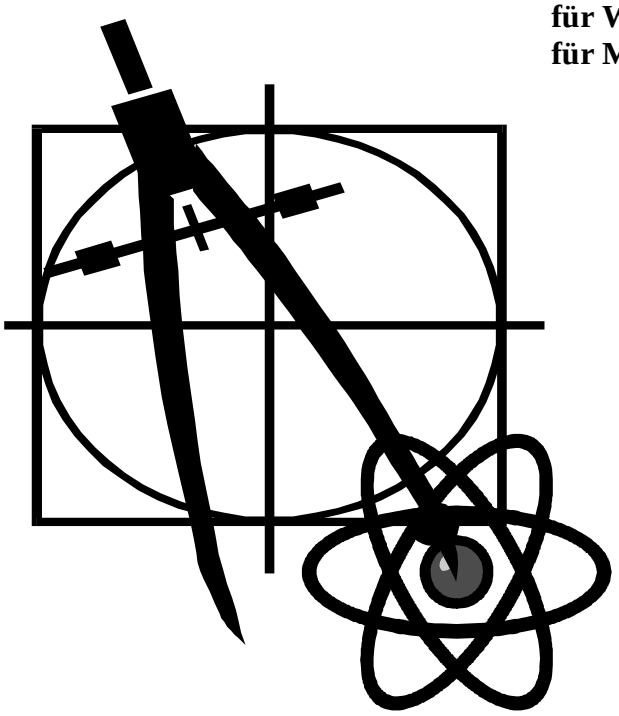
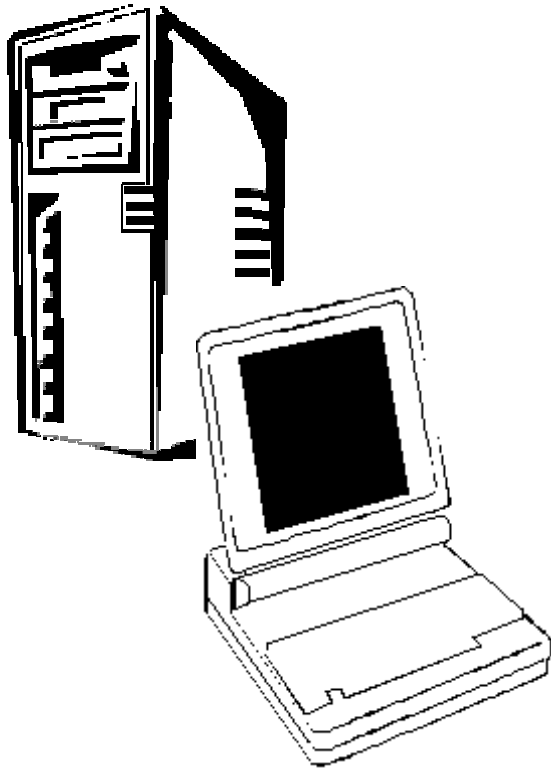
VIERTE DIMENSION

2 / 2003

Das FORTH-Magazin

**für Wissenschaft und Technik, für kommerzielle EDV,
für MSR-Technik, für den interessierten Hobbyisten.**





In dieser Ausgabe:

19. Jahrgang - **Ausg. 2/2003** - 4,00 €

Organ der FORTH-Gesellschaft e.V.

Dienstleistungen und Produkte fördernder Mitglieder des Vereins

tematik GmbH
Technische Informatik

Feldstrasse 143
D-22880 Wedel
Fon 04103 – 808989 – 0
Fax 04103 – 808989 – 9
mail@tematik.de
www.tematik.de

Gegründet 1985 als Partnerinstitut der FH-Wedel beschäftigten wir uns in den letzten Jahren vorwiegend mit Industrieelektronik und Präzisionsmeßtechnik und bauen z.Z. eine eigene Produktpalette auf.

Know-How Schwerpunkte liegen in den Bereichen Industriewaagen SWA & SWW, Differential-Dosierwaagen, DMS-Messverstärker, 68000 und 68HC11 Prozessoren, Sigma-Delta A/D. Wir programmieren in Pascal, C und Forth auf SwiftX86k und seit kurzem mit Holon11 und MPE IRTC für Amtel AVR.

LEGO RCX-Verleih

Seit unserem Gewinn (VD 1/2001 S.30) verfügt unsere Schule über so ausreichend viele RCX Komponenten, dass ich meine privat eingebrachten Dinge nun Anderen, vorzugsweise Mitgliedern der Forthgesellschaft e.V., zur Verfügung stellen kann.

Angeboten wird: Ein komplettes LEGO-RCX-Set, so wie es für ca. 230,- € im Handel zu erwerben ist.

Inhalt:

1 RCX, 1 Sendeturm, 2 Motoren, 4 Sensoren und ca. 1.000 LEGO Steine.

Anfragen bitte an

Martin.Bitter@t-online.de

Letztlich enthält das Ganze auch nicht mehr als einen Mikrocontroller der Familie H8/300 von Hitachi, ein paar Treiber und etwas Peripherie. Zudem: dieses Teil ist 'narrensicher' !

Dipl.-Ing. Arndt Klingelnberg

Tel.: ++32 +87 -63 09 89 (Fax: -63 09 88)
Waldring 23, B-4730 Hauset, Belgien
akg@aachen.kbbs.org

Computergestützte Meßtechnik und Qualitätskontrolle, Fuzzy, Datalogger, Elektroakustik (HiFi), MusiCassette HighSpeedDuplicating, Tonband, (engl.) Dokumentationen und Bedienungsanleitungen.

Forth Engineering
Dr. Wolf Wejgaard

Tel.: +41 41 377 3774 - Fax: +41 41 377 4774
Neuhöflirain 10
CH-6045 Meggen http://holonforth.com

Wir konzentrieren uns auf Forschung und Weiterentwicklung des Forth-Prinzips und offerieren HolonForth, ein interaktives Forth Cross-Entwicklungssystem mit ungewöhnlichen Eigenschaften. HolonForth ist erhältlich für 80x86, 68HC11 und 68300 Zielprozessoren.

KIMA Echtzeitsysteme GmbH

Tel.: 02461/690-380
Fax: 02461/690-387 oder -100
Karl-Heinz-Beckurtz-Str. 13
52428 Jülich

Automatisierungstechnik: Fortgeschrittene Steuerungen für die Verfahrenstechnik, Schaltanlagenbau, Projektierung, Sensorik, Maschinenüberwachungen. Echtzeitrechnersysteme: für Werkzeug- und Sondermaschinen, Fuzzy Logic.

FORTecH Software
Entwicklungsbüro Dr.-Ing. Egmont Woitzel

Budapester Straße 80 a D-18057 Rostock
Tel.: (0381) 46 13 99 10 Fax: (0381) 4 58 34 88

PC-basierte Forth-Entwicklungswerkzeuge, comFORTH für Windows und eingebettete und verteilte Systeme. Softwareentwicklung für Windows und Mikrocontroller mit Forth, C/C++, Delphi und Basic. Entwicklung von Gerätetreibern und Kommunikationssoftware für Windows 3.1, Windows95 und WindowsNT. Beratung zu Software-/Systementwurf. Mehr als 15 Jahre Erfahrung.

Ingenieurbüro
Dipl.-Ing. Wolfgang Allinger

Tel.: (+Fax) 0+212-66811
Brander Weg 6
D-42699 Solingen

Entwicklung von µC, HW+SW, Embedded Controller, Echtzeitsysteme 1-60 Computer, Forth+Assembler PC / 8031 / 80C166 / RTX 2000 / Z80 ... für extreme Einsatzbedingungen in Walzwerken, KKW, Medizin, Verkehr / >20 Jahre Erfahrung.

Ingenieurbüro
Klaus Kohl

Tel.: 08233-30 524 Fax: - 9971
Postfach 1173
D-86404 Mering

FORTH-Software (volksFORTH, KKFORTH und viele PD-Versionen). FORTH-Hardware (z.B. Super8) und Literaturservice. Professionelle Entwicklung für Steuerungs- und Meßtechnik.

Inhalt

Diese Ausgabe der VD wird vier bis sechs Wochen nach dem Erscheinen der Druckausgabe im Internet auf der Web-Seite der Forthgesellschaft e.V. veröffentlicht.

http://www.forth-ev.de

Eine PDF-Version dieser Ausgabe wird ab dem Zeitpunkt der Veröffentlichung im Internet ebenfalls zur Verfügung stehen. Bitte wenden Sie sich hierzu über die oben angegebene Adresse an den Webmaster der Forthgesellschaft e.V. oder an die Redaktion der „Vierte Dimension“.

fep

In der nächsten Ausgabe finden Sie voraussichtlich:

- Triceps spielt Solitär
- Solitair, Spieletheorie
- OOP – Prelude‘s Konzept
- 4 gewinnt, Spieletheorie
- Ein GCC codiert Forth
- Forth entwickelt Schallhörner
- µCore Simulator

Editorial

IMPRESSUM

Name der Zeitschrift

Vierte Dimension

Herausgeberin

Forth-Gesellschaft e.V.
Postfach 19 02 25
80602 München
Tel.: (0 89) 1 23 47 84
E-Mail:
SECRETARY@FORTH-EV.DE
DIREKTORIUM@FORTH-EV.DE

Bankverbindung: Postbank Hamburg
BLZ 200 100 20
Kto 563 211 208

Redaktion & Layout

Friederich Prinz
Hombergerstraße 335
47443 Moers
Tel.: (0 28 41) 5 83 98
E-Mail:
VD@FORTH-EV.DE

Anzeigenverwaltung

Büro der Herausgeberin

Redaktionsschluß

März, Juni, September, Dezember
jeweils in der dritten Woche

Erscheinungsweise

1 Ausgabe / Quartal

Einzelpreis

4,00 € + Porto u. Verpackung

Liebe Leser,

die Forth-Tagung in Lambrecht war ein Fest. Die vorliegende Ausgabe der VD enthält selbstverständlich einen ausführlicheren Bericht über die Tagung und zur Mitgliederversammlung. Diese Berichte werden Sie hoffentlich dazu bewegen, im nächsten Jahr zur Tagung auf die Insel Fehmarn zu fahren.

Für mich persönlich stellt sich nach den Treffen mit den forthigen Kollegen immer aufs Neue die Frage, was mich im jeweiligen Jahr am meisten beeindruckt hat. Die Herzlichkeit der Gastgeber, der Tagungsort, die Landschaft und Umgebung des Tagungsortes, das Rahmenprogramm oder die Beiträge, Themen, Diskussionen und Begegnungen sind nur einige Rahmen zu einer Menge wirklicher Schlaglichter. Eine Auswahl zu treffen, fällt schwer.

Aber die Anwesenheit der jungen Leute auf der Tagung hat mir viel Spaß gemacht. Ihre Ideen, ihre Vorträge und die forthige Neugier mit der sie sich auf die Spielereien der „Alten“ eingelassen haben, läßt mich denken, daß Forth eine weitaus bessere Zukunftsperspektive hat, als etabliertere Sprachen das hoffen dürfen.

Auch die Informationen über die langsamen aber stetigen Fortschritte verschiedener Projekte in der FG stimmen mich positiv. Die neue FG-CD, das „Scan-Projekt“ (alte Ausgaben der VD im PDF-Format digitalisieren) und nicht zuletzt die Angebote und Beschlüsse zu µCore und b16 in FPGA zeigen deutlich, daß unsere Gesellschaft so aktiv und engagiert ist wie ihre Mitglieder selbst.

Das Büro der Forthgesellschaft mußte in diesem Frühjahr von Rostock nach München umziehen. Auch das ist dank des Engagements aller Beteiligten reibungslos und zügig vonstatten gegangen. Sie können das direkt dem nebenstehenden, aktualisierten Impressum entnehmen. Es ist gut, zu wissen, daß die FG über engagierte Mitglieder verfügt, die bereit sind, die für den Verein notwendigen Arbeiten zu übernehmen.

Ebenso verfügt die FG nach wie vor über eine ausreichend große Anzahl von Mitgliedern, die sich mit ihrem Einsatz für Forth und für die Forthgesellschaft für eine Nominierung für den Drachenpreis qualifizieren. Auch darüber wird an anderer Stelle in dieser Ausgabe mehr zu lesen sein.

Der Fußzeile dieser Seite können Sie entnehmen, daß es die VD, und damit die FG, in 2003 bereits seit 19 Jahren gibt. Im kommenden Jahr wollen wir auf Fehmarn unser 20-jähriges Jubiläum feiern. Ich bin heute sicher, daß wir dann immer noch feststellen werden, daß die Forthgesellschaft jung ist.

Ihr

Friederich Prinz

Quelltext-Service

Die Quelltexte in der VD müssen Sie nicht abtippen. Sie können diese Texte auch direkt bei uns anfordern und sich zum Beispiel per E-Mail schicken lassen. Schreiben Sie dazu einfach eine E-Mail an die Redaktionsadresse. *fep*

Die Forthgesellschaft wird durch ihr Direktorium vertreten:

Prof. Dr.	Fred Behringer	Kontakte: Direktorium@Forth-ev.de
Dr.	Ulrich Hoffmann	
Dipl.Inf.	Bernd Paysan	

Datum: Thu, 13 Mar 2003 13:50:48 GMT
Von: fsala@t-online.de (Filippo Sala)

...
In ISORT habe ich einen Fehler entdeckt, der sich aber nur auf die MC680X0-Version auswirkt.

```
Wort : ISORT ( ..... -- )

-Zeilentext alt     dtype 1 =  i 3 =  and      \ wenn Zahlen
                                               \ mit Vorzeichen

-Zeilentext neu     dtype 1 =  i 3 =  and      \ relative Zahlen
                                               \ (für 680X0-CPU 0 statt 3)
```

Hat jemand nach dem Listing gefragt? Wenn jemand nach dem Listing fragt, nehme ich Ihnen gerne die Arbeit ab. Schicken Sie mir bitte dessen Email-Adresse.

Herzliche Grüsse
Filippo Sala

Nun, Filippo Salas E-Mail Adresse ist hiermit veröffentlicht. Wenden Sie sich bitte zu allen Fragen bezüglich der in der letzten Ausgabe der VD vorgestellten Sortieralgorithmen direkt an den Autor.

Darüber hinaus beschäftigen sich auch unsere englischen Freunde in der FIG UK aktuell mit dem Sortieren von Listen. Chris Jakeman hat in der Forthwrite (März 2003; S. 11) einen längeren Beitrag hierzu veröffentlicht.

fep

Betreff: Meinung zum Leserbrief von Gerard Baecker
(VD 1/2003)
Datum: Wed, 05 Mar 2003 20:56:28 +0000
Von: Ulrich Paul <upaul@paul.de>

Hi Freunde,

nun, den "Handschuh" greife ich gerne auf, muß aber gleich dazusagen, daß ich ihn nicht als einen Fehdehandschuh auffasse. Ich muß nämlich Gerard Baecker weitgehend recht geben. Vom Standpunkt der Informatik aus ist Forth nur eine von vielen Progammiersprachen. Wie jede hat sie ihre Vor- und Nachteile. Kein vernünftiger Mensch käme auf die Idee, eine aufwendige GUI in Forth, oder in C, oder in Pascal, oder ... zu schreiben, wenn es dazu fertige Tools gibt (auch wenn dieses Tool dann vielleicht Java generiert, obwohl man in dieser Sprache eigentlich gar nicht programmiert). Was ich dem Leserbrief entnehmen kann, ist einfach folgendes:

Eine klare Verteidigung eines Nicht-Forthlers gegen die nach außen getragene Überzeugung der Forthler, daß alle anderen unterlegen sind und das einzig und allein dadurch, daß sie nichts mit Forth am Hut haben. Diese Verteidigungsrede (besser Brief) sitzt! Und ich kann sie weitgehend unterstützen.

Aber eines möchte ich doch klarstellen:

Forth wird von Praktikern verwendet. Für Theoretiker ist Forth recht uninteressant - wie beinahe jede Programmiersprache. Einen gegebenen Algorithmus kann ich in fast jeder Sprache implementieren. Wer sich also primär für Algorithmen interessiert, braucht nicht programmieren zu können. Da hat Gerard Baecker absolut recht und keiner wird ihm widersprechen. Aber ein Algorithmus per se löst nicht ein Problem der realen Welt. Dazu muß er sinnvoll und gewinnbringend angewendet werden.

Und hier liegt der Knackpunkt an dem beide Seiten immer wieder aneinander vorbeireden:

Wer ein Projekt, das recht eng mit der physikalischen Welt kommuniziert und das leicht adaptierbar und portierbar sein soll, durchziehen will, der kann sich auf keine so große Toolbasis stützen wie ein Programmierer für z.B. Office- oder Wissenschafts-Anwendungen, deren I/O nur über Maus, Tastatur, Bildschirm, Festplatte und ev. noch über das Netzwerk erfolgt. Die anstehenden Probleme werden in einer HW-nahen Umgebung nicht primär durch Algorithmen gelöst. Das "Wie" ist dann nicht unbedingt das ausschlaggebende, sondern mehr das "Wann" und das "Unter welchen Einschränkungen".

Diese Umgebung erzwingt andere Vorgehensweisen als die Theorie beschreibt - und sie auch nie vollständig beschreiben kann. Andererseits ist ein reiner Theoretiker mit funktionalen Sprachen wie Haskell, ML Miranda oder Prolog besser bedient. Wie diese Sprachen praktisch untauglich zur Lösung eines Steuerungsproblemes sind, so ist Forth genauso untauglich als Ersatz für diese Sprachen. Das sind zwei verschiedene Welten. Und daher gibt es auch keine Konkurrenz und schon gar keine Fehde zwischen ihnen. Wer trotzdem da dem anderen irgend etwas vorwirft, ist einfach nur ein bornierter Streithammel!

Abschließend möchte ich aber die Forthler auch ein bißchen in Schutz nehmen: Lieber Gerard Baecker, die Forthler sind nicht im Herzen so wie sie sich nach außen geben. Ihr jahrzehntelanger Kampf mit den "Anderen" hat sie so gemacht: Ein kleiner verschworener Trupp aus lauter Einzelkämpfern, die sich in lockerer Aufstellung in den Kampf werfen und dabei durchaus auf Mannen der eigenen Seite einschlagen. Das drückt sich schon allein dadurch aus, daß wohl keine zwei Forthsysteme gleich sind, außer sie wurden als schlüsselfertige SW einem Anwender geliefert. Es ist der Stolz eines Forthlers ein eigenes Forth zu haben - und da bin ich auch keine Ausnahme.

Was hat mich dazu bewegt ein eigenes Forth zu bauen? Einmal schon allein die Leichtigkeit mit der man das machen kann. Dann aber auch meine Faulheit, denn ich lasse Aufgaben lieber vom Computer erledigen, als daß ich es machen muß. In welcher anderen Sprache, die aber ebenfalls interaktiv und laufzeiteffizient ist, kann man so leicht den Compiler on-the-fly modifizieren wie in Forth? Ich kenne Keine. Aber diese Fähigkeit spart bei richtiger Anwendung unwahrscheinlich viel Zeit beim Programmieren und der Fehlersuche. Sicher, Makros in C sind auch recht mächtig, aber eben doch nicht so wie die CREATE-

DOES>-Konstruktion in Forth. Aber das lernt man erst zu schätzen, wenn man damit echte Probleme elegant gelöst hat. Mein Tip daher: Einfach irgendein Forth installieren (gibt es für so ziemlich jede Plattform, allerdings mit kleinen Unterschieden, da die meisten nicht einfach eine Portierung sind, sondern ein eigenes Forth eines Forthlers, s.o.) und herumspielen. Aber immer im Kopf behalten, daß das keine Konkurrenz zu Basic, Pascal, Cobol, usw. sein soll. Und richtig lustig wird die Sache dann, wenn man irgendwelche Hardware damit ansteuert.

CU, *Uli*

PS: Ich programmiere zur Zeit fast ausschließlich in Perl und C, weil die anstehenden Probleme eben damit gut gelöst werden können und nur deshalb.

Paul Elektronik, Erlenweg 18, D-86391 Stadtbergen
mailto:upaul@paul.de http://www.paul.de
Tel: +49-821-156888-0 Fax: +49-821-156888-7

Betreff: Eine kleine Aufgabe,
wie von Gerard Baecker angeregt
Von: Ulrich Paul <upaul@paul.de>

Hi Fritz,
was hältst Du von einem dauerhaften "Wettbewerb", der zum Ziel hat, das verblüffendste Ergebnis zu produzieren? Im Prinzip ein Ergebnis "for fun only". Was mir da so einfällt, wäre ein Deutsch auf "Hundhammer-Englisch"-Übersetzer, also einer, der z.B. die Phrase "gleich geht es los" auf "equally goes is loose" übersetzt (Leser der Süddeutschen Zeitung kennen das als die "Filserbiefe").

Da steckt mehr dahinter als man so einfach sieht. Im ersten Durchlauf sollten wörtliche Übersetzungen erlaubt sein, aber in den weiteren Durchläufen sollte auf den Kontext mehr und mehr eingegangen werden.
Forth hätte da einige Vorteile, vor allem weil man on-the-fly Worte generieren und sie dann hinterher "forget"ten kann, was die Sache am Ausufern hindert.

Das Problem ist kein so rein akademisches, aber ich will hier den Spaß-Effekt in den Vordergrund stellen, weil sonst die meisten Fortler nicht darauf reagieren werden

CU, *Uli*

Na, ich hoffe, daß viele Leser hierauf reagieren. In der Vergangenheit haben das eine oder andere Rätsel, oder entsprechende Aufgaben gelegentlich große Resonanz gefunden. Manchmal hängt die Begeisterung für solche Aufgaben aber direkt proportional von der Jahreszeit, bzw. vom Wetter ab.

fep

Betreff: Eine kleine Aufgabe,
wie von Gerard Baecker angeregt
Von: Ulrich Paul <upaul@paul.de>

...hier eine kleine Aufgabe:

Gegeben sei ein Mikrokontroller mit serieller Schnittstelle (wie SPI oder µWire). An dieser Schnittstelle hängen 4 Portchips, die alle von dem selben CE aktiviert werden. Als Beispiel diene hier der 68HC68P1-Portchip. Er ist 8 Bit breit und wird in dieser Aufgabe nur zur Ausgabe verwendet. Um ein Byte auszugeben, muß eine Zwei-Byte-Sequenz geschickt werden. Das zweite Byte sind die eigentlichen Nutzdaten und das erste enthält Steuerinformationen. Hier relevant sind nur die obersten zwei Bit, die die ID des gewünschten Chips enthalten. Jeder Chip hat also eine eigene ID, die durch Beschaltung zweier Pins des Chips festgelegt wird. Aber das nur am Rande. An den 8 Ausgängen ist jeweils eine 7-Segment-Anzeige mit Dezimalpunkt angeschlossen.

Die Aufgabe besteht nun darin, einen übergebenen Zahlenwert auf das Display zu bringen, also eine Routine zu schreiben, die folgende Deklaration hat: >DISPLAY (N D --). Wobei als Schikane die Randbedingung gelten soll, daß das Minuszeichen durch den Dezimalpunkt der Einerstelle angesteuert wird. Die Ausgabe soll dezimal erfolgen, womit der Wertebereich von N gleich -9999 bis +9999 ist. Der oberste Wert auf dem Stack gibt die Position des Dezimalpunktes an und zwar gezählt von rechts nach links, also aktiviert P=1 den Punkt zwischen der Einer- und der Zehnerstelle, P=2 den zwischen Zehner- und Hunderterstelle, usw. Vorsicht! P=0 darf nicht das Minuszeichen aktivieren!

Klar, die Aufgabe ist schon beinahe trivial. Aber mir geht es hier mehr um die Eleganz. Die kann aber durchaus verschieden betrachtet werden. So z.B. kann man sie unter dem Gesichtspunkt der klaren Verständlichkeit der Lösung sehen. Oder auch unter dem der Effizienz (Zeit und/oder Platz) oder unter dem der leichten Erweiterbarkeit auf mehr Stellen oder unter dem der leichten Portierbarkeit oder ...

Ich möchte hier keinen Wettbewerb mit einem Sieger haben, sondern ein möglichst großes Sortiment von verschiedenen Lösungen sehen. Als Sprache ist jede zugelassen, die es in freier Verbreitung gibt. Bei Exoten muß daher eine Adresse angegeben werden, von der man diese Sprache (Interpreter oder Compiler) herunterladen kann.

Da es kein Wettbewerb sein soll, werde ich mir auch etwas einfallen lassen und es einreichen. Ich habe dieses Problem nicht vorher gehabt, sondern es ist mir gerade vorhin erst eingefallen. Es steht auch nicht in der Zukunft an, da es dafür dedizierte Chips gibt und mit denen habe ich dieses "Problem" natürlich gelöst.

CU, *Uli*

Betreff: smartcards
Von: Rafael Deliano <Rafael_Deliano@t-online.de>

Die FAQ von alt.technology.smartcards von Scott Guthery ist recht lesenswert:
...
Europay International (http://www.europay.com) hat ebenso eine Spezifikation für Terminals zusammengestellt, die "Open

Terminal Architecture" (OTA) genannt wird. OTA enthält eine virtuelle Forth-Maschine. Die OTA VM ist ein Derivat der FORTH VM, die von MicroProcessor Engineering (www.mpeltd.demon.co.uk) für das SENDIT Esprit Projekt designt wurde. Die VM arbeitet in einer 2-Stack-Architektur, die aus Forth abgeleitet und sprachenneutral erweitert wurde, so daß Kompilate aus den Quellen auch anderer Sprachen als Forth erzeugt werden können. Tatsächlich wird C häufiger genutzt als Forth.

...

Keycorp (www.keycorp.com.au) hatte eine SmartCard mit dem Namen OSSCA (Operating System for Smart Card Applications) auf dem Markt, die Sie mit Forth programmieren konnten. Dies war vermutlich die erste SmardCard mit einer virtuellen Maschine.

...

Kann man inzwischen wohl selber basteln:

...

Blank Cards, White Cards and Soft Masks

==========

Mit diesen Karten können Sie ausführbaren Code direkt in das EEPROM der SmardCard laden. Die Karten enthalten einen kleinen Lader im ROM, der Motorola S-Records oder Intel Hex-Records oder andere Binärcodeimages laden kann. Nach Beendigung des Ladevorgangs kippen Sie ein Bit, das den Chip anweist, nach dem nächsten Reset nicht den Lader auszuführen, sondern Ihr kleines Programm.

Es ist ganz klar, daß dies aus der Sicht des Entwicklers die flexibelsten Karten sind, die Sie bekommen können. Sie sind aber auch am schwersten in den Griff zu bekommen.

Ein heftiges Aufstöhnen bezüglich sicherheitlicher Bedenken bei Blank Cards ist nicht zu überhören. Allerdings können Sie mit einer Blank Card nichts anstellen, was Sie nicht auch mit einer Java Card oder eine Windows Card tun könnten.

Atmel verkauft ein Entwicklerwerkzeug mit dem Sie Ihre eigene Smard Card aufbauen können und von den Flash Memory Chips wegkommen.

...

MfG, *JRD*

Hinweis: Die kursiv gedruckten Textstellen der vorangehenden Meldung wurden aus dem Englischen übersetzt.

fep

Wo will Ulrich Hoffmann wohl hin?

Aufruf des Direktoriums der Forthgesellschaft e.V.

In einem Verein, der dezentral organisiert ist, ist es nicht uninteressant zu erfahren, wer wo wohnt und was betreibt. Kontakte werden in der Nachbarschaft schneller geschlossen als über weitere Entfernungen hinweg. Wir möchten nach langer Zeit wieder einmal eine Mitgliederliste herausgeben. Zur Abkürzung der Vorbereitung und um den datenrechtlichen Anforderungen gerecht zu werden, möchten wir das Ausschliessungsprinzip anwenden: Jeder, der (oder die) mit der Veröffentlichung seiner (ihrer) Daten in Bezug auf eine der folgenden Rubriken nicht einverstanden ist, möge das dem Forthbüro möglichst bald mitteilen. Nichtantworten gilt als Zustimmung. Als Rubriken kommen in Frage: Name, Beruf, Wohnort, Straße und Hausnummer, Telephonnummer, E-Mail-Adresse, forthbezogene und verwandte Interessen.

Herzliche Grüße

Ihr

Fred Behringer

Neues vom Forth-Büro

Das Büro ist umgezogen,- von Rostock nach München. Die neue Adresse erfahren Sie im Impressum. E-Mail-Adresse und Bankverbindung haben sich nicht geändert.

<Die e-mail Adresse des Forth-Büros bleibt witerhin ***secretary@forth-ev.de***

fep>

Bisher wurden Sie von Ute Woitzel bestens betreut. Ich danke ihr herzlich, auch dafür, dass sie die Übergabe der Geschäfte so vorbildlich gestaltet hat. Nun bin ich Ihr Ansprechpartner.

Mein Name ist Rolf Schöne. Forth-Anhänger bin ich seit 1978, bin es also schon sechs Jahre vor Bestehen der Forth-Gesellschaft gewesen, die im nächsten Jahr ihr 20-jähriges Jubiläum feiern kann.

Das ist alles schon eine Weile her, und wenn Sie jetzt annehmen, dass ich Rentner bin, dann haben Sie Recht. Daher habe ich auch die Zeit, das Büro zu verwalten. Ich hoffe, dass ich das als Nachfolger von Ute Woitzel (fünf Jahre hat sie das gemacht) und Ulrike Schnitter (die es gar auf 10 Jahre gebracht hat) zu Ihrer Zufriedenheit tun kann. Ich bitte Sie um Nachsicht, wenn nicht alles gleich so klappt, wie Sie es gewohnt sind. Momentan ist mein Stack an Erfahrungen noch ziemlich leer. Füllen Sie ihn, bitte. ok

Rolf Schöne

OO ist (k)eine Antwort

Der nachfolgende Text ist die deutsche Übersetzung eines "Postings" von Tim Daneliuk im Forum comp\lang\forth. Vorausgegangen waren mehrere Postings und Re-Postings zur Verleihung des ***Turing-Preises*** *an* ***Dahl und Nygaard*** *in 2001. Tim Daneliuk hätte – auch als Nicht-Forther –* ***Charles Moore*** *gerne als Preisträger gesehen. Aber schon der bloße Hinweis darauf hat eine Diskussion über Programmiersprachen und Konzepte entfacht, die Tim im Keim ersticken wollte. Das Ersticken hat nicht funktioniert. Aber die Vierte Dimension hat einen Artikel, der ihre Leser zu eigenen Äußerungen anregen soll.*

Tim Daneliuk

tundra@tundraware.com

Betreff: Silver Bullets (Long) - Was: Turing award for Moore?

Elizabeth D. Rather:
Der "A.M. Turing Award 2001", der auch "Nobelpreis der Computerwissenschaften" genannt wird, wurde von ACM an Ole-Johan Dahl und Kristen Nygaard verliehen; für ihre Rolle bei der Entwicklung der objektorientierten Programmierung, des Programmiermodells, das heute am weitesten verbreitet ist.

Daneliuk:
Es ist nur schwer einzusehen, daß Moore, der mit FORTH einen tiefen Einschnitt in Programmier-Paradigmen vorgenommen hat, nicht ebenso anerkannt ist. Aber noch einmal: das ACM (Association for Computing Machinery) neigt dazu, größeren Gefallen an "akademischen" Veränderungen zu finden, als an praktischen (Thompson & Ritchie ausgenommen). Wenn Moore seine Arbeit mit einer Menge unwesentlicher und überdeckender Mathematik vollgestopft hätte, dann wäre er vielleicht ernsthaft bedacht worden.

aph@redhat.invalid:
Um fair zu Dahl und Nygaard zu sein: es ist völlig falsch, die objektorientierte Programmierung für C++ verantwortlich zu machen: Sofern es korrekt genutzt wird, vereinfacht OO Programmieraufgaben mehr als daß es sie vernebelt.

Elizabeth D. Rather:
Auch wenn die Applikation objektähnliche Merkmale hat, nutzen rein-prozedurale Applikationen (wie die meisten eingebetteten Systeme) davon nichts.

<**Knurr-Modus**>

Daneliuk:
Ist denn das wahr? Ich habe jüngst indirekt einen Brückenschlag angeboten zwischen "OO ist nicht die Antwort auf die Probleme dieser Welt" auf der einen Seite und der Frage "Welche Programmiersprache ist die beste", die in jeder Nanosekunde erneut im USENET aufgeworfen wird, auf der anderen Seite.

http://www.tundraware.com/Technology/How-To-Pick-A-Programming-Language/

<siehe nachfolgenden Beitrag, fep>

Nachdem ich über 20 Jahre im Beruf bin, höre ich nie auf, vor der impliziten Annahme von so vielen unserer Mitpraktiker zu erstaunen, daß eine "silberne Kugel", gerade um die Ecke herum, darauf wartet gefunden zu werden, damit unsere Softwareproduktivitäts- und Qualitätsprobleme für alle Zeit mit ihrer Hilfe behoben werden können. Vermutlich sind wir der einzige Berufszweig, der ein beträchtliches Maß an Hirnschmalz in den Versuch investiert, jegliche Praxis auf ein einzelnes Werkzeug oder eine einzelne Methode zu reduzieren. Ich kann mir keinen Automechaniker vorstellen, der versucht seine Arbeit so elegant zu erledigen, daß er mit einem Schraubendreher als einzigem Werkzeug für jede beliebige Arbeit auskommt. ;-)

Nach meiner Beobachtung besitzen **OOs** nominale Tugenden, aber sie **machen** auch **Entwicklungsbemühungen häufig komplexer, länger in ihrer Dauer** und zuletzt **das Produkt schwieriger in der Wartung**. Nach meiner Auffassung resultiert das daraus, daß OOs niemals die Netz-Komplexität reduzieren, sondern diese lediglich transformieren, wo immer diese Komplexität erscheint.

Außerdem führt der Versuch, ein nicht triviales Problem mit einem einzigen Software-Paradigma zu bewältigen, dazu, daß neue Probleme erzeugt werden, die nichts mit Ihrer vordringlichen Aufgabe gemein haben. Die Designer/Programmierer vergeuden 80% ihrer Zeit und Kraft damit, dieses Paradigma den letzten 20% die Kehlen hinunter zu würgen, weil diese letzten 20% nicht wirklich gut in das gewählte Paradigma hineinpassen.

Tatsächlich denke ich, daß dies für **alle** Design- und Programmieransätze gilt, und keineswegs nur für OO.

Ich nutze gerne die folgende Matrix, (*Abb. 1*) um den noch wenig verbildeten jungen Menschen zu erklären, warum das, was sie in der Schule gelernt haben, nicht die wirkliche Art sein kann, Dinge anzupacken.

Ich behaupte, daß in jedem dieser Ansätze nur die jeweiligen Paradigmen transformiert wurden, wo auch immer die wirklichen Probleme aufgetaucht sind, gerade so als würde man eine Laplace Transformation von einer Differential-Gleichung nach Algebra transformieren, unter Inkaufnahme (neue + wirkliche Probleme) der Kosten für die Versuche, eine inverse Transformation hinzubekommen.

Der Versuch, die wahrhaft Gläubigen zu überzeugen, wird vergebens sein, weil diese aktuell ein signifikant komplexes System aufbauen müssen, das KI einschließt, Daten Speicher, Transaktions-Schutz, Fehler-Sicherheit usw.. Das Erste worüber die OO-Masse stolpern wird, ist die immense "Scheinwiderstand-Abweichung" zwischen der Art wie OOs Vererbung die Welt sieht, und wie DBMS die Welt sieht. Fügen Sie dazu noch die Notwendigkeit für Transaktions-Schutz-Schlüssel-Ereignisse hinzu, und das OO-Volk rennt "wird noch geliefert" murmelnd herum, sich beziehend auf JAVA oder ein neues .NET Merkmal.

Einführung des Paradigmas, Problems	Paradigma	Quelle der Komplexität/ Hauptgedanke
1950er	algorithmische Gliederung und -design	Erfassen/Reduzieren von Komplexität.
1960er	Heuristische Algorithmen für Hardwareprobleme	Suche nach Verfahren, die gut zu dem Problem „Speicher“ passen und bestimmen, ob das Verhalten im schlimmsten Fall akzeptabel ist.
1960er	KI	Definition eines Satzes von Regeln, die gut zu dem Problem „Speicher“ passen und dann daraus solche Regeln finden, die geeignet sind, nicht-triviale Probleme zu lösen.
1970er	DBMS Ansätze	Ordnen der Kategorien und der Semantik von Daten, so daß ein vollständiges und handhabbares Schema entsteht.
1980er	Prozedurale Programmierung	Wartung der Kohärenz und Konsistenz zwischen all den gegliederten Code-Teilen.
1980er	Objekt-Orientierung	Definition einer leistungsfähigen Vererbungs-Hierarchie, die generische Objekte faktorisiert, passend zu abgeleiteten Eigenschaften.
1980er	Methoden-Beweise	Suche nach mathematischen Analogien zur realen Welt, die für alles zu gebrauchen sind, außer zur Lösung des Spielzeugproblems „Speicher“.
1980er	Funktionale Programmierung	Transformierung von Problemen der realen Welt in funktionale Äquivalente, mit denen die FP arbeiten kann.

Abb.: 1

Die Wirklichkeit ist, das reale Probleme immer ereignis- oder bedingungsgesteuert sind. Deshalb sind die Strukturen der Zeit und der Ereignisse außerhalb des Programms sehr viel wichtiger als die interne Organisation des Codes selbst, oder welche Programmier-Paradigmen gewählt wurden. In der Echtzeit sind es Hardware-Ereignisse, die dem Programm seine Struktur und seinen Fluß diktieren. In Transaktions-Systemen ist es typischer Weise eine geschäftliches Ereignis, welches die Arbeit eines Programms initiiert.

In Betriebssystemen ist es eine Kombination von Hardware, Prozessen und Nutzer-Ereignissen, die diktieren, wie die Dinge laufen sollen.

Es ist schon erstaunlich, daß bei all dem Fortschritt in der Sprachentheorie, in Designtools usw. so wenig Wert gelegt wird auf die Unterstützung von (a)synchroner Datenübertragung, konkurrierende Prozesse, Locking und Rendezvous als die wichtigsten Eigenschaften der Sprachen selbst.

FORTH hat hier klar den Weg gewiesen, aber jüngere Sprachen wie Erlang beginnen, diesen Weg auch zu denken.

Ich behalte mir vor, mich zu irren, aber mir scheint, als entstünde gerade eine Welt, in welcher Programmierer komplexer Systeme ihre Codes in Meta-Sprachen wie Python oder Ruby schreiben – die OO umfassen, Prozedurales, DBMS und funktionale Programmierung – deren Laufzeitcodes aber besser von FORTH oder Erlang abgeliefert würden. Unterschiedliche Software-Paradigmen könnten austauschbar genutzt werden, weil diese die hervorstechendsten Eigenschaften der jeweiligen Sprache sind, aber der tatsächliche Code-Ausstoß erfolgte in eine hoch effiziente, einfach optimierte, ereignis-sensitive Laufzeitumgebung. Das ist womöglich ein Alptraum, weil der Versuch, die Semantik einer spät-bindenden, dynamischen Sprache wie Python an eine glatte FORTH-Laufzeitroutine anzupassen, vermutlich einfach zu teuer werden wird in Bezug auf Laufzeit und Codegröße. Oder etwa nicht?

Dann noch einmal, ich knurre gerade ein wenig...
</Knurr-Modus>

Tim Daneliuk

Übersetzung: Friederich Prinz

OO ist (k)eine Antwort

Wie man eine Programmiersprache auswählt

Tim Daneliuk
(tundra@tundraware.com)

(Originally posted on comp.lang.python, 1 May, 2002)

Dies wurde geschrieben als Antwort auf die unvermeidlich wiederkehrende Frage "Warum ist die Programmersprache x besser als y?". Diese Frage taucht im Usenet mit großer Regelmäßigkeit auf. Ich würde sonst etwas dafür geben, wenn sich diese Frage ein für alle Male klären lassen könnte...

Es sei **L** die Menge aller Programmiersprachen, die jemals erdacht wurden, in der Vergangenheit, in der Gegenwart, in der Zukunft. Die Metafrage **M** heißt dann:

> Welches ist das "beste" Mitglied der Menge **M** und deshalb: welches sollte ich lernen und nutzen?

Eine Variante von **M**, lassen Sie uns diese **M'** nennen, ist:

> Ist **L(x)** "besser" als **L(y)**?

Es sei **P** der Zweck zu dem Sie die "beste" Sprache lernen wollen.

Es sei **A** die Antwort auf **M** oder **M'** mit der Bedingung **P** als Einschränkung. **A** wird dann wie folgt lauten, für:

1) **P** = "Mich interessiert das Design von Sprachen und/ oder Sprachentheorie"

   A = Lernen Sie so viele Mitglieder aus **L**, wie Ihre Zeit das erlaubt. Verunreinigen Sie **L** aber nicht mit Ihren eigenen Ideen, bis Sie wirklich verstehen was Sie tun, und was bereits getan ist. Weichen Sie davon nur dann ab, wenn Sie restlos davon überzeugt sind, daß Sie durch eine originelle und nützliche Arbeit Substantielles zum Bestand beitragen können. Die Welt braucht keine weitere Sprache. Das werden Sie erkennen, wenn sie die Grammatik aller **L**R(1) verstanden haben.

2) **P** = "Meine aktuelle Sprache ist zu langsam/groß für die ihr zugedachte Aufgabe"

   A = Meistens haben Probleme dieser Art sehr viel mehr mit Ihrer schlechten Programmiererei und mit Algorithmen und Datenstrukturen zu tun, als mit der Sprache. Beschäftigen Sie sich mit Ihrem Programmierstil und Ihren Designansätzen, bevor Sie sich ausgerechnet von der Sprache trennen, die Sie aktuell am besten kennen. Lesen Sie Bentley's "Programming Pearls". Lernen sie, einen Profiler zu nutzen. Gehen Sie an Optimierungen dergestalt heran, daß Sie lediglich langsame/große Codeteile durch solche ersetzen, die in einer "natürlicherweise" schnelleren/kleineren Sprache geschrieben werden.

3) **P** = "Ich habe ein besonderes Problem zu bearbeiten, das durch eine der üblichen Sprachen nur unzureichend abgedeckt wird."

   A = Finden Sie heraus, was Andere hierzu bereits erledigt haben. Sie finden oft Lösungen zu Aufgabenstellungen, die denen Ihres Problems nahe genug kommen, um Ihnen nützlich zu sein. Wenn es wirklich überhaupt nichts Nützliches da draußen für Sie gibt, dann sollten Sie eine der Skript-Sprachen verwenden, um sich selbst eine "kleine Sprache" zu schreiben, die Ihre Anforderungen erfüllt. Wenn alles andere fehlschlägt, dann ignorieren Sie den Punkt 1) und erfinden Sie ihre eigene Sprache, aber halten Sie diese so klein und einfach wie möglich. Anderenfalls werden Sie weniger Zeit für die Lösung Ihres speziellen Problems aufwenden können, als Sie für die Pflege Ihrer neuen Sprache benötigen.

4) **P** = "Ich muß in ein Arbeitsverhältnis kommen/bleiben."

   A = Eine gute Übung ist sicher, wenn Sie für den Bereich Ihres Berufes/Aufgabengebietes die gebräuchlichsten Betriebssysteme, GUIs und Sprachen ermitteln, und die wahrscheinliche Entwicklung der nächsten paar Jahre abschätzen. Auf der Grundlage dieser Informationen sollten Sie eine prozedurale, eine objektorientierte, eine Skript-Sprache und, wenn möglich, noch einen Assembler auswählen, die Ihre Kernaufgaben abdecken. Stellen Sie dabei sicher, daß ausreichend Unterstützung für diese Sprachen zur Verfügung steht, wie Standardbibliotheken, Debugger, sprachensensitive Editoren, Profiler usw.. Wenn Sie zwischen verschiedenen Sprachen auswählen können, entscheiden Sie sich am besten für jene, die ein gemeinsames Link-Format unterstützen und zumindest einige sprachübergreifende Aufrufe (APIs) erzeugen. Wenn Sie sich zwischen gleichwertigen Sprachen entscheiden müssen, sollten Sie immer die ältere Sprache nehmen, weil diese auch die älteren Fehler hat!

5) **P** = "Ich möchte "portierbaren" Code schreiben."

Achtung: Es folgt eine computerwissenschaftliche Häresie. Ich habe mir damit schon überreichlich die Zunge verbrannt. Also lassen Sie Ihre Streichhölzer ruhig in den Taschen.

A = Code-Portabilität wird überbewertet. Viel wichtiger ist die Portabilität Ihrer programmiertechnischen Fertigkeiten. Sprachen kommen und gehen (nun, sie gehen nicht wirklich, nicht wahr?), aber ein guter Programmierer gibt seinen Fertigkeiten in jeder Sprache sehr guten Ausdruck! Das kommt mit der Zeit, mit der Erfahrung, durch das Lesen sehr guten Codes anderer Leute und durch Fehler, die man wieder und wieder macht. Die portabelste Sprache der Welt kann Ihnen nicht helfen, wenn Sie Algorithmen, Datenstrukturen und Objektaufteilung nicht verstehen und nicht wissen, wie Sie sich dazwischen zurecht finden. Wirklich portierbaren Code schreiben zu können, hängt mehr von der unterliegenden Umgebung ab (Hardware, OS, Netzwerk, I/O und GUI) als von der Programmiersprache, die Sie auswählen.

Selbstverständlich gibt es auch Fälle, in denen Portabilität sehr wichtig ist. So zum Beispiel, wenn Sie kommerzielle Software schreiben sollen, die auf einer großen Anzahl unterschiedlicher Plattformen erfolgreich für Ihr Unternehmen arbeiten muß. In diesem Fall schauen Sie zuerst nach, ob Sie plattformübergreifende Bibliotheken finden, damit Sie nicht mehr alle Arbeit allein tun müssen.

6) **P** = "Ich möchte viel Geld verdienen."

A = Hören Sie auf zu programmieren. Werden Sie ein Rock-Star, ein Schauspieler oder ein korrupter Politker. Die Möglichkeiten in diesen Feldern reich zu werden, sind erheblich besser als beim Programmieren.

7) **P** = "Ich habe gehört, daß OO der richtige Weg ist, Probleme zu lösen, darum möchte ich die beste OO Sprache."

A = OO ist ein Weg, Probleme zu lösen. Es ist nicht der einzige Weg. Viel wichtiger: es ist oft kein guter Weg. Sie sollten eine solide OO Sprache “in petto” haben. Das ist aber nicht alles was Sie brauchen (*werden*).

8) **P** = "Ich habe gehört, daß KI der richtige Weg ist, Probleme zu lösen, darum möchte ich die beste KI Sprache."

A = Siehe 7) ...

9) **P** = "Ich möchte Code schneller produzieren können."

A = Hauen Sie schneller in die Tasten. Es spielt keine Rolle, was Sie eintippen. Wenn Geschwindigkeit Ihr Ziel ist, dann werden Sie eben Experte in der Erzeugung fehlerhafter Codes; warum sich noch mit Syntax und Semantik herumärgern? Korrekter Code wird stets von Leuten produziert, die über das nachdenken, was sie tun. Oh, und auf lange Sicht sind diese Leute immer auch schneller in dem was sie tun.

10) **P** = "Ich möchte, daß <Geschlecht/Art> meiner Wahl mich mögen und beeindruckt genug sind, um mit mir auszugehen."

A = Siehe 6) ...

11) **P** = "Das System, das ich nutzen muß, schreibt mir die Sprache und Tools des Herstellers/Verkäufers vor. Ich will andere Möglichkeiten haben.”

A = Sehen Sie nach, ob es “Open Source” oder Angebote anderer Hersteller gibt. Wenn nicht, meistern Sie die Sprachen und Tools dieser Plattform. Danach erweitern Sie ihren Horizont, begründet auf Ihrem beruflichen Stolz und wegen Ihres Arbeitsverhältnisses (Siehe 4 ...).

12) **P** = "Ich bin fix und fertig. Alles was ich will, ist Code erzeugen, für so viele Plattformen wie möglich und mit so vielen Programmiersprachen wie möglich. Mein Ausbildungsverzeichnis ist 14 Seiten lang. Ich war seit 1982 nicht mehr aus dem Haus. Bitte helfen Sie mir.”

A = Nehmen Sie sich Zeit. Lesen Sie ein wenig Poesie, oder besser: schreiben Sie welche (ohne eckige Klammern oder Semikolons zu nutzen). Lernen Sie ein Musikinstrument das keinerlei Elektrizität benötigt. Wählen Sie sich ein nicht technisiertes Hobby wie Gartenarbeit oder Nähen. Sie sind ein Süchtiger – ich fühle Ihren Schmerz. Als Erholungsstrategie sollten Sie ins Auge fassen, Ihre Fertigkeiten an etwas hinzugeben, das Ihre Zeit und Ihren Einsatz auch wert ist, und das es erforderlich macht, daß Sie mit Menschen zusammenkommen, die absolut keine Ahnung davon haben, wie ein Computer arbeitet. Lassen Sie sich nicht davon aufwühlen, wenn der lokale (*Innen*)minister die Eleganz Ihres erst jüngst getunten LaTeX-Makros zur Erzeugung des BILD Bulletins nicht aufgreift. Tun Sie etwas Gutes für Ihren Körper und Ihren Geist. Ich rate dringend zu wenigstens einem Hochleistungssport (Channel-Surfing reicht definitiv nicht) und zum Schießen mit halbautomatischen Handfeuerwaffen (auf Papierziele, wegen der Legalität) als ausgezeichnetes Mittel zum Freimachen des Kopfes. Sammeln Sie etwas anderes als Computer-Handbücher.

13) **P** = "Ich möchte berühmt sein."

A = Berauben Sie eine Bank und lassen Sie sich fassen, ODER schreiben Sie eine Tonne wirklich nützlichen Codes und verschenken Sie diesen.

14) **P** = "Ich habe Langeweile, ich brauche eine neue Herausforderung."

A = Wenn Sie mehr als 5 Programmiersprachen kennen, überlegen Sie, den Ledernacken der Navy beizutreten. Die sind zur Zeit sehr beschäftigt und halten immer

Ausschau nach Abenteurern wir Ihnen. Wenn Sie älter als 19 Jahre sind – das scheint mir aber nicht der Fall zu sein – heiraten Sie und setzen Sie Kinder in die Welt. Sie werden nie wieder Langeweile haben.

Tim Daneliuk

"Bach, Sig-Sauer, Gibson Gitarren, Python, & meine Tischsäge haben aus mir gemacht, was ich heute bin."

Übersetzung: Friederich Prinz

Ansprechpartner der Mitglieder: das Forthbüro

Rolf Schöne ist seit der Übernahme der Amtsgeschäfte während der Mitgliederversammlung im Jahr 2003 für die Mitglieder, Freunde und Förderer der Forthgesellschaft e.V. – aber auch für alle „Außenstehende" - DER Ansprechpartner. Rolf Schöne stellt sich uns hier kurz vor.

Rolf Schöne
Erzgießereistraße 27
80335 München,
Tel./Fax (0 89) 1 23 47 84
E-Mail: rolf@rolf-schoene.de
secretary@forth-ev.de

1941 geboren in Stockach/Baden
1957 Mittlere Reife
1960 Facharbeiterprüfung als Feinmechaniker
Tätigkeit als Elektroniker
1963-1971 Bundeswehr (Flugabwehrraketen)
letzter Dienstgrad Feldwebel
1968 Gesellenprüfung als Radio- und Fernsehtechniker, dienstbegleitend
1971 Staatlich geprüfter Techniker der Fachrichtung Elektronik
1972-2001 Tätigkeit am Institut für Angewandte Mathematik der TUM
1976 Fachabitur, berufsbegleitend
2002 im Ruhestand, aber nicht ganz inaktiv.

Mein Zugang zu Forth.

Die Angaben in meiner Bewerbung für diese Tätigkeit sind nicht ganz richtig. Nicht erst 1980, sondern schon 1978 gab es die F.I.G. Schon 1976 gab es den KIM, den mit dem 6502-Prozessor und sattem 1 KB RAM, den ich für die TUM beschaffte. Privat war er mir zu teuer. Also selber bauen (das war damals noch billiger). Schon 1977 hatte ich ein home brewed 6502-System mit (etwas später) sagenhaften 48 KB RAM – aber kein Betriebssystem! Da kam mir die F.I.G. 1978 gerade recht. Reinhacken, auf Maschinencode-Ebene. Was sonst, einen Assembler gab es noch nicht. Aber danach! Und alles lief. Auf einem Mikroprozessor! Gerade mal 9 Jahre nach dem Geniestreich von Charles Moore, der 1969 noch keine Ahnung von Mikroprozessoren haben konnte, hatte ich ein FORTH. Das war privat.

Dienstlich lief leider alles anders.

Schuld daran war Fred Behringer, der 1971 zur Wartung und Reparatur des Hybridrechners der TUM ein Dialogsystem entwickelte und dazu die Sprache DISPRA (Dialogsprache) erfand. Leider hatte der Rechner keinen Stack. Können Sie sich vorstellen, wie man damals Unterprogramme aufgerufen hat? Wenn es damals schon E-Mail und News gegeben hätte, so hätte er von FORTH erfahren, flugs einen Software-Stack implementiert und zusammen mit Charles Moore mindestens ein FIFTH gemacht. So blieb es bei DISPRA.
Es war neben FORTRAN (FOR-WHAT? hat mal jemand gefragt) die erste Hochsprache, die ich kennen lernte, nachdem mich Fred Behringer im Januar 1972 an der TUM einstellte. Und DISPRA war so gut, und war so gut dokumentiert, dass ich es leicht auch für einen neuen Analogteil des Hybridrechners erweitern konnte, nachdem Fred Behringer sich wieder lieber der Wissenschaft zuwandte und prompt Professor wurde. Nicht für Physik, nicht für EDV, auch nicht für Informatik (die gerade im Entstehen war), sondern für Mathematik. Aber das wissen Sie ja.

Was tun?

Können Sie sich vorstellen, in welchem Dilemma ich war? Dilemma ist verkehrt, weil es die Wahl zwischen zwei eher unangenehmen Dingen bedeutet. Aber in einer Zwangslage war ich. Privat freute ich mich über FORTH, wurde auch Mitglied der FG, dienstlich aber klebte ich genauso gern an DISPRA. Und unangenehm war beides natürlich nicht.

Zu entscheiden war nichts mehr, nachdem ein Blitzschlag mit einer Potentialdifferenz von einigen KV zwischen Messerde und Schuko dem Digitalteil den Garaus machte. Er wurde durch einen selbst entwickelten 68000-Rechner mit VME-Bus ersetzt, für den ich zwar ein FORTH hatte (kennt jemand noch den pfälzischen Zahnarzt Dr. Blaich?), aber Echtzeit und C war angesagt, und damit das Aus für DISPRA und FORTH. Dumm wie ich war, bin ich aus der FG ausgetreten.

Eingetreten bin ich wieder im Januar 2002, nachdem mich Fred Behringer darüber informiert hat, dass es für den Roboterbaukasten von LEGO ein FORTH gibt.

01.04.2003
Rolf Schöne

Triceps spielt Solitaire

Auf der Forth-Tagung in Lambrecht hat Karsten Roederer den Solitair-spielenden Triceps-Roboter gefilmt. Einen einminütigen Filmausschnitt gibt es auf der Roboter-Projektseite der Forthgesellschaft zu sehen.

http://www.forth-ev.de/Projekte/roboter.html

U.Hoffmann

Holländisch ist gar nicht so schwer. Es ähnelt sehr den norddeutschen Sprachgepflogenheiten. Und außerdem ist Forth sowieso international. Neugierig? Werden Sie Förderer der

HCC-Forth-gebruikersgroep.

Für 10 € pro Jahr schicken wir Ihnen 5 oder 6 Hefte unserer Vereinszeitschrift 'Het Vijgeblaadje' zu. Dort können Sie sich über die Aktivitäten unserer Mitglieder, über neue Hard- und Softwareprojekte, über Produkte zu günstigen Bezugspreisen, über Literatur aus unserer Forth-Bibliothek und vieles mehr aus erster Hand unterrichten. Auskünfte erteilt:

Willem Ouwerkerk
Boulevard Heuvelink 126
NL-6828 KW Arnhem
E-Mail: ***w.ouwerkerk@kader.hobby.nl***

Oder überweisen Sie einfach 10 € auf das Konto 525 35 72 der HCC-Forth-gebruikersgroep bei der Postbank Amsterdam. Noch einfacher ist es wahrscheinlich, sich deshalb direkt an unseren Vorsitzenden Willem Ouwerkerk zu wenden.

Gustav & SWAP

Gustav ist ein kleines Dankeschön der Forthgesellschaft an Ute Woitzel für fünf Jahre Arbeit im Forthbüro.

Lambrecht 2003
Kultur und Spannung ;-)

FIG UK

(Englische Forth-Gesellschaft)

Treten Sie unserer Forth-Gruppe bei.
Verschaffen Sie sich Zugang zu unserer umfangreichen Bibliothek.
Sichern Sie sich alle zwei Monate
ein Heft unserer Vereinszeitschrift.
(Auch ältere Hefte erhältlich)
Suchen Sie unsere Webseite auf:
www.users.zetnet.co.uk/aborigine/Forth.htm
Lassen Sie sich unser Neuzugangs-Gratis-Paket geben.
Der Mitgliedsbeitrag beträgt 12 engl. Pfund.
Hierfür bekommen Sie 6 Hefte unserer
Vereinszeitschrift Forthwrite.
Beschleunigte Zustellung (Air Mail)
ins Ausland kostet 20 Pfund.
Körperschaften zahlen 36 Pfund,
erhalten dafür aber viel Werbung.

Wenden Sie sich an:

Dr. Douglas Neale
58 Woodland Way
Morden Surrey
SM4 4DS
Tel.: (44) 181-542-2747
E-Mail: dneale@w58wmorden.demon.co.uk

Protokoll

Protokoll der
Mitgliederversammlung der Forth-Gesellschaft
in Lambrecht am 13.04.2003

Die Versammlung beginnt pünktlich um 09:00 Uhr. Die Tagesordnungspunkte aus der Einladung werden noch einmal bekannt gegeben.

TOP 1: Ulli Hoffmann, Direktoriumsmitglied, begrüßt die versammelten Mitglieder der FG. Danach wird der Swap-Drachen verliehen an Thomas Beierlein, übergeben vom Drachenträger des Vorjahres, Hans Eckes. Mit diesem Preis wird alljährlich ein Mitglied der FG geehrt, das sich sehr um den Verein und die Verbreitung von Forth verdient gemacht hat.

Top 2 und 3: Vorgeschlagen werden als Protokollant Michael Kalus und als Versammlungsleiter Heinz Schnitter, die beide annehmen. Es waren 22 von 115 stimmberechtigten Mitgliedern anwesend, womit Beschlussfähigkeit gegeben war.

TOP 4: Zum Punkt "Verschiedenes" werden Beiträge und Fragen auf der Tafel aufgelistet: CD-ROM-Projekt, Website, Prototyping-Board zum uCore, neues VolksForth?, Mitglieder- und Mailing-Listen, weitere Förderprojekte? Forth-Bücher-Neuauflagen. Andere Änderungen der Tagesordnung wurden nicht gewünscht.

TOP 5: Bericht des Direktoriums. Die jeweiligen Beauftragten tragen vor. Auf den gemeinnützigen Charakter der Aktivitäten wird hingewiesen.
- Friederich Prinz, Editor der VD, berichtet über seine ehrenamtliche Tätigkeit bei der Herstellung der VD. Derzeit keine gravierenden Probleme. Er setzt seine Arbeit fort. Die Versammlung dankt ihm herzlich für seine umfangreiche Arbeit. Friederich Prinz bat besonders die Vortragenden der Tagung um ihre Beiträge für die VD.
- Egmont Woitzel stellt die Ein- und Ausgaben gegenüber und berichtet zu den einzelnen Posten (Einen ausführlichen Finanzbericht erhalten Mitglieder im Forthbüro). Eingenommen wurden rund 12000 Euro, ausgegeben nicht ganz 11000 Euro, Jahrestagung und VD eingeschlossen. An Rücklagen haben sich inzwischen rund 22000 Euro angesammelt. Die Prognose für das nächste Jahr ist gut, ein kleiner Überschuss wird erwartet. Der Haushalt ist ausgeglichen. Die Kassenprüferin Liesl Rohrmayer fand eine ordentliche Buchführung vor, die Zahlen waren sämtlich belegt und nachvollziehbar.
- Pause, 10:00 Uhr: Gruppenfoto.
- Fred Behringer: Ehrung von Ute Woitzel für die Arbeit im Forthbüro. Überreichung des Abschiedsgeschenkes - der Bär "Gustav". Zum Nachfolger im Forthbüro bestellte das Direktorium Rolf Schöne, der sich der Versammlung vorstellte und sich bereit erklärte.
- Ulli Hoffmann berichtete von der Homepage des Vereins, welche seit Jahren stabil und ohne Störungen gehostet ist bei Schlund und Partner. Transfervolumen derzeit ca. 10% der 50 GB. Technisch gesicherte Präsenz. Weitere Inhalte sind gefragt. Eine E-Mail-Adresse für Vereinsmitglieder wird mit angeboten.
- Fred Behringer berichtete von der Auslandsarbeit. Enge Kontakte nach USA (H. Vinerts, C.H. Ting, SVFIG) und England (Ch. Jakeman, Forthwrite, G. Dunbar, Buchverleih). Die Kontakte nach Holland sind gut (W. Ouwerkerk). Fred Behringer hat das ANS-Forth-Kursbegleitbuch von Albert Nijhof (holländische FG) übersetzt. Letzterer und Rolf Schöne haben die elektronische Fassung vorbereitet und Korrektur gelesen. Es soll ohne Gewinn bei Books on Demand und kostenlos übers Internet verbreitet werden. Wechselseitige Mitgliederwerbung in der VD und in den Vereinsblättern in England und Holland. Regelmäßige Besprechung der VD in der Forthwrite (Joe Anderson) und der Forthwrite und des Vijgeblaadjes in der VD. Der Kontakt zu Marc Petremann müsste aufgefrischt werden. Keine französische FG.
- Bernd Paysan berichtet vom Micro-Controller-Verleih: "... keine Nachfrage mehr." Es wurden daher keine Neuanschaffungen getätigt. Derzeit wird versucht, die Rechte am Forthlehrbuch "Starting Forth" von E. Rather zu bekommen, um eine aktualisierte Auflage für Einsteiger zu machen. U. Hoffmann und B.Paysan wollen für eine gedruckte Fassung sorgen. Die Rechte an der deutschen Übersetzung des bisherigen nicht aktualisierten Buches liegen beim Hanser-Verlag und sind für die Neufassung noch unklar.

TOP 6: Die Versammlung entlastet das Direktorium ohne Gegenstimmen bei zwei Enthaltungen.

TOP 7: Das bisherige Direktorium wurde mit einer Gegenstimme und einer Enthaltung bestätigt.

TOP 8: Mitgliedsbeiträge: Ulrich Hoffmann schlägt vor, die Jahresbeiträge nach der Senkung vom letzten Jahr nun stabil zu lassen. Der Haushalt sei damit ausgeglichen. Fred Behringer weist auf das Vermögen hin, möchte die Beiträge senken. Die Teilnehmer weisen auf den 4%igen Mitgliederschwund im letzten Jahr hin, der Trend schwächt sich allerdings derzeit ab. Nach Diskussion wird über den Antrag von Klaus Schleisiek abgestimmt: Der Mitgliedsbeitrag soll bestehen bleiben. Dies wird angenommen mit 2 Gegenstimmen und ohne Enthaltung.

TOP 9: Verschiedenes.
- Bericht zur Steuererklärung, VD-Behandlung in ermäßigter Mehrwertsteuer nach wie vor. Fragen zum Verkauf der VD wurden vom Direktorium beantwortet.
- Es wird gewünscht, das Direktorium im Impressum der VD namentlich aufzuführen. Dies wird mit Mehrheit befürwortet.
- Nächster Tagungsort wird sein auf der Insel Fehmarn in der Ostsee. Ulli Hoffmann wird das 20. Treffen dort ausrichten, voraussichtlich Ende April des Jahres. (Bitte nicht in die Oster-Ferien legen!).
- CD-ROM Projekt. Ulli Hoffmann stellt den bisherigen Stand vor. Die versammelten Mitglieder erhalten je ein Exemplar. Weitere Sammlung zu der Idee: VD-Archiv komplettieren, CD kostenlos abgeben, auch an Ehemalige, auch an die Interessenten im Ausland.

Kritik: Es sei eine Retrospektive. Gegenrede: Geschichte der FG zum 20. Jahr ist allen wichtig.
Das Direktorium soll für das Scannen einen Endtermin setzen. Falls bis dahin nicht genug Freiwillige gefunden werden, soll jemand beauftragt werden, das restliche VD-Archiv zu scannen; der Vorschlag wird zur Abstimmung gebracht und mit 2 Gegenstimmen und 2 Enthaltungen angenommen.
- Die Mitgliederliste soll wieder bereit gestellt werden. Es wird dabei auf de.comp.lang.forth hingewiesen, um Forth Kontakte zu finden, und auf die Homepage; Ulrich Hoffmann bietet an, die Verbindung zwischen Neuigkeiten in der Homepage und dem Usenet herzustellen. Doch auch die Papierform soll wieder erstellt werden. Das Forthbüro wird gebeten die Liste zu erneuern. Friederich Prinz will mit der nächsten VD wieder die Mitgliederliste versenden, mit Telefonnummern und E-mail Adresse wie bisher.

Sodann wurden weitere Förderprojekte vorgestellt/vorgeschlagen.
- Knoppix-Forth-Projekt, Ulli Hoffmann: Er stellt die technische Studie dazu vor; diese ist gelungen. Unterstützung wird gesucht; einstimmig wurde eine finanzielle Unterstützung durch die FG beschlossen.
- uCore-Projekt, Klaus Schleisiek: Er regt die Diskussion dazu an; verschiedene Überlegungen werden vorgestellt. Es wird schließlich beschlossen mit 1 Gegenstimme und 1 Enthaltung, den Start des Projekts zu fördern durch 20 Platinen des Evaluation-Boards; diese sollen vorfinanziert werden; Größenordnung 2K-Euro. Als Warenzeichen soll "uCore" auf den Verein eingetragen werden. Auch dieser Vorschlag wurde mit 1 Gegenstimme und 6 Enthaltungen angenommen. (Rechtsprüfung durch den Vorstand vorbehalten.)
- Soll es wieder ein "VolksForth" geben? Herr Reilhofer plädiert stark für die Rückkehr zu einfachem Forth auch auf den modernen Oberflächen und projektiert, ein solches im nächsten Jahr vorzustellen (Applaus!).
- Forth-Systeme: Als derzeitige Referenz vorgeschlagen wurde das Gforth. Aber auch Bedenken dagegen wurden vorgetragen, weil GForth keine 'Blinkware' sei, daher BigForth als solches besser geeignet scheint. Manfred Mahlows "At Last" wird vorgeschlagen als ein "Starting Forth", minimal und mit file interface.
- Stand der GForth-Erfahrungen, Jens Wilke: Der Trend geht auch dort zu einem simplen Startup-System. Fragen sind offen bezogen auf die Portabilität. Die Unterstützung von Windows stellt sich als besonders problematisch heraus. Der "ecKernel" ist Public Domain. Dies könne auch nicht anders sein, da es sich ohnehin um die Aufarbeitung von Forth-Allgemeingut handele. Zur Compilierung werde allerdings der Cross-Compiler von Gforth verwendet. Dieser sei und bleibe GPL. Das ist aber kein Hindernis zur Verwendung des "ecKernel", denn die erzeugten Systeme seien frei von jeglichen Lizenzen.
- Es wird vorgeschlagen, das Forth-Lehrbuch "Starting Forth" solle als "Book on Demand" hinterlegt werden. Es stellte sich heraus, dass die Technik dazu bereitsteht. Die Versammlung beschließt einstimmig, das Direktorium solle prüfen, ob die Rechte erworben werden können und zu welchen Kosten, und gegebenenfalls entscheiden.
- Mikrocontroller: Nach einiger Zeit "Ruhe an der Front" mangels geeigneter Systeme scheint sich nun wieder was zu tun, z. B. in der Forthquelle bei Flesch gibt es ein neues Board.
- Anregung weiter Projekte, Martin Bitter: Die FG habe ca. 180 Euro pro Mitglied zur Verfügung, könne daher z.B. irgendeine forthige Jahresgabe (ähnlich ADAC-Karten etc) (auch Freds Übersetzung des Forth-Kurses von Albert Nijhof?) herausgeben; Lizenzen für z.B ByteForth der niederländischen Freunde kaufen und verbilligt an Mitglieder weitergeben - dito für das F11-FIGUK-Board; regionalen Forth-Gruppen (z.B. die um Ewald Rieger) könnten Ausgaben erstattet werden; die FG finanziert die Herstellung von Bausätzen (ähnlich dem von Ewald), die dann unter dem Herstellungspreis an (gemeinnützige Träger) Schulen usw. abgegeben werden. Das Direktorium nahm die Anregungen dankend auf, es wird darauf hingewiesen, dass Anträge auf Förderung solcher Projekte gestellt werden können.
- Triceps-Roboter, Ewald Rieger: Dieser kann als Bausatz zur Verfügung gestellt werden; Förderung durch die FG; mit 5 Enthaltungen angenommen.
- Ein Anschreiben zur Wiederanwerbung an die ehemaligen Mitglieder wird gewünscht, diskutiert und zum Antrag gebracht; dies wurde mit 6 Ja-Stimmen und 5 Enthaltungen abgelehnt.

Abschließend sprach Ulrich Hoffmann im Namen der FG den herzlichsten Dank aus an Ewald und Andrea Rieger für die Ausrichtung der Tagung in Lambrecht (anhaltender Applaus der Versammlung).

Ende der Sitzung um 12:30 Uhr.

Michael Kalus, 26.04.2003

SWAP fährt nach Sachsen

Der nachfolgende Beitrag basiert vermutlich auf einer der ältesten Veröffentlichungen zu Forth überhaupt. In den "Proceedings of the IEEE, Vol. 61, No. 9, September 1973" wurde der Artikel "The FORTH Program for Spectral Line Observing" von Charles H. Moore und Elizabeth D. Rather abgedruckt. Elf Jahre vor der Gründungsveranstaltung der Forthgesellschaft e. V. haben Moore und Rather ihr FORTH und ihre Arbeit für das NRAO beschrieben. Die Übersetzung dieses Artikels, den Rafael Deliano für die Forthgesellschaft aus seinem Fundus ausgegraben hat, ist aber keineswegs ein Blick zurück im Sinne von Vergangenheitsbewältigung. Tatsächlich ist die von Moore und Rather beschriebene Vorgehensweise, die Strukturierung der Arbeit und Aufgaben am NRAO und die Arbeit mit Forth heute so aktuell wie nie. Die in der Zusammenfassung gegebene Begründung für den Einsatz von FORTH ist heute ebenso zutreffend, wie sie es "damals" war.

Die Übersetzung wurde von Friederich Prinz angefertigt. Der Originaltext kann bei der Redaktion der VD im GIF-Format (Scans des Originaldrucks) angefordert werden.

Das FORTH Programm zur Beobachtung von Spektrallinien

Charles H. Moore
und
Elizabeth D. Rather

Abstrakt – Das 11-m-Teleskop des National Radio Astronomy Observatory (NRAO) auf dem Kitt Peak wird ausgiebig zur Beobachtung von Spektrallinien im Millimeterbereich genutzt. Die Computer, die das Teleskop und den Empfänger steuern, sind in FORTH programmiert. Diese Sprache bietet dem Astronomen ein ausführliches Vokabular zur Steuerung seiner Beobachtungen und zur Auswertung seiner Daten. Hier beschreiben wir zusammenfassend die Fähigkeiten der gebräuchlichsten Worte dieses Vokabulars; Fähigkeiten, die von der Führung des Teleskops bis zur Bearbeitung von Profilen für Spektrallinien reichen. Es muß aber erwähnt werden, daß die Anzahl der Worte, die für eine hochwertige Beobachtung benötigt werden, bereits groß ist und weiter wächst.

Einführung

Die Untersuchung von Spektrallinien interstellarer Moleküle ist aktuell ein wichtiger und aufregender Bereich der Radioastronomie. Die Linien im Wellenlängenbereich mehrerer Millimeter, werden von Astronomen vieler Organisationen mit dem 11-m-Teleskop des NRAO auf dem Kitt Peak in Arizona beobachtet. Obwohl verschiedene Empfänger für die Beobachtungen eingesetzt werden, teilen sich alle das gleiche System zum Empfang, zur Speicherung und zur Bearbeitung der Daten. Die dabei verwendete Ausrüstung wird in der Abbildung 1 dargestellt.

Als das System 1970 entworfen wurde, war man sich darüber im Klaren, daß die Möglichkeit, die anfallenden Daten direkt am Teleskop zu kalibrieren und zu analysieren, die Effizienz der Beobachtungen wesentlich steigern würde. Tatsächlich bot das on-line-Programm SPECTRA solch außerordentliche Möglichkeiten zur Bearbeitung der Daten, daß viele Anwender die off-line-Bearbeitung der Daten als überflüssig ansahen. Obwohl die Spektren auf Magnetbänder aufgezeichnet wurden, war die bevorzugte Datenausgabe darum ein 8 ½ * 11 Zoll großer Bildschirmabdruck des Tektronix 4002 Graphikterminals. Der Computer kennzeichnet das ausgegebene Bild. Der Anwender kann es zusätzlich auf dem Bildschirm kommentieren, so daß Spektren in exakt dem jeweils gewünschten Format dokumentiert werden können. Die Abbildungen 2 bis 4 sind direkte Wiedergaben der Systemausgaben (*Anm. d. Übers: aus Gründen der Treue zum Original wurden die Abbildungen nicht geändert oder aufgearbeitet*).

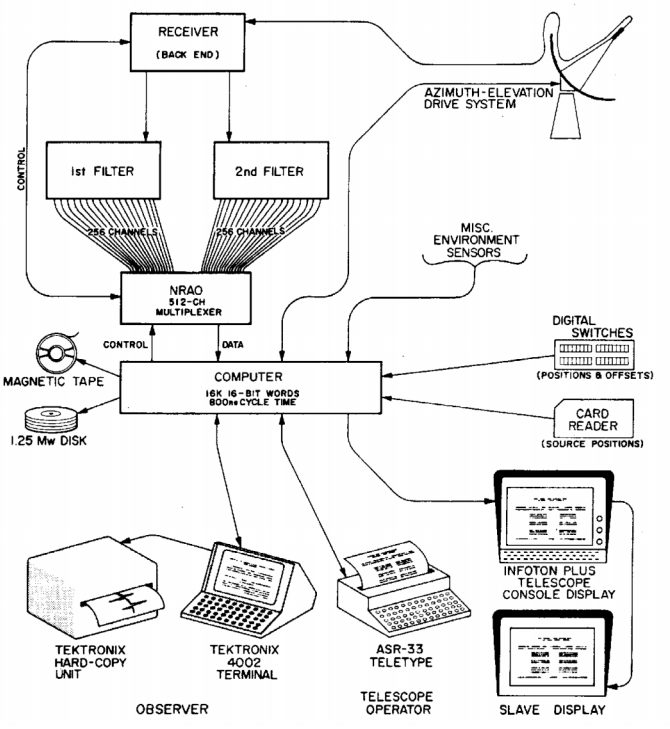


Abb.1 NRAOs Steuerung des 11-m-Teleskops und das System zur Beobachtung der Spektren im Millimeterbereich.

Der Computer ist in FORTH programmiert, in einem Software-System, bzw. in einer Sprache, die insbesondere für interaktive Applikationen entwickelt wurde [1]. Das ermöglicht eine Multi-Programm-Umgebung aus vier miteinander konkurrierenden Tasks: Teleskop-Steuerung, Daten-Erfassung, ein Drucker-Terminal für den Teleskop-Fahrer und ein graphisches Terminal für den Astronomen. In diesem Artikel werden wir beschreiben, wie einige der einzigartigen Merkmale von FORTH dazu beitragen, daß die Nutzer mit der gesamten Ausrüstung interagieren und die Datenerfasssung und -auswertung flexibel und effizient gestalten können. Detaillierte Informationen können einem internen Bericht des NRAO entnommen werden [2].

Besonders bei einem Programm von der Komplexität SPECTRAs ist eine anwenderorientierte Dokumentation essentiell. Unsere Dokumentation ist so organisiert, daß das dem Astronomen bekannte Vokabular allmählich wächst, wobei in gleichem Maße die Präzision fortschreitet, mit welcher er den Computer steuern kann. Die erste Stufe ist eine zweiseitige Zusammenfassung des minimalen Vokabulars, das für Beobachtungen benötigt wird. Sobald sich die Anfangskonfusion ein wenig gelegt hat, kann er Worte kennenlernen und ausprobieren, die mit dem Display, der Datenbearbeitung und der Beobachtung des Himmels verbunden sind. Weil mehrere Programme gleichzeitig arbeiten können, kann der Anfänger Worte ausprobieren, Daten auf den Bildschirm ausgeben, Abkürzungen für Worte definieren – all dies, ohne mit laufenden Beobachtungen oder Steuerungen des Telekops in Konflikt zu kommen.

Benutzer-Schnittstelle

Eine Hauptaufgabe des Computers ist es, Schnittstelle zwischen Anwender und Ausrüstung zu sein. Die meisten unserer Beobachter kommen nur wenige Male im Jahr zu uns. Das erfordert eine Schnittstelle, die leicht zu erlernen, leicht zu bedienen und leicht zu merken ist. Eine konfigurierbare Steuerkonsole ist eine der möglichen Optionen, und wird tatsächlich auch für die Teleskopsteuerung benutzt. Aber das ist wirklich schwierig handhabbar und unflexibel - und ungeeignet für eine frei zu definierende Datenverarbeitungsaufgabe. Deshalb nutzen wir die Tektronix-Tastatur als Eingabegerät für den Astronomen. Dieses Terminal stellt ein sehr großes Vokabular zur Verfügung, das hauptsächlich aus englischen Worten besteht (CALIBRATE, RANGE) und aus einfachen Abkürzungen (SEC für SECOND). Das Vokabular ist für eine einfache Steuerung des Computers durch Worte und Phrasen ausgelegt. FORTH ist dabei gegenüber Tipfehlern tolerant und ermöglicht dem Operator, komplette Eingaben zu verwerfen oder mit der Rücktaste zu korrigieren.

SPECTRA arbeitet mit Worten, die vollständig ausgeschrieben werden. Gewöhnlich ist die tatsächliche Kürze von Abkürzungen ein direktes Maß für die Schwierigkeit, sich eine Abkürzung ebenso gut zu merken wie das eigentliche Wort selbst. Dessen ungeachtet ist die Möglichkeit, Abkürzungen beim Aufbau von Definitionen zu bilden, ein integraler Bestandteil von FORTH. Tatsächlich ist der größte Teil von SPECTRA unter Verwendung von High-Level Worten definiert worden. Und der Anwender wird ermutigt, diesen Definitionen solche hinzuzufügen, die ihm selbst zusagen. Zum Beispiel könnte er sich entscheiden, wiederholt eine Sequenz von 4 ein-aus-Messungen durchzuführen. Um eine Sequenz zu starten würde er schreiben

```
4 ON-OFF
```

Wenn er möchte, kann er das Wort “G” definieren

```
: G  4  ON-OFF ;
```

woraufhin die Eingabe von “G” die darunter definierte Sequenz starten wird. Die Syntax zum Aufbau einer Abkürzung ist einfach: Dem zu definierenden Wort wird ein Doppelpunkt vorangestellt und die Befehlssequenz angefügt. Abgeschlossen wird das Ganze durch ein Semikolon.

Teleskop Steuerung

Das 11-m-Teleskop verfügt über eine Azimuth-Höhenverstellung. Das bedeutet, daß der Computer äquatoriale Koordinaten in lokale Koordinaten umrechnen muß. Das Teleskop selbst wird von einer Konsole aus gesteuert. Diese Konsole beinhaltet Schalter, eine Tastatur und einen Kartenleser. Der Kartenleser kann Eingabetexte ebenso interpretieren, wie FORTH Eingabetexte der Tastatur interpretiert. Ein hierfür definiertes Vokabular eröffnet dem Beobachter eine außergewöhnlich weitgehende Kontrolle über die Bewegungen des Teleskops.

Eine der parallel arbeitenden Aufgaben ist die kontinuierliche Bewegung des Teleskops mit einem bestimmten Himmelskörper. Richtungskorrekturen werden periodisch und automatisch durchgeführt. Dabei wird eine Spurgenauigkeit von +- 5” erreicht. Jeder Empfänger hat eine ihm eigene Strahlabweichung, die bei seiner Montage angegeben und spezifisch kompensiert wird. Der daraus resultierende Führungsfehler des Teleskops wird angezeigt.

Die Datenaufnahme wird automatisch unterbrochen, wenn dieser Fehler eine vorgegebene Grenze überschreitet (10”). Damit wird auch verhindert, daß z.B. Windböen eine lange andauernde Beobachtung verderben. Die Beobachtungen werden automatisch gestartet (oder fortgesetzt), wenn das Teleskop seine vorgegebene Position erreicht hat.

Das einfachste Steuerungskommando wird wie folgt über die Tatstatur eingegeben (oder in den Kartenleser gestanzt),

```
20:38:02.2   42:13:32.   CURRENT   DR 21
```

womit die aktuelle Rektaszension und Deklination (*siehe Glossar, der Übers.*) der Quelle DR21 beschrieben wird. Das Teleskop wird sich auf diese eindeutig angegebene Adresse ausrichten und diese so lange verfolgen, bis ein Änderungskommando gegeben wird. Alternativ beschreibt

```
20:37:14.2   42:09:07.   1950   EPOCH   DR21
```

die Position der gleichen Quelle in 1950er Koordinaten, auf welche die Präzessions-, Nutations- und Aberrationskorrekturen (*siehe Glossar*) addiert wurden.

Ebenso können galaktische Koordinaten spezifiziert werden,

```
81.6794   0.5391   GALACTIC   DR21
```

sofern diese exakt zur Verfügung stehen.

Um einen Planeten zu verfolgen, benötigt man dessen Position und seine Umlaufdaten, die in den amerikanischen Ephemeriden aufgeführt sind. Entsprechende Tabellen stehen für das jeweils aktuelle Jahr zur Verfügung.

Die Worte ZENITH und SERVICE bewegen das Teleskop an festgelegte Positionen.

Zusätzlich zur Verfolgung einmal fixierter, sich gleichförmig bewegender Quellen sind verschiedene, anspruchsvollere Beobachtungsabläufe möglich.

Das Wort ON-OFF weist das Teleskop an, zwischen dem zu beobachtenden Objekt und dem jeweils nahen, leeren Himmel hin und her zu schalten, um systematische und atmosphärische Fehler zu kompensieren. Die Weite des Abstandes dieser Meßbewegungen wird vom Beobachter vorgegeben und kann so gewählt werden, daß sie für Zweistrahl-Empfänger paßt, wobei dann die Strahlen wechselseitig auf die Quelle weisen.

Eine weit verbreitete Art der Beobachtung ist das Zeichnen von Karten einer bestimmten Himmelsregion. MAP führt darum eine Raster-Abtastung in äquatorialen oder galaktischen Koordinaten relativ zu einer vorgegebenen Quellen-Position durch. Die Abmessungen des Rasters und die zeitliche Integration der Aufnahme werden vom Beobachter gesteuert. Die dabei erhaltenen Daten werden kalibriert und formatiert und in einer angenehm lesbaren Form auf den Bildschirm ausgegeben.

Um eine punktförmige Quelle möglichst exakt zu lokalisieren, zeichnet das Wort FIVE Messungen auf, die bei fünf Punkten nördlich, südlich, zentriert, östlich und westlich der Quelle durchgeführt werden. Das kann dem Beobachter zumindest genauere Informationen zu Azimuth, Steigungswinkel, Strahlweite und Peak-Temperatur zu einer gegebenen Quelle verschaffen. Gewöhnlich wird das für Kontinuum-Messungen genutzt, kann aber natürlich auch hilfreich sein, um Ausrichtungskorrekturen bei der Messung von Spektrallinien durchzuführen.

Ähnliche Abtastmuster können leicht definiert werden, was Anpassungen an speziellere Aufgaben ermöglicht.

Die atmosphärische Extinktion wird von EXTINCTION gemessen. Zunächst werden elf Messungen bei sechs unterschiedlichen Höheneinstellungen durchgeführt. Eine Funktion mit einem linearen Term über die Abweichung der Empfängerleistung und einem exponentialen Term für die Signal-Abschwächung, passen die Datenaufnahme an die aktuellen Gegebenheiten an. Danach werden alle eingelesenen Daten automatisch an die Extinktion der aktuellen Beobachtung angeglichen.

Alle diese Routinen sind als high-level Definitionen kodiert. Jede steuert das Teleskop entsprechend den jeweiligen Erfordernissen und verarbeitet die Rohdaten bereits signifikant. Die Folge davon ist eine effiziente Beobachtung und eine sofortige Rückmeldung an den Beobachter bezüglich der Qualität seiner Daten.

Empfänger Steuerung

Der Astronom kann aus einer Reihe verschiedener Ausrüstungen und Techniken wählen. So wie hier beschrieben, wurde die Spektrallinien-Beobachtung im Jahr 1973 mit zwei Filterbänken mit 256 1-MHz und 256 0,25-MHz Kanälen durchgeführt. Diese nahmen Signale von Empfängern mit austauschbaren Mixern auf (von denen einige von den Besuchern mitgebracht wurden), die das 2 bis 7 mm Band abdecken.

Spektrallinien deren Antennen-Temperaturen nur ungefähr 1 K betragen, werden von einem 1.000 K Rauschen überlagert. Darum müssen Verfahren angeboten werden, mit denen sich dieses Rauschen unterdrücken läßt.

Signal und Referenzspektren wechseln einander mit 10 Hz ab. Der Computer subtrahiert das Referenzspektrum vom Signal. Referenzspektren werden entweder durch das Aufschalten der Frequenz einer heißen Quelle erzeugt, oder durch Bewegung des Strahls über die Ansteuerung eines nutierenden Subreflektors.

Zusätzlich kann die Antenne auf die Quelle gerichtet werden, und mit einer Frequenz von 0,1 Hz zwischen der Quelle und einer Position unmittelbar daneben wechseln.

In der Praxis wurden die besten Meßergebnisse durch eine Kombination der Antennenbewegungen und der Frequenzschaltungen erzielt. Das erfordert von dem Programm allerdings die Integration von vier 512-Wort-Arrays und die Kombination der darin enthaltenen Daten zur Erzeugung der Referenzspektren.

Empfänger lassen sich grob in zwei Kategorien bezüglich ihrer Kalibrierungen aufteilen. Einige stellen 400 K Temperatur auf einem Einzel-Seitenband dadurch zur Verfügung, daß zwischen “leerem Himmel” und “heißem Punkt” mit einem Flügelrad sehr schnell “umgeschaltet” wird. Andere stellen 100 K Temperatur zur Verfügung, wenn der Computer eine Rauschquelle ein und aus schaltet. In wieder anderen Fällen wird das Spektrum auf Band aufgezeichnet, einfach als kalibrierter Datenstrom in jedem Empfangskanal.

Zusätzlich stehen zwei Formeln zur Verfügung, um Temperaturen zu berechnen. Die Auswahl hängt teilweise von der Art des verwendeten Empfängers ab, ist aber ein wenig auch Geschmacksache. Drei Kombinationen der Kalibrierungstechniken und der Temperaturberechnungen werden standardmäßig angeboten. Die Eingabe von NRAO oder CHOPPER oder BTL treffen die gewünschte Auswahl. Eine neue Kombination könnte in kurzer Zeit definiert und zur Verfügung gestellt werden, allerdings nicht von dem durchschnittlichen Beobachter.

Um eine Beobachtung zu beginnen, spezifiziert der Astronom die Position der Quellen, die gewöhnlich in den Kartenleser gestanzt wird, sowie die Spektrallinien-Frequenz und die Geschwindigkeit der Quelle, die meist über die Tastatur in das System eingegeben werden. Der Computer berechnet die Geschwindigkeit der Erde relativ zu der Quelle und stellt die lokalen Oszillatoreinstellungen zur Verfügung (oder überprüft diese Einstellungen).

Der Astronom muß nur den Empfänger einstellen, einige Schalter setzen und CALIBRATE über die Tastatur eingeben, damit eine Kalibrierung durchgeführt und deren Ergebnisse auf

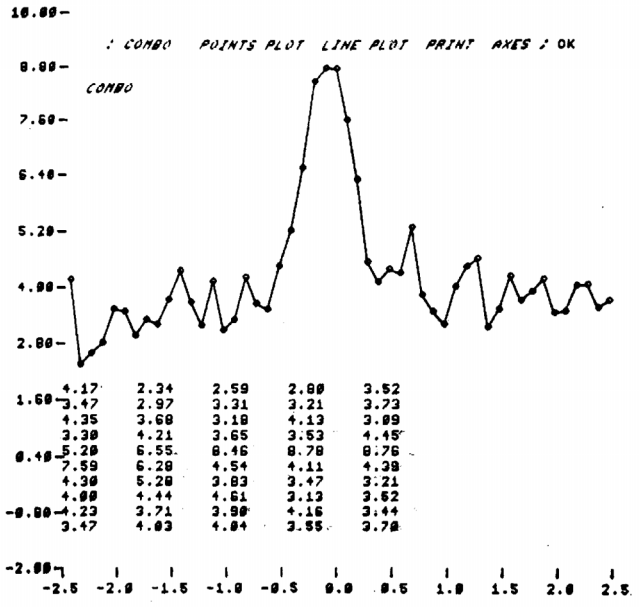


Abb.2 Bildschirmausgabe, produziert von dem Wort COMBO: ein Spektrum, dargestellt als Punkte mit verbindender Linie, mit Vertikalachse in Kelvin und Horizontalachse in Megahertz. Die Werte der Punkte werden zusätzlich tabellarisch ausgegeben.

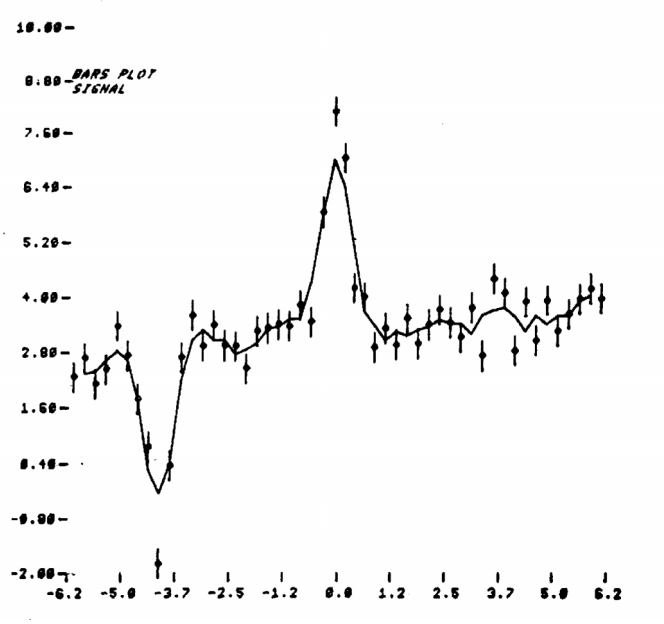


Abb. 3 Ein Spektrum, dargestellt mit der Standardabweichung. Die Ausgabe von SIGNAL ist überlagert: Die Punkte werden geplottet und mit einer Linie verbunden, die bereits der Fehlerkorrektur entsprechend angepaßt wurde.

dem Bildschirm ausgegeben werden. Das Ergebnis ist ein Spektrum des Systemrauschens. Danach setzt er das System zurück und schreibt "1 ON-OFF" um eine Messung zu beginnen, die sich sowohl auf eine bestimmte Position bezieht, als auch auf eine bestimmte Frequenz.

Verschiedene Parameter beeinflussen die Beobachtung. Die Werte dieser Parameter werden von dem Wort STATUS ausgegeben und können leicht verändert werden. Zum Beispiel,

```
30 SEC !
```

setzt die Integrationszeit (Zeit der Datenaufnahme) auf 30 Sekunden. Wichtige Parameter werden gemeinsam mit den Daten aufgezeichnet.

Ausgaben

Die gleiche Bedeutung wie der korrekten Spezifizierung einer Messung kommt der Prüfung der Gültigkeit dieser Messung zu. Eine Hauptaufgabe von SPECTRA ist deshalb die Ausgabe von Daten in einer lesbaren und der Mehrzahl der Beobachter entgegenkommenden Weise.

Der Anwender kann deshalb aktuell aufgezeichnete Daten auf dem Bildschirm anzeigen lassen, ebenso wie er früher aufgezeichnete Daten anzeigen lassen kann. Er kann auch aufgezeichnete Spektren miteinander kombinieren, verbunden über die Zeit oder die Frequenz.

Weil zwei Spektren gleichzeitig gültig sein können, wird das tatsächlich relevante Spektrum durch den ersten oder den zweiten Filter des 1-MHz, bzw. des 0,25 MHz Kanals spezifiziert.

Die Eingabe von PLOT wird das letzte aufgezeichnete Spektrum neu zeichnen, wenn sich dessen Parameter verändert haben, oder wenn die Bildschirmausgabe zerstört wurde (Röhren-Bildschirme überlagern Spektren oft vollständig). Das Wort BOTH stellt beide Spektren (mit unterschiedlichen Auflösungen) auf der gleichen horizontalen Frequenzachse dar. Diese Achse wird durch 1ST FREQUENCY und 2ND VELOCITY gesteuert, die entweder Frequenz- oder Geschwindigkeitsskalen vorgeben. Ebenso setzt die Anweisung 0 1000 RANGE den vertikalen Temperaturbereich auf 0 bis 10,00 Kelvin. Gleichzeitig mit den Daten wird die Standardabweichung des einlesenden Kanals aufgezeichnet. Diese wird aus der Bandweite des Kanals und der Integrationszeit errechnet. Das Wort BARS zeichnet die eingelesenen Daten darum mit den Fehlerbalken der Standardabweichung. Diese Fehler werden in den Analyseworten, die weiter unten diskutiert werden, korrekt weitergeführt. Alternativ kann der Beobachter POINTS oder LINE oder HISTOGRAM definieren, um ein entsprechendes Plotformat auszuwählen. Gewöhnlich spezifiert jeder Beobachter selbst ein Ausgabeformat, das er mag. Zum Beispiel,

```
: COMBO   POINTS PLOT LINE PLOT PRINT AXES ;
```

erzeugt die Ausgabe, die in der Abbildung 2 dargestellt wird.

Spektren, die über mehrere Tage hinweg aufgenommen wurden, können über ihre Aufnahmenummern in einer Tabelle miteinander kombiniert werden. Die Eingabe von SHOW stellt alle ausgewählten Spektren in Tabellenform dar. Diese sollten einander sehr ähnlich sein, trotz des permanenten Hintergrundrauschens, und unsaubere Daten sollten deutlich sichtbar sein. Das Wort COMBINE wird diese Aufnahmen zu einem einzigen Spektrum zusammenfassen, entsprechend der Aufnahmezeit

und ausgerichtet an den Frequenzen jedes einzelnen Spektrums. Dieses Spektrum ist dann das Endergebnis von SPECTRA. Es kann gekennzeichnet werden, einen Titel beigestellt bekommen, mit Kommentaren versehen werden, und dies alles in einem prinzipiell bereits veröffentlichbaren Format (Lesbarkeit bedeutet nicht die Einhaltung von Standards bei Veröffentlichungen).

Das letzte gezeichnete Spektrum wird in einem Ausgabe-Puffer zwischengespeichert. Es gibt verschiedene Worte, die mit diesem Puffer arbeiten, um zu versuchen, von seinen Inhalten weitere Informationen zu erhalten. SMOOTH wird das Spektrum im Puffer ordnen und die Ergebnisse auf den Bildschirm ausgeben. SMOOTH bearbeitet die Daten im Puffer mit einer 7-Punkt Funktion. Die Eingaben von HANNING oder BOXCAR oder BAND-LIMIT wählen ein Dreieck mit drei Punkten aus, ein Rechteck mit drei Punkten oder eine 7-Punkt-Abschätzung für sin(x)/x. Der Beobachter kann auch eine eigene Funktion definieren. Die Definition des Wortes

```
: SIGNAL   POINTS PLOT LINE SMOOTH AXES ;
```

wird die anzuzeigenden Daten auf den Bildschirm plotten, mit einer den Daten folgenden Linie, wie in der Abbildung 3 sichtbar.

Wie auch immer, die erfolgreichste Vorgehensweise um Informationen aus den aufgenommenen Daten zu gewinnen, ist, ein Modell der Daten entweder auf der Basis bereits bekannter Informationen zu entwickeln, oder auf der Basis visueller Mustererkennung. Die Möglichkeiten der Linienanpassung von SPECTRA sind das deutlichste Beispiel für den Wert der Interaktion zwischen Beobachter und Computer, herbeigeführt durch eine mächtige Sprache. Ein besonderes Vokabular ermöglicht es, eine Funktion zu definieren und an die gewichteten Daten des Puffers anzupassen. Zum Beispiel spezifiziert die Eingabe von PEAK eine einfache Gauss-Verteilung mit einer konstanten Grundlinie.

```
0  100  1000  0  GUESS
```

beschreibt Eingangswerte für die Position, die halbe Höhe, die Amplitude und die Grundlinie. Die Eingabe von CURVE zeichnet die so definierte Funktion, vermutlich aber vollständig verdeckt von Daten, die zuvor ausgegeben wurden. Wenn das Ergebnis aber überwiegend gut mit den Daten übereinstimmt, wird FIT eine einfache Mindestabweichung als Differenz zu den Daten hinzufügen, und die neue Funktion zeichnen. Weil auch eine einfache Gauss-Verteilung eine nichtlineare, von der Grundlinie abhängige Funktion ist, möchte der Beobachter vielleicht AGAIN eingeben, damit eine weitere Iteration berechnet und dargestellt wird. Dies kann er solange tun, bis er mit der Konvergenz seiner Ergebnisse einverstanden ist – oder seine Beobachtung neu startet.
RESULT druckt schließlich die Endergebnisse aller Parameter und deren Standard-Abweichungen, so wie diese aus den Daten entnommen werden.

Um mit der enormen Vielfalt der möglichen Spektren zurecht zu kommen, haben wir Basisfunktionen und mögliche Kombinationen dieser Funktionen definiert. Mehrfache Linien mit

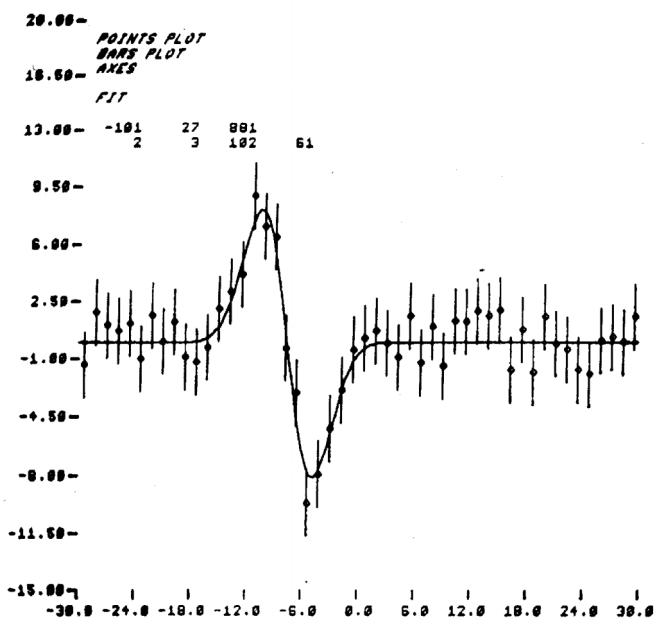


Abb. 4 Ein Spektrum, dargestellt mit Punkten, einschließlich der Standardabweichung (Fehlerbalken), verbunden mit einer Linie, die bereits gegenüber der Normalverteilung korrigiert wurde. Die Zahlen geben die Position, Weite und Höhe des positiven Spitzenwertes wieder, so wie die jeweilige Standardabweichung in der Zeile darunter. Der negative Spitzenwert hat die gleiche Kontur und die gleichen Abweichungen.

Gauss-Verteilungen, gesättigte Normalverteilungen, Lorentz-Profile und konstante, lineare, quadratische oder sinussoidale Grundlinien können ebenso festgelegt werden, wie Kombinationen aus Linien mit festen Trennbereichen und Amplitudenbegrenzungen für sehr feine Beobachtungen. Die einzige Begrenzung für Funktionen ist die Vorgabe von maximal 10 Parametern. Zum Beispiel ist die Funktion der Abbildung 4 wie folgt definiert:

```
: PEAK   4 PARAMETRS ASSIGN START GAUSSIAN
         CONSTANT ;
```

Ein Beispiel einer frequenzbestimmten Normalverteilung (feste Trennung, gleiche Amplituden) wird in der Abbildung 4 gezeigt.

FORTH

Selbstverständlich hätte jedes Merkmal von SPECTRA auch mit konventioneller Technik programmiert werden können. Aber es ist unwahrscheinlich, daß sich das gesamte Programm in einer einzigen Sprache hätte realisieren lassen, außer eben in FORTH. Dafür gibt es verschiedene Gründe.

FORTH ist “kern-effizient”. SPECTRA läuft in weniger als 16k 16-b Worte, ohne Overlays, und enthält, als Erweiterung des Beobachtungs-Vokabulars, den kern-residenten Assembler, Kompiler und Interpreter, die es möglich machen, neue Worte zu definieren. Ein Parameterstapel erlaubt den Verzicht auf

Call-Sequenzen, was den Code sehr kompakt werden läßt. Auf seiner höchsten Ebene arbeitet FORTH mehrfach effizienter im Kern als Assembler-Code.

FORTH ist schnell. Nach außen ist FORTH eine interpretierende Sprache – die Worte, die ein Anwender eingibt, werden in einem Wörterbuch nachgeschlagen und gegebenenfalls ausgeführt. Aber im Gegensatz zu anderen Sprachen kompiliert FORTH seine neuen Definition, so daß deren Ausführung keine wiederholten Zugriffe auf das Wörterbuch benötigt. Deshalb ist der zeitliche Überhang für High-Level-Definitionen äquivalent zu einem Subroutinen-Aufruf, gerade einige, wenige Mikrosekunden lang. Das wiederrum macht es möglich, auch zentrale Teile des Systems, wie die Teleskop-Steuerung und die Datenbearbeitung, als High-Level-Definition zu kodieren, wodurch die Flexibilität und die Verfügbarkeit dieser Definitionen erheblich gewinnen.

FORTH ist interaktiv. Die Bearbeitung der eingelesenen Daten erfordert eine Interaktion zwischen dem Beobachter und dem Computer. Weil FORTH *inhärent* interaktiv ist, ist keine besondere Programmierung notwendig, um die Interaktion zu ermöglichen. Vielmehr ziehen viele Systemfunktionen, bei denen die Interaktion eigentlich nicht notwendig ist, wie zum Beispiel die Teleskopsteuerung, eigenen Nutzen aus diesem Merkmal. So wird es ungeheuer einfach, Fehlfunktionen der Ausrüstung zu diagnostizieren oder neue Ausrüstung zu testen – durch direkte Kommunikation mit dieser Ausrüstung über das Terminal.

FORTH ist reaktionsschnell. FORTH-Programme werden als Quelltexte in der Größenordnung von 100 kByte auf einer Diskette oder auf Band gespeichert. Sie können schnell modifiziert und rekompiliert werden (in Minuten), ohne die Kosten und den üblichen Aufwand für separate Kompiler, Assembler, Lader und Betriebssysteme. Auf diese Weise haben sich über die Jahre, in denen SPECTRA jetzt in Gebrauch ist, die Vorstellungen der Anwender in das Programm aufnehmen lassen, so daß die Summe vieler kleiner Annehmlichkeiten das Programm komfortabel sein läßt. Es kann einfach an neue Hardware angepaßt werden, reagiert auf Änderungen bei der Nutzung der Empfänger und läßt auch Zugriffe auf eigentlich inoperable Ausrüstung zu.

Zusätzlich zu diesen Attributen, die essentiell für SPECTRA sind, sind FORTH's High-Level Definitionen unabhängig vom verwendeten Computer. Eine wachsende Anzahl anderer Observatorien entwickelt zur Zeit FORTH-Programme, sowie für die dortigen Anwender eine ebenfalls wachsende Bibliothek diverser Techniken, einschließlich der hier vorgestellten Routinen.

[1] C.H. Moore,
"FORTH: A new way to program a minicomputer,"
veröffentlicht in Astron.Astrophys. Suppl. Ser., 1973

[2] E.D. Rather und C.H. Moore,
"FORTH programmer's guide,"
NRAOComputer Division Int.Rep. 11, 1972

C.H. Moore wurde 1938 in Pennsylvania geboren. Den Grad eines "Bachelor of sience" erlangte er bei der Arbeit in der Gammastrahlen-Astronomie in 1960 am Massachusetts Institue of Technologie, in Cambridge, Massachusets. Er hat zusätzlich Mathematik und Computer-Wissenschaften an der Stanford Universität in Stanford, Kalifornien studiert. Als Programmierer hatte er mit verschiedenen Applikationen, Sprachen und Computern zu tun. Seine Arbeit beinhaltet ein Programm zur Bestimmung von Satellitenorbits am Smithonian Astrophysical Observatory in Cambridge, in 1959; ein Programm zur optischen Umsetzung von Elektronenstrahlen am Stanford Linear Accelerator Center, 1963; ein Programm zur Steuerung eines Gas-Chromatographen bei Chem Systems, Inc. New York, 1965; ein Management-Informationssystem bei Mohasco Industries, Inc. Amsterdam, N.Y., 1969; und ein Programm zur Datensammlung und Teleskop-Steuerung am National Radio Astronomy Observatory, Charlottesville, Va., 1972. Er ist der Schöpfer der Programmiersprache FORTH für Minicomputer.

Elizabeth D. Rather erhielt ihre A. B. Graduierung in Geschichte von der Universität von Kalifornien in Berkeley, 1967, und die M.A Graduierung in Geschichte mit einer Dissertation über "17th Century Bastardy Proceedings" von der Universität von Kalifornien in Berkeley in 1971. Sie hatte bereits seit 1962 am Oak Ridge National Laboratory mit Computer-Programmierung zu tun, wo sie mit der Datenverarbeitung bei Flug-Experimenten betraut war. Sie hat mehrere Jahre in der astronomischen Abteilung der Universitätr von Kalifornien gearbeitet, wo sie Orbits von Kometen und künstlichen Satelliten der Erde berechnet hat. Anschließend programmierte sie ein komplexes System von Programmen zur Analyse studentischer, akademischer Aufzeichnungen, als Anforderungen für die A.B. Graduierung für das College of Letters and Science in Berkeley. Seit 1971 arbeitet sie am National Radio Astronomy Observatory, Tucson, Arizona, wo sie zur Zeit damit beauftragt ist, Software für das 11-m-Teleskop für Millimeter-Wellenlängen auf dem Kitt Peak in Arizona zu entwickeln.
Frau Rather ist Mitglied der Association for Computing Machinery und Phi Beta Kappa.

Anmerkung der Redaktion: Die vorangestellten Lebensläufe sind dem Aufsatz von Moore und Rather und der Veröffentlichung von 1973 entnommen und entsprechen deshalb selbstverständlich ebenso wenig dem heutigen Stand, wie es die Photographien auch nicht tun. Die Photographien können von 'Sammlern' bei der Redaktion der VD abgefragt werden.

Glossar:

Azimuth: Richtungswinkel zwischen geographisch Nord und der Richtung, in welcher man ein Objekt sieht (auch Marschrichtung). Der Azimut wird im Uhrzeigersinn gemessen, meistens ausgehend von geographisch Süd. Die Richtung NW = 45° links von geographisch Nord entspricht einem Azimuth 135°.

Äquatoriales Koordinatensystem: Ein Koordinatensystem, das es ermöglicht, einen Stern oder einen Satelliten "blind" mit den Teilkreisen der Teleskopmontierung im Teleskop einzustellen. Die für das äquatoriale System bekannten Koordinaten müssen topozentrisch (auf die lokale Position des Beobachters) umgerechnet werden. Letzteres ist einfachere Arithmetik und Arbeit mit diversen Tabellen. Zugehörige Koordinaten sind Rektaszension und Deklination.

Deklination: Die geographische Breite der Erde; entspricht dem Winkelabstand, den ein Himmelskörper zum Himmelsäquator hat.

Rektaszension: Vereinfacht - Höhenwinkel, gibt den von West nach Ost gemessenen Winkelabstand eines Himmelskörpers zum Frühlingspunkt an.

Präzession: Ein nicht kräftefreier Kreisel führt eine Präzessionsbewegung aus, um, vereinfacht gesagt, der zugeführten Kraft senkrecht zur Drehachse auszuweichen. Die Erde ist keine vollkommene Kugel, sondern aufgrund der Rotationskräfte ein Ellipsoid. Sonne und Mond üben auf die "auswölbenden" Bereiche des Körpers (Äquator) Kräfte aus, welche die Erde aus ihrer zur Ekliptik geneigten Drehachse aufrichten und senkrecht zur Ekliptik stellen wollen. Die Erde reagiert auf diese Kräfte mit einer Präzessionsbewegung, die ca. 25.700 Jahre dauert. Die Erdachse "eiert" um einen Kegel mit einem Öffnungswinkel von 23,5° herum.

Nutation: Der Einfluß des Mondes, der die von der Sonne ausgelöste Präzession der Erde stört, wird Nutation genannt. Der Störeintrag der Nutation folgt einem Zyklus von 18,6 Jahren.

Aberration: Die scheinbare Verschiebung eines Himmelskörpers von seiner tatsächlichen Position. Die Bahnbewegung der Erde um die Sonne bewirkt einen Effekt, der auf der konstanten Geschwindigkeit des Lichtes beruht (299.792,5) km/s. Das Licht von z.B. Sternen scheint "unter einem bestimmten Winkel" einzutreffen, der eine scheinbare Verschiebung der tatsächlichen Position von mehr als 20" ausmachen kann.
Ekliptik: Die Bahnebene der Erde um die Sonne.

Ephemerieden: Tabellenwerke mit den Positionen bestimmter Himmelskörper zu einem bestimmten Zeitpunkt. Die Ephemerieden werden auf der Basis genauer Kenntnis von Form, Größe und räumlicher Lage der Umlaufbahn einzelner Himmelskörper (z.B. von Satelliten) errechnet.

Extinktion: Abschwächung von Strahlungsintensität beim Durchlauf von Materie. Sonnenstrahlen sind einer deutlichen Extinktion auf dem Weg durch die Atmosphäre der Erde ausgesetzt. Extinktion ist abhängig von der Frequenz und Intensität der Strahlung und von der zu durchlaufenden Materie (Schwächung durch Absorption, Streuung, Beugung und Reflexion).

Das Glossar wurde zusammengestellt von Friederich Prinz und erhebt keinen Anspruch auf fachliche Exaktheit.

Gehaltvolles

zusammengestellt und übertragen
von Fred Behringer

FORTHWRITE der FIG UK, Großbritannien

Nr. 119, Januar 2003

1 Editorial
Chris Jakeman <cjakeman@bigfoot.com>

Diesmal etwas verspätet, dafür aber ein sehr "dickes" Heft (55 Seiten). Chris begrüßt Philip Eaton (langjähriger Forth-Enthusiast und interessiert in Arcade-Games) als neues Mitglied und übermittelt die traurige Nachricht vom Ableben des (seit mehr als 17 Jahren) aktiven Forthwrite-Autors Ed Hersom.

2 Forth News

In dem kürzlich erwähnten Wettbewerb der "2002 International Conference on Functional Programming" ist Bernd Paysans Beitrag zehntbester unter 168 Einsendungen; Howard Oakford verbessert CWeed (v2.1); kurzes Programm zum Empfang von E-Mails und News und zum Senden von E-Mails von Marcel Hendrix für iForth unter WinNT 4.0 und Linux 2.0; 4ePost (Mailer für E-Mails und News) von Jos v.d. Ven (Windows. MSIE, Win32Forth); 3D Chess Update; Gleitkomma-Erweiterungen für 16-Bit- und 32-Bit-ANS-Forth-Systeme von Brad Eckert; HolonX von Wolf Wejgaard; David Williams: ^Forth nach C ; Forth an der Vrije Universiteit Brussel; Jodell Bumatai: SVFIG-Projekt "Musical Forth Words"; Richard Collins: ColorForth for Windows; b16-FPGA-Prozessor von Bernd Paysan; Francois Vignon: Forth for iPag PDA; Win32Forth v6.01; Krishna Myneni: kForth Updates; Samuel Tardieu: PicForth v0.7; Forth Inc.: SwiftX-SC für AT90SC.

5 Rectangles in Win32Forth
Dave Pochin

Dave "zähmt das Monster Win32Forth" seit geraumer Zeit auf http://www.sunterr.demon.co.uk . "Das Arbeiten mit dem Graphikpaket in Win32Forth kann frustrierend sein." Er bringt Beispiele für den Umgang mit Rechtecken.

13 Culver Consultancy -
An Interview with Barry Culver

Barry hat sein Unternehmen 1998 als das gegründet, was im allerneuesten Deutsch als Ich-AG bezeichnet werden würde. Er schwört auf Forth und erklärt anhand der üblichen Pro-Argumente, warum. Er befürchtet, dass Forth bald nur noch als Sprache für Enthusiasten übrig bleibt. Der Rezensent: War sie das nicht schon immer - oder habe ich die Materie von Grund auf falsch verstanden?

16 F11-UK
Jeremy Fowell <jeremy.fowell@btinternet.com>

Diese Platine wurde von FIGUK-Mitgliedern entwickelt, um eine leichte Möglichkeit, die Welt der microcontrollergetriebenen Einrichtungen zu erkunden, bereitzustellen. Das Paket enthält Hardware und Software und wird von einer privaten Firma vertrieben und an Mitglieder der FIG UK zum Selbstkostenpreis abgegeben.

18 From the 'Net
Chris Jakeman <cjakeman@bigfoot.com>

Chris präsentiert eine Auswahl von Themen aus der comp.lang. forth: Bernd Paysan zeigt, dass bei der Umwandlung einer zweistellige BCD-codierten Zahl die Verwendung on 2/ sehr viel schneller ist als */ . Anton Ertl gibt : aligned (c-addr -- a-addr) 3 + -4 and ; als gute Lösung für die Aufrundung zur nächsten 4-Byte-Grenze. Bernd Paysan gibt anhand von Bubble-Sort ein Beispiel für Metaprogrammierung (Programme, die Programme schreiben). Wil Baden definiert zwei Worte, N>R und NR>, zur Returnstackmanipulation. Wil Baden macht darauf aufmerksam, dass mit Hilfe von MARKER aus ANS-Forth DO-LOOPs und IF-THENs auch interpretativ verwendet werden können. Jerry Avins gibt einen Bericht über die manuelle Entzifferung verstümmelter Nachrichten.

22 Solution to Puzzle

"Gratulation an Fred Behringer, der als erster eine korrekte Lösung lieferte - MAX". Es wird der Prozess des Auffindens der Lösung über eine mathematische Analyse aufgezeigt.

24 euroFORTH 2002 - Conference Report
Bill Stoddart

In den Räumen der Technischen Universität Wien von Anton Ertl organisiert. Acht Vorträge (zwei davon von Anton Ertl, der als perfekter Gastgeber gelobt wird) und zwei Workshops. Die nächste Tagung, euroFORTH 2003, wird in Großbritannien, in Ross on Wye, von Nick Nelson und Micro-Ross veranstaltet.

27 Word Completion for Quikwriter Project
Chris Jakeman

Chris experimentiert mit "MinType", einem Algorithmus zur Vervollständigung bereits eingegebener Buchstabenfolgen zu einem Wort. "Für das Quikwriter-Projekt (behindertengerecht)." "Kann für das F11-UK-Board verwendet werden." Chris experimentiert mit VBA für Word.

30 Nominations for the FIG UK Awards - 2002

Aufruf, 2002-Kandidaten für die beiden jährlich vergebenen Preise zu benennen. Für 2001 wurden die beiden Preise von Chris Hainsworth und Dave Pochin gewonnen. Der Achivement Award wird für die beste Leistung in Forth vergeben, der Forthwrite Award für den besten Beitrag in der Forthwrite.

31 FIGUK AGM Report
Chris Jakeman

Keine Veränderungen in der Führungs-Crew. Etwa 110 Mitglieder. Viele davon aktiv. Neue Aktive: Henry Vinerts und Joe Anderson. (Der Rezensent: Beide gehen auf die Bemühungen der FG zurück.) Finanzen ausgeglichen. Die erste globale Liste von Forth-Quellen in der Welt auf der FIGUK-Website.

32 Using Wordlists for Many[
Jenny Brien and Chris Jakeman

Im letzten Forthwrite-Heft wurde MANY: definiert. Diesmal kommt MANY[. eine alternative Definition, die von kurzen Wortlisten Gebrauch macht. "Datengetriebene Programmierung, in Forth besonders einfach zu handhaben, häufig sehr viel einfacher als eine entsprechende IF...THEN...ELSE-Kette."

35 Across the Big Teich
Henry Vinerts

Drei Berichte von Henry über SVFIG-Aktivitäten in Originalfassung (September, Oktober, November 2002). Wir kennen sie in der Übersetzung von Thomas Beierlein.

40 Vierte Dimension 3/2002
Joe Anderson

Joe hat die Aufgabe der regelmäßigen Besprechung der VD in der Forthwrite, die bisher Alan Wenham inne hatte, übernommen. Wir danken Joe, der ja mit seiner Tätigkeit auch sehr im Interesse der Forth-Gesellschaft wirkt, herzlich.

43 Dutch Forth Users Group

Anzeige zur Mitgliederwerbung der holländischen Forth-Anwendergruppe (Forth-gebruikersgroep). Die Preise werden immer noch in Gulden angegeben :-)

44 Forthwrite Index

Jenny Briens Titelliste sämtlicher Forthwrite-Artikel von 1990 bis 2002. Unsere eigene VD-Titelliste ist Jennys Liste nachempfunden.

Gehaltvolles

VIJGEBLAADJE der HCC Forth-gebruikersgroep, Niederlande

Nr. 36, Februar 2003

HX
De schoolmeester

Der Schulmeister (Albert Nijhof) spricht wieder einmal, indem er sich des Frage-und-Antwort-Spiels zwischen Theo und Leo bedient. Ein Lehr- und Übungsstück. Ein Dollar ($) vor einer Zahl soll diese immer als Hexadezimalzahl ausgeben, unabhängig davon, in welcher Zahlenbasis das System gerade steht. Allgemein geht es um Veränderungsworte, die genau auf das nachfolgende Wort, und auf kein anderes, wirken. Leicht getan. Was aber passiert, wenn mitten in der Handlung ein Fehler auftritt? Wie soll das System, das über $ beispielsweise kurzzeitig nach Hex geschaltet wurde, auch im Fehlerfall wieder sauber auf die ursprüngliche Zahlenbasis zurückgesetzt werden? CATCH und THROW ist eine Lösung. Wirkt zwar wie Kanonen auf Spatzen, aber sie wirkt.

Ushi krijgt ogen
Marc Hartjes

Auf den HCC-Hobby-Elektronik-Tagen war ein Roboter zu sehen, der sich mit Hilfe einer elektronischen Kamera bewegte. (Ushi aus dem oben stehenden Titel ist das von Willem Ouwerkerk initiierte Roboter-Selbstbauprojekt der Fgg. Wir erinnern uns an Willems Gastvortrag auf unserer Tagung 2002 in Garmisch-Partenkirchen.) "Ushis Augen" sollen mit preiswerten Komponenten aufgebaut werden. Marc berichtet über die ersten Schritte auf dem Fgg-Treffen am 8. Februar 2003. Mehr davon wird auf dem nächsten Treffen am 12. April 2003 zu hören sein.

Algemene ledenvergadering
Het bestuur

Die Mitgliederversammlung der Fgg wird am 12. April 2003 abgehalten. Albert Nijhof, der Redakteur, und Coos Haak, Vorstandsmitglied, beenden ihre Amtszeit.

FORTHWRITE der FIG UK, Großbritannien

Nr. 120, März 2003

2 Editorial
Chris Jakeman <cjakeman@bigfoot.com>

Chris begrüßt drei neue Mitglieder. Er macht auf eine neue Rubrik aufmerksam: Forth Inside. Schließlich gratuliert er den diesjährigen Preisträgern.

3 Forth News

euroForth; "Ugly Home Page" von Neil Bawd; blockorientierte nach textorientierten Dateien; public e-mail gateway; Forth-Werkzeuge für den Internet-Zugang; MacForth Forum; Charles Montgomery übernimmt die Forth Scientific Library; charakteristische Maschinendaten in kForth; Big Number Packages von Marcel Hendrix; Hans Bezemer und 4th-Version 3.3d; Enth-Version 0.4 von Sean Pringle; MinForth, ein kleines ANS-Forth für DOS, mit und ohne DPMI, Windows und Linux, von Andreas Kochenburger; GForth wird schneller; kForth-Version 1.0.13; neue Version von nnCron.

6 Forth inside: Forth and the Neuron Chip
Chris Jakeman and Bill Powell

Echelon Corporation, Kalifornien. Seit 20 Jahren existent. Spezieller Prozessor mit leistungsfähiger Netzwerk-Software. Verwendet Neuron-Chips von Motorola, Toshiba oder Cypress. 24 Millionen Neuron-Chips bisher installiert, bis zu 3,-- $ billig. Eindeutiger 48-Bit-ID-Code. 3-Prozessoren-Parallel-Betrieb. Peer-to-Peer-Verbindungen. Forth-ähnlicher Aufbau (2-Stack-Maschinen). Forth-Basis jedoch bestgehütetes Geheimnis. Keine Dokumentation. Für den Anwender aus Marketinggründen nur C-Compiler.

9 F11-UK FIG Hardware Project
Jeremy Fowell <jeremy.fowell@btinternet.com>

Es wird über die F11-UK-Mailing-Liste berichtet. Die Konstruktion einer Palette von Erweiterungsplatinen wurde vorgeschlagen. Graeme Dunbar liefert Einzelheiten über die Entwicklung vergleichbarer Platinen an der Robert-Gordon-Universität. Informationen: Direkt von Jeremy.

11 Sorting a List
Chris Jakeman

Ein einführender Artikel über das Sortieren einer Liste von Ganzzahlen. "Forth bietet eine verführerisch große Menge an Möglichkeiten" - "Viel Literatur, aber nichts Einfaches in ANS-Forth." - "Dank für Korrespondenzen mit Graeme Dunbar und Leo Wong."

24 euroFORTH 2003

Die Konferenz findet diesmal in England statt, in Ross-on-Wye vom 17. bis 19. Oktober 2003. Alle "Officers" (Funktionäre) der FIG UK werden anwesend sein und Vorträge halten. (euroForth wird erst wieder 2006 in England ausgerichtet werden.)

26 Forth inside: nnCron

Ein Scheduler, Scripting Tool und Automation Manager für Windows-PCs von Nicholas Nemtzev (nnSoft, http://www.nncron.ru). In SP-Forth geschrieben, dem UNIX-Cron nachempfunden.

27 Feedback on Forth Code Index

265 Einträge bisher. Wird ständig erweitert. Artikel von älteren Ausgaben der Forthwrite und anderes. Elekronisch und Papier. Papiernachfragen 60 pro Jahr (extrapoliert). http://www.figuk.plus.com/codeindex/index.html

28 Presenting The FIG UK Awards of 2002

Ed Hersom (kürzlich verstorben) bekam posthum den Preis für die beste Leistung in Forth, Howerd Oakford den für die besten Beiträge in der Forthwrite (für seine Berichte über euroForth).

29 Across the Big Teich
Henry Vinerts <Volvovid@aol.com>

Drei Berichte von Henry über SVFIG-Aktivitäten in Originalfassung (Dezember 2002, Januar und Februar 2003). Wir kennen sie in der Übersetzung von Thomas Beierlein.

34 Deutsche Forth-Gesellschaft

Wieder unsere Anzeige zur Mitgliederwerbung. Die Beiträge werden immer noch in DM angegeben. Ich müsste noch einmal nachfassen.

35 Letters

Drei Briefe an die Redaktion. Boris Fennema ergänzt die Routine zur Bitspiegelung von Julian Noble (Forthwrite 113 und Übersetzung durch den Rezensenten in VD 2/2002 sowie Ergänzung des Rezensenten in Forthwrite 114) um eine High-Level-Forth-Routine, die einer C-Routine aus "Hacker's Delight", Addison-Wesley 2002, nachempfunden ist. Zweitens ein Mailaustausch zwischen Chris Jakeman und Phil Burk über pForth, GForth und eine portable KEY-Routine. Als dritten Brief schließlich hat Chris Jakeman den Leserbrief von G. Baecker (VD 1/2003) in einer Übersetzung von Henry Vinerts abgedruckt. Chris dankt Friederich Prinz für die Erlaubnis, das "Material" aus der VD zu übernehmen.

Fred Behringer

Fragen und Antworten aus .../comp/lang/forth

Die Frage

... eine Frage, zu der ich bisher keine wirkliche Antwort gefunden habe: Woran erkennt man eine gute/moderne FORTH Implementierung?
Bei der Suche im Web und in Bibliotheken bin ich über eine Menge Informationen zu FORTH-Implementierungen gestossen. Leider beziehen sich diese Detailinformationen aus den 70er und 80er Jahren auf alte FIG Forth Varianten (z. B. die FORTH Encyclopedia). Für den modernen Sprachumfang gibt es den ANSI Standard (wie immer man hierzu stehen mag), er gibt einen Leitpfaden vor, an dem sich Entwickler orientieren können. Aber der ANSI Standard behandelt nicht die Implementierung von FORTH Systemen.
Gibt es allgemeine Regeln oder Quellen, die (systemunabhängig) erklären, was eine gute und moderne FORTH Architektur ausmacht?

Carsten Strotmann

1. Antwort

Man muß sich klar werden, worauf das FORTH laufen soll:
- a) 8 Bit CPU/Controller
- b) 16/32 Bit CPU/Controller
- c) PC.

Stackbreite
In a) wird man den Stack 16 Bit auslegen,
in c) zu 32 Bit in b) in vielen Fällen auch 32 Bit.

Letztlich neigt man immer zu 32 Bit wenn die MemoryMap grösser 64k ist. Auch Verwendung von Float kann 32 Bit erzwingen.

Ausführung der Grundbefehle

Es gäbe
1) bytecode
2) threaded code
   also 16 Bit Adressen a la figFORTH

3) JSR-threaded code
   Assemblerunterprogramme
4) native code
   direkter Assembler

Speicherverbrauch steigt von 1) nach 4) aber Geschwindigkeit steigt auch. Für a) & b) sind 2) & 3) üblich. Für c) eher 4). Ein anderer Aspekt wäre ein völlig maschinenunabhängiger compilier & testbarer Code. Da wäre 1) & 2) sinnvoll, aber eben langsam. Für z.B. Protokollsoftware denkbar.

Compilierung

Auf c) kann der Compiler beliebig kompliziert werden.
Wenn man von c) nach a) & b) crosscompiliert dito.
Wenn der Compiler in Targets wie a) & b) laufen soll, wird man einen sehr simplen Compiler ohne viel Optimierungen bevorzugen.

Sprachumfang

Der Sprachumfang ist von figFORTH nach FORTH83 nach ANSI munter gewachsen. Auf einem Target a) & b) genügt meist simpler FORTH-Sprachumfang à la figFORTH. Andererseits hat man dort gerne Assembler für die CPU integriert.
MfG JRD *(Rafael Deliano)*

2. Antwort
Gut: Das kommt auf Deine Qualitaetskriterien an. Ueblicherweise fliessen die folgenden Kriterien in die Beurteilung der Qualitaet ein (ohne bestimmte Ordnung):

- auf welchen Plattformen es laeuft
- Ist das System hosted (arbeitet unter einem OS) oder native (Forth ist das OS)
- Kompatibilitaet mit vorhandenem Code
- Kompatibilitaet mit vorhandenem Wissen ueber Forth-Implementierungen
- welchen Standard das System unterstuetzt
- Features zur Unterstuetzung bei der Entwicklung
- Geschwindigkeit bei der Ausfuehrung
- Geschwindigkeit beim Compilieren
- Groesse
- Dokumentation
- Erweiterungen
- Zugriff auf das Betriebssystem oder Libraries

Modern: Dieses Wort wird meist in folgendem Zusammenhang gebraucht:

A: Forth kann nicht X.
B: Das war vor 20 Jahren. Moderne Forth-Systeme koennen X.

An den ‚X', mit denen das typischerweise gebraucht wird, könntest Du vielleicht die Modernitaet festmachen. Typischerweise werden genannt:

- Zeilenorientierte Dateien fuer Source-code
- Gleitkomma-Arithmetik
- Locale Variablen
- Lange Namen

Vielleicht suchst Du also Informationen, wie man heutzutage Forth implementieren kann.

Sowas findest Du u.a. auf

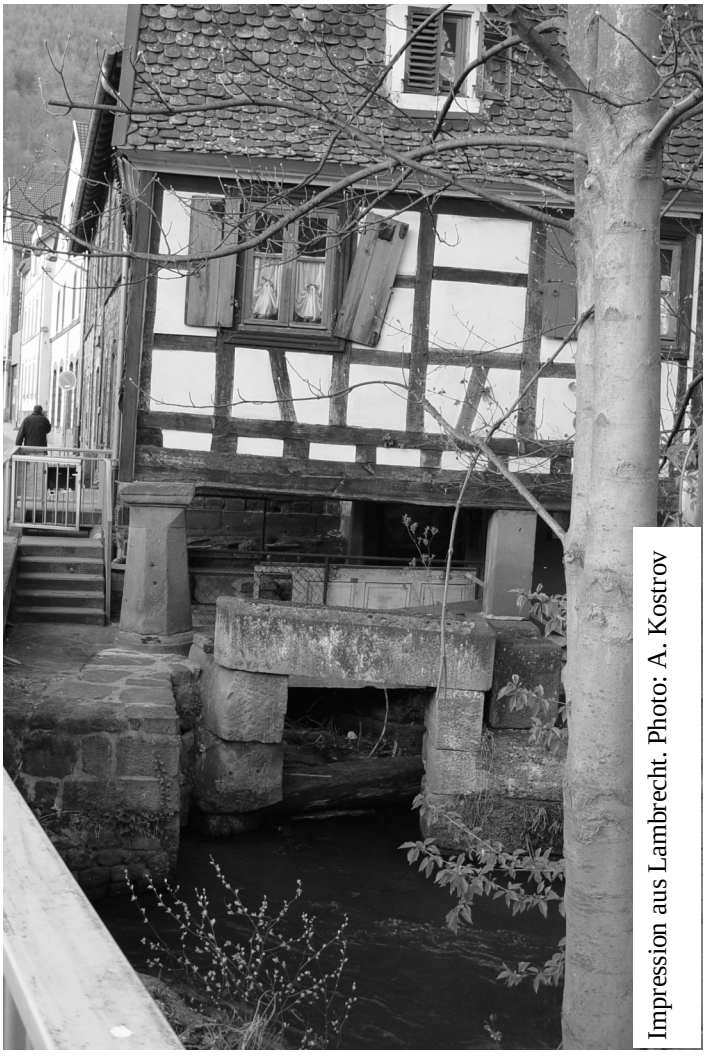
http://www.complang.tuwien.ac.at/projects/forth.html

unter Projects; Gforth orientiert sich eher an traditionellen Methoden, waehrend Du unter "Forth native code generation" einen weniger traditionellen Ansatz findest. Jedenfalls findest Du dort diverse Papers zu diesen Themen, und ueber die Literaturlisten noch andere.

M. Anton Ertl

Eine kluge Frage – und zwei kompetente Antworten; kurz aber ausreichend; eben „forthig". Trotzdem wäre ein umfassender Aufsatz über die Implementierung eines Forth nicht nur für die VD ein wirklicher Höhepunkt. Implementierer hat die Forthgesellschaft genug. Vielleicht findet sich sogar ein Autorenteam, das uns einen solchen Aufsatz liefert? Nicht nur die „ Neulinge" würden das dankbar aufnehmen.

fep



Impression aus Lambrecht. Photo: A. Kostrov

Debugging mit einem Mixed Signal Oszilloskope am Beispiel I²C-Bus

Klaus Zobawa
Hamburg

"Ohne In-Circuit-Emulator, kurz ICE, embedded Software debuggen und testen, ist das denn möglich?" "Ja, mit FORTH!" (ok, auch mit diversen Monitortools soll es gehen) Wer jedoch einmal ComFORTHabel interaktiv mit seinem Target "kommuniziert" hat, möchte diesen Luxus nicht mehr missen.

Wie die Praxis bei der Entwicklung eines chirurgischen Gerätes mit dem FieldFORTH der "Rostocker" zeigt, kann ich meine obige Aussage nur nochmals wiederholen. Ich bin froh den "emulatorspezifischen" Ärger los zu sein – unflexibel in der Handhabung, hoch sensibel bezüglich Kontaktierung zum Target u.v.a.m.. Eine Sache, über die ein gut ausgestatteter ICE verfügt, fehlt leider bei einer FORTH-Umgebung: nämlich die Möglichkeit unter "Volldampf" einen Trace aufzuzeichnen. Die Tracefunktion zeichnet die letzten z.B. 1024 Programmschritte auf und stellt sie nachträglich textuell zur Ansicht, sodass in aller Ruhe nachvollzogen werden kann, was genau zum Zeitpunkt "der Katastrophe" geschah. Gute ICE bieten weiterhin die Möglichkeit, externe Digitalsignale im Trace mit aufzuzeichnen. Spätestens wenn man komplexere Datenübertragungsprotokolle realisieren möchte, braucht man so etwas. Wie etwa bei der Implementierung einer Multimaster-Kommunikation über I²C-Bus. Ziemlich schnell stellt man dabei fest, dass alles mögliche im 849-Seitigen Datenbuch des µController beschrieben ist, nur nicht die Details, über die man dann stolpert, wenn zwei und irgendwann drei oder vier µC zeitgleich I²C-Busmaster spielen wollen. Mal ganz abgesehen von den Bugs, die aufs eigne Konto gehen ☹. Guter Rat ist teuer – ein gutes Messmittel auch. Wie z.B. das "Mixed-Signal-Oszilloskop" von Agilent (ehemals HP). Wie der Bezeichnung zu entnehmen ist, handelt es sich dabei um ein Oszilloskop mit zusätzlichen Logic Analyzer Funktionen. Oder handelt es sich doch um einen Logic Analyzer mit Analogkanälen? Wie dem auch sei, es bietet zwei Analogkanäle und 16 Logiksignaleingänge, die gleichzeitig dargestellt werden können. Da es sich um ein Digital-Oszilloskop handelt, darf ein Signalspeicher nicht fehlen. Dieser ist wirklich üppig vorhanden und umfasst sage und schreibe 2 MB Speichertiefe **pro** Kanal. Was dies für die Praxis bedeutet, möchte ich anhand der Screen-shuts in den *Bildern 1 – 4* zeigen.

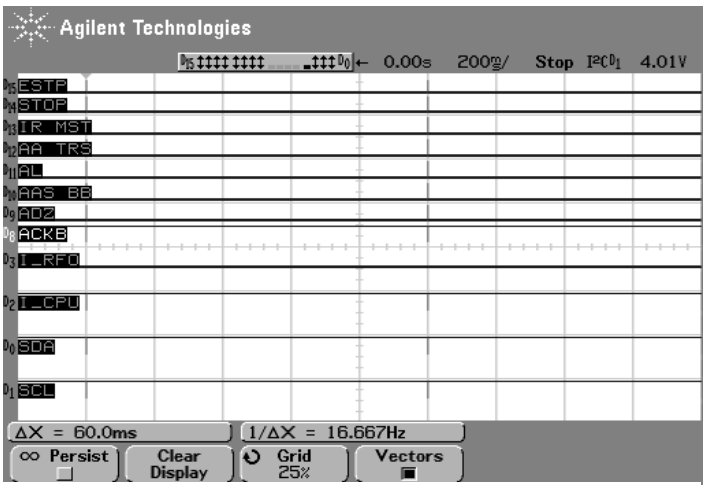


Bild 1 ^ „Fliegendreck" ^ Abstand = 1 Sekunde

"Wie ihr seht, könnt ihr nichts sehen. Was ihr nicht seht, dass werdet ihr gleich sehen."

So (oder so ähnlich) lautet ein Spruch der auf *Bild 1* zutrifft. D.h., einiges ist doch sehr deutlich zu erkennen. Wie z.B. die Zeile unter dem Schriftzug "Agilent Technologies". Sie gibt den aktuellen Gerätestatus wieder. Das Gerät befindet sich im Logic Analyzer Modus. Von Links her betrachtet werden die aktuellen Signalpegel angezeigt: D15 ..... D0 (D4 – D7 sind inaktiv). 0.00s gibt den zeitlichen Abstand zwischen der Zeitreferenz und dem Triggerpunkt des aufgezeichneten Signals an. Im vorliegenden Fall liegen Triggerpunkt und die Zeitreferenz in Deckung. Die Zeitbasis ist auf 200ms pro Division eingestellt. Bei einer Anzahl von 10 Division werden somit insgesamt 2 sec aufgezeichnet. Getriggert wird auf die Startbedingung des I²C-Signals. Die Low-/High-Schwelle liegt bei 4,01V.

Am linken Rand sind die physikalischen Kanalnummern, verknüpft mit dem frei wählbaren Signalnamen, dargestellt. Die eigentlichen I²C-Signale liegen auf D0 (SDA = Datensignal) und D1 (SCL = Clocksignal). Die Kanäle D2 , D3 und D8 – D15 stellen vom Programm generierte Signale dar. Sie dienen dazu, den genauen Zustand des im µController integrierten I²C-Bus (D8 –D 15) und den Interrupt-Status (D2 u. D3) anzuzeigen. Damit die Signale erzeugt werden konnten, musste ein I/O-Port des µController herhalten. Konkret die Schnittstelle zum Ansteuern der Relais. Durch das Programm werden die einzelnen Bits des Ports, entsprechend den vorhandenen Statusbits, gesetzt. Bei dem Rest des Monitorbildes scheint es sich wohl um Druckfehler oder Fliegendreck zu handeln. Zu erkennen sind kurze, etwas breitere senkrechte Linien, die sich vom Gitter leicht abheben.

In *Bild 2* ist zu sehen, was in *Bild 1* nicht zu sehen ist. Durch einfachen Dreh am Oszilloskop wird das aufgezeichnete Signal um Faktoren gedehnt. Dank der hohen Speichertiefe auf eine zeitliche Auflösung von 50µs/ Division, was einem Faktor von 4.000 entspricht. Der vermeintliche "Fliegendreck" entpuppt sich also bei genauerem Hinsehen, als eine wohldefinierte Folge von Pegelwechseln der einzelnen Signalkanäle. Mit einem weiteren Knopf am Oszilloskop kann eine andere zu untersuchende Stelle ins Blickfeld gerückt werden. In *Bild 3* wird davon Gebrauch gemacht – der Triggerpunkt wurde gegenüber der Zeitreferenz um 999ms "nach links" verschoben, sodass nun das 2. Signalbündel betrachtet werden kann.

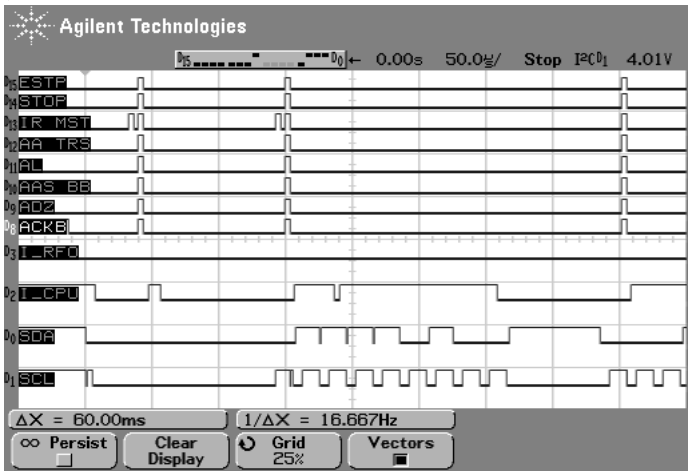


Bild 2

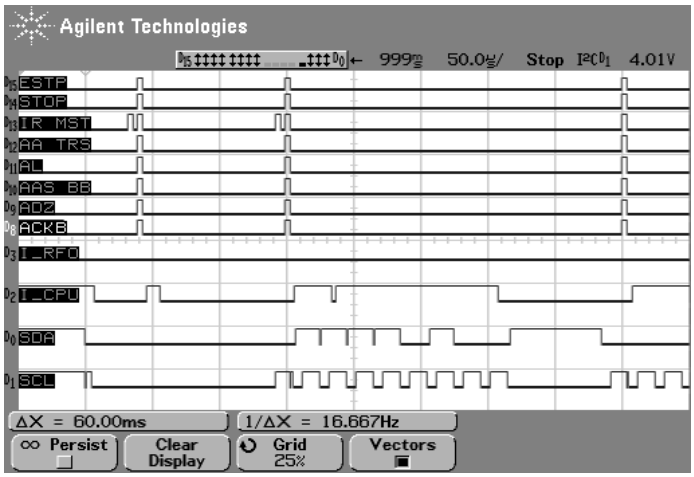


Bild 3

Auf diese Art lässt sich die gesamte Aufzeichnungsdauer von 2 Sekunden mit der hohen Auflösung betrachten. *Bild 4* zeigt die 10.000-fache Zeitdehnung.

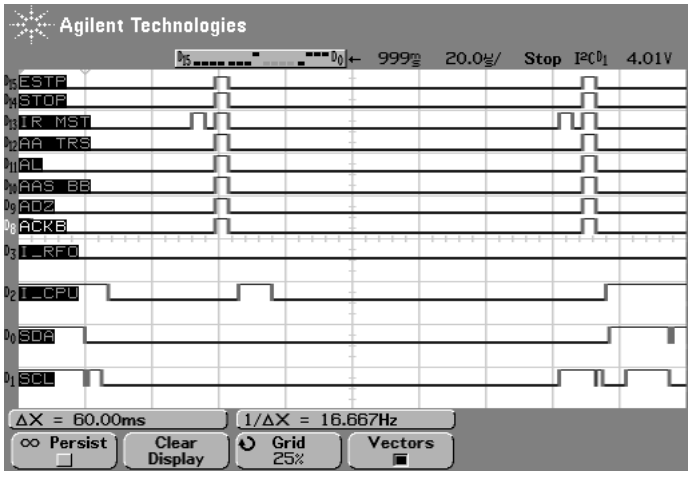


Bild 4

Fazit: nicht in allen Punkten kann die beschriebene Methode ein ICE vollständig ersetzen. Die Kombination aus interaktivem Zugang zum Target durch FORTH und der Möglichkeit, mit hoher Speichertiefe die relevanten Signale aufzuzeichnen, ermöglichen jedoch ein hohes Maß an Flexibilität beim Debuggen und Testen.

Wer mehr über das "Deep Memory Oszilloskop" mit Mega-Zoom-Technologie erfahren möchte, kommt mit der genauen Bezeichnung: **54622D 2+16 Channel 100 MHz Mixed Signal Oscilloscope** unter www.Google.de garantiert weiter.
Falls ihr Fragen zu diesem Thema habt, meldet euch einfach bei Klaus.Zobawa@t-online.de

Klaus Zobawa
Söring Medizintechnik GmbH

Triceps

Ein Roboter mit hohem Spaßfaktor

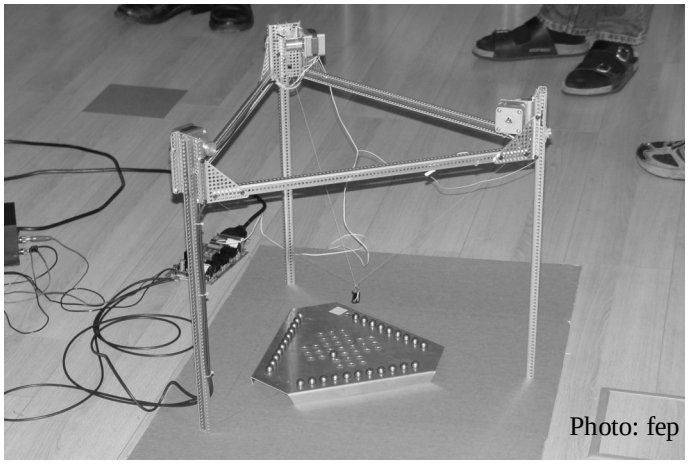
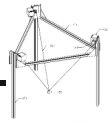


Photo: fep

So wie auf dem Bild hier oben, war Triceps in Lambrecht in Aktion zu erleben (nachdem M. Kalus, und Andere, Triceps Netzteil repariert hatten). Ein Roboter? Das sieht ja irgendwie nicht so aus. Der eigentliche Aktor (der Magnet) wird über Seile bewegt? Und davon braucht es dann gleich drei? Damit soll


Photo: fep

sozusagen feinmechanisch gearbeitet werden? Ewald Rieger hat den Teilnehmern der Tagung gezeigt, daß Triceps sehr wohl diese und andere Anforderungen bestens erfüllt. Danke, für einen spannenden Vortrag! *fep*



Triceps
Ein einfacher Roboter zur Demonstration von Pick and Place Aufgaben

Ewald Rieger
(eMail: ewald.rieger@t-online.de)
Littersheimer Weg 10
D-67240 Bobenheim-Roxheim
28.03.2003

Zusammenfassung

Ein einfaches dreiachsiges System zum Transportieren von eisenmetallischen Teilen wird kurz vorgestellt. Anhand des Programmtextes wird gezeigt, wie man die Motoren des Systems ansteuert und eine Beschleunigungs- und Bremsrampe programmiert; und wie man von kartesischen Positionsangaben zur Achseninterpolation kommt. Zum Schluß werden ein paar grundlegende Funktionen zur Bedienung des Roboters vorgestellt.

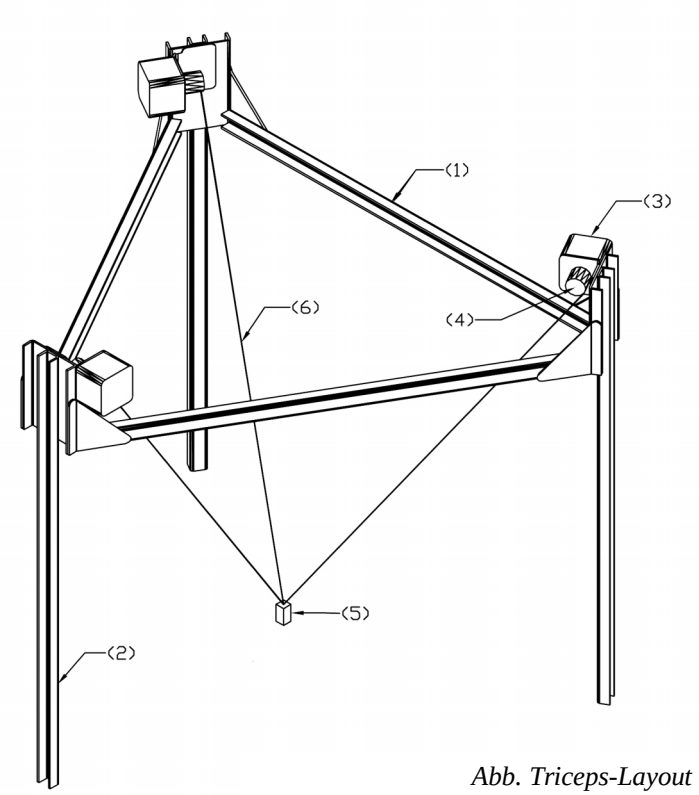


Abb. Triceps-Layout

1. Aufbau des Roboters

Abbildung 1 zeigt den schematischen Aufbau des Roboters. Drei gleichlange Aluminium-U-Profilschienen (1) sind zu einem gleichschenkligen Dreieck zusammengeschraubt, das von drei Beinen (2) gleicher Länge getragen wird. In jeder Ecke des Dreiecks ist ein Schrittmotor (3) mit horizontal liegender Achse montiert und so ausgerichtet, daß er quer zum Schnittpunkt der Winkelhalbierenden des Dreiecks liegt. Auf jede Motorachse ist eine kleine Spule (4) geschraubt, an der das Ende einer Schnur (6) befestigt ist. Die anderen Enden der drei Schnüre sind in der Mitte des Dreiecks verknotet und tragen einen kleinen Elektromagneten (5). Die Länge der Schnüre sind so bemessen, daß jeder Punkt im Raum unterhalb des Dreiecks durch Auf- und Abwickeln erreicht werden kann. ([fig] ([fig] Triceps-Roboter) Zur Ansteuerung des Triceps-Roboters verwenden wir eine preisgünstige Schrittmotorenansteuerung, die bei Conrad Elektronik bezogen wurde. Ein 24V-Netzteil mit 3A Leistung versorgt die Schrittmotorenkarte mit Energie. Die Karte erlaubt maximal 3 Schrittmotoren anzusteuern. Sie verfügt über eine Schnittstelle zum Standard-Parallelport und wird mit einem Druckerkabel direkt an einen PC angeschlossen. Aus Kostengründen beschränken wir uns auf den Transport von eisenmetallischen Teilen. Zum Tragen der Teile dient ein Elektromagnet aus einem 24V-Kleinrelais. Zur Ansteuerung der Spule erweitern wir die Schrittmotorenkarte mit einer kleinen Zusatzschaltung. Ihr Eingang wird mit der freien Parallelportleitung Pin14 (beim Druckerport ist Pin 14 das Autofeed-Signal) an der Centronicsbuchse auf der Motorenkarte verbunden. Die Versorgungsspannung entnehmen wir der Motorenkarte und sichern mit einer Feinsicherung gegen einen drohenden Kurzschluß ab.

2. Eigenschaften des Programms

Das in diesem Artikel beschriebene Programm wurde mit Bigforth unter Linux und Windows entwickelt und getestet. Das Programm kommuniziert mit der Motorenkarte, indem es direkt auf die Register des Parallelports zugreift. Einige Betriebssysteme verweigern jedoch diesen Zugriff. So läßt sich das Programm zur Zeit nicht unter Windows XP und Windows NT ausführen. Auf Windows 95, 98, und ME kann man das Programm jedoch ausführen. Allerdings läuft der Roboter unter diesen Betriebssystemen sehr langsam. Denn in regelmäßigen Abständen muß das Forth die Kontrolle an das Betriebssystem abgeben. Ungeachtet von kurzen angeforderten Wartezeiten von z.B. 1 ms gibt das Betriebssystem erst nach mehreren Millisekunden zurück. Das Problem läßt sich wahrscheinlich durch Nutzen anderer Betriebssystemtimer beheben. Unter Linux ist für den Zugriff auf den Parallelport ein Anmelden als Benutzer Root notwendig. Was das Timing unter Linux anbelangt, sieht man deutlich, daß dieses Betriebssystem die Heimat von Bigforth ist. Eine aufwendige Timerprogrammierung sorgt für akzeptable Schrittfrequenzen bis zu 2 Khz. Für Demozwecke ist das allemal ausreichend. Ein richtiger Schrittmotorenbetrieb ist jedoch nur unter einem echtzeitfähigen Betriebssystem möglich (z.B. Forth auf einem Mikrocontroller mit Timerinterrupt getriebenem Task). Das Compilieren des Programms wird durch Compilerdirektiven gesteuert und ermöglicht ein korrekte betriebssystemabhängige Compilation. Das Programm stellt einen einfachen Befehlssatz zum Bewegungen des Magneten, der als Greifer dient, im Kartessischen Koordinatensystem bereit. Eine trapezförmige Beschleunigungs- und Bremsrampe sorgt für sanftes Starten und Stoppen einer Bewegung. Beim Verfahren

von einer Start- zur Zielposition werden in kleinen Schritten die neuen Positionen der X-, Y-, Z-, Achsen im Kartesischen System berechnet und danach in die drei Seillängen umgerechnet, welche die neue Postion festlegen. Das Raster wurde so fein gewählt, daß bei jeder Berechnung je Motor höchstens ein Schritt verfahren wird. Damit erreichen wir eine saubere Interpolation der Achsen.

3. Die Programmierung

Der nachfolgende Text erklärt den Quelltext des Roboterprogramms

```
include comclass.fb
\needs float include float.fb float also forth

2 &19 thru
```

Nach dem Laden des Fließkommapakets und einigen hilfreichen Definitionen aus dem File: comclass.fb wird das Programm compiliert.

3.1 Die Schnittstelle zur Schrittmotorenkarte

```
$378 Constant port   \ LPT1
$379 Constant pt     \ LPT1
$37A Constant stb
```

Für den Zugriff auf den Druckerport werden die drei Adressen des Parallelports als Konstanten definiert.

```
[IFDEF] win32
: bitset ( adr 8b -- ) over pc@ or swap pc! ;
: bitclr ( adr 8b -- ) $FF xor over pc@ and swap pc! ;
: motorout ( 8b -- ) port pc! stb 1 bitset stb 1 bitclr ;
[THEN]
```

Zur Manipulation der Portregister definieren wir die Forth-Worte *bitclr* und *bitset*. Bitclr erwartet auf dem Datenstack die Portadresse *adr*, gefolgt vom Wert *8b*, der ein Bitmuster darstellt. Alle gesetzten Bits im Bitmuster werden durch Ausführen von *bitclr* im Register gelöscht. Das Forth-Wort *bitset* erwartet die gleichen Argumente auf dem Datenstack, setzt aber diese Bits im Register. *Motorout* übergibt das oberste Stackelement als 8b-Wert an den Parallelport und erzeugt danach den Datastrob zur Datenübernahme durch die Motorenkarte.

```
[IFDEF] unix
also dos
Create devicename ," /dev/port"
Variable lphandle
Create lpinbuf 20 allot
: closelp lphandle @ ?dup IF close-file THEN ;
: openlp  devicename count r/w open-file drop
  lphandle ! ;
: position-lp ( ud -- ior )
  lphandle @ filehandle @ rot fseek ior ;
: lp> ( -- )
  lpinbuf 1+ 1 lphandle @ read-file drop lpinbuf c! ;
[THEN]
```

Unter Linux greifen wir auf die Parallelportregister über /dev/port zu. Dieser Device wird wie ein File behandelt. Vor dem ersten Zugriff auf /dev/port wird mit *openlp* das Device geöffnet und am Ende mit *closelp* wieder geschlossen. Das Wort *position-lp* positioniert durch Ausführen von *fseek* auf das zu lesende oder zu schreibende Register. *Lp>* holt einen 8Bit-Wert vom Register ab und schreibt ihn in einen Buffer.

```
[IFDEF] unix
Create lpoutbuf 2 allot  1 lpoutbuf c!

: lptype ( adr len -- ) pause lphandle @ write-file drop ;

\ : rsemit? true ;

: lpemit ( 8b -- )  lpoutbuf 1+ c!  lpoutbuf count lptype ;
  previous
[THEN]
```

Zum Schreiben in ein Register wird *lpemit* definiert, das den obersten Stackwert als 8Bit-Wert behandelt und in das zuvor durch *position-lp* adressierte Register schreibt.

```
[IFDEF] unix
openlp

: ppc@ ( adr -- 8b )
  0 swap position-lp drop lp> lpinbuf 1+ c@  ;
: ppc! ( 8b adr  )
  0 swap  position-lp drop lpemit ;
```

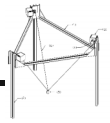
Die Definitionen von *ppc@* und *ppc!* entsprechen den Funktionen von *pc@* und *pc!*.

```
: bitset ( adr 8b -- ) over ppc@ or swap ppc! ;

: bitclr ( adr 8b -- ) $FF xor over ppc@ and swap ppc! ;

: motorout port ppc! stb 1 bitset stb 1 bitclr ;
```

Nun erfolgen die Definitionen von *bitclr*, *bitset* und *motorout* für die Linuxvariante.

```
[THEN]
```

Damit sind die betriebssystemabhängigen Definitionen für den Zugriff auf den Parallelport abgeschlossen.



3.2 Greiferansteuerung

```
: magnet stb %10 ;
```

Zum Aufnehmen und Ablegen von Metallteilen müßen wir den Magneten ansteuern. Das Forth-Wort *magnet* legt dazu die zugehörige Registeradresse und sein Bitmuster auf den Datenstack. Das Wort *magnet* wird, gefolgt von *bitclr* oder *bitset*, zum Ein- und Ausschalten des Magneten benutzt.

3.3 Motorenansteuerung

```
: motor: Create 0 , c, does> dup cell+ c@ swap ;
```

Das Wort *motor:* ist ein definierendes Forth-Wort. Beim Ausführen von *motor:* wird ein neuer Motor im Vocabulary erzeugt. Mit *0* , wird die Zelle für den Schrittzähler angelegt. Während c, das Bitmuster für die Adressierung dieses Motors ablegt.

```
$00 motor: ma
$40 motor: mb
$80 motor: mc
```

Für den Triceps-Roboter werden drei Motoren definiert. Wird im Programm das Wort eines Motors ausgeführt, z.B. *ma*, legt dieses Wort die Adresse seiner Datenstruktur und das Motorbitmuster auf den Datenstack.

```
Create mottab $27 c, $2D c, $1C c, $0D c,
              $03 c, $09 c, $38 c, $29 c,
```

Zur Kommutierung der Motoren brauchen wir ein Array von 8 Bit-Mustern für aufeinanderfolgende Voll- und Halbschritten. Diese Werte sind dem Handbuch der Motorenkarte von Conrad-Elektronik entnommen. Bei einem bipolaren Schrittmotor mit 200 Vollschritten pro Umdrehung erhalten wir damit 400 Schritte pro Umdrehung.

```
: right  dup -1 swap +! @ 8 mod mottab + c@ or motorout ;
: left   dup  1 swap +! @ 8 mod mottab + c@ or motorout ;
```

Die Worte *right* und *left* erwarten die Argumente eines Motors auf dem Datenstack. Beim Ausführen bewegt sich der Motor jeweils einen Schritt in die gewünschte Richtung. Je nach Richtung wird der Schrittzähler inkrementiert oder dekrementiert und aus dem aktuellen Zählerstand, durch Ausführen von *8 mod*, ein Index für die Kommutierungstabelle berechnet. Das entnommene Kommutierungsbitmuster wird nun mit dem Motorbitmuster ‚verodert' und an die Motorenkarte abgegeben.

```
: arelstep ( n -- )
   ?dup 0= ?EXIT 0<
   IF  ma left ELSE  ma right THEN ;

: brelstep ( n -- )
   ?dup 0= ?EXIT 0<
   IF  mb left ELSE  mb right THEN ;

: crelstep ( n -- )
   ?dup 0= ?EXIT 0<
   IF  mc left ELSE  mc right THEN ;
```

Die Worte *arelstep*, *brelstep* und *crelstep* haben die Aufgabe, den betreffenden Motor bei Bedarf um einen Schritt, in die gewünschte Richtung zu drehen. Beim Ausführen nimmt das Wort *n* vom Datenstack. Ist n<0 bewegt sich der Motor einen Schritt nach links und bei n>0 einen Schritt nach rechts. Ist n=0 geschieht nichts.

```
Variable delay  &20 delay !
: langsamer &10 delay +! ;
: schneller -&10 delay +! ;
: action key? IF key $FF52 case? IF schneller false exit
                                    THEN
                          $FF54 case? IF langsamer false exit
                                    THEN
                          $1B case? IF true exit
                                    THEN
   THEN false ;
: tl BEGIN delay @  wait ma right action until ;
```

Das Wort *tl* steht für Testlauf und hilft in diesem Beispiel bei der Inbetriebnahme des Motors *ma*. Nach dem Starten von *tl* bewegt sich der Motor mit der durch *delay* vorgegebenen Geschwindigkeit. Durch Drücken der Pfeiltasten [unten] oder [oben] verändert sich die Geschwindigkeit, während die ESC-Taste das Programm abbricht.

3.4 Beschleunigen und Bremsen

```
Variable laststep
```

Die Variable *laststep* hält die Zielposition, die am Ende der Beschleunigungs- und Bremsrampe erreicht wird.

```
Variable step
```

Die Variable *step* zeigt beim Verfahren auf die aktuelle Position und wird ebenfalls zur Generierung der Beschleunigungs- und Bremsrampe benötigt.

```
Variable speed  &30 speed !
```

Die Variable *speed* enthält die kleinste Wartezeit von Schritt zu Schritt und bestimmt damit die Höchstgeschwindigkeit der Achsen. Der Wert *30* entspricht einer Verzögerungszeit von 30/50 msec pro Schritt.

```
: rampe: Create 2dup - 1- 0 max ,
   ?DO 2dup I um/mod nip , LOOP 2drop ;
```

Das Forth-Wort *Rampe:* ist ein definierendes Wort. *Rampe:* erwartet drei Werte auf dem Datenstack, gefolgt von dem Namen einer neuen Rampe, und erzeugt daraus ein Array der Verzögerungszeiten von Schritt zu Schritt für die Brems- und Beschleunigungsrampe.

```
: .rampe ' >body dup @ cells bounds ?DO I @ . ?cr cell
  +LOOP ;
```

Das Wort *.rampe*, gefolgt von dem Namen einer Rampe, druckt deren Werte auf dem Terminal aus.

```
&8000. &300 &6 rampe: rampe
```

Erzeugt eine Rampe mit 1. Wert = 8000/6=1333 und dem letzten Wert = 8000/300 = 26. Das heißt der Motor startet mit einer Verzögerung von 1333/50 msec beim ersten Schritt, beschleunigt linear und erreicht nach 240 Schritten mit einer Verzögerungszeit von 26/50 msec seine Höchstgeschwindigkeit, sofern sie nicht schon vorzeitig durch den Wert in der Variablen speed auf einen größeren Wert begrenzt wurde (siehe nachfolgenden Text).

```
Variable wartezeit
Variable zdif  25 zdif !

: mwait ( 1/50msec -- )  wartezeit @ till
                         timer@ + wartezeit ! ;
```

Das vom Forth-Standard definierte Wort *wait* erwartet sein Argument in Millisekunden. Für die Schrittmotoranwendung wäre diese Timerauflösung viel zu gering. Deshalb wurde *mwait* mit einer Timerauflösung von 1/50 msec definiert. Vor dem ersten Aufruf von *mwait* muß in die Variable *wartezeit* die Zielzeit geschrieben werden. Nachdem *till* auf diesen Zeitpunkt gewartet hat, könnte nun einfach die gewünschte Zeitdifferenz des nächsten Schrittes zur Variablen *wartezeit* addiert werden und dann beim nächsten Ausführen von *mwait* auf diesen Zeitpunkt gewartet werden. Leider geht das von Zeit zu Zeit schief, weil das Betriebssystem zu spät zurück gibt. Deshalb berechnen wir die neue Zielzeit aus dem aktuellen Timerstand mit *timer@*.

```
: waiting ( -- ) step @
   laststep @ over - min rampe @ min 1+ cells rampe
   + @ speed @ max mwait  1 step +! ;
```

Das Wort *waiting* berechnet bei seinem Aufruf in Abhängigkeit vom Inhalt der Variablen *step* und *laststep* den Index für einen Verzögerungswert in der Rampe, entnimmt diesen Wert der Rampe, begrenzt ihn auf den Wert in der Variablen *speed* und wartet im nächsten Wartezyklus, bis diese Zeit verstrichen ist.

3.5 Positionen berechnen

Für den einfachen Umgang mit Vektoren benötigen wir noch ein paar Forth-Worte

```
[IFUNDEF] 3dup

: 3dup ( n1 n2 n3 -- n1 n2 n3 n1 n2 n3 )
       2 pick 2 pick 2 pick ;
[THEN]
```

3dup dupliziert den auf dem Datenstack liegenden Vektor.

```
: v- ( x1 y1 z1 x2 y2 z2 -- dx dy dz )
     >r rot r> swap - >r     \ dz
     rot - >r                \ dy
     swap - r> r> ;
```

v- bildet die Differenz von zwei Vektoren.

```
: v+ ( x1 y1 z1 x2 y2 z2 -- x3 y3 z3 )
      >r rot r> + >r     \ z
      rot + >r           \ y
      + r> r> ;
```

Und *v+* summiert zwei Vektoren auf.

Schließlich brauchen wir noch ein paar Positionsangaben.

```
Create ma-pos   0 ,   0 , 0 ,
Create mb-pos 5200 ,   0 , 0 ,
Create mc-pos 5200 dup s>f pi f2* !6 f/ fcos f* f>s ,
                              s>f pi f2* !6 f/ fsin f* f>s , 0 ,
```

Die Worte *ma-pos*, *mb-pos* und *mc-pos* enthalten die Vektoren für die räumliche Anordnung der Motoren. Genau genommen ist das der Punkt, an dem das Seil auf die Trommel gewickelt wird.

```
Create gr-pos 5200 dup 2/ ,
       s>f pi f2* !6 f/ fsin f* !3 f/ f>s , 3400 ,
```

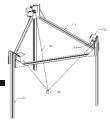
Die Vektorvariable *gr-pos* enthält die aktuelle Position, an der die drei Seile zusammen geknotet sind.

```
Create gr-org 5200 dup 2/ ,
       s>f pi f2* !6 f/ fsin f* !3 f/ f>s , 3400 ,
```

Die Vektorvariable *gr-org* enthält die Referenzposition für den Seilknoten beim Systemstart. Bei einer Reinitialisierung wird *gr-pos* mit dem Wert von *gr-org* überschrieben und das System manuell positioniert. Dadurch sparen wir uns den hohen Aufwand für eine Referenzfahrt ein. Alle Positionen sind in 1/10 mm angegeben und müssen sich innerhalb eines Quadranten befinden.

```
: x-pos ;
: y-pos cell+ ;
: z-pos cell+ cell+ ;
: pos@ ( adr -- x y z ) dup >r x-pos @ r@ y-pos @ r>
       z-pos @ ;
: pos! ( x y z adr -- ) dup >r z-pos ! r@ y-pos ! r>
       x-pos ! ;
```

Die Forth-Worte *pos@* und *pos!* vereinfachen das Lesen und Schreiben von Vektoren in und aus ihren Variablen.



3.6 Vom 3D-Raum zur Bewegung der Motoren

```
: ds ( dx dy dz -- s )
   dup * >r dup * >r dup * r> + r> +
   s>f fsqrt f>s ;
```

Das Wort *ds* berechnet die kürzeste Distanz zwischen zwei Punkten im Raum und dient zur Berechnung von Seillängen. Bei seinem Aufruf nimmt das Wort eine Vektordifferenz, von zwei Punkten im Raum, vom Datenstack und übergibt den Wert des Abstandes.

```
: schritte ( 1/10mm -- schritte )
   400 355 */ ;
```

Die Aufgabe von *schritte* ist unsere Längenangaben von unserer Einheit 1/10 mm in Motorschritte umzurechnen. Für eine Seillänge von 35,5 mm dreht sich der Motor 400 Schritte. Das entspricht einer Motorumdrehung.

Zur simultanen Bewegung der Motoren in unserem System, müssen nun noch alle Seillängen schrittweise berechnet werden.

```
: seillaengen ( x y z -- sa sb sc )
   3dup mc-pos pos@  v- ds schritte >r
   3dup mb-pos pos@  v- ds schritte >r
   ma-pos pos@  v- ds schritte r> r> ;
```

Das Wort *seillaengen* erwartet eine Greiferposition auf dem Datenstack und berechnet daraus die Seillängen zu den drei Motoren *ma*, *mb* und *mc*.

```
: init-seillaengen ( -- ) gr-org pos@ 3dup
   gr-pos pos! seillaengen mc nip ! mb nip ! ma nip ! ;
   init-seillaengen
```

Bei einem Neustart des Roboters sorgt *init-seillaengen* für die Reinitialisierung der Variablen.

```
: positioniere ( x y z -- )
   seillaengen
   mc nip @ swap - crelstep
   mb nip @ swap - brelstep
   ma nip @ swap - arelstep waiting ;
```

Danach bewegt *positioniere* die Motoren um einen Schritt weiter.

3.7 Vorbereitungen zur OpenGL-Simulation

Alle vorausgegangenen Programmteile können zur Simulation des Roboters durch eine OpenGL-Graphik genutzt werden. Lediglich das Wort *positioniere* wird dazu redefiniert. Da ein virtuelles System keine Beschleunigungs- und Bremsrampe benötigt, benutzen wir keine Rampe, sondern lesen die Verzögerungszeit aus der Variablen *zdif*. Wir ermöglichen so die Geschwindigkeit über ein Dialogfeld im Simulator zuverändern.

```
[IFDEF] simulation
Variable actualized?
```

Die Variable *actualized?* teilt dem Simulator (Der Simulator arbeitet als Beobachter) mit, daß der Roboter eine neue Position eingenommen hat und die Graphik aktualisiert werden muß.

```
Create greiferpos   0 ,   0 , 0 ,
gr-org pos@ greiferpos pos!
Variable magnet
```

Weiter darf der Magnet nicht mehr geschaltet werden. Sein Zustand wird in der Variablen Magnet gespeichert.

```
: positioniere ( x y z -- )
     wartezeit @ zdif @ wartezeit +! till
     greiferpos pos! actualized? on ;
  [THEN]
```

3.8 Das Benutzerinterface

Nach der Definition von vielen programminternen Forth-Worten kommen wir nun zu den Worten, die der Benutzer zur Bedienung des Roboters benötigt.

```
: action key? IF key
   $FF52 case? IF  1 gr-pos y-pos +! false exit THEN
   $FF54 case? IF -1 gr-pos y-pos +! false exit THEN
   $FF51 case? IF  1 gr-pos x-pos +! false exit THEN
   $FF53 case? IF -1 gr-pos x-pos +! false exit THEN
   $0075 case? IF -1 gr-pos z-pos +! false exit THEN
   $0064 case? IF  1 gr-pos z-pos +! false exit THEN
   $1B case? IF true exit THEN THEN false ;

: mmm BEGIN pause gr-pos pos@ positioniere
   action until ;
```

Das Wort *mmm* erlaubt das manuelle, schrittweise Verfahren der Achsen und dient zur Erprobung des Systems und zum Fahren auf die Referenzposition.

```
do: : dx
do: : dy
do: : dz
```

Die Objekte *dx*, *dy*, *dz* werden von der Klasse *do*: erzeugt. Ihre Aufgabe besteht darin, zwei Strecken schrittweise in einem bestimmten Verhältnis zu teilen. Die Methode >*ratio* übergibt zunächst zwei Strecken an das Objekt. Mit jedem Aufruf der Methode *go* wird intern um eine Streckeneinheit erhöht. Beim Aufruf der Methode *do?* erhalten wir Schritte in dem vorgegebenen Streckenverhältnis zurück. Das Objekt gewährleistet eine Teilung der Strecke ohne Rundungsfehler und ermöglicht eine fehlerfreie Interpolation beim Achsenvorschub.

```
: absbewegen ( x y z -- )
   gr-pos pos@ v- 3dup ds dup 0= IF 2drop 2drop EXIT
                                 THEN
   under dz >ratio under dy >ratio under dx >ratio
   dup laststep ! step off  timer@ wartezeit !
   0 ?DO dx go dy go dz go
   dx do? dx counter @ 0> IF negate THEN
   dy do? dy counter @ 0> IF negate THEN
   dz do? dz counter @ 0> IF negate THEN
   gr-pos pos@ v+  3dup gr-pos pos!
   positioniere LOOP ;
```

Alle Bewegungen des Systems leiten sich von dem zentralen Wort *absbewegen* ab. Bei seiner Ausführung nimmt es die drei absoluten Positionsangaben (x y z) vom Datenstack und bewegt den Greifer auf dem kürzesten Wege von der momentanen Position zur neuen Position xyz.

```
: down ( dz -- )
   0 0 rot gr-pos pos@ v+ absbewegen ;
```

down bewegt den Greifer auf der Z-Achse um dz Streckeneinheiten nach unten,

```
: up   ( dz -- )
   negate down ;
```

während *up* den Greifer in die entgegengesetzte Richtung, also nach oben bewegt.

```
: left ( dx -- )
   0 0  gr-pos pos@ v+ absbewegen ;

: right ( dx -- )
   negate left ;

: forw  ( dy -- )
   0 swap 0  gr-pos pos@ v+ absbewegen ;

: back ( dy -- )
   negate forw ;
```

Entsprechend den Worten *up* und *down*, bewegen die Worte *left* und *right* den Greifer auf der X-Achse und die Worte *forw* und *back* tun das gleiche auf der Y-Achse.

4. Schlußbemerkung

Das zuvor beschriebene Programm zeigt die grundlegenden Funktionen zur Steuerung einer dreiachsigen Maschine. Richtige Roboter sind oft mit zwei weiteren Achsen zum Drehen und Neigen von Werkzeugen ausgestattet. Je nach Bauweise sind ganz unterschiedliche Algorithmen zur Berechnung der Positionen und zum Bewegen der Achsen notwendig. Entwickeln diese Maschinen größere Kräfte, müssen entsprechende Maschinenrichtlinen und Normen beachtet werden. Als wichtige Beispiele möchte ich den Notaus und den Kollisionsschutz nennen. Die Programmierung wird dadurch um ein Mehrfaches aufwendiger.

Anm. d. Redaktion: Eine „wirkliche“ Maschine [Triceps gilt hier sicher im Sinne der EG-Maschinenrichtlinie als Ausstellungsstück mit geringem Gefährdungspotential] muß nicht nur die Anforderungen der EG-Maschinenrichtlinie erfüllen, sondern mindestens auch die Anforderungen der EG-Niederspannungsrichtlinie und der EG-EMV-Richtlinie, die in Deutschland als EMV-Gesetz in nationales Recht umgesetzt worden ist. Für verschiedene Einsatzfälle – Chemie, Kohlen-Bergbau, Mühlen u.a. – müssen zusätzlich ab dem 1. Juli 2003 die Anforderungen der ATEX-Richtlinie erfüllt werden (Ex-Schutz). Hierbei muß der Hersteller bereits bei der Planung und Konstruktion sicherstellen, daß sein „Stück Maschinenbau“ keine potentiell explosionsfähigen Atmosphären zünden kann.
Für „uns Programmierer“ bedeutet das konkret, daß auch die Programme von z.B. Triceps daraufhin überprüft werden müssen, ob sie eine technisch einwandfreie und in jedem denkbaren Fall ungefährliche Funktion gewährleisten. Denkbar (konstruierbar) wäre z.B. eine Arbeitssituation, in der ein großer Triceps schwere, metallene Gegenstände in einer Halle bewegt, in der sich brennbare Gase sammeln können. Eine elektro-magnetische Störeinstrahlung veranlaßt den Triceps, seine Last so unsanft an die Hallenwand zu donnern, daß Funken geschlagen werden. Die Halle, einschließlich der darin befindlichen Menschen, sind danach ein unangenehmer Teil der Firmengeschichte (sofern diese das Desaster wirtschaftlich überlebt).
DAS will natürlich Niemand (das Unternehmen verlieren). Darum ist das Kosten– und Aufwandsargument nicht gar so griffig. Maschinenbauer UND Programmierer, die sich bisher um solche Dinge – im Gegensatz zu den Elektrotechnikern – meist nicht zu kümmern brauchten, fühlen sich allerdings häufig von einer komplexen Gesetzeslage erschlagen.
Es muß aber nichts „weh tun“. Europäische Normen (DIN EN) helfen aus den meisten Verlegenheiten heraus. Und in den Fällen, in denen es solche Normen gar nicht gibt, ist nach wie vor der Einäugige (Hersteller) König unter den Blinden.

fep

Haben Sie für die Mitglieder der Forthgesellschaft Hinweise auf Share- und Freeware? Können Sie uns Erfahrungsberichte geben? Wie kommen Sie zum Beispiel mit Knoppix zurecht? Oder würden Sie den Lesern der VD lieber eine ganz andere Distribution empfehlen? Wo lassen sich welche Treiber herunterladen? Mit welcher Hardware haben Sie in den letzten zwölf Monaten welche Erfahrungen gemacht?

Schreiben Sie uns doch bitte zu diesen und anderen Fragen einfach Ihre Antworten. *fep*

Andrea und Ewald Rieger, die ‚Ausrichter‘ der Tagung in Lambrecht.

Forth-Gruppen regional

Moers

Friederich Prinz
Tel.: (0 28 41) - 5 83 98 (p) (Q)
(Bitte den Anrufbeantworter nutzen!)
(Besucher: Bitte anmelden!)
Treffen: 2. und 4. Samstag im Monat
14:00 Uhr, **MALZ, Donaustraße 1**
47441 Moers

Mannheim

Thomas Prinz
Tel.: (0 62 71) - 28 30 (p)
Ewald Rieger
Tel.: (0 62 39) - 92 01 85 (p)
Treffen: jeden 1. Mittwoch im Monat
Vereinslokal Segelverein Mannheim e.V.
Flugplatz Mannheim-Neuostheim

München

Jens Wilke
Tel.: (0 89) - 8 97 68 90
Treffen: jeden 4. Mittwoch im Monat
Ristorante Pizzeria Gran Sasso
Ebenauer Str. 1
80637 München

Hamburg

Küstenforth
Klaus Schleisiek
Tel.: (**0 40) - 37 50 08-13 (g)**
kschleisiek@send.de
Treffen 1 mal im Quartal
Ort und Zeit nach Vereinbarung
(bitte erfragen)

Gruppengründungen, Kontakte

Hier könnte Ihre Adresse oder Ihre Rufnummer stehen – wenn Sie eine Forthgruppe gründen wollen.

µP-Controller Verleih

Thomas Prinz
Tel.: (**0 62 71) - 28 30** (p)
micro@forth-ev.de

Forth-Hilfe für Ratsuchende

Jörg Plewe
Tel.: (**02 08) - 49 70 68** (p)

Jörg Staben
Tel.: (**0 21 03) - 24 06 09** (p)

Karl Schroer
Tel.: (**0 28 45) - 2 89 51** (p)

Spezielle Fachgebiete

Arbeitsgruppe MARC4	**Rafael Deliano** Tel./Fax: (**0 89) - 8 41 83 17** (p)
FORTHchips (FRP 1600, RTX, Novix)	**Klaus Schleisiek-Kern** Tel.: (**0 40) - 37 50 08 03** (g)
F-PC & TCOM, Asyst (Meßtechnik), embedded Controller (H8/5xx// TDS2020, TDS9092), Fuzzy	**Arndt Klingelnberg**, Consultants akg@aachen.kbbs.org Tel.: **(00 32)(0 87) - 63 09 89** (pgQ) Fax **- 63 09 88**
KI, Object Oriented Forth, Sicherheitskritische Systeme	**Ulrich Hoffmann** Tel.: (**0 43 51) - 71 22 17** (p) Fax: **- 71 22 16**
Forth-Vertrieb vlksFORTH ultraFORTH RTX / FG / Super8 KK-FORTH Ingenieurbüro	**Klaus Kohl** Tel.: (**0 82 33) - 3 05 24** (p) Fax : (**0 82 33) - 99 71** mailorder@forth-ev.de

Möchten Sie gerne in Ihrer Umgebung eine lokale Forthgruppe gründen, oder einfach nur regelmäßige Treffen initiieren? Oder können Sie sich vorstellen, ratsuchenden Forthern zu Forth (oder anderen Themen) Hilfestellung zu leisten? Möchten Sie gerne Kontakte knüpfen, die über die VD und das jährliche Mitgliedertreffen hinausgehen?

Schreiben Sie einfach der VD - oder rufen Sie an - oder schicken Sie uns eine E-Mail!

Hinweise zu den Angaben nach den Telefonnummern:

Q = Anrufbeantworter
p = privat, außerhalb typischer Arbeitszeiten
g = geschäftlich

Die Adressen des Büros der Forthgesellschaft und der VD finden Sie im Impressum des Heftes.

Forth-Tagung 2003
Lambrecht, Neustadt/Weinstraße

11. bis 13. April 2003

Photo: A. Kostrov

Photo: F. Prinz

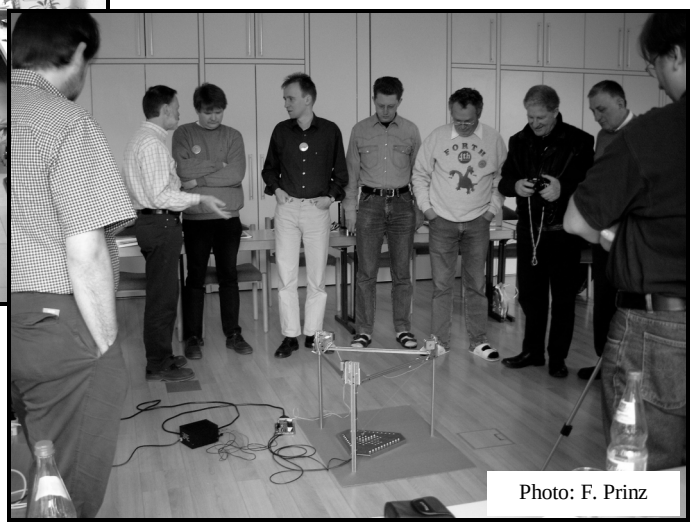


Photo: F. Prinz

Mehr Photos finden Sie auf dem WEB-Server der Forthgesellschaft