headerless markierten Worte mit CLEAR entfernt werden.

Bevor wir uns wieder der eigentlichen Problemstellung zuwenden, zeigt das Beispiel einer Tastaturabfrage auf CTRL-Tasten hin, wie dieses CASE-Konstrukt einzusetzen ist. Wichtig ist hierbei, daß das OF selbst die Gleichheit der beiden vorliegenden Werte prüft und in diesem Fall die Anweisungen zwischen OF und ENDOF ausführt.

```
\ CASE  OF  ENDOF  ENDCASE  BREAK

: CASE ( n1 -- n1 n1 )   dup ;

: OF  [compile] IF compile drop ; immediate restrict

: ENDOF    [compile] ELSE 4+ ; immediate restrict

: ENDCASE    compile drop
   BEGIN
      3 case?
   WHILE
      >resolve
   REPEAT ; immediate restrict
```

Bild 9

Eine Implementierung für das F83 in traditioneller Form und eine weitere Diskussion dieses Themas finden Sie in [3]. Die oben gezeigte Implementierung des Eaker-CASE soll mit Hilfe eines Wortes CONTROL überprüft werden. Damit wäre bereits die Grundlage für beispielweise eine WordStar-kompatible Cursorsteuerung geschaffen (siehe Bild 7).

Zurück zu unserem Würfelspiel. Mit der CASE-Anweisung läßt sich die Zuordnung der sechs Möglichkeiten zu den sechs Anweisungen ähnlich wie in PASCAL schreiben, lediglich Bereiche wie 0..255 als Fall-Konstanten sind nicht erlaubt (siehe Bild 8).

Nun, nach einer zufriedenstellend programmierten Auswertung, kann unser kleines Programm so hingeschrieben werden:

```
: craps ( -- )

   cr Anfrage cr
   input#
   Auswertung
   cr Glückwunsch
;
```

Nun hat aber Wil Baden in [1] ausgeführt, daß eine CASE-Anweisung nur syntaktischer Zucker für ein Programm ist und letztendlich nichts weiter ist, als das Kompilieren einer verschachtelten IF...THEN-Anweisung. Eine solche Implementierung für das volksFORTH83 wurde von Herrn Klaus Schleisiek-Kern geschrieben (siehe Bild 9):

Wil Badens Implementierung hält sich sehr eng an die logischen Grundlagen, wobei der Unterschied zum EAKER-CASE hauptsächlich darin besteht, daß hier jedes TRUE-Flag den Anweisungsteil zwischen OF und ENDOF ausführt; das OF nimmt keine Prüfung auf Gleichheit vor, sondern beliebige Ausdrücke können zu einem Flag führen, das dann von OF ausgewertet wird. So ist das Auswerten des Fall-Index variabler als beim EAKER-CASE (siehe Bild 10):

Hier bei dieser Konstruktion steht die Plausibilitätsprüfung ganz vorn, um den ELSECASE-Fall durch ein EXIT aus dem Wort zu erreichen. Eine andere Lösung für den Fall,

Behandlung einer CASE-Situation

daß keines der Worte aus der Auswahl-Liste ausgeführt wird, läßt sich mit BREAK erreichen:

```
: BREAK     compile exit
    [compile] THEN ; immediate
restrict
```

Mit diesem Wort und dem Baden-CASE kann nun die siebte Variante der Auswertung vorbildlich übersichtlich gestaltet werden. Dadurch, daß BREAK ein EXIT aus dem Wort darstellt, kann man ein (implizites) ELSECASE erreichen, indem man die Anweisungen der Auswahl-Liste mit OF und BREAK klammert und die Anweisungen für den ELSE-Fall nach ENDCASE aufführt (siehe Bild 11):

Die Fortsetzung dieses Artikels folgt voraussichtlich in der nächsten 'Vierten Dimension'.

```
: Auswertung.6  ( Augenzahl -- )
   dup
      1 6 between not
         IF invers ." Betrug!" normal drop exit ENDIF
   CASE 1 = OF nehmen   ENDOF
   CASE 6 = OF trinken  ENDOF
   CASE 4 < OF links schieben      ENDOF
   CASE 3 > OF rechts schieben     ENDOF
   ENDCASE ;
```

Bild 10

```
: Auswertung.7  ( Augenzahl -- )

   CASE 1 =   OF nehmen BREAK
   CASE 2 = 3 =or  OF links schieben     BREAK
   CASE 4 = 5 =or  OF rechts schieben    BREAK
   CASE 6 =   OF trinken  BREAK
   ENDCASE
   invers ." Betrug!" normal ;
```

Bild 11

[1] Ultimate CASE-Statement, Wil Baden, VD2/87 S.40 ff.

[2] Just in CASE, Dr. Charles Eaker, FORTH DIM II/3

[3] FORTH 83, R. Zech, S.98ff/S.318f.

[4] FORTH Handbuch, E. Floegel, S.109

[5] Menüs in FORTH, W. Wejgaard, Elektroniker 9/88, S.109 ff.

Inserentenverzeichnis:

Bericht von der FORTH-Tagung 1989 in Aachen

M.Kalus

Vom 7. bis 9. April war die diesjährige Tagung der FORTH-Gesellschaft e.V. - diesmal in Aachen. Dieses jährliche Treffen verfolgt zwei Ziele: Zum einen Austausch über den aktuellen Entwicklungsstand von FORTH-Programmiertechnik, Programmierumgebung und Anwenderberichte - und zum anderen Versammlung des Vereins mit Geschäftsbericht und Beschlüsse für das Jahr. Über den Verlauf der Vereinsversammlung wird an anderer Stelle berichtet.

Die Tagung wurde in diesem Jahr in Zusammenarbeit mit dem Lehr- und Forschungsgebiet für Verfahren der Prozeßdatenverarbeitung und Prozeßführung an der Rheinisch Westfälischen Technischen Hochschule (RWTH) Aachen veranstaltet.

Durch die Hilfe des Instituts gab es ein "Tagungshandbuch zur FORTH-Tagung 1989" unmittelbar nach den Vorträgen am Samstag Abend. Interessierte Leser können sicher noch ein Exemplar bekommen - vermutlich über den Kopierservice der FORTH-Gesellschaft.

Im folgenden gebe ich einige dieser Beiträge kurz wieder, um zu zeigen, was in etwa abgehandelt wurde. Die Auswahl traf ich ganz subjektiv und sie sagt daher nichts über die Bedeutung des einen oder anderen Themas aus. Ich möchte damit eher anregen, sich im Tagungshandbuch ausführlicher zu informieren.

Mit fünfzehn Vorträgen war die zur Verfügung stehende Zeit vollständig ausgeschöpft, weil dabei oft bis ins Detail bestimmter Programmiertechniken vorgedrungen wurde. Die Teilnehmer zeigten sich durchweg fachkundig. In den verschiedenen Beiträgen wurde wiederum die große Spannbreite der Einsatzgebiete für FORTH deutlich.

Einordnung der Programmiersprache FORTH

Daß FORTH eine ganz besondere Stellung bei der Verwendung von sehr schnellen Mikroprozessoren in der Echtzeitverarbeitung einnimmt, hat Prof. Pleßmann in seinem Einführungsvortrag "Einordnung der Programmiersprache FORTH" deutlich aufgezeigt. Mittlerweile wird das Sprachkonzept FORTH auf Hardwareebene unterstützt - wohl weil FORTH wie keine andere Sprache seinerseits eine Rechnerarchitektur direkt unterstützt.

Ausführlich vorgestellt wurde der RTX2000 Real Time Express Microcontroller, Harris Semiconductor. Er ist solch ein FORTHChip. Angekündigt wurde außerdem der RTX4000. Die Firmen Flesch, Brühl und Paul stellten dann ihre RTX2000-Karten vor, die verfügbar waren und bereits in Applikationen ihren Dienst taten.

Herr Neubert von der physikalisch-technischen Bundesanstalt in Berlin berichtete über den erfolgreichen Einsatz des RTX2000 bei biometrischen Aufgaben - so bei der Überprüfung der Güte von EKG Geräten. Sein Team benutzte den RTX2000 auf einem AT/FORCE-Board von Silicon Composer in einem 386-AT 16 Mhz. Daten von einem 100Khz A/D Wandler wurden Interrupt-getrieben eingelesen. Eine Fast Fourier Transformation von 1K darüber dauerte beispielsweise 100ms. Diese Lösung wurde bevorzugt, weil so eine hohe Flexibilität der Anwendungen für das System erreicht werden konnte bei genügend schneller Echtzeitverarbeitung und Anzeige der Resultate. Der Hostrechner wurde übrigens programmiert mit PC/FORTH 3.2 von LMI und der RTX2000 mit der FCompiler Language, einem optimierenden FORTH-Compiler, der mit dem Silicon Composer Board geliefert wird.

Doch die Tagung stand nicht nur unter dem Zeichen des RTX2000, der übrigens mit einem Preis von einigen hundert Dollar pro Stück (!) noch nicht gerade zu den breit einsetzbaren Chips gehört. Aber gerade im Bereich kleinster und billiger Rechner und Rechnernetze hat FORTH seine Stärken gezeigt. Wie FORTH in verteilten Rechnersystemen vorteilhaft verwendet werden kann, zeigten die Vorträge von Zöller, Dahm, Schnitter und Pfüller.

Herr Zöller stellte sein Konzept der Kommunikation zwischen Rechnern vor - realisiert in einer haustechnischen Anwendung auf der

Basis preiswerter Einzelkomponenten mit simpler Zweidrahtverbindung. Dabei wird FORTH-Quelltext zwischen den Rechnern asynchron mit 9600 Baud übertragen. Der Kommunikationsablauf und die Reaktionsweise von FORTH darauf wurde dargestellt.

Herr Dahm stellte eine bildverarbeitende Anwendung vor. Ein Forschungsschwerpunkt des Lehrstuhls für Meßtechnik (LfM) an der RWTH Aachen ist PACS - Picture Archieving and Communication System. Als Teil davon wurde am LfM eine dezentrale Bildverarbeitungs-Workstation entwickelt, die auf dem VME-Bus aufbaut. Die bildbezogene Kommunikation zwischen den Stationen läuft über das industriell verfügbare Glasfaser-Netzwerk ImageNet. Als Betriebssystem wurde ein 32-Bit FORTH erstellt. Der FORTH-Kern wurde ergänzt um die Verwaltung des lokalen Speichers für Bilder, eine graphische Benutzeroberfläche, sowie die Verwaltung des virtuellen Massenspeichers über ImageNet. Dazu wurde ein Bildverarbeitungspaket zur Unterstützung einer schnellen Implementation und Testung bildverarbeitender Algorithmen eingebunden, das von FORTH aus aufgerufen werden kann.

Workstation mit IMAGE FORTH

Diese Workstation mit IMAGE FORTH wurde im Vorraum vorgeführt und konnte ausprobiert werden. Es wurde die Übermittlung und Bearbeitung von Röntgenbildern des Menschen gezeigt, wie sie zur Zeit zusammen mit dem Klinikum Aachen erprobt wird. (Diese Station hat mich übrigens besonders begeistert. Die Bilder kamen ohne erkennbare Aufbau- oder Wartezeiten auf den Schirm, eine Funktion "Lupe" lief im Bild ohne sichtbare Verzögerung, die ganze Handhabung der Bilder war mittels Maus simpel und ohne große Erklärungen ersichtlich. Das wird die Kommunikation im Krankenhaus beleben! Sollte sich jemand über dieses Lob mitten im Bericht wundern ... ich bin Arzt.)

Über ein anderes Netzwerk berichtete Herr Schnitter aus dem Beschleunigerlabor der Universität und der Technischen Universität München. Dort in Garching wird der Beschleuniger mit FORTH über ein verteiltes System von Rechnern gefahren. Von mehreren PC Zenith 386 Stationen aus wird per Maus und je sechs Winkelschrittgebern über ein offenes Netz mit kleinen Rechnern kommuniziert, an die die Aggregate des Beschleunigers angeschlossen sind. Die Rechner bestehen aus ECB-Baugruppen, die je nach Bedarf zusammengestellt sind aus A/D und D/A-Wandler oder parallelen, IEEE 488 oder V.24 Schnittstellen, FSBC Z280-Prozessoren und den ARCNET-Prozessoren. Die Baugruppen werden in FORTH programmiert. Die Verbindung wird durch das ARCNET hergestellt. Über dieses Netzt wird in FORTH-Quelltext kommuniziert - Request, Respond, Alarm, Download sind die Botschaften. Verwendet wird Open Network FORTH (ONF), eine Entwicklung des Labors mit dem Metacompiler und dem UR/FORTH von LMI.

Hartmut Pfüller zeigte mit dem comFORTH weitere Wege, wie über Cross Compiler ganz unterschiedliche Prozessoren als preiswerte Rechner in Netzen programmiert werden können. Das System wurde entwickelt an der Wilhelm-Pieck-Universität Rostock, Sektion Technische Elektronik, Wissenschaftsbereich Automatische Steuerung. Dort wurde ein FORTH vom Kern her neu gebaut nach gründlicher Analyse der gewünschten Primitives und dann mit den nötigen Werkzeugen der Programmierung versehen wie eingebundener Assembler, Editor, Rückübersetzer, Tracer und Cross Compiler. Interessant die Einführung weiterer Modi im Cross Compiler wie Controlmode, Crossmode, Makromode. Die Portabilität auf verschiedene Betriebssysteme und Hardware-Konfigurationen hatte von vornherein eine hohe Priorität.

Früher oder später wünscht sich jeder Programmierer solch eine komfortable Umgebung. Es wurden zwei weitere Entwicklungswerkzeuge vorgestellt: SwissFORTH, eine Entwicklung der Firma Flesch für das "Rapid Prototyping" und Projektmanagement und dann das "Evolution Development System for microprocessor applications" (EDS) der Firma Prinzen.

Zu diesem Thema gehörte auch der Vortrag von Fiedler, Lehr und Forschungsgebiet für Prozeßdatenverarbeitung, über Objekt-orientierte Programmierung unter FORTH. Unter Verwendung der Wörterbuchstruktur eines figFORTH zeigte er eine elegante Möglichkeit auf, abstrakte Daten zu definieren. Daten und Funktionen werden zusammengefaßt. Die Daten sind nur den ihnen zugeordneten Funktionen zugänglich.

Eine Reihe praktischer Anwendungen

Des weiteren wurden eine ganze Reihe praktischer Anwendungen für PCs geschildert. Ich will hierauf nicht mehr näher eingehen. Der interessierte Leser findet weiter unten die komplette Liste der Themen und Autoren.

Leider war es mir nicht mehr vergönnt am Sonntagnachmittag die Demonstration von Schleisiek-Kern zu erleben. Er hatte das "Dingsbums" mitgebracht, welches in USA beim Wettbewerb zum schnellsten Programmierer als Objekt gedient hatte. Zwar wurde mir erzählt, was es damit auf sich hat, doch will ich das hier nicht verraten. Vielleicht haben Sie ja mal eine Gelegenheit, es selbst zu besichtigen. Zum Beispiel auf der EuroFORML-Konferenz in diesem Jahr in Forth.

Unserem Mitglied Rolf Kretzschmar gebührt der besondere Dank des Vereins - er war der Motor für die Organisation der Tagung. Er hat erreicht, daß in diesem Jahr die Tagung in Zusammenarbeit mit Prof. Dr. Ing. Pleßmann veranstaltet werden konnte, dem Leiter im Lehr- und Forschungsgebiet für Verfahren der Prozeßdatenverarbeitung und Prozeßführung an der RWTH. Dank der Hilfe von Prof. Pleßmann wurde von der RWTH im Karman-Auditorium ein Hörsaal für die

FORTH-Tagung 1989 in Aachen

Vorträge und die Versammlung zur Verfügung gestellt und dazu in den Vorräumen genügend Platz für die Aussteller. Der Dank gebührt auch allen Mitarbeitern des Institutes und der RWTH, die zum Gelingen der Tagung beigetragen haben und allen Ausstellern für ihre fesselnden Demonstrationen.

Das Veranstaltungsprogramm

- **Einführungsvortrag Einordnung der Programmiersprache FORTH**
  Prof.Pleßmann, Lehr- und Forschungsgebiet für Prozeßdatenverarbeitung, RWTH Aachen
- **Kommunikationsfähiges FORTH-System CHESS**
  H.Zöller, Lehr- und Forschungsgebiet für Prozeßdatenverarbeitung, RWTH Aachen
- **IMAGE FORTH - eine dezentrale Systemarchitektur für die digitale Bildverarbeitung**
  M.Dahm, Lehrstuhl für Meßtechnik, RWTH Aachen
- **ONF - Open Network FORTH**
  H.Schnitter, Beschleunigerlabor der Universität und der Technischen Universität München
- **Entwicklungssystem comFORTH für verteilte Systeme**
  Pfüller, Universität Rostock, DDR
- **RTX-2000, Real-Time Express Microcontroller**
  Fa.Brühl, Nürnberg; Fa.Flesch, Breisach; Fa.Harris, München; Fa.Paul, Leitershofen
- **Projektmanagment, Rapid Prototyping, SwissFORTH**
  Flesch, Breisach
- **EDS: Evolution Development System for microprocessor applications**
  Princen, Pijnburg, NL
- **Objekt-orientierte Programmierung unter FORTH**
  Fiedler, Lehr- und Forschungsgebiet für Prozeßdatenverarbeitung, RWTH Aachen
- **Postscript + FORTH = FORTHscript**
  Implementation auf dem RTX-2000
  Krinninger, München
- **Interaktive Hinweiszettel auf dem PC**
  Hoffmann, Delta-t, Hamburg
- **CAMP - Computer Aided Music Processing**
  Nieberle, TU Berlin
- **Buchhaltung, FORTH typisch einfach**
  Schleisiek, Aachen
- **Rechnen mit Dimensionierten Größen**
  Verarbeitung unterschiedlicher Maßsysteme
  Dr. J. Storjohann, BTM, Hamburg
- **Echtzeit-Signal-Verarbeitung mit dem RTX-2000**
  Neubert, Physikalisch - Technische Bundesanstalt, Berlin
- **FORTH im Unterricht**
  Kretzschmar, BBS Alsdorf
- **FORTH in den USA**
  Schleisiek-Kern, Delta-t, Hamburg

Ausstellende Firmen/Institute

- **RTX2000, MINIBEE**
  Brühl Elektronik, Hegelstr.10, D-8500 Nürnberg 10
- **Meßsystem MC-32**
  Dr. Schetter BMCIGmbh, Boschstr.10, D-8039 Puchheim
- **RTX2000, swissFORTH auf PCs**
  FORTH-SYSTEME Angelika Flesch, Kühnheimerstr.21, D-7814 Breisach
- **RTX2000-Demonstration**
  HARRIS SEMICONDUCTOR, Putzbrunnerstr.69, D-8000 München
- **IMAGE-FORTH**
  Lehrstuhl für Meßtechnik, RWTH, Templergraben 55, D-5100 Aachen
- **RTX2000**
  Ulrich Paul, Erlenweg 18, D-8901 Leitershofen

Autoren gesucht!

Im Rahmen der Podiumsdiskussion bei der Mitgliederversammlung an der RWTH Aachen wurden Ziele der FORTH-Gesellschaft e.V. erörtert. Veröffentlichungen von FORTH-Anwendungen auf breiter Basis in wissenschaftlichen Zeitschriften würden die Leistungsfähigkeit von FORTH zeigen und den Bekanntheitsgrad erhöhen. Es haben bereits einige Verlage Interesse signalisiert. Die Koordination erfolgt in Zusammenarbeit von der RWTH Aachen und der FORTH-Gesellschaft e.V.

Um abschätzen zu können, was die FORTH-Gesellschaft e.V. zu diesem Vorhaben beisteuern kann, bitten wir alle Interessierten sich mit dem FORTH-Büro in Verbindung zu setzen.

PS: Vom Tagungband 1989 existiert noch ein Restbestand. Bestellungen nimmt das FORTH-Büro in Unterschließheim gerne an.

Noch einmal: STACK

Frank Stüss, 6369 Schöneck 2
Jörg Staben, 4010 Hilden

Eines der Hauptprobleme im ersten Umgang mit der Programmiersprache FORTH ist die Art der Parameterübergabe an Prozeduren. Im Gegensatz zu den Sprachen der PASCAL-Familie werden in FORTH die Argumente für eine Prozedur implizit über den Stack übergeben. Die einzelnen Argumente tragen keine Namen, sondern werden über ihre Position auf dem Stack identifiziert. So wird z.B. bei TYPE das oberste Stackelement als Länge der auszugebenden Zeichenkette und das zweite Element als Startadresse der Zeichenkette interpretiert. Werden mehrere Parameter an ein Wort übergeben und diese innerhalb des Wortes noch weiter bearbeitet, so kommt es schnell zu kaum noch zu durchschauenden Stackmanipulationen.

Wie das folgende Beispiel aus einem Lehrbuch für Einsteiger zeigt, ist FORTH für diese SWAPeratoren nicht zu Unrecht berüchtigt (siehe Bild 1):

Durch diese Parameterübergabe beim Aufruf von VIELFACHE in dieser Form vorgesehen:

3 10 1 Vielfache

Bei dieser Reihenfolge der Argumentenübergabe wird kaum deutlich, daß es sich bei dem Wert DREI um den Faktor handelt, dessen Vielfache in den Grenzen von 1 bis 10 berechnet werden sollen. Eine solche Parameterübergabe zusammen mit ihrer Verarbeitung durch sieben (!) aufeinanderfolgende Stack-Operatoren hat sicherlich einen großen Anteil an der mangelnden Akzeptanz von FORTH und ist dazu in einer Problemsituation kaum noch beherrschbar. Ein erster Ansatz zur Verbesserung dieses zweifelsfrei "richtigen" Programmes liegt in einer anderen Organisation der Argumentenübergabe.

Quelltext Service

Vom Sprachgefühl her wird man Grenzen bevorzugt in der Reihenfolge STARTWERT ... ENDWERT übergeben und den Faktor direkt vor das aufzurufende Wort setzen:

1 10 3 mal

MAL ist sicherlich nicht das schönste Wort, aber die Routine ist von der Argumentenübergabe her sicherer einzusetzen. Auch wird mit dieser geänderten Reihenfolge die Anzahl der Stackoperatoren drastisch reduziert. Zusätzlich kann man im Quelltext auch deutlich machen, daß z.B. die DUPs inhaltlich eindeutig der Textausgabe zuzuordnen sind. Auch werden hier bereits Stack-Kommentare in jeder Zeile eingesetzt, um im Fehlerfall die Posi-

```
\ Vielfache I                                   S.98-101

: cls  full page ;

: Vielfache  ( Faktor Endwert Startwert -- )
   3 arguments cls

   swap rot
   dup ." Aha, Sie wollen Vielfache von " . ." haben."
   rot
   rot over
   DO
      over over dup
      cr . ." -faches: "
      * .
      1 +
      STOP? IF leave THEN
   LOOP cr ." Fertig! " cr drop drop ;
```

Bild 1

Stichworte

- **Parameterübergabe,**
- **Stackmanipuation**

Noch einmal: Stack

```
\ .s mit TOS rechts

: .s
    depth 0 = exit
     sp@ s0 @
    over - $1E umin
    bounds swap 2- ?DO
        I u?
    -2 +LOOP ." <tos " ;
```

Bild 2

tion der Parameter auf dem Stack vergleichen zu können, wobei sich ein .S bewährt, das die Stackwerte mit TOS nach rechts ausgibt (siehe Bild 2):

Dieses Wort .S ist deutlich verabscheuungswürdig, weil im Falle nur eines Stackelementes nichts ausgegeben wird (siehe Bild 3).

Eine wirkliche Entlastung gerade für den FORTH-Neuling oder für jemanden, der nur gelegentlich in FORTH programmiert, ist ein Wort, das die Stackmanipulationen sozusagen nach einer bildlichen Darstellung durchführt. Ein solches Wort STACK wurde bereits in [1] vorgestellt, so daß hier die Umsetzung in Assembler für das volksFORTH gezeigt wird (siehe Bild 4):

Mit diesem Wort STACK lassen sich die gängigsten Stackoperatoren nach [1] neu definieren (siehe Bild 5):

```
: drop    Stack A> ;
: dup     Stack A>AA ;
: swap    Stack AB>BA ;

: rot     Stack ABC>BCA ;
: 2swap   Stack ABCD>CDAB ;
```

Bild 5

Zugleich bieten diese Definitionen eine Möglichkeit, die korrekte Arbeitsweise von STACK zu prüfen. Beim Einsatz von STACK muß man zwar immer noch die benötigten Parameter innerhalb der DO..LOOP-Schleife kennen, aber über deren Positionierung braucht man sich nun keine Gedanken mehr zu machen.

```
\ Stack nach Parametern (stack fnk 23jan89

\ siehe Vierte Dim. Vol2/1 April 86


 label rret 2 allot    \ ein LABEL ist eh temporär

                \ deshalb nicht headerless

| Code (stack ( n1 ... nj -- ni ... nk )     \ run-time STACK
    R
rret    #) mov   \ R retten
    D push I ) C- mov
    dp #) W mov
    W R mov
    [[ A pop word stos C- dec 0= ?]

    I inc
    I ) C- mov
    A+ A+ xor I inc C C or
    0= not
?[  [[ byte lods A W mov
        W R i) D mov
        D push
        C- dec 0= ?] ]?
    rret #) R mov

    D pop Next end-code
\ char> STACK

| : char> ( --> n ) \ holt nächsten ASCII-char
    von input-stream
    >in @ 1 >in +! blk @ ?dup IF block ELSE tib THEN
    + c@ ;

 : stack \ bereitet Stack auf
    state @ 0 = exit
    compile (stack
    BEGIN char> bl - UNTIL
    0
    BEGIN 1+ char> ASCII > = UNTIL
    dup >r c, >in @ -1
    BEGIN 1+ char> bl = UNTIL
    c, >in !
    BEGIN char> dup bl -
    WHILE 31 and r@ swap - 2* c, REPEAT
    r> 2drop ; immediate restrict
```

Bild 4

```
\ Vielfache II          S.98-101              jrg 27feb89

: mal ( Startwert Endwert Faktor -- )
    3 arguments cls
    dup
    ." Aha, Sie wollen Vielfache von " . ." haben."

                                          \ start end fak
    swap rot
    under      \ fak start end start
    DO         dup cr . ." -faches: "     \ fak start
       2dup * .
        1+             \ fak start+1
       stop? IF leave THEN                \ sollte man in der Testphase
                                          \ immer drin haben !
    LOOP
    cr ." Fertig! " cr 2drop ;
```

Bild 3

Schlußendlich sieht das kleine Programm, das Vielfache einer gegebenen Zahl innerhalb zweier gegebener Grenzen berechnet, so aus (siehe Bild 6):

Frank Stüss, 6369 Schönbeck
Jörg Staben, 4010 Hilden

Bibliographie

[1] VD Vol2/1 April 1986

```
\ Vielfache III          S.98-101          jrg 27feb89

: mal ( Startwert Endwert Faktor -- )
   3 arguments cls

   dup ." Aha, Sie wollen Vielfache von " . ." haben."
                                    \ start end fak
   STACK
   ABC>CABA                         \ fak start end start
   DO
      dup
      cr . ." -faches: "            \ fak start
      2dup * .
      1+        \ fak start+1
      stop? IF leave THEN           \ sollte man in der Testphase
                                    \ immer drin haben !
   LOOP
   cr ." Fertig! " cr 2drop ;
```

Bild 6 vom Artikel 'Neues vom Stack'

Die FORTH-Box München stellt sich vor

von Christoph Krinninger

Seit 18. Mai 1989 ist die FORTH-Box München am Netz. Damit steht der FORTH-Gesellschaft e.V. nach langer Wartezeit wieder eine eigene Mailbox zur Verfügung. Dankenswerterweise hat die Fa. Flesch uns die Mailboxsoftware "PC-Board" gespendet, die auch in ihrer eigenen Mailbox zuverlässig ihren Dienst verrichtet.

Jetzt heißt es also das Sparschwein geschlachtet und endlich ein Modem gekauft. Für diejenigen, die schon telekommunikationsfähig sind, hier die allererste Einführung.

Die Mailbox arbeitet 24h am Tag mit 300/1200 Baud 8N1 und ist unter der Nummer 089/7259625 erreichbar.

Wenn man sich das erste Mal einloggt, so werden Vorname, Name und einige persönliche Angaben abgefragt, bevor das Hauptmenü erscheint.

Nach der Zeile "Main Board Command?" können Sie ihren ersten Befehl eingeben. Als erstes wird man die hinterlegten Nachrichten lesen wollen, dies geschieht mit der Eingabe "R" für READ. Darauf erscheint die Zeile "(H)elp, (1-47), Message Read Command?". Wenn sie jetzt "H" eingeben, so erhalten Sie den erläuternden Hilfstext zum READ-Befehl. Bitte machen Sie von dieser Online-Hilfe ausgiebig Gebrauch! Wollen Sie die Meldungen lesen, die neu sind, also nach ihrem letzten Besuch in der Mailbox dazugekommen sind, so geben sie stattdessen "S" für "Since" ein. Verkürzt können Sie auch vom Hauptmenü aus die Kombination "R S" eingeben. Der Rechner bietet Ihnen dann der Reihe nach alle Neuigkeiten an, frägt Sie aber auch an geeigneter Stelle, ob Sie nicht lieber abbrechen wollen. Machen Sie im Zweifelsfall auch hier von der Online-Hilfe Gebrauch. Wenn Sie eine eigene Meldung in der Box hinterlassen wollen, so geben Sie im Hauptmenü "E" für "Enter a Message" ein. Der Rechner möchte dann den Empfänger wissen, dies können alle sein (=ALL), eine spezielle Person (z.B. CHRISTOPH KRINNINGER), oder der SYSOP. Diese Meldung kann zunächst nur der Sender und der Empfänger lesen. Wenn die Meldung für die Allgemeinheit bestimmt ist, so muß (leider) noch eine Vorzensur durch den SYSOP erfolgen, bevor sie für alle lesbar ist. Anschließend möchte der Rechner noch einen Titel für diese Nachricht wissen, und schon befinden Sie sich im Online-Editor. Es handelt sich um einen völlig unkomplizierten Zeilen-Editor, der für die wichtigsten Bedürfnisse ausreichend sein dürfte. Wenn Ihre Nachricht zu Ende ist, so drücken Sie einfach zweimal die RETURN-Taste. Anschließend können Sie die Nachricht noch editieren oder mit "S" für "SAVE" endgültig auf Platte speichern.

Wenn Sie erfolgreich die ersten Schritte gemacht haben, so probieren Sie doch einmal die Befehle der Reihe nach durch. Wie schon erwähnt, können sie sich mit "H" jederzeit die nötigen Erläuterungen holen.

Interessant sind die Befehle "Bulletin Listing", "Download", "Upload" und "Join a Conference". Das Hauptmenü können Sie mit "X" für "eXpert" abschalten, dies ist besonders im 300 Baud Modus manchmal wünschenswert.

Wenn Sie sich dann genügend in der Box umgeschaut haben, so bleibt nur noch der Befehl "G" für "Goodbye".

Fraktale Berge

von Christoph Krinninger

Rechtzeitig zur nächsten Reisesaison möchte ich allen denjenigen, die vielleicht zuhause bleiben müssen, wenigstens die Illusion von Sonne, Meer und Inseln ermöglichen.

Das englische Wort 'Fractal' kommt vom Begriff "Fractional Dimension". Man kann mit diesen Fraktalen verschiedene geometrische Formen beschreiben, die nicht streng genommen ein-, zwei- oder dreidimensional sind. Um eine bessere Vorstellung zu bekommen, muß man sich nur eine Küstenlinie betrachten. Aus großer Entfernung, etwa aus dem Weltall, sieht sie vielleicht wie eine Gerade aus, kommt man näher, so sieht man Einzelheiten, wie Buchten und Halbinseln. Aus einem eindimensionalen Objekt wurde also ein zweidimensionales.

Wie kann man mit diesem Konzept eine künstliche Landschaft berechnen? Nehmen wir zuerst eine Gerade und ziehen den Mittelpunkt nach einer Seite weg. Anschließend halbieren wir die beiden Abschnitte und ziehen wieder die Mittelpunkte zur Seite. Sehr schnell erzeugt dieser rekursive Prozeß eine wandernde Linie. Besonders schön wird dieser Effekt, wenn man das Maß der Auslenkung auch immer in jeder Rekursionsrunde halbiert.

In dem vorgestellten Programm wird statt einer Linie ein zweidimensionales Feld bearbeitet. Jede Zelle enthält die Höhe des Gitterpunktes über oder unter Meereshöhe. Das Wort CALCULATE-SURFACE zerteilt das Feld in immer kleinere Quadrate, SET-HEIGHTS ermittelt die neuen Höhen für die Mittelpunkte der Seitenlinien und der Mitte des Quadrates und zerteilt anschließend das Quadrat in vier Unterquadrate. Wenn für jeden Gitterpunkt eine Höhe ermittelt wurde, korrigiert das Wort SEA-LEVEL alle negativen Höhen und das Objekt wird gezeichnet.

Quelltext Service

Um einen dreidimensionalen Eindruck der Berge zu bekommen, werden von hinten nach vorne einzelne Segmente des Feldes gezeichnet, nicht sichtbare Linien werden automatisch gelöscht. Diese Methode produziert eine überzeugende dreidimensionale Grafik, ohne Nachteile bei der Geschwindigkeit in Kauf nehmen zu müssen.

Das Programm wurde wie immer in volksFORTH auf dem Atari ST geschrieben. Die wesentlichen Grafikbefehle stammen aus dem GEM-VDI Paket, nähere Erläuterungen sind der einschlägigen Fachliteratur zu entnehmen.

Zu beachten ist noch folgendes: Das Wort COORDINATES hinterlegt auf dem Stack je nach Auflösung bis zu 140 (!) Parameter. Die normale Stacktiefe ist nicht auf so extensive Benutzung ausgelegt. Beim volksFORTH kann die Größe des Stacks mit dem Wort RELOCATE (in RELOCATE.SCR) festgelegt werden. Man gibt am besten folgendes ein: INCLUDE RELOCATE.SCR <return> R0 @ 500 RELOCATE <return> Die beiden Stacks werden somit auf eine Größe ausgedehnt, die auch ausgedehnteste Benutzung erlaubt.

Genauso muß das Punktearray für das VDI vergrößert werden. Ändern Sie in GEM\BASICS.SCR SCR#2 Zeile 2 den Ausdruck CREATE PTSIN &60 ALLOT in CREATE PTSIN &256 ALLOT um (siehe auch Leserbrief von Martin Josef Warken). Anschlieend muß das FORTH-System neu zusammengestellt werden.

Die im Sourcecode angegebene Auflösung ist doppelt so groß, wie auf den ausgedruckten Beispielen. Der Grund liegt in der begrenzten Aufnahmekapazität von GEM-Metafiles.

Der Sourcecode zu diesem Artikel kann über den Diskettenservice oder die neu eingerichtete Mailbox FBM in München bezogen werden.

Bibliografie:

- Fractal Landscapes, Phil Koopman, Jr. Dr. Dobb's Toolbook of FORTH, Vol. II S. 357ff
- Dynamische Systeme und Fraktale, Becker, Dörfler Vieweg 1988

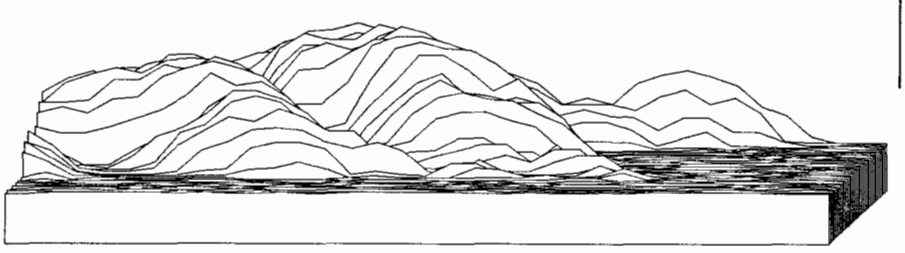


Bild 1

Stichworte

» **Fraktale,**
» **Pseudo 3D-Grafik**

Fraktale Berge

Screen# 0

```
\ Fractal Landscapes                    25may89 ck
```

Das Word FRACTAL kommt vom Begriff FRACTIONAL DIMENSION. Man kann damit verschiedene geometrische Formen beschreiben, die nicht streng ein-, zwei- oder dreidimensional sind. Interessanterweise können die meisten natürlich vorkommenden Objekte (wie z.B. Küstenlinien oder die Brownsche Molekülbewegung) mit fraktaler Geometrie beschrieben werden.

Hier also die FRAKTALEN BERGE.

Screen# 1

```
\ Loadscreen                            25may89 ck

Onlyforth gem also      \needs fillarea  2 loadfrom vdi.scr
Onlyforth gem also      \needs overwrite 8 loadfrom vdi.scr

Onlyforth gem also

\needs it       : it ;
forget it       : it ;

2 $F thru
```

Screen# 2

```
\ Zufallsgenerator                      25may89 ck

2Variable seed          $2B44463C. seed 2!

: rndf  ( -- n )
        [ seed 1+ ] Literal @ &434123 um* 1. d+ dup rot rot
        seed 2! ;

: +-    ( n1 n2 flag -- n4 )
        0< IF - ELSE + THEN ;

: cell*       2 * ;             : cell+       2+ ;

5 Constant #levels              &65  Constant size
&150 Constant ymax
```

Screen# 3

```
\ Fraktale Datenbank                    25may89 ck

Create square-p1        size size * cell* allot
size 1-         cell*  square-p1 + Constant square-p2
size size * 1-  cell*  square-p1 + Constant square-p3
size size 1- *  cell*  square-p1 + Constant square-p4

: ave   ( u1 u2 -- uave )
        0 swap 0 d+ 2 um/mod swap drop ;


8 Constant scale
```

Screen# 16

```
\                                       25may89 ck
```

Bibliographie:

Fractal Landscapes

von Phil Koopman, Jr.

Dr. Dobb's Toolbook of FORTH, Vol.II

S. 363ff

Screen# 17

```
\                                       25may89 ck
```

Aus dem VDI-Packet des volksFORTH werden die Ausgabe- und Attributfunktionen benötigt.

Kleiner Trick, um wiederholt compilieren zu können.

Screen# 18

```
\                                       25may89 ck
```

SEED Die oberen 16-bit enthalten die eigentliche Zufallsnummer.
RNDF Sehr einfacher Zufallsgenerator, erzeugt aber sehr interessante und runde Berge

+- Addition/Subtraktion abhängig von FLAG

CELL* Größe einer Speicherzelle, ggf. anpassen
CELL+
#LEVELS Anzahl der Rekursionen
SIZE Größe des Arrays (= 1 + 2^(#levels+1))
YMAX Maximale Berghöhe in Pixel

Screen# 19

```
\                                       25may89 ck
```

SQUARE-P1 2D Array, enthält die Höhen der Gitterpunkte
SQUARE-P2 Eckpunkte P1---X---P2
SQUARE-P3 | | |
SQUARE-P4   X---X---X
                         | | |
                         P4---X---P3

AVE Mittelwert

SCALE Horizontaler Vergrößerungsfaktor

Screen# 4

```
\ Initialisierung                              25may89 ck

: initialize-square     ( -- )
      square-p1  size size * cell*  $81 fill
      -60 square-p3 !          -60 square-p4 !
      -20 square-p1 !          -20 square-p2 ! ;
```

Screen# 5

```
\ Diverse Parameter für die Ausgabe            25may89 ck

Variable arrayoffset
Variable x-shift        Variable y-shift

&200 Constant y-screenoffset   &100 Constant x-screenoffset
x-screenoffset size scale * +       Constant x-offsetbase
y-screenoffset &30 +                Constant y-offsetbase

-1 Constant x-screenshift
 1 Constant y-screenshift

: x-shift+      ( x -- x )      x-shift @ + ;

: y-shift+      ( y -- y )      y-shift @ + ;
```

Screen# 6

```
\ Sockelberechnung für Pseudo 3D-Ausgabe       25may89 ck

: x-offset      ( -- x )
      x-screenoffset x-shift+ ;

: y-offset      ( -- y )
      y-screenoffset y-shift+ ;


: x-offbase     ( -- x )
      x-offsetbase x-shift+ ;

: y-offbase     ( -- y )
      y-offsetbase y-shift+ ;
```

Screen# 7

```
\ Koordinatenermittlung                        25may89 ck

: coordinates   ( -- x1 y1 .. xn yn n )
      x-offset   y-offset     ( first point )
      size 0 DO
            I scale * x-offset +   ( x )
            arrayoffset @
            I cell* + @ y-offset swap -   ( y )
      LOOP
      x-offbase   y-offset
      x-offbase   y-offbase
      x-offset    y-offbase
      x-offset    y-offset
      size 5 + ;
```

Screen# 20

```
\                                              25may89 ck

INITIALIZE-SQUARE       Array initialisieren

            Die Startwerte für die Eckpunkte bestimmen das
            Verhältnis Land/Wasser.
```

Screen# 21

```
\                                              25may89 ck

Diverse Hilfsvariablen und Konstanten
```

Screen# 22

```
\                                              25may89 ck

X-OFFSET        Ermittelt relative x-Koordinate für links oben

Y-OFFSET        Ermittelt relative y-Koordinate für links oben


X-OFFBASE       Ermittelt relative x-Koordinate für rechts unten

Y-OFFBASE       Ermittelt relative y-Koordinate für rechts unten
```

Screen# 23

```
\                                              25may89 ck

COORDINATES     Ermittelt die Koordinaten für einen Umriß


                                    Koordinaten für Sockel
```

Fraktale Berge

Screen# 8

```
\ Umriß zeichnen                                    25may89 ck

: draw-surface  ( -- )
      page
      x-shift off  y-shift off
      overwrite
      1 sl_width     0 sf_interior   0 sf_style
      square-p1
      size 0 DO
            dup arrayoffset !
            coordinates fillarea
            coordinates pline
            x-screenshift x-shift +!
            y-screenshift y-shift +!
            size cell* +
      LOOP drop ;
```

Screen# 9

```
\ Set height of a point for recursive processing     25may89 ck

: set-height    ( dh level px value py value -- dh level )
      rot + 2/ rot rot ave
      dup @ $8181 =
            IF swap 3 pick rndf +- swap !
            ELSE 2drop THEN ;

: set-heights   ( p1 p2 p3 p4 delta-h level# -- <unchanged> )
      stop? Abort" set-heights"
      ( ave p1/p2 ) 5 pick dup @ 6 pick dup @ set-height
      ( ave p2/p3 ) 4 pick dup @ 5 pick dup @ set-height
      ( ave p3/p4 ) 3 pick dup @ 4 pick dup @ set-height
      ( ave p1/p4 ) 5 pick dup @ 4 pick dup @ set-height
      ( ave p1/p3 ) 5 pick dup @ 5 pick dup @ set-height ;
```

Screen# 10

```
\ words to set up parameters for sub-squares 1-2     09sep88 ck

: square1       ( p1 p2 p3 p4 delta-h level# -- <2.copies> )
      5 pick dup 6 pick ave
      over 6 pick ave 8 pick 6 pick ave
      5 pick 2/ 5 pick 1- ;

: square2       ( p1 p2 p3 p4 delta-h level# -- <2.copies> )
      5 pick 5 pick ave 5 pick
      dup 6 pick ave over 6 pick ave
      5 pick 2/ 5 pick 1- ;
```

Screen# 11

```
\ words to set up parameters for sub-squares 3-4     09sep88 ck

: square3       ( p1 p2 p3 p4 delta-h level# -- <2.copies> )
      5 pick 4 pick ave 5 pick 5 pick ave
      5 pick dup 6 pick ave
      5 pick 2/ 5 pick 1- ;

: square4       ( p1 p2 p3 p4 delta-h level# -- <2.copies> )
      5 pick 3 pick ave 5 pick 4 pick ave
      5 pick 5 pick ave 5 pick
      5 pick 2/ 5 pick 1- ;
```

Screen# 24

```
\                                                    25may89 ck

DRAW-SURFACE    Löscht Hintergrund und zeichnet einen Umriß

                      Zeichenmodus
                      Strichstärke und Füllfarbe


                      Hintergrund löschen
                      Umriß zeichnen
```

Screen# 25

```
\                                                    25may89 ck

SET-HEIGHT      Ermittelt Höhe für einen Gitterpunkt

SET-HEIGHTS     Ermittelt die Höhen für alle "X"-Punkte, um
                Untergruppen zu bilden.
```

Screen# 26

```
\                                                    25may89 ck

SQUARE1         Unterquadrat 1 ermitteln

SQUARE2         Unterquadrat 2 ermitteln
```

Screen# 27

```
\                                                    25may89 ck

SQUARE3         Unterquadrat 3 ermitteln

SQUARE4         Unterquadrat 4 ermitteln
```

Fraktale Berge

Screen# 12

```
\ recursive procedure to set heights for random      20may89 ck
\ 3-d terrain

: calculate-surface     ( p1 p2 p3 p4 delta-h level# -- )
      recursive
      set-heights
      dup
      IF
            square1 calculate-surface
            square2 calculate-surface
            square3 calculate-surface
            square4 calculate-surface
      THEN
      2drop 2drop 2drop ;
```

Screen# 13

```
\ set sea-level for negative height points          24may89 ck

: sea-level     ( -- )
      cr ." Computing sea-level"
      square-p1 size 0 DO
            size 0 DO
                  dup @ dup 0<
                  IF
                        1 and over !
                  ELSE drop THEN
            cell+  LOOP
      LOOP drop ;
```

Screen# 14

```
\ new hills                                  25may89 ck

: new-heights   ( -- )
      cr ." Computing new heights"
      square-p1 square-p2 square-p3 square-p4
      ymax 2/ #levels calculate-surface ;


: hills ( -- )
      initialize-square
      new-heights
      sea-level ;
```

Screen# 15

```
\ master procedure to draw a random 3-d fractal      25may89 ck

: landscape     ( -- )
      page
      BEGIN
            seed 2@
            hills draw-surface
            cr ." Seed =" d.
      REPEAT ;
```

Screen# 28

```
\                                            25may89 ck

CALCULATE-SURFACE     Werte für gesamtes Feld neu berechnen
```

Screen# 29

```
\                                            25may89 ck

SEA-LEVEL       Alle Höhenpunkte, die unter dem "Meeresspiegel"
                liegen, werden korrigiert.


                Zufälliger Wert für "Meeresrauschen"
```

Screen# 30

```
\                                            25may89 ck

NEW-HEIGHTS     Neue Höhen berechnen


HILLS           Zentrale Routine für Neuberechnung
```

Screen# 31

```
                                             25may89 ck

LANDSCAPE       Eigentliches Programm

                Kann mit ESC gestoppt werden
```

Schneller Swopiler

Nachtrag zu meinem letzten Artikel: Swopiler - Ein Generator für Stackoperatorworte (VD Vol. V. Nr. 1, S. 21 ff)

von Andreas Findewirth

Ich bin von verschiedener Seite auf meinen Swopiler angesprochen worden, so daß ich hier kurz darauf antworten möchte. Rolf Kretzschmar verdanke ich den Hinweis, daß die von mir vorgeschlagene Notation der bereits 1986 von Doncil Hoekman (FORTH Dimensions, Vol. VII, No. 5, S. 25 'A Universal Stack Word') entwickelten Schreibweise entspricht. Auf den Gebrauch des Begriffes 'Swopperator' in Zusammenhang mit meinem Generator für Stackoperatorworte verzichte ich künftig, um nicht Verwechslungen mit anderen Konzepten heraufzubeschwören.

Quelltext Service

habe ich die Version 1.1 fertiggestellt, die eine ganze Reihe von Schwächen des ursprünglichen Programmes behebt. Der eigentliche Quelltext wurde von 22 Screens auf 15 Screens gestrafft, systematisiert und durchgehend mit Shadow-Screens kommentiert. Der Swopiler belegt jetzt außerdem etwa 30% weniger Platz im Hauptspeicher. Die Syntax wurde vereinfacht und verallgemeinert (siehe Abb. 1), der Swopiler akzeptiert nun auch Worte wie beispielsweise **?(A2=2A)**. Das Konzept des determinierten endlichen Automaten wurde beibehalten, jedoch erfolgt die Umsetzung der Zustände in FORTH-Worte jetzt eleganter und unter Verwendung "sprechender Namen".

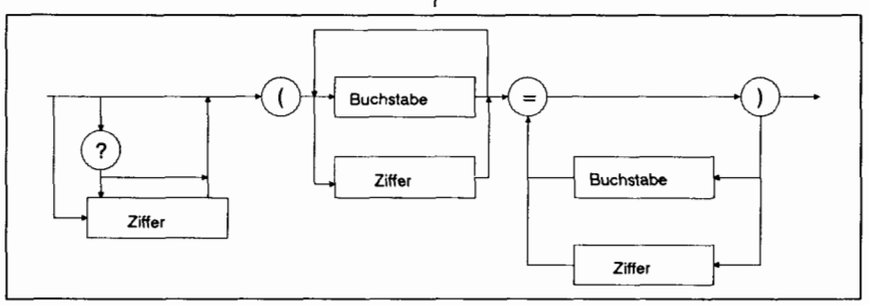


Abb1:Syntaxgraf für Stack-Operatornamen

Ich wies selbst bereits in meinem Artikel darauf hin, daß der veröffentlichte Quelltext aus Zeitgründen nicht ganz sauber ausprogrammiert wurde. Inzwischen

Die vom Swopiler erzeugten Stack-Operatoren bestehen jetzt aus regulärem High-Level-Code und können auf die übliche Weise 'getract' werden, um ihren Aufbau zu studieren und die korrekte Funktionsweise zu überprüfen. Ihre Ausführungsgeschwindigkeit wurde dabei gegenüber der Version 1.0 um den Faktor 5 - 10 verbessert (siehe Tabelle 1).

Aufgrund der stärkeren Systematisierung des Programmes können Änderungen und Erweiterungen leicht vorgenommen werde. So genügte die Anpassung von 4 Worten in 2 Screens, um eine Maschinensprachversion (1.11) für den Prozessor 68000 zu erstellen. Dies führt gegenüber der High-Level-Version zu einer weiteren Verbesserung der Ausführungsgeschwindigkeit um den Faktor 10 - 20. Die so erzeugten Stackoperatorworte sind nur noch um den Faktor 1,4 - 1,8 langsamer als die original volks-FORTH Operatoren, was darauf beruht, daß zum Zwischenspeichern nicht die Prozessorregister verwendet werden. Gegenüber regulären High-Level-Definitionen (siehe Beispiel **4DUP** in Tabelle 1) weisen die swopilierten Operatoren einen deutlichen Geschwindigkeitsvorteil auf. Eine weitere Variante (1.12) ermöglicht den Gebrauch unbekannter Stackoperatorworte beim Kompilieren neuer Worte, automatisch wird der notwendige High-Level-Code eingefügt. So kann beispielsweise die Definition

: SUM (ABC=CBA) + ;

korrekt übersetzt werden, ohne daß **(ABC=CBA)** vorher oder nachher im Dictionary vorhanden ist. Eine Kombination dieser Variante mit der Erzeugung schneller Worte (Version 1.11) befindet sich noch in Bearbeitung.

Stichworte

- **Stackoperatoren**
- **SWOPILER**

Bezeichnung des Stackoperators		Ausführungszeit in Mikrosekunden		
Standardname	Systemat. Name	Original volksFORTH	Swopiler 68000-Code	Swopiler High-Level-Code
DROP	(A=)	6,1	8,2	79,5
DUP	(A=AA)	6,6	10,7	151,3
OVER	(AB=ABA)	7,1	10,7	151,2
SWAP	(AB=BA)	8,7	12,8	294,8
2DROP	2(A=)	6,1	8,2	79,4 *
2DUP	2(A=AA)	7,7	13,3	157,4 *
2OVER	2(AB=ABA)	8,2	13,3	157,4 *
2SWAP	2(AB=BA)	11,3	20,5	313,3 *
NIP	(AB=B)	6,6	10,7	151,2
UNDER	(AB=BAB)	9,7	18,4	366,6
ROT	(12=21)	11,3	15,3	366,6
-ROT	(21=12)	11,3	15,3	366,6
4DUP	4(A=AA)	33,3 **	18,4	169,7 *
?DUP	?(A=AA) false	7,6	-	32,3
?DUP	?(A=AA) true	8,7	-	172,3

Alle Angaben wurden mit volksFORTH 83 rev. 3.80 auf Atari 1040 ST F ermitttelt. Die Ausführungszeiten wurden als Durchschnitt aus 100.000 Aufrufen ermittelt. Die Zeitmessung erfolgte mit einer Genauigkeit von 5 Millisekunden.

* Der Swopiler erzeugt 68000er Maschinencode nur mit 16-Bit-Transportbefehlen. Er übersetzt derzeit Mehrfachworte nur, wenn sie in einer geeigneten Notation vorgegeben werden, z.B.: nicht 2(AB=ABA) sondern (ABCD=ABCDAB).

** 4DUP wurde definiert als 2OVER 2OVER.

Tabelle 1

Den vollständigen Quelltext der verbesserten Version 1.1 habe ich dem FORTH-Kopier-Service (Hamburg) und dem Sourcecode-Disketten-Service (Mering) übersandt; ich hoffe es gibt beim Bezug keine Probleme. Die Varianten 1.11 und 1.12 können direkt bei mir angefordert werden.

Ein einfacher Textinterpreter der Rechnen und auch Formeln verarbeiten kann

von Peter Saupe

Einleitung

Bei der Arbeit mit Textverarbeitungsprogrammen hat es mich immer gestört, daß man zwar Zahlen eingeben, jedoch nicht direkt damit rechnen konnte. Das Problem hat mich sehr beschäftigt und am Ende zu dem hier wiedergegebenen Programmauszug geführt.

Das Prinzip ist denkbar einfach: nach der Eingabe einer Textzeile wird festgestellt, ob es sich um Text oder eine Berechnung handelt. Ist es Text, so wird die Eingabe unmittelbar fortgesetzt, ist es eine Berechnung, wird der Inhalt der Textzeile Wort für Wort geprüft, ob das Wort eine gültige Zahl, ein Rechenoperator oder der Text die variable Eingabe einer Zahl erwartet, die z.B. für die Berechnung einer Formel erforderlich ist. Daraus ergeben sich zwei Möglichkeiten des Programmablaufes: mit TC oder TEXTCALC kann zeilenweise entweder Text oder eine Berechnung verarbeitet werden, mit FC oder FORMELCALC wird eine Formel eingegeben, deren variabler Text durch die Eingabe einer Zahl ersetzt und die Formel dann berechnet wird.

Das Programm wurde mit UR/FORTH 1.0 von LMI geschrieben und läuft mit MS-DOS auf PC, XT, AT oder kompatiblen Rechnern und, bei Verwendung von UR/FORTH 1.1, auch unter OS/2.

Programm-Beschreibung

Bei der Eingabe mit INPUT-LINE, wird jedes einzelne Zeichen auf bestimmte Anforderungen getestet, bevor es in den LINE-BUF übergeben wird. Handelt es sich bei dem Zeichen um dezimal 27 - Escape - ist das Programmende angezeigt, bei einer Korrektur des Textes am Bildschirm mit Backspace - dezimal 8 - wird das auf dem Stack liegende Zeichen gelöscht und durch ein anderes ersetzt, mit dezimal 13 - Carriage Return - ist das Zeilenende erkannt und automatisch mit dezimal 10 - Linefeed - ergänzt.

Für die Adressierung der Ein- und Ausgabe von Zeichen gibt es zwei Möglichkeiten: (1) man verwendet den Stack, auf dem die Adresse abgelegt und auch dort manipuliert wird oder aber (2) eine Variable als Pointer. Bei dem hier wiedergegebenen Programm habe ich mich für die Pointer-Variable - als Adressierung an der "Oberfläche" - entschieden. Das bietet die Möglichkeit, den Programmablauf besser zu übersehen und zu kontrollieren.

LPTR dient dabei als Pointer zur Adressierung von LINE-BUF. Sie übernimmt am Beginn einer Zeile die Adresse von LINE-BUF und wird nach jeder Zeicheneingabe um 1 erhöht - bzw. bei Korrekturen mit Backspace um 1 vermindert. Die Pointer-Variable WPTR - weiter unten - arbeitet analog.

Quelltext Service

Zur Steuerung der einzelnen Programmschleifen dienen Flags. LINEFLAG signalisiert das Ende einer Eingabezeile, WRITEFLAG zeigt das Programmende an.

Nach Eingabe einer Textzeile prüft CHECK-LINE den Zeilenanfang nach folgendem Muster: ist das erste Zeichen der Zeile größer als dezimal 57 (ASCII 9) oder kleiner dezimal 91 (ASCII a), handelt es sich um Text, andernfalls um eine Berechnung.

```
: CALCULATION
   BEGIN WORT PUT-E
      CALCULATE
      IS-FORMEL?
      LINEFLAG @
   UNTIL ;
```

CALCULATION zerlegt dabei, in einer Schleife, die Zeile in die einzelnen Wörter und interpretiert WORT - nach Vervollständigung durch PUT-E - mit CALCULATE, ob es sich um eine Zahl, einen Operator oder um variablen Text handelt. Ist das letzte WORT aus dem LINE-BUF ausgelesen und interpretiert, beendet der Zustand von LINEFLAG die Schleife.

Stichworte

» Textinterpreter
» UR/FORTH

Da die in LINE-BUF stehenden Wörter - wie bei jedem normalen Text - durch Blanks voneinander getrennt und am Zeilenende durch CR abgeschlossen sind, werden diese auch als Delimiter verwendet und unterbrechen bzw. beenden das Lesen aus dem LINE-BUF. Ist mit der DO-WORT Schleife ein Blank - oder am Ende der Zeile ein CR - in LINE-BUF erreicht, wird die Kopie des Wortes in WORT-BUF abgeschlossen und die Länge des Strings an der ersten Position des WORT-BUF als Count-Byte abgelegt.

PUT-E beendet den String, für eine korrekte Umwandlung in eine Floating Point Number, mit einem ASCII E.

WORT-BUF steht jetzt der weiteren Verwendung, als ein "counted String", zur Verfügung.

Ergibt die Umwandlung des Strings eine Zahl, wird sie auf dem Stack abgelegt, ist es keine Zahl, sucht FIND im Dictionary nach einer Übereinstimmung mit dem in WORT-BUF enthaltenen counted String, um, wenn gefunden, dessen Code Field Adresse zu EXECUTEn, in diesem Fall eine Rechenoperation auszuführen. Ist die Suche von FIND erfolglos, dann handelt es sich um einen variablen Text, der - bei Verwendung von FORMELCALC - durch die Eingabe einer Zahl erledigt wird.

Eine Textzeile sieht danach folgendermaßen aus:

- entweder als normaler Text
  " Tante Agatha und ihr Hund"
- oder als Berechnung
  " 12 12 + 133 6 * + 2 /"
  mit Ausgabe des Ergebnisses
- oder als Formel mit FORMELCALC
  " länge breite * höhe *"
  mit der anschließenden Aufforderung, die entsprechenden Zahlen für den variablen Text einzugeben.

LPTR dient hier der Adressierung des Lesens in LINE-BUF, WPTR der des Schreibvorgangs in WORT-BUF; während WORTFLAG zur Steuerung der WORT-Schleife benutzt wir.

Die Form der Berechnung - wie sollte es bei FORTH anders sein - erfolgt in der Postfix Notation.

Ansonsten sollten die Screens, für weitere Erläuterungen, hinreichend kommentiert sein.

Zusammenfassung

Text ist eines der grundlegenden Werkzeuge bei der Arbeit mit dem Computer. Text der entweder so belassen wird, wie man ihn eingibt oder der mit entsprechenden Programmen übersetzt oder in anderer Form manipuliert wird. Der hier gezeigte Text/Kalkulator ist nur ein Beispiel für eine mögliche Interpretation oder Verarbeitung von Text.

Peter Saupe
Friedrichstraße 29
8000 München 40
Tel. 089/396581

Screen #0

```
\ SMALLTC1.SCR                    PS  10:45 02/25/89

   mit URFORTH 1.0 von
   Laboratory Microsystems Inc.

      Peter Saupe
      Friedrichstrasse 29
      D-8000 Muenchen 40
      Tel. 089/396581
```

Screen #1

```
\ Preliminarien                   PS  10:45 02/25/89

SFP  \ Load SFP.BIN oder auch I8087.BIN

256 EQU $L      \ Groesse von LINE-BUF
 32 EQU $W      \ Groesse von WORT-BUF

 VARIABLE LINE-BUF $L ALLOT    LINE-BUF $L BLANK
 VARIABLE WORT-BUF $W ALLOT    WORT-BUF $W BLANK
 VARIABLE WORTFLAG
 VARIABLE LINEFLAG
 VARIABLE WRITEFLAG
 VARIABLE FORMELFLAG
 VARIABLE LPTR
 VARIABLE WPTR
-->
```

Screen #2

```
\ Line, verschiedenes                    10:45 02/25/89

: LINE-BEGINS      ( --- ) LINE-BUF LPTR ! ;

: PUT-LINE-CHAR    ( c -- ) LPTR @ C! ;

: GET-LINE-CHAR    ( -- c ) LPTR @ C@ ;

: INCR-LPTR        ( --- ) 1 LPTR +! ;

: DECR-LPTR        ( --- ) -1 LPTR +! ;

: BEEP             7 EMIT ;
-->
```

Screen #3

```
\ INPUT-LINE                      PS  10:45 02/25/89

\ Eingabe einer Textzeile
: INPUT-LINE
  LINE-BEGINS             \ initialisiere Addr_Pointer
  LINEFLAG OFF            \ Zeilenflag
  BEGIN KEY DUP  27 =     \ Eingabe - ist es Escape ?
  IF DROP WRITEFLAG ON EXIT \ wenn ja, beende Programm
  ELSE DUP DUP EMIT 8 =   \ ist es Backspace fuer Korrektur ?
   IF DROP                \ wenn ja, drop Charakter
     32 EMIT 8 EMIT       \ bereinige Zeichen auf Monitor
     DECR-LPTR            \ dekrementiere Addr_Pointer
     LINE-BUF LPTR @ =    \ wenn Addresse = Addr_Pointer
     IF BEEP THEN         \ dann BEEP
-->
```

Screen #4

```
\ INPUT-LINE  cont.                    PS  10:45 02/25/89

   ELSE DUP 13 =          \ ist es Carriage Return ?
    IF PUT-LINE-CHAR      \ wenn ja, store CR,
      INCR-LPTR           \ inkrement Addr_Pointer und
      10 PUT-LINE-CHAR    \ store fuer Zeilenende ein LF,
      INCR-LPTR           \ inkrementiere Addr_Pointer
      CR  LINEFLAG ON     \ Zeilenende ist erreicht
    ELSE PUT-LINE-CHAR    \ andernfalls, store Charakter
      INCR-LPTR           \ und inkrement Addr_Pointer
  THEN THEN THEN
  LINEFLAG @              \ ist Zeilenende erreicht ?
  UNTIL ;                 \ wenn ja, beende die Schleife
-->
```

Screen #5

```
\ Wort - verschiedenes                 PS  10:45 02/25/89

: WORT-BEGINS    ( --- )  WORT-BUF 1 + WPTR ! ;

: PUT-WORT-CHAR   ( c -- ) WPTR @ C! ;

: INCR-WPTR      ( --- )  1 WPTR +! ;

: DECR-WPTR      ( --- ) -1 WPTR +! ;

: STORE-COUNT
  WORT-BUF WPTR @     \ hole Addressen,
  OVER -              \ berechne Count und
  SWAP C! ;           \ store als erstes Zeichen des Strings
-->
```

Screen #6

```
\ INPUT-WORT                           PS  10:45 02/25/89

\ Eingabe eines Wortes als counted String
: INPUT-WORT  ." = "
  WORT-BEGINS
  BEGIN KEY DUP DUP EMIT 8 =
  IF DROP 32 EMIT 8 EMIT
    DECR-WPTR
    WORT-BUF 1 + WPTR @ = IF BEEP THEN
  ELSE DUP 13 =
    IF DROP STORE-COUNT EXIT
    ELSE PUT-WORT-CHAR
      INCR-WPTR
  THEN THEN
  AGAIN ;
-->
```

Screen #7

```
\ .WORT IS-BLANK?                      PS  10:45 02/25/89

: .WORT              \ Textausgabe von WORT-BUF
  WORT-BUF COUNT 1-   \ reduziere Wortlaenge fuer ASCII E
  TYPE SPACE ;        \ um 1 und zeige es auf dem Monitor

: IS-BLANK?          \ Leerzeichenpruefung
  BEGIN GET-LINE-CHAR 32 = \ ist es ein Blank ?
  WHILE INCR-LPTR     \ wenn ja, ist es mehr als eins ?
  REPEAT ;
-->
```

Screen #8

```
\ DO-WORT                              PS  10:45 02/25/89

: DO-WORT
  GET-LINE-CHAR DUP              \ lies ein Zeichen
  INCR-LPTR  13 =                \ ist es ein CR ? Wenn ja,
  IF DROP                        \ Zeilenende ist erreicht
    STORE-COUNT                  \ dann ..
    WORTFLAG ON  LINEFLAG ON     \ und setze Flags
  ELSE DUP 32 = WORTFLAG @ NOT AND  \ andernfalls Wortende ?
    IF DROP                      \ wenn ja, dann
      STORE-COUNT                \ ...
      WORTFLAG ON                \ und setze Flag
    ELSE PUT-WORT-CHAR INCR-WPTR     \ andernfalls store Cha
  THEN THEN ;
-->
```

Screen #9

```
\ PUT-E                                PS 10:45 02/25/89

\ Fuer die Umwandlung eines Strings - mit FNUMBER? - in eine
\ Floating Point Number, ist es notwendig, das Wort mit
\ einem " E " zu beenden, also ASCII E als letztes Zeichen
\ in WORT-BUF zu speichern: 1234.321E

: PUT-E
  WORT-BUF COUNT  ( -- Addr Count )
  1-              \ reduziere Count um 1 und
  +               \ berechne absolute neue Addresse
  ASCII E SWAP C! ;  \ store E
-->
```

Screen #10

```
\ WORT                                 PS 10:45 02/25/89

: WORT                \ Initialisiere Wortanfang
  WORTFLAG OFF        \ Text von WORT beginnt mit Addr
  WORT-BEGINS         \ + 1, Addr + 0 enthaelt Countbyte
  IS-BLANK?
  BEGIN  DO-WORT      \ lies Zeichen fuer Zeichen,
       WORTFLAG @     \ bis Wortende erreicht.
  UNTIL ;
-->
```

Screen #11

```
\ Rechenoperatoren                     10:45 02/25/89
\ die von CALCULATE verwendet werden
\ kann entsprechend erweitert werden

: +E    F+ ;    \  +
: -E    F- ;    \  -
: *E    F* ;    \  *
: /E    F/ ;    \  /
: SINE  FSIN ;  \  SIN
: COSE  FCOS ;  \  COS
: TANE  FTAN ;  \  TAN
: LOGE  FLOG ;  \  LOG
: SQRTE FSQRT ; \  SQRT
: DUPE  FDUP ;  \  DUPliziere Zahl
-->
```

Screen #12

```
\ CALCULATE                              10:45 02/25/89

: CALCULATE
  FORMELFLAG OFF
  WORT-BUF COUNT STRPCK    \ ist der Counted String
  FNUMBER?                 \  eine gueltige Zahl ?
  IF                  \ wenn ja, OK
  ELSE FDROP               \ wenn keine Zahl, drop die Null
   WORT-BUF FIND           \ finde Code Field Addr von WORT,
   IF EXECUTE              \ wenn gefunden, wird sie EXECUTEd
   ELSE DROP               \ andernfalls ist Wort Variable
   FORMELFLAG ON
  THEN THEN ;
-->
```

Screen #13

```
\ J/N  IS-FORMEL?                        10:46 02/25/89

: J/N  ( -- flag )
  ." Formel beenden <J/N> "
  KEY DUP ASCII j = SWAP  \ ist es gleich j oder J fuer Ja
  ASCII J = OR ;

: IS-FORMEL?
  BEGIN FORMELFLAG @
  WHILE .WORT              \ Zeige das nicht zuinterpret. Wort
       INPUT-WORT          \ Gieb dafuer eine Zahl ein
       PUT-E CR            \ fuer Fnumber
       CALCULATE           \ Rechne
  REPEAT ;
-->
```

Screen #14

```
\ CALCULATION                            10:46 02/25/89

: CALCULATION
  LINEFLAG OFF             \ neue Zeile kann gelesen werden
  BEGIN WORT               \ lies Wort fuer Wort
       PUT-E               \ ASCII E fuer FNUMBER?
       CALCULATE           \ interpretiere das Wort und wenn
       IS-FORMEL?          \ nicht gefunden, variabler Text
       LINEFLAG @          \ bis die Zeile zu Ende ist
  UNTIL ;
-->
```

Screen #15

```
\ CHECK-LINE                             10:46 02/25/89

: CHECK-LINE
  LINE-BEGINS              \ Linepointer aktivieren
  GET-LINE-CHAR  13 =      \ 1. Zeichen der Zeile, ist es CR
  IF NOOP CR               \ es ist nur CR fuer eine Leerzeile
  ELSE IS-BLANK?           \ oder ein BLANK ?
     GET-LINE-CHAR         \ hole Zeichen nach Blank auf Stack
     DUP 57 > SWAP 91 < AND \ ist Zeichen Uppercase ?
     IF NOOP               \ wenn ja, tu nichts, es ist Text
     ELSE CALCULATION      \ andernfalls fuehre Berechnung aus
          F. CR            \ und zeige das Ergebnis
     THEN
  THEN ;
-->
```

Screen #16

```
\ TEXTCALC                               10:46 02/25/89

: INFO1 ." Bearbeitung mit <ESC> beenden !" ;
: INFO2 ." Das war's dann" ;
: STACK?  DEPTH 0 ?DO DROP LOOP ;

: TEXTCALC
  WRITEFLAG OFF
  CLS INFO1 CR
  BEGIN STACK?             \ Stack leer ?
       INPUT-LINE          \ Gib Textzeile ein.
       WRITEFLAG @ NOT     \ Solange nicht Null,
  WHILE CHECK-LINE         \ pruef Zeile
  REPEAT ;
: TC       TEXTCALC INFO2 CR ;
```

Screen #17

```
\ FORMELCALC                             10:46 02/25/89

: INFO3 ." Formelberechnung" ;

: FORMELCALC
  WRITEFLAG OFF
  CLS INFO3 CR
  ." Formel eingeben " CR
  ." ( zum Beispiel Pythagoras : a DUP * b DUP * + SQRT )" CR
  INPUT-LINE               \ Gib Textzeile ein.
  BEGIN LINE-BEGINS
       IS-BLANK? CALCULATION F. CR
       J/N CR              \ Beenden ?
  UNTIL ;
: FC       FORMELCALC INFO2 CR ;
```

Über das Wesen der UPN

von Jörg Plewe

Stichworte

- UPN
- Infixnotation

In der Ausgabe der VIERTEN DIMENSION vom Dezember 1988 wurde im Artikel 'Wie das Titelbild entstand' von Christoph Krinninger ein Infix-Parser von Dick Pountain vorgestellt. Mit diesem Parser ist es möglich, numerische Ausdrücke in FORTH statt in umgekehrt polnischer Notation (UPN) in der geläufigeren Infixnotation darzustellen. Dieser Artikel soll ein Anlaß sein, einmal über Sinn und Unsinn der UPN, die so viele FORTH-Neulinge abschreckt, nachzudenken.

Die klassische Infixnotation ist den Menschen in die Wiege gelegt worden. Lange bevor es Worte wie 'plus' gab, hat man Dinge zusammenzählen können. Der Sprachgebrauch hierbei erscheint uns natürlich:

"Eins und eins macht zwei!"

Als die Mathematik dazu überging, die Welt zu formalisieren, wurde daraus entsprechend ('entsprechend' enthält 'sprechen'!):

"1 + 1 = 2"

Dies ist ein einfacher Ausdruck, der von links nach rechts auszuwerten ist. Man bemerkte schnell, daß diese einfache Art der Darstellung nicht weit trug. Kompliziertere Ausdrücke mit verschiedenen Operationen, ließen sich so nicht schreiben. Um sich zu behelfen, führte man Klammern ein:

((3 + 4)*7)/(((1 + 1)*2) + (3*4))

Zur Vereinfachung stellt man weitere Regeln auf, die jeder Grundschüler heute lernen muß; wie z.B. "Punktrechnung geht vor Strichrechnung":

(3 + 4)*7/((1 + 1)*2 + 3*4)

Warum schlägt die einfache Auswertung von links fehl? Der klammerlos geschriebene Ausdruck hat kein 'Gedächtnis' und kann auch nicht nach vorn schauen, um zu sehen, was da noch kommt.

Der geklammerte Ausdruck wird nach Klammerebenen ausgewertet und das Ergebnis jeder berechneten Klammer muß, wenn auch nur kurzzeitig, für die weitere Verwendung gespeichert werden. Wie man sieht, ist die Auswertung, so natürlich sie uns auch erscheint, eine recht komplizierte Angelegenheit.

Gehen wir sie einmal im Geiste Schritt für Schritt durch: Wir addieren 3 und 4 und merken und uns das Ergebnis 7. Dieses multiplizieren wir mit 7 und merken uns 49. Nun addieren wir 1 und 1 und multiplizieren das Resultat mit 2 und addieren dazu noch das Ergebnis der Multiplikation von 3 und 4. Als letzten Schritt dividieren wir noch die am Anfang gewonnene 49 durch das letzte Ergebnis. Fertig!

So etwa dürfte der Ablauf in unserem Gehirn vor sich gehen. Haben Sie es gemerkt? Wir haben bei der Auswertung eigentlich die UPN bereits benutzt. Wir haben einen inneren Stapel zum Speichern unserer Zwischenergebnisse benutzt und immer zuerst die Operanden festgestellt und dann erst die Operation durchgeführt. Konsequent hingeschrieben haben wir also folgendes getan:

3 4 + 7 * 1 1 + 2 * 3 4 * /

Durch schrittweises Verfolgen von Gedankengängen haben wir einen typischen UPN-Ausdruck erhalten. Als nächstes erkennt man, daß hier keine Klammern mehr notwendig sind, und da wir auch die in der Grundschule gepaukten Operatorhierarchien vergessen dürfen. Damit ist alles aus der Welt geschafft, was künstlich und willkürlich festgesetzt werden mußte, um die durch den ursprünglichen Sprachgebrauch indizierte Infixnotation aufrecht erhalten zu können.

Die UPN ist damit vom mathematischen Standpunkt aus die natürlichere Schreibweise für Rechenausdrücke und ich glaube, sie kommt unseren inneren Denkstrukturen auch besser entgegen. Man sollte die UPN nicht als hinzunehmender Makel der Sprache FORTH auffassen und versuchen, sich mit künstlichen Hilfsmitteln, wie dem oben erwähnten Infixparser ins Gewohnte zu retten. Es mag am Anfang umständlich (weil ungewohnt) erscheinen, trägt aber auf lange Sicht Vorteile in sich, die zu einem leichteren Umsetzen von Gedachtem in Programmtext führen.

Ich denke alte FORTH-Hasen wissen, was ich meine ...

Der Kopierservice der FORTH-Gesellschaft e.V.

Das Archiv der FORTH-Gesellschaft befindet sich in Hamburg. Hier werden alle Publikationen erfaßt und gelagert, die uns von den Mitgliedern zur Verfügung gestellt werden. Auf dieses Archiv baut der

Kopierservice als Dienstleistung für die Mitglieder

auf. Wichtige Artikel werden von uns in einer Bestelliste geführt und können von den Mitgliedern bestellt werden. Außerdem versenden wir Inhaltsverzeichnisse von FORTH-Publikationen. Einzelne Artikel aus diesen Publikationen können bei uns bestellt werden.

Bestellungen werden **innerhalb einer Woche** bearbeitet. Die Seite kostet DM 0,30. Zur Verwaltungsvereinfachung bearbeiten wir nur Bestellungen, die per Vorauskasse bezahlt wurden. Je Bestellung wird eine Bearbeitungs-/Versandkostenpauschale von DM 3,- berechnet. Die Bezahlung erfolgt entweder durch Zusendung von Briefmarken oder durch Überweisung auf das Postgirokonto des Kopierservice:

FORTH-Gesellschaft e.V.
Sonderkonto K
Postgiroamt Hamburg
Nr. 5226 46 - 203
BLZ 200 100 20

Anschrift:
Klaus Schleisiek
Uhlenhorster Weg 3
2000 Hamburg 76

Im folgenden der erste Bestellzettel des Kopierservice. Bei Bestellungen bitte neben der gewünschten Position die Anzahl der Seiten eintragen, am Ende die Gesamtanzahl der Seiten zusammenzählen, mit DM 0,30 multiplizieren, DM 3,00 Versandkosten hinzuzählen und auf das Kopierkonto überweisen. Bei Überweisung von einem Girokonto kann der Bestellzettel unmittelbar an den Überweisungsauftrag angeheftet werden. Sonst bitte Überweisungsbeleg - oder einfach den Betrag in Briefmarken beilegen.

Sollten Sie wichtige Artikel, Zeitschriftenausschnitte etc. haben, die auch für andere Mitglieder interessant sein könnten, so schicken Sie uns diese bitte zu. Wir werden sie ins Archiv und den Kopierservice aufnehmen.

BES#	TITEL	QUELLE	AUTOR	SEIT.
LS001	Forth auf F65F11	MC 1 '85	R.Zech	5
LS002	Multitasking in Forth	Micro 9 '84	Butterfield	6
LS003	Structure Trees	Micro 9 '84	M.Dougherty	3
LS004	Faster Forth	Byte 6 '84	R.L.Greene	7
LS005	Forth-83 Evolution	Byte 8 '84	C.K.McCabe	10
LS006	polyFORTH/pcFORTH Review	Byte 11 '84	E.Tello	9
LS007	Roboter programmiert seine Bahnkurven selbst	Elektronik 22 '84	H.Weidner	4
LS008	Forth Computer	Wireless-World '83	B.Woodroffe	26
LS009	Forth Language	Wireless-World '83	B.Woodroffe	8
LS010	Silizium-Software F65F11	Elek-Indust. '85	C.Streicher	2
LS011	FORTH	Elektronik-Industrie '85	G.Meyer	5

BES#	TITEL	SYSTEM	AUTOR	SEIT.
FK001	Fast Array Indexing	6502	C.Springer	4
FK002	Forward Referencing	MVP	R.Koluvek	5
FK003	Spooler/Multitasker	FIG 8080	FTS	1
FK004	DIGIT Contest	FIG	Or. County	1
FK005	Sorting WORDS	F83	W.Baden	10
FK006	VIC-20 Terminal Emulation	FIG	D.E.Legan	2
FK007	CP/M in Forth	Forth-83	K.Schleisiek	5
FK008	GUARD control structures	FIG	C.Springer	12
FK009	FMODEM	FIG	Z.Thomas	9
FK010	Modem Program	F83	D.Dillon	3
FK011	MODEM7	FIG	E.Ramm	5
FK012	XMODEM Protocol	PC/FORTH	R.Taylor	9
FK013	CASE	polyFORTH	W.Baden	3
FK014	nützliche Utilities	FIG / Forth-83	W.Baden	4
FK015	Number I/O	FIG	T.Almy	1
FK016	UPPER case/Parameters	FIG	D.Doudna	1
FK017	Ascii Printer Graphic	FIG	W.Baden	2
FK018	Turtle Graphic	FIG	J.W.Brown	8
FK019	Converting FIG - Forth-83	FIG	R.Duncan	7
FK020	Disassembler 6502	volks4th	G.Rehfeld	2
FK021	Cordic	F83	A.T.Furman	2
FK022	64 Bit Arithmetic	Forth-83	D.A.Beers	5
FK023	Floating Point	Forth-79	M.Jesch	4
FK024	C.Moore's BASIC Compiler	FIG	M.Perry	4
FK025	8080 Assembler	FIG	J.J.Cassady	2
FK026	High Level Interrupts		R.L.Keck	2
FK027	Stringstack	volks4th	K.Schleisiek	5
FK028	Forth Inc. Line Editor	FIG	S.H.Daniel	9
FK029	Forth Database Design		G.B.Haydon	8
FK030	ISAM	F83	M.C.Stolowitz	4
FK031	Indexer for Data Base Model		R.N.Watkins	7
FK032	Userstack	8080	P.H.Helmers	3
FK033	Stack Diagram Utility	FIG	B.A.Cole	10
FK034	New Syntax for Def. Words	FIG	B.Ragsdale	8
FK035	Forth Gedichtgenerator	FIG	B.Ragsdale	1
FK036	Quick Text Formatter QTF	Forth-79	L.Brodie	12
FK037	QTF Extensions	Forth-79	R.Koluvek	5
FK038	Database Field Definitions	Forth-79	E.W.Fittery	5
FK039	Quicksort	FIG	W.Baden	2
FK040	SELECT ORDER PERFORM	Forth-79	W.Baden	4
FK041	68000 Assembler	F83	M.A.Perry	10
FK042	+LOAD +THRU Shadow -Screens	Forth-83	K.Schleisiek	1
FK043	ZEN Floating Point	F83	M.Tracy	1
FK044	Forth Slide Rule (Float.)	Forth-79	N.Grossman	7
FK045	Floating Point Standard Words		M.Tracy	13
FK046	High Level Adress Interpreter	F83	L.Craymer	6
FK047	Names for record fields	Forth-83	W.Carpenter	7
FK048	Report of Forth Non-Stand. Team		C.Curley	10
FK049	ENCLOSE MATCH for 6301	FIG		2
FK050	Nonce Defining Words	F83	W.Baden	6
FK051	8087 Floating Point	F83	S.Pollack	17
FK052	Simple Metacompiler	FIG	G.M.Kelly	9
FK053	Naming Conventions		K.Harris	4
FK054	Ackermann Funktion	FIG	U.Hoffmann	1
FK055	Simple Expert System	FIG	J.J.Cassady	9

Inhalsverzeichnisse von FORTH-Publikationen

BES#	PUBLIKATION	SEITEN
I01	Journal of Forth Application and Research	15
I02	Rochester Conference Proceedings	20
I03	Forth Dimensions, alle Ausgaben	50

Die Pals der RTX 2000-Mini-BEE

von Ulrich Paul

Mit der Beschreibung der Logikgleichungen der drei PALs auf der Karte wird die Reihe über das RTX-2000 System fortgesetzt. Gleichzeitig geht die Schaltung ins Public-Domain über, sie darf also zu nicht kommerziellen Zwecken nachgebaut werden.

Betrachten wir zuerst den einfachsten Baustein, den RTXMDEC. Er besorgt die Dekodierung der Adreßsignale in Abhängigkeit von PCLK, R/W und !DESELECT. Zur Notation ist zu bemerken, daß ein Ausrufezeichen vor einem Namen angibt, daß das betreffende Signal als aktiv low zu behandeln ist. Die einzige Besonderheit in diesem Chip ist die Selektion der RAM-Bank0 und des EPROMs. Solange BOOT aktiv ist, geht jeder Lesezugriff in Bank0 auf das EPROM und jedes Schreiben ins RAM.

Quelltext Service

Im RTXMTIM ist noch eine fehlende Gleichung des RTXMDEC enthalten, da diese im anderen Chip keinen Platz mehr hatte. Der RESET des RTX wird ebenfalls hier erzeugt. Zum größten Teil ist jedoch ein Automat in dieses Bauteil eingebaut, auf Neudeutsch: eine State-Maschine. Sie erzeugt den ICLK des Prozessors unter Beachtung einiger Eingänge. Mit BOOT wird grundsätzlich auf die niedrigste Taktrate umgeschaltet, d.h. es werden alle Zustände von der State-Maschine durchlaufen. In Abhängigkeit von ASYM und PLUS_ON wird im Zustand S4 mehr oder weniger weit zurückverzweigt, die Anzahl der nötigen Takte, um wieder nach S4 zu kommen, variiert also. In S6, dem letzten Zustand vor der Ausgabe von ICLK, wird der HALT-Eingang abgefragt und solange in dieser Abfrageschleife verblieben, wie dieser Pin aktiv ist. Man kann damit jeden ICLK-Puls gezielt verzögern und WAIT-States in Inkrementen von 25ns erzeugen. Allerdings werden die internen Timer des RTX dadurch auch beeinflußt.

Der letzte im Bunde ist der RTXMINIO, der die Mini-Schnittstelle des Boards darstellt. Der Pin UB_MODE bewirkt eine Invertierung der Signale an der RS232 Seite, um mit und ohne folgende Treiber (z.B. MAX232) arbeiten zu können. Die Schnittstelle belegt, kompatibel zum Harris-Minimal-System, die Datenbits D0 und D1 der ASIC-Bus-Adresse 7. Ein Interrrupt wird vom Chip bei jedem Wechsel der Empfangsdaten von logisch 0 auf 1 ausgelöst (wenn die Schnittstelle durch MINIDIS enabled wurde), bzw. das Signal vom EI3I-Eingang nur durchgeschaltet (wenn disabled).

In der Bedienungsroutine für die Schnittstelle wird dieser Interrrupt erlaubt, wenn auf ein Startbit gewartet wird. Nach Empfangen von diesem werden die weiteren Bits

```
Date      20.10.88  ;
Name      rtxminio  ;
Device    g16v8     ;
Assembly rtxcpucard     ;
$INCLUDE      head2.txt

/*
Dieses Pld stellt die RS232 Schnittstelle des Minimalsystemes dar. Sie ist mit
auf der CPU-Karte untergebracht. Durch zwei Pins kann ihre Funktion beeinflusst
werden: MINI_DIS schaltet das Port generell ab; UB_MODE waehlt den Modus ohne
RS232-Treiber, die Pegel muessen dann invertiert werden.

*/

/** Inputs **/

Pin  1 =  CLK       ;
Pin  2 =  EI3I      ; /* EI3 from outside of card                    */
Pin  3 =  !GIO      ; /* GIO von RTX                                 */
Pin  4 =  GRW       ; /* GRW con RTX                                 */
Pin  [5..7] =  [GA0..2] ; /* GA0..GA2 von RTX                        */
Pin  8 =  !UB_MODE      ; /* Wenn auf 0V selektiert den Modus ohne Treiber*/
Pin  9 =  !MINIDIS  ; /* Disabled das gesamte Port, ist dann frei    */
Pin 11 =  !OE       ;
Pin 12 =  CTS       ; /* Eingang von CTS                             */
Pin 15 =  RXD       ; /* Eingang der seriellen Daten                 */


/** Outputs **/

Pin 13 =  RTS       ; /* Ausgang von RTS                             */
Pin 14 =  TXD       ; /* Ausgang der seriellen Daten                 */
Pin 16 =  EI3       ; /* EI3 zu RTX, Interrupt, wenn RXD toggelt     */
Pin 19 =  STROBE_IT     ; /* Mit Pin 1 zu verbinden, um Daten zu latchen*/

/** Bidirectional Pins **/

Pin 18 =  GD1       ; /* GD1 von RTX                                 */
Pin 17 =  GD0       ; /* GD0 von RTX                                 */

/** Declarations and Intermediate Variable Definitions **/

read  =  !MINIDIS & GA0 & GA1 & GA2 & GIO &  GRW ;
write =  !MINIDIS & GA0 & GA1 & GA2 & GIO & !GRW ;


/** Logic Equations **/

EI3   =  !MINIDIS & ( !RXD & !UB_MODE # RXD & UB_MODE )
      #  MINIDIS & EI3I ;

TXD.d     = ( GD0 & !UB_MODE # !GD0 & UB_MODE ) ;
RTS.d     = ( GD1 & !UB_MODE # !GD1 & UB_MODE ) ;

STROBE_IT     = !write ;

GD0 = ( CTS & !UB_MODE # !CTS & UB_MODE ) ;
GD1 = ( RXD & !UB_MODE # !RXD & UB_MODE ) ;
GD0.oe    = read ;
GD1.oe    = read ;

/* Stock for useful lines */

$IFDEF NEVER /* is hopefully NEVER defined */

Pin 14 = ; /* */
Pin 15 = ; /* */
Pin 16 = ; /* */
Pin 17 = ; /* */


$ENDIF
```

LISTING RTXMINIO.PLD

Die Pals der RTX 2000-Mini-BEE

```
Name     rtxmtim;
Partno   88072;
Date 15.09.88;
Revision  02;
Designer  U.Paul;
Company THUP-Systems;
Assembly RTX-CPU-Card;
Location  IC2;
Device    p16r4;

/*****************************************************************************/
/*
     Main-Timing-Chip for RTX 2000 CPU-Card
     This chip supplies all the clock signals for the CPU. It gets
     40MHz from a crystal oscillator at its CLK-pin. The output is
     a 1:2 duty-cycle clock with 6,666MHz (BOOT low) or a 1:4 duty-
     cycle clock with 4,444MHz (BOOT high). This allows 120ns EPROMs
     to be used for bootstrapping without wait-states. The content
     of the ROMs is to be copied into RAM after RESET for faster
     execution.
                                                                  */
/*****************************************************************************/
/*  Allowable Target Device Types:  PAL16R4D (10ns!)                 */
/*****************************************************************************/

/** Inputs **/

Pin  1 =  CLK       ; /* Clock from 40MHz oscillator                 */
Pin  2 =  BOOT      ; /* BOOT input from RTX-chip (high after RESET) */
Pin  3 = !DESELECT  ; /* deselect input, disables on-card accesses   */
Pin  4 =  PCLK      ; /* PCLK from CPU, needed to add micro-wait-states */
Pin  5 =  HALT      ; /* halts ICLK at the end of PCLK-low until released */
Pin  6 =  PLUS_ON; /* adds 25ns to both cycles see End of file      */
Pin  7 =  ASYM      ; /* adds 25ns to PCLK low cycle                 */
Pin  8 = !RESETL    ; /* external RESET input, active low            */
Pin  9 =  RESETH    ; /* external RESET input, active high           */
Pin 11 = !OE        ; /* fixed OE, tied to GND externally            */

/** Outputs **/

Pin 12 = !ROMCS       ; /* CS for the EPROMS                         */
Pin 13 =  ICLK      ; /* ICLK to RTX-chip                            */
Pin 14 =  NC_A        ; /* do not connect!!                          */
Pin 15 =  NC_B        ; /* do not connect!!                          */
Pin 16 =  NC_C        ; /* do not connect!!                          */
Pin 17 =  LOSPEED     ; /* high, if low speed is selected            */
Pin 18 =  RTX_RESET   ; /* active high output to RTX                 */
Pin 19 = !HALT_STATE  ; /* active, when in HALT state                */

/** Declarations and Intermediate Variable Definitions **/

Field TIMING = [ NC_C, NC_B, NC_A ] ;

$DEFINE S0 'b'000
$DEFINE S1 'b'001
$DEFINE S2 'b'010
$DEFINE S3 'b'011
$DEFINE S4 'b'100
$DEFINE S5 'b'101
$DEFINE S6 'b'110
$DEFINE S7 'b'111

DO_HALT     =  !PCLK & HALT;
EXTRA_LOW   =  !PCLK & ASYM;

/** Logic Equations **/

ROMCS       = !DESELECT & BOOT ;

LOSPEED.D   =  RESETL # RESETH # BOOT ;

RTX_RESET   =  RESETL # RESETH ;

SEQUENCE TIMING {
    PRESENT S5
                                NEXT S0;
    PRESENT S0
                                NEXT S1;
    PRESENT S1
                                NEXT S3;
    PRESENT S3
                                NEXT S2;
    PRESENT S2
                                NEXT S6;
    PRESENT S6
        IF  HALT                NEXT S6;
        IF !HALT                NEXT S4;
        OUT HALT_STATE ;

    PRESENT S4
        IF  LOSPEED                         NEXT S5;
        IF  PCLK &  PLUS_ON & !LOSPEED                  NEXT S2;
        IF  PCLK & !PLUS_ON & !LOSPEED                  NEXT S6;
        IF !PCLK & !ASYM &  PLUS_ON & !LOSPEED          NEXT S2;
        IF !PCLK & !ASYM & !PLUS_ON & !LOSPEED          NEXT S6;
        IF !PCLK &  ASYM &  PLUS_ON & !LOSPEED          NEXT S3;
        IF !PCLK &  ASYM & !PLUS_ON & !LOSPEED          NEXT S2;
        OUT ICLK;

}
```

LISTING RTXMTIM.PLD

```
Name     rtxmdec;
Partno   88071;
Date 05.07.88;
Revision  01;
Designer  U.Paul;
Company THUP-Systems;
Assembly RTX-CPU-Card;
Location  IC1;
Device    p16l8;

/*****************************************************************/
/*                                                               */
/*  Adress and chip control PAL for RTX-CPU-Card                 */
/*  This chip synchronizes the access to the memory chips with   */
/*  PCLK and decodes the proper address. It is designed to work  */
/*  in a basic system with 2x32k word-wide RAM and 32k word-wide */
/*  ROM. Two chips of RAM/ROM are connected in parallel to get   */
/*  a 16 Bit word.                                               */
/*                                                               */
/*                                                               */
/*****************************************************************/
/*  Allowable Target Device Types: PAL16l8D (10ns!)              */
/*****************************************************************/

/** Inputs **/

Pin  1 =  PCLK        ; /* PCLK from RTX chip                    */
Pin  2 =  MA16        ; /* Memory adress line 16 from RTX        */
Pin  3 =  MA17        ; /*          17                           */
Pin  4 =  MA18        ; /*          18                           */
Pin  5 =  MA19        ; /*          19                           */
Pin  6 =  UDS      ; /* UDS (upper data strobe) from RTX         */
Pin  7 =  LDS      ; /* LDS (lower data strobe) from RTX         */
Pin  8 =  BOOT        ; /* BOOT from RTX                         */
Pin  9 =  MRW         ; /* MRW from RTX                          */
Pin 11 = !DESELECT    ; /* disables on-card memory RAM & ROM     */

/** Outputs **/

Pin 12 = !RAMLBWR     ; /* WR to low byte of the RAMs            */
Pin 13 = !RAMHBWR     ; /*      high                             */
Pin 14 = !RAMLBOE     ; /* OE to low byte of the RAMs            */
Pin 15 = !RAMHBOE     ; /*      high                             */
Pin 16 = !RAMB0CS     ; /* CS to lower 32k of RAM                */
Pin 17 = !RAMB1CS     ; /*      upper                            */
Pin 18 = !DBUFOE ; /* OE to the data buffers                     */
Pin 19 = !ROMOE       ; /* OE                                    */

/** Declarations and Intermediate Variable Definitions **/

FIELD ADR = [MA19..16];

VALACC      =   !PCLK & !DESELECT;

$DEFINE ONE 'B'1        /* Dummy to get a logical 1              */
$DEFINE ZERO 'B'0

/** Logic Equations **/

RAMLBWR     =   VALACC & !MRW & ADR:[00000..1FFFF] & LDS ;
RAMHBWR     =   VALACC & !MRW & ADR:[00000..1FFFF] & UDS ;

RAMLBOE     =   VALACC &  MRW & ADR:[00000..1FFFF] & LDS ;
RAMHBOE     =   VALACC &  MRW & ADR:[00000..1FFFF] & UDS ;

RAMB0CS     =   VALACC & ADR:[00000..0FFFF] & ( !BOOT # !MRW );
RAMB1CS     =   VALACC & ADR:[10000..1FFFF] ;

ROMOE       =   VALACC & ADR:[00000..0FFFF] & BOOT & MRW ;

DBUFOE      =   DESELECT
            #   !(ADR:[00000..1FFFF]) ;
```

LISTING RTXMDEC.PLD

eines Zeichens in Abhängigkeit von einem Timer abgetastet. Am Ende der 8 Bit und dem Stop-Bit wird wieder mit eingeschaltetem Interrupt auf den Start eines neuen Zeichens gewartet.

Abschließend noch eine Bemerkung zu den Konsequenzen des Überganges dieses Boards in den PD:

Es wird ab etwa dem 3.Quartal '89 die Version 2 geben, die einige Änderungen und Verbesserungen gegenüber der beschriebenen Version 1 aufweisen wird. Eine Aufwärtskompatibilität wird gewährleistet sein. Auf der anderen Seite werden ab sofort keine Peripheriekarten mehr für diese Version 1 von der Firma Ulrich Paul Elektronik verkauft, dafür ist nun ausschließlich die Firma Brühl Elektronik, Nürnberg zuständig. Bei Erscheinen der nächsten Version werden Kunden mit dem alten Board ein befristetes Angebot zum Aufrüsten auf den neuen Standard erhalten. Sie geben dabei ihr altes System zurück und erhalten gegen einen geringen Aufpreis das neue.

BÜCHERECKE

Jörg Staben

Mein Bücher-Beitrag in der VD 2/3 '88 ist, begünstigt durch den Wechsel in der Redaktion der VD, irrtümlich gedruckt worden. Dieser Beitrag gehörte in einen ganz anderen Sinnzusammenhang und sollte in dieser Form nie veröffentlicht werden. Als Rohkonzept sollte diese Liste von sehr subjektiven Buchkritiken interessierten Neumitgliedern eine erste Orientierung in der FORTH-Literatur ermöglichen.

So möchte ich hier meinen Beitrag korrigieren und ergänzen. Diese Korrektur beruht auf der inzwischen gewonnenen Erkenntnis, daß keine Programmiersprache vernünftig einzusetzen ist, wenn man die dahinterstehende Sprachphilosophie nicht kennt und akzeptiert.

So muß das Buch von A. Goppold, 'FORTH: Ein Programmiersystem ohne Grenzen', hervorgehoben werden, in dem Goppold auch auf die Methodik der FORTH-Programmierung eingeht und Grundsätzliches zur Programmierung in FORTH sagt, was in der Literatur sonst kaum zu finden ist. Diese Überlegungen halte ich inzwischen für viel wichtiger, als die Diskussion irgendwelcher Compiler-Interna. Auch seine Betrachtung des CREATE...DOES> Konstruktes weist die Richtung, in die sich die Diskussion über das Programmieren in FORTH entwickeln sollte.

Das gleiche gilt auch für L. Brodie, 'Denken in FORTH'. Hier werden die meisten Merkmale der FORTH-Programmierung zur Diskussion gestellt und die Hintergründe bestehender Regeln und Empfehlungen erläutert.

Leider zeigen solche Bücher ihre Qualitäten erst, wenn man schon einige Programmiererfahrungen in FORTH gesammelt hat.

Eine andere Problematik tritt bei einem recht neuen Buch auf, P. Kail, FORTH, Oldenburg Verlag. Dieses Buch kommt wie viele andere auch aus dem Englischen und wirkt wie ein Fragment übersetzt. Teilweise ohne Zusammenhang kommen durch die lieblose Übersetzung unfreiwillig komische Aussagen zustande: "Die Wörter 1+, 1-, ... 2*, 2/ sind etwas schneller auszuführen und werden etwas schneller verarbeitet." Diese Aussage ist als alleinstehender Merksatz ohne Hintergrundwissen kaum zu verstehen. "Hat man TASK im Dictionary kann man mit FORGET ein ganzes Programm löschen" vermittelt keinen Zusammenhang zwischen TASK und dem zu löschenden Programm und "Die hervorragende Eigenschaft eines Computers ist, daß er auf Informationen von der Umwelt hin reagieren kann. Die Grundstruktur für eine solche Funktion in FORTH ist IF...ELSE...THEN." als Merksatz spricht für sich. So lernt man die Übertragung der Bücher von Brodie ins Deutsche zu schätzen.

Fazit dieses Nachtrages ist, daß heute zum Jahreswechsel 1988/89 keine brauchbare Einführung in FORTH83 für Programmieranfänger zur Verfügung steht. Das Buch von R. Zech, FORTH 83, ist zwar ein gutes Arbeitsbuch und bietet auch eine knappe Einführung, verlangt aber einem Neu-Einsteiger wohl zuviel ab.

Ungeachtet dieses Mankos im deutschen Sprachraum, möchte ich noch einige englischsprachige Bücher mit ihren ISBNs nachtragen:

- ISBN 1-55851-010-0
  DR. DOBB'S TOOLBOOK OF FORTH, M. Ouverson, M&T Books,Vol. I
- ISBN 0-89303-660-9
  MASTERING FORTH, Anderson/Tracy Brady
- ISBN 0-471-88235-6
  COMPLETE FORTH, A. Winfield, John Wiley
- ISBN 0-13-326331-2
  FORTH, Kelly/Spies, Prentice-Hall
- ISBN 0-13-843079-9
  STARTING FORTH - 2nd ed., L. Brodie, Prentice-Hall
- ISBN 0-13-559957-1 Nov.88
  MASTERING FORTH, M. Tracy, Brady

Die nächste 'Vierte Dimension' erscheint im September '89.

FORTH-Bibliothek, Teil 3

Dies ist eine auszugsweise Übersicht der FORTH-Bibliothek der Münchner Gruppe. Bei Interesse an einem Artikel kann man sich an Christoph Krinninger wenden. Diese Serie wird in den folgenden 'Vierten Dimensionen' fortgesetzt werden. Diesesmal werden eine Reihe von Anwendungen vorgestellt. (verw. Abkürzungen: DDToF = Dr. Dobb's Toolbook of FORTH, DD Journal = Dr. Dobb's Journal)

'WITH'	Ein Wort, um alle Worte eines Vokabulars 'MIT' bestimmten Buchstaben auszudrucken	Andrew Waters	Firmenschrift MPE
A compiler for Programmable Logic in FORTH	Ein Compiler für PAL's	Michael Stolowitz	MMI
A Disk Operating System for FORTH	FORTHDOS is a powerful, efficient and easy-to-use DOS for single-user FORTH systems	Peter Reece	BYTE, 04/82
A FORTH Decompiler	Decompiler für MacForth	Jörg Langowski	Best of MacTutor, Vol. 1
A FORTH Lisp	Implementation von Lisp	Martin Tracy	DDToF, Vol. 2
A FORTH Spreadsheet	Eine Tabellenkalkulation in FORTH	Craig A. Lindley	DDToF, Vol. 2
A FORTH Standard Prelude	Einige Worte zur Vereinheitlichung	Martin Tracy	DDToF, Vol. 2
A Generic Application	Application template in Neon	Jörg Langowski	The Complete MacTutor, Vol. 2, S. 413
A LISP-Kernal for the NC4000	Implementation von LISP in FORTH-83	Ulrich Hoffmann	euroFORML '87
A proposal for strings in FORTH	String-Packet in fig-FORTH	Ralph Deane	DDToF
A relocating loader in FORTH	Schnelle Ladetechnik für FORTH-Code	Joe Barnhart	DDToF
A Text File Syntax for Screen File Users	Einige Worte zum Laden und Verwalten von Source-Code aus Textfiles	Stephen Pelc & Andrew Waters	euroFORML '88
ABUNDANCE	Artikel über eine public-domain Datenbank in FORTH	Roedy Green	BYTE, 10/86
Acquisition and Real-Time Display of Spectrophotometry	Artikel über ein Photospektrometer in FORTH	Martin V. Thomas	euroFORML '88
Adding Record Structures to FORTH	Records with local field names	Jörg Langowski	The Complete MacTutor, Vol. 2
Adreßkartei unter FORTH	Adreßverwaltung mit FORTH	Wolfgang Zweygart	65xx Micro Mag
An Augmented Transition Network Compiler and Sta	Artikel über ATN's und State-Machines in FORTH	Rod Crawford	euroFORML '88
An Edit Task for FORTH	Editor und viele nützliche Wörter für MACH2	Jörg Langowski	The Complete MacTutor, Vol. 2
An efficient Algorithm for Large Priority Queues	Artikel über queues	Robert Jay Brown	DD Journal, 6/87
Analyzing Large FORTH-Programs	Implementation des 'Structure Tool', um Programme besser analysieren zu können.	Kim R. Harris	DDToF, Vol. 2
Animated Hanoi Towers in NEON	Objekt-orientiertes Türme von Hanoi	Jörg Langowski	The Complete MacTutor, Vol.2, S.410
APD C-to-FORTH Translator	Datenblatt über ein C in FORTH		Firmenschrift
Applications of Multitasking	Artikel über Multitasking	Howard Oakford	euroFORML '88
Applications of the Harris RTX Processor in Medical Applications	Artikel über den Einsatz von FORTH bei der NMR-Tomografie	R.J. Marriott	euroFORML '88
Batch Text File Transfer by XMODEM	XModem Protokoll für Macintosh/MACH2	Jörg Langowski	The Complete MacTutor, Vol. 2
Bresenham Line-Drawing Algorithm	Gerade Linien in FORTH	Phil Koopman, Jr.	DDToF, Vol. 2
CASES	Und noch ein CASE in fig-FORTH		65xx Micro Mag
Charting FORTH	Bessere Lesbarkeit von FORTH-Programmen mit Flußdiagrammen	Wil Baden	DDToF, Vol. 2
Command interpreter for peripheral devices	Implementation eines Spezial-QUIT's	Klaus Schleisiek	euroFORML '87
Converting FORTH Blocks to ASCII Text	Block-Format zu ASCII und umgekehrt.	Jörg Langowski	Best of MacTutor, Vol. 1
Digitale Oszilloskopkamera	Kurzbericht über eine Diplomarbeit mit FORTH		

Gruppen

Lokale FORTH-Gruppen, die sich regelmäßig treffen:

1000 Berlin	Claus Vogt, Tel.: 030/2168938. Treffen am letzten Donnerstag des Monats um 19.30 Uhr in der Technischen Universität Berlin, Mathematikgebäude, 6.Stock im Raum MA 621
2000 Hamburg 13	Karsten Roederer, Tel. 040/4104446, tagsüber 412 329 84, Treffen im Geomatikum Raum 1438 (14. Stock), Bundesstr. 55, am 28. Juni, 27. September, 25. Oktober jeweils um 19.30 Uhr
4130 Moers 1	Friederich Prinz, näheres Tel: 02841/583 98
Rhein-Ruhr	Jörg Plewe, Tel: 0208/423514, Treffen nach Absprache. Der nächste Termin kann bei Jörg Plewe erreichbar unter obiger Telefonnummer erfragt werden.
6100 Darmstadt	Andreas Soeder, Tel. 06257/2744. Treffen an der VHS an einem Mittwoch in der Mitte des Monats.
6800 Mannheim	Lokale Gruppe Rhein-Neckar, Thomas Prinz, Tel.: 06271/2830. Treffen jeden ersten Mittwoch im Monat im Vereinslokal des Segelflugvereins Mannheim e.V. Flugplatz, Mannheim-Neuostheim.
8000 München	Heinz Schnitter, Tel. 089/3103385 und Christoph Krinninger 089/7259382. Treffen jeden 4. Mittwoch im Monat 19 Uhr 30 im Vereinsraum 1 im Bürgerhaus Unterschleißheim am Rathausplatz (S-Bahnhaltepunkt S1 Unterschleißheim).

FORTH-Fachgruppen:

8000 München	RTX 2000 Gruppe, Koordinator Max Diez, Treff- und Zeitpunkt wie oben bei der lokalen Münchner Gruppe.
6800 Mannheim	FIS (FORTH Integriertes System) - Datenbank, Textverarbeitung, Kalkulation, Postadresse: Dr. med. Elemer Teshmar, Danziger Baumgang 97, 6800 Mannheim 31

Es möchten in ihrer Region eine Gruppe gründen:

7000 Stuttgart 31	Wolf-Helge Neumann, Huttenstr. 27, Tel. 0711/882638.
8500 Nürnberg 20	Thomas G. Bauer, Fichtestr. 31, Tel. 0911/538321.
5000 Köln 60	Michael Heycke, Boltensternstr.
4830 Gütersloh 1	Ludwig Röver, Holzheide 145A
4900 Herford	Andreas Findewirth, Im Großen Vorwerk 48, Tel.: 05221/23504

Eine Fachgruppe will gründen:

7000 Stuttgart 80	Grafik/Arithmetik, Jörg Tomes, Anweilerweg 56, Tel. 0711/7802293.
8000 München 70	Btx u. FORTH, Christian Schwarz, Lindenschmitstr.30, 8000 München 70

Hier kann man um Rat fragen:

02103/556 09	Jörg Staben, Dienstag und Freitag, 20.00 - 22.00 Uhr
02845/28951	Karl Schroer

Ansprechpartner zu bestimmten Interessengebieten:

volksFORTH/ultraFORTH:	Klaus Kohl, Tel.: 08233/30524 Bernd Pennemann, Tel. 0228/640979 und Klaus Schleisiek-Kern, Tel. 040/2202539.
32-Bit Systeme:	Robert Jones, Tel. 02434/4579
Künstliche Intelligenz:	Ulrich Hoffmann, Tel. 0431/678850
NC4000 Novix Chip:	Klaus Schleisiek, Tel. 040/6449412
Realtime relationale Netze:	Wigand Gawenda, Tel. 040/446941
Gleitkomma-Arithmetik:	Andreas Döring, Tel. 02631/52786
32FORTH	Rainer Aumiller, Tel. 089/6708355
PostScript/FORTHscript	Christoph Krinninger, Tel: 089/725 93 82

FORTH-Gesellschaft e.V. - Postfach 1110 - D-8044 Unterschleißheim
Tel.089/3173784, FORTH-Mailbox Tel.: 089/7259625
Postgiroamt Hamburg, Kontonr.: 563211-208 BLZ 20010020

Ergänzungen, Änderungen bitte dem Büro der FORTH-Gesellschaft e.V. mitteilen.

FORTH-SYSTEME

Software-Vertrieb
Beratung und Schulung

ANGELIKA FLESCH

Software-Entwicklung
Hardware-Entwicklung

UR/FORTH

- Forth-83 Standard
- Für MS-DOS, OS/2, 80386, 68000 UNIX und XENIX
- Direkt gefädelte Code Implementationen mit dem obersten Stackwert im Register um größtmögliche Ausführungsgeschwindigkeit zu erreichen
- Segmentiertes Speichermodell mit Programm, Daten, Headers und Dictionary Hash Table jeweils in einem getrennten Segment
- Komplett gehashtes Dictionary führt zu extrem schneller Übersetzung
- Mächtige neue String Operatoren (Suche, Extraktion, Vergleich und Addition) sowie einen dynamischen String-, Speichermanager
- Kann mit Objektmodulen, die in Assembler oder anderen Hochsprachen erzeugt wurden, gelinkt werden
- Native Code Optimizer zur direkten Umsetzung in 80 x 86 Code im Lieferumfang

HARRIS RTX 2000

Informieren Sie sich über diesen Prozessor, der auch von uns unterstützt wird.

DSP APPLIKATIONEN

DSP Anwendungen mit dem AT & T DSP-32. Informieren Sie sich über unser Angebot.

FORTH MAIL BOX

Für alle FORTH-Interessierten hat unsere Firma eine Mailbox eröffnet. Sie ist unter der Nummer 0 76 67 556 zu erreichen und akzeptiert 300, 1200 und 2400 Baud, 8N1. Außer einer offenen Hauptkonferenz und einigen Fileareas enthält sie auch Supportkonferenzen für unsere komplette Produktlinie.

LMI FORTH-83 Metacompiler

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

● 8086/8088	● 8096/97
● Z80	● HD64180
● 8080/8085	● 8031/32/535
● 68000	● 6303
● Z8	● 6502
● 1802	● 6802
● 6809	● 68HC11
● 65816/65802	● RTX 2000

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

Forth Programme, die mit einem LMI interaktiven Forth System z. B. PC/FORTH oder Z80 Forth geschrieben und getestet wurden, werden im Normalfall mit nur geringen Änderungen übersetzt.

Serieller ROM/RAM Simulator

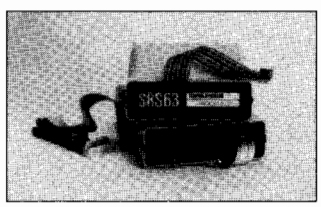
Entwickeln Sie romfähige Programme ?

Müssen Sie neu entwickelte Einplatinencomputer testen ?

Setzen Sie 2764, 27128, 27256, 27512 oder 4364, 43256 oder kompatible ROM/RAM-Bausteine ein ?

Wollen Sie diese Bausteine mit bis zu 38 400 Baud über die serielle Schnittstelle laden ?

Können Sie eine zusätzliche serielle Schnittstelle über den Speichersockel zum interaktiven Programmieren gebrauchen ?



Dann ist unser SRS63 die optimale Ergänzung Ihres Arbeitsplatzes.

Sie werden vom Preis-Leistungsverhältnis überrascht sein.

Unsere ROM-Compiler liefern direkt verwendbare Dateien, wir akzeptieren auch Intel-Hex oder Motorola-S-Formate.

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

Kühnheimerstraße 21 · Postfach 1103 · D-7814 Breisach · Tel. (0 76 67) 5 51 · Fax 07667 555