# VIERTE DIMENSION

1/1993

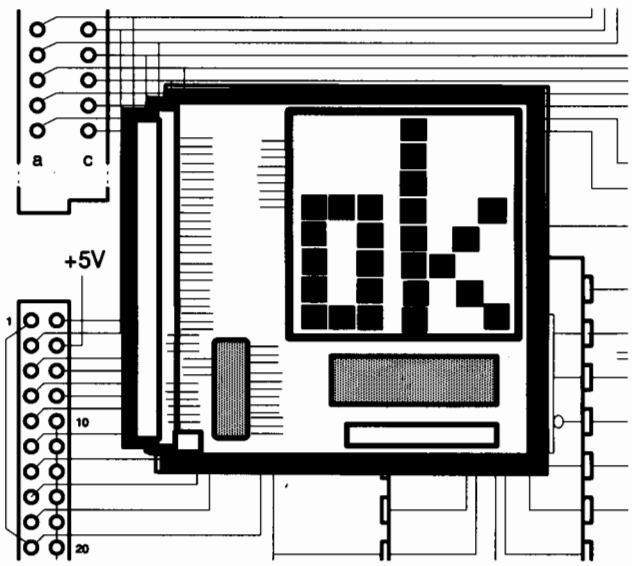
9. Jahrgang 1993 1. Quartal DM 10,-


# Forth++

## Hard DisCo

## fis(c)hing Forth

## KeyBaB




## DFU Eins

## FILE DUMPen

## Forth spinnt

## Schweinekiste

Bücher, Briefe, Infos, ANS-Forth


FORTH MAGAZIN

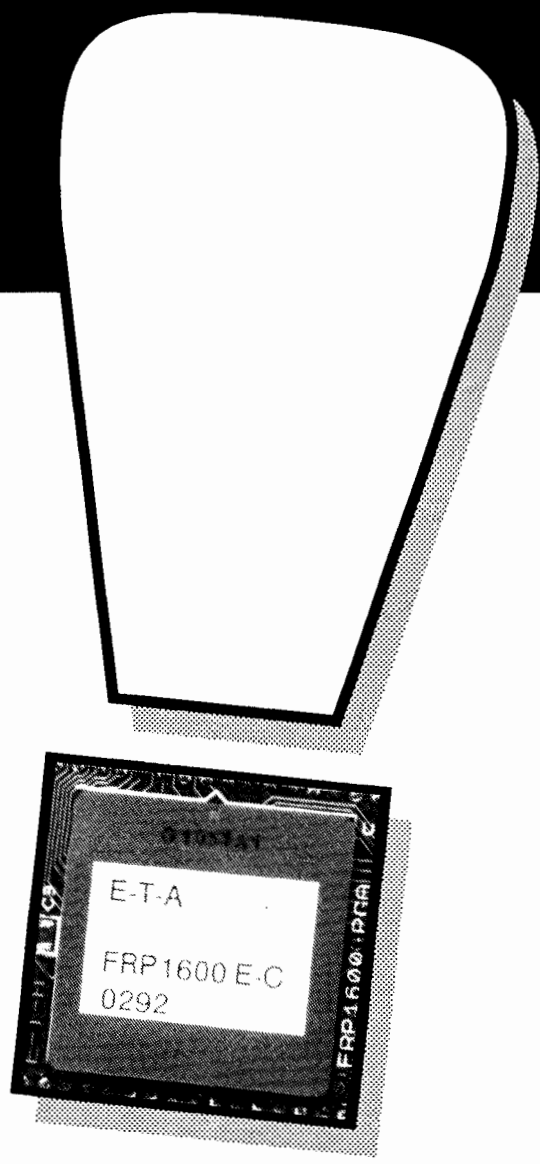
Organ der FORTH-Gesellschaft e.V.

**Herstellung und Vertrieb:**

E-T-A®
Elektrotechnische Apparate GmbH
Industriestraße 2-8
W-8503 Altdorf/Nürnberg
Postfach 1061
Tel. 09187/10-0 Tlx. 624 461
Fax. 09187/10397

**Design:**

FhG IIS
Institut für
Integrierte
Schaltungen




# Der fränkische FORTH-RISC-PROZESSOR FRP1600 ist da

**Kundenbetreuung:**

SysMik GmbH DRESDEN
Systemlösungen mit Mikroelektronik
Holbein Straße 2
O-8019 Dresden
Tel. (0351) 4 59 82 12
Fax. (0351) 2 36 15 40

MIKROMAK GmbH
Am Weichselgarten 7
W-8520 Erlangen
Tel. 0 91 31/69 12 50
Fax. 0 91 31/69 11 11

DELTA t
Entwicklungsgesellschaft
für computergesteuerte Echtzeitsysteme mbH
Uhlenhorster Weg 3
W-2000 Hamburg 76
Tel. 040/2 29 64 41

## IMPRESSUM

### Name der Zeitschrift
VIERTE DIMENSION
FORTH MAGAZIN
Organ der Forth-Gesellschaft e.V.

### Herausgeber
Forth-Gesellschaft e.V.
Postfach 1110
W-8044 Unterschleißheim
Tel.: 089-3173784 oder
Forth-Mailbox Tel. 089-8714548 8N1

### Redaktionsleitung
Rolf Kretzschmar (rk), (verantwortlich)
Rote Gasse 7, W-5112 Baesweiler
(Redaktionsadresse)
Tel/Fax: 02401-88891

### Redaktion
Arndt Klingelnberg (akg), Alsdorf
Tel.: 02404-61648 Fax: 02404-63039
Klaus-Peter Schleisiek (kps), Aachen
Tel/Fax: 0241-873462

### Layout, Satz, Herstellung
ORGA Sport, Rilkestr. 8, W-5110 Alsdorf
Tel/Fax: 02404-61425

### Grafik, Illustration, Layout
Rolf Kretzschmar (rolf)

### Anzeigenverwaltung
Arndt Klingelnberg
Straßburger Str. 12
5110 Alsdorf
Tel.: 02404-61648 Fax: 02404-63039

### Redaktionsschluß
Feb., Mai, Aug., Nov.

### Erscheinungsweise
vierteljährlich

### Auflage
1000

### Preis
Einzelheft DM 10,-, Abonnementpreis DM 50,-, bei Auslandsadresse DM 55,- inklusive Versandkosten




# Das Fossil

von Rolf Kretzschmar

Wußten Sie schon, daß die VD sich überwiegend mit vorzeitlichen, versteinerten Gegenständen beschäftigt? Glaubt man der Februar-Ausgabe der Zeitschrift AMIGA PLUS, so handelt es sich bei dem dort getesteten Helios-Forth um einen interessanten "Versuch, das Sprachfossil Forth in ein amigagemäßes Gewand zu kleiden". Ich denke, die Leser der VD haben da eine etwas andere Sicht der Dinge. Gesteht nicht selbst ein so kritischer Forther wie *Ulrich Paul* (Leserbrief in 4/92) der Sprache Forth noch einen gewissen Vorsprung zu, wenn er bemerkt "Es ist keine Frage von Jahren mehr, daß z.B. C++ in dieser Richtung (Erweiterbarkeit, d.R.) Forth voll überholt.."? Echsen und Schildkröten werden schon mal als "lebende Fossilien" bezeichnet. Gut, damit bin ich einverstanden: Forth, das lebende Fossil; F-PC, der quicklebendige Saurier! Leser der VD wissen, in welchen Nischen unser geliebtes, verfluchtes Biest sich quietschvergnügt tummelt. Robotik und Forth könnten zu Synonymen werden. *Rafael Deliano* bewältigt den offensichtlichen Stau an Informationen zum Thema Robotik dadurch, daß er ein Heft verlegt (Creeping FORTH), in dem Interessenten zahlreiche Anregungen, Informationen und die Bitte um Mitarbeit finden. Wir werden seine Arbeit interessiert verfolgen und wohl auch darüber berichten.

Dem lebenden Fossil wachsen auch noch neue Extremitäten: Die Crew um *Prof. Pleßmann* zeigt in diesem Heft, wie man ihm zum ++-Appendix verhelfen kann.

Die Aussage: "Für Lernzwecke (...) ist Forth allgemein wenig geeignet, da heute immer mehr Wert auf strukturierte Programmierung gelegt wird" ist wohl die unüberlegteste im eingangs erwähnten Artikel: Vermutlich hat der Verfasser (*Oliver Wagner*) etwas anderes im Sinn; denn strukturierter als Forth kann kaum noch eine Sprache sein. Und nicht zuletzt deswegen stehen wir nicht alleine mit der Meinung, daß Forth die Lehr-Sprache schlechthin ist. So stellen *Friederich Prinz* und *Klaus-Peter Schleisiek* in diesem Heft Hardware vor, die das Lehren und Lernen mit Forth zum Vergnügen werden lassen! Ein lebendes Sprachfossil läßt grüßen! Ich auch!

Euer

R. Kretzschmar

# Forth und der Rest der Welt

von Andreas Goppold

Unterherrmhauser Str. 13 D-8196 Eurasburg Tel (++49) +8179 1479

Ich möchte mich erst einmal recht herzlich bei euch bedanken. Die Resonanz auf meine letzte Kolumne war sehr positiv, obwohl ich ja hier und da in Bezug auf Forth eine Lippe riskiert habe. Ich freue mich, daß ich mit meinen Ansichten doch bei den Forthern etwas berühre und ausspreche, das viele auch bewegt. Und es ist noch einiges an Bewegung im Kommen. Dieses Jahr ist mein Jubiläumsjahr. Vor zehn Jahren bin ich Jünger der "Wahren Leere vom Forth" geworden. Damals, ja damals, als wir noch in einer Schnitzeljagd über verschiedene Adressen, wo die Sitzung nicht stattfand, dann endlich zu diesem ehemaligen Kino in Hamburg gelangt sind, wo wir dann auf den Theaterstühlen hockend uns in die Mysterien von DUP ROT SWAP, CREATE und DOES von *Klaus Schleisiek*, seines Zeichens Prophet der frohen Botschaft des Wortes von Forth, einweihen ließen. Zu solchen Jubiläumsgelegenheiten fällt es dann auch leicht, mit feuchten Augen und bewegter Stimme über die Unstimmigkeiten und Zwiste der Vergangenheit wegzusehen und die verbindenden Gemeinsamkeiten wiederzufinden.

Ich habe heute leider nur Platz für eine Druckseite, und so muß ich mich ein wenig kurz fassen. Das fällt mir schwer. Ich sehe nämlich einige sehr relevante Dinge auf uns zukommen, die der Idee von Forth eine sehr leuchtende Zukunft bringen können.

"Wot dem eenen sien Schiet iss dem annern sien Dünger" ist ein altes ostfriesisches Sprichwort, das schon lange mein Wahlspruch ist. In der Tat ist es so: Die anderen stecken jetzt tief in ihrem Schiet und das kann sehr wohl der Dünger für die Ansätze von Forth sein. Die Spatzen pfeifen es von den Dächern, und es fühlen jetzt auch die Blinden mit dem Krückstock, daß die Computer-Industrie bis über die Nasenspitze in jenem besagten Schiet steckt. Darum sage ich es jetzt laut und deutlich: Die Technologie der Objekt-Compilierung und damit aller herkömmlichen Compilersprachen, hat sich an ihrem eigenen Erfolg totgelaufen. Die Zeit ist gekommen für die Metacode- P-Code- und Tokencode-Systeme. Forth ist ein Tokenlisten-Interpreter System. Die Software-Technologie der Zukunft liegt eindeutig bei den Metacode- Systemen. Die heutigen RISC-Prozessoren sind ideal geeignet für solche Systeme. Und die Forth-Programmierer haben den Metacode ja sozusagen mit der Muttermilch aufgesogen und kennen "every nook of granny".

Wie das gemeint ist, kann man an Postscript sehen, welches im Augenblick die einzige (wenn auch nicht sehr effiziente) Möglichkeit für eine device-unabhängige Graphik-Darstellung ist. Das X-Windows System dient den Hardware-Herstellern nur dazu, jedes Jahr noch mehr MIPS verkaufen zu können und bindet Millionen von kostbarsten Programmierer-Mannstunden, um dieses Monster nur irgendwie zu handhaben. Leider hat die Computerindustrie ihre Rechnung ohne den Wirt, sprich ohne die Welt-Ökonomie gemacht. In unserer Welt ist Rezession angesagt. Seit der kalte Krieg beendet ist, fließen die Dollarmilliarden aus dem Pentagon nicht mehr, und damit sitzt das gesamte Silicon Valley auf dem Trockenen. Das Lebensblut dieser ungeheuer aufgeblähten, mit den Ressourcen um sich schmeißenden Industrie trocknet aus. Das Zeitalter von "Small is Beautiful" bricht nun für die Computerindustrie an. Und small, das sind vor allem Tokencode-Systeme. Mit Tokencode-Systemen kann man typischerweise auf 1/3 oder weniger des Ressourcenbedarfs im Vergleich zu object- compilierten Systemen kommen. Ich werde auf der CBIT in Hannover meine Version dieser Idee vorstellen. Vor zehn bis 15 Jahren hatte Forth seine historische Chance in der Computerindustrie. Sie wurde aus diversen Gründen vertan. Die Stunde der Tokencode-Prozessor Systeme aber kommt noch. Forth als Sprache ist meiner Ansicht nach ein toter Arm der Evolution. Forth als Idee ist so aktuell wie nie zuvor.

❑

## Termine

akg

**Hannover (Industrie) Messe**
1993april21--28 Hannover

**FORTH-Tage 93**
Kongress und MitgliederVersammlung
1993apr23--25 Nürnberg

**Münchener Elektronik Börse**
Großer Elektronik Flohmarkt
1993may02 München-Theresienhöhe
Pschorr-Keller

**ComputerShow / HobbyTronic**
1993may12--16 Dortmund

**The Forth Professional WorkShop '93**
1993may27--30 BadenBaden
Klaus Felsch, Steven Pelc

**echtzeit/iNET**
1993juni(14) 15--17 Karlsruhe
Ausstellung und Kongress
Fuzzy-Controller
embedded systems
(Feld-)BusSysteme
ProgrammierWettbewerb

**ATARI**
1993aug20--22 Düsseldorf

**Internationale Funkausstellung**
1993aug27--sep05 Berlin

**MessComp '93**
7. KongressMesse für industrielle Meßtechnik
1993sep07--09

**Europäische WerkzeugMaschinen Ausstellung 'EMO'**
1993sep14--22 Hannover

(bitte geben Sie uns ihren Input zu für Forthler sinnvollen Veranstaltungen: Forth-Kurse, Forth-Anwendungen, Forth-Kunden, Forth-Konkurrenten, Hardware für Forthler)

# Inhalt

# Forth++ (1)

von Burkhard Golf, Rolf Schönlau,
Franz Angenendt, Klaus W. Pleßmann

pdv, RWTH Aachen, Beverstr. 46, 5100 Aachen, Tel. 0241-500749

Ein Objekt-orientierter Forth-Sprachinterpreter als Kern eines Informationssystems

Teil 1: Forth++: Eine Erweiterung von Forth um Typen und Objekt-orientierte Strukturen

Teil 2: INFOSYS-106: Ein Informationssystem auf der Basis eines Objekt-orientierten Forth-Sprachinterpreters

Die vorliegende zweiteilige Veröffentlichung stellt schwerpunktmäßig die Forth-Komponente des Informationssystems INFOSYS-106 vor, das im Rahmen des Sonderforschungsbereiches 106 "Korrelation von Fertigung und Bauteileigenschaften bei Kunststoffen" [Pleß92] implementiert wurde. Der Sonderforschungsbereich 106 setzte sich mit der Problematik auseinander, Kunststoffe in der Verarbeitung gezielt beeinflussen zu können, um gewünschte Eigenschaften vorhersagen und erreichen zu können. Die Aufgabe des Informationssystems lag in der Erfassung und Aufbereitung anfallender Informationen. Die Art der Erfassung reichte von persönlichen Notizen, über strukturierende Aussagen bis zur Dokumentation und Begründung von Ergebnissen der bearbeiteten Forschungsschwerpunkte.

Aus diesen Anforderungen folgte die Entscheidung über eine interpretierende Auswertung von Informationseinheiten und Informationsbasis. Um nicht in der Verarbeitungsgeschwindigkeit beeinträchtigt zu werden, wurde dem Informationssystem ein Interpreter zugrundegelegt, für dessen Realisation ein Forth83-System gewählt wurde, da mit Forth die anfallenden Aufgaben, durch die besondere Technik der Fädelung von Adreßverweisen des zu verarbeitenden Codes, sehr effizient gelöst werden konnten.

Durch die Interaktivität des Interpreters wurde eine lose Kopplung von Informationseinheiten, die mit individuellen Daten besetzt werden können, erreicht. Die heutzutage vorhandenen Möglichkeiten an Programmiersicherheit und -komfort machten eine Verwaltung der Informationseinheiten mit den Mitteln der Objekt-orientierten Programmierung erstrebenswert, denn die Informationseinheiten können vorteilhaft als Objekte im Sinne der Objekt-orientierten Programmierung aufgefaßt werden. Mit dem ersten Teil der Veröffentlichung wird nun eine Objekt-orientierte Erweiterung von Forth vorgestellt, die wir Forth++ genannt haben und die sicherlich nicht nur für die im zweiten Teil vorgestellte Anwendung interessant sein dürfte.

## Teil 1: Forth++:

Eine Erweiterung von Forth um Typen und Objekt-orientierte Strukturen

### 1. Zielsetzung

Die Idee Forth "aufzubohren", um es komfortabler und sicherer zu machen, ist sicherlich nicht neu. Es gab in der Vergangenheit schon mehrere Ansätze, die allerdings versuchten, über trickreiche Wortdefinitionen neue Formulierungsmöglichkeiten zu bieten. Beispielsweise in [POUT87] werden solche Möglichkeiten beschrieben, mit denen sich Datenstrukturen und Objekt-orientierte Mechanismen realisieren lassen.

Hier wurde ein grundsätzlich anderer Weg eingeschlagen, nämlich der klassische Weg des Compilerbaus. Ausgangspunkt war nicht die Überlegung, wie man mit besonders raffinierten Wortdefinitionen hochsprachenähnliche Konstrukte implementiert, sondern das Design einer Sprachdefinition für ein Objekt-orientiertes Forth, ohne zunächst durch Überlegungen der Umsetzung im Denkprozeß blockiert zu werden. Diese grundsätzliche Entscheidung ergab sich auch aus dem Problem, die Sprachsemantik sonst nicht exakt beschreiben zu können.

Von der Sprachdefinition einer Programmiersprache hängt es nämlich ab, welche Art von semantischen Fehlern, die der Programmierer macht, bereits vom Compiler abgefangen oder auch zur Laufzeit gemeldet werden. Man denke nur an die Kontrolle der Parameterübergabe bei Unterprogrammaufrufen oder die Typüberprüfungen bei Wertzuweisungen. Das Dilemma einer Sprachdefinition besteht aus der Forderung nach viel Sicherheit, aber auch der Forderung nach Effizienz, guter Pragmatik und nicht allzuvieler Restriktionen, wegen der Einschränkung der Ausdrucksmöglichkeiten.

An dieser Stelle sollen nur kurz die wichtigsten Diskussionspunkte wiedergegeben werden, die das Design des Objekt-orientierten Forths letztendlich motiviert haben. Ausgangspunkte sind die charakteristischen Eigenschaften von Forth und die Fragestellung, inwieweit diese Eigenschaften übernommen werden sollen bzw. können. Für detailliertere Informationen sei der interessierte Leser hier und im folgenden auf [Schö92] verwiesen.

- Die Interaktivität war für die Wahl von Forth einer der entscheidenden Faktoren und darf nicht in Frage gestellt werden.
- Die Erweiterbarkeit und das Wörterbuchprinzip, die bei der Bottom-

***Die Autoren:***
*Dipl.-Inform. Burkhard Golf, Dipl.-Inform. Rolf Schönlau, Dr. rer nat. Franz Angenendt, Prof. Dr.-Ing. Klaus W. Pleßmann;*
*Lehr.- und Forschungsgebiet für Verfahren der Prozeßdatenverarbeitung und Prozeßführung Rheinisch-Westfälische Technische Hochschule Aachen.*

up Realisierung von Programmen unschätzbare Dienste leisten, werden ebenfalls beibehalten.

- Die an Forth geschätzte Maschinennähe kann in der Form nicht aufrechterhalten werden. Die Maschine darf nicht über das direkte Arbeiten mit Systemadressen erreichbar sein, sondern nur über klar definierte Schnittstellen.
- Die Anforderungen an die Kompaktheit eines Forthsystems sind im Zeitalter immer leistungsfähiger und preiswerter werdender Prozessoren und Speicherbausteine neu zu überdenken. Es würde keinen Sinn machen, bei einem Objektorientierten, komfortablen Forth mit 16kB Speicher auskommen zu wollen. Auf der anderen Seite soll es der Philosophie von Forth entsprechend immer noch relativ kompakt sein.
- Die Stackorientierung wurde beibehalten, weil die Komplexität um Größenordnungen angestiegen wäre, die einen vernünftigen Kompromiß bezüglich der Kompaktheit ausgeschlossen hätten.
- Durch die Objekt-Orientierung ist eine Verschlechterung der Performance zu erwarten. Eine Laufzeitverschlechterung in der Größenordnung von 10 scheint realistisch, aber auch vertretbar zu sein.

## 2. Die Sprachkonstrukte von Forth++

Im folgenden Kapitel werden die Sprachkonstrukte von Forth++ in der erweiterten Backus-Naur-Form dargestellt. Eine ausführliche Einführung in Forth83 findet der Leser in [Zech87].

### 2.1. Das Typkonzept von Forth++

Ein Forthprogramm (eine Sitzung im Forthsystem) besteht aus einer Folge von Definitionen und Befehlen, die mit dem Befehl BYE abgeschlossen wird.

```
Prog ::=
   { Def | Instr } BYE
```

Definitionen (Def) können Typ-, Wort-, Konstanten-, Variablen- oder Wörterbuchdefinitionen sein.

```
Def ::=
   Typedef |
   Worddef |
   Constdef |
   Vardef |
   Vocdef
```

Eine der bedeutsamsten Erweiterungen ist das Typsystem, das eine Reihe von Basistypen und auch Konstruktoren für Typneudefinitionen zur Verfügung stellt, so daß Felder, Verbunde, Aufzählungstypen, Unterbereichstypen und Klassen definiert werden können..

Zu den Basistypen gehören Integer, Real, Boolean, Zeichen, Adressen und Strings.

```
Basetype ::=
   INT16 | INT32 |
   REAL | BOOLEAN |
   CHAR | ADR | STRING
```

Ein Typbezeichner ist entweder ein Basistyp oder ein bereits dem System bekannter Bezeichner, dem eine Typdefinition zugrunde liegt.

```
Typeid ::=
   Basetype | Id_type
```

Typdefinitionen beginnen mit dem Schlüsselwort TYPE gefolgt von dem Bezeichner für den neuen Typ und einer Typbeschreibung.

```
Typedef ::=
   TYPE Id Typespez
```

Typbeschreibungen umfassen Felder, Aufzählungstypen, zwei Arten von Unterbereichstypen, Pointertypen und Klassendefinitionen.

```
Typespez ::=
   Arraydef |
   Enumdef |
   Subtypedef |
   Newtypedef |
   Pointerdef |
   Classdef
```

Ein Feldtyp wird von den Feldgrenzen, die durch Integer16-Konstanten angegeben werden, beschrieben und von dem Typ der einzelnen Elemente.

```
Arraydef ::=
   ARRAY "[" Int16const
         "," Int16const
         "]" of Typedescr
```

Der Typdeskriptor umfaßt bekannte Typbezeichner und Stringbeschreibungen, die durch die maximale Anzahl von Zeichen charakterisiert sind.

```
Typedescr ::=
   Typeid |
   STRING"["Int16const"]"
```

Aufzählungstypen werden durch die zu ihnen zugehörigen Aufzählungsbezeichner definiert, die von dem Schlüsselwort ENUM und einem Semikolon eingerahmt werden.

```
Enumtype ::=
   ENUM Id { Id } ";"
```

Unterbereichstypen lassen sich mit (SUBTYPE) und ohne (NEWTYPE) Zuweisungskompatibilität definieren. Der Wertebereich läßt sich bezüglich des Bezugtyps einschränken.

```
Subtypedef ::=
   SUBTYPE Typeid
   [LIMITS Numconst
           Numconst] ";"
Newtypedef ::=
   NEWTYPE Typeid
   [LIMITS Numconst
           Numconst] ";"
```

Mit Pointertypen wird das Arbeiten mit Maschinenadressen in eine sichere Bahn gelenkt. Zu jedem Typ läßt sich ein entsprechender Pointertyp definieren.

```
Pointerdef ::=
   POINTER TO Typeid
```

Die tiefgreifendste und auch umfangreichste Erweiterung im Typkonzept umfaßt die Definition von Klassen. Mit dem Klassentyp lassen sich Objekte definieren, die einen eigenen Datenbereich und die zulässigen Operationen auf diese Daten umfassen. Pate bei der Festlegung der Klassendefinition stand C++.

Eine Klasse kann als Ableitung einer schon bestehenden Klasse definiert werden. Mit dem Schlüsselwort PUBLIC wird festgelegt, ob die ererbten Features nach außen hin sichtbar sein sollen oder nur für den internen Gebrauch bestimmt sind. Mit Typedef lassen sich optional lokale Typen definieren. Die Features bestehen aus Datendefinitionen und Methodendefinitionen. Methoden sind Wortdefinitionen, die nur innerhalb der Klassendefinition sichtbar sind und exklusiven Zugriff auf die Daten eines Objektes haben.

Features lassen sich in drei Sichtbarkeitskategorien einteilen. PUBLIC-Features, die nach außen hin sichtbar sind, PRIVATE-Features, die nur innerhalb der Klassendefinition

sichtbar sind; sie sind auch in abgeleiteten Klassendefinitionen unsichtbar. Und im Gegensatz dazu stehen die PROTECTED-Features, die zwar nach außen hin auch unsichtbar sind, aber in Unterklassen verwendet werden können. Mit SHARE lassen sich Klassenvariablen definieren, d. h. Variablen, die allen Objekten einer Klasse zugänglich sind, aber ansonsten im System unsichtbar bleiben. Mit Initmeth kann man optional eine Methode definieren, die genau einmal bei der Erzeugung eines Objektes ausgeführt wird und für Initialisierungen gebraucht werden kann.

```
Classdef ::=
   CLASS
   [CHILD OF [PUBLIC]
                  Typeid]
   { Typedef }
   [SHARE {Comp}]
   [[PRIVATE] Features]
   [PROTECTED Features]
   PUBLIC Features
   [Initmeth]
   ENDCLASS
```

Features sind Daten- oder Methodendefinitionen. Der Name des Nichtterminalsymbols Comp soll an Komponenten einer Verbunddefinition erinnern, da die Definition der Objektvariablen nichts anderes ist.

```
Features ::=
   { Comp } { Methdef }
Comp ::= Typedescr Id
```

## 2.2. Das Type-Checking in Forth++

Eine weitere grundlegende Änderung ergibt sich bei den Wortdefinitionen. Bei Forth-Programmierern ist es Sitte, zu jeder Wortdefinition als Kommentar dazuzuschreiben, was das Wort auf dem Stack erwartet und was es dort nach seiner Ausführung hinterläßt. Diese Beschreibung ist in Forth++ Bestandteil der Wortdefinition. Sie ist am ehesten mit einem formalen Parameter in bekannten Hochsprachen vergleichbar. Sie wird Zusicherung (Assertion) genannt und ist auch für Wörter des Forth++-Kerns bekannt.

Eine Zusicherung besteht aus einer Vor- und einer Nachbedingung. Die Vorbedingung ist eine Liste von Typangaben, die beschreibt, was das Wort auf dem Stack erwartet. Die Nachbedingung ist eine Liste, die beschreibt, was nach der Ausführung auf der Stackspitze zurückgelassen wird. Da das Stackverhalten für alle vorhandenen Wörter bekannt ist, läßt sich die vom Programmierer formulierte Zusicherung mit dem Rumpf der Wortdefinition vergleichen und Fehler sind somit frühzeitig erkennbar. Diese Aufgabe wird vom Typechecker im Forth++-System übernommen.

Mit diesem Mechanismus ist eine große Fehlerquelle in Forth entschärft. Um schwer nachvollziehbare Stackmanipulationsbefehle wie Rotation, Vertauschung usw. zu vermeiden, kann man an die Typbezeichner in den Zusicherungslisten auch Bezeichner binden, die in der Wortdefinition wie Variablen verwendet werden können. Die Paare Typname:Bezeichner werden Parameter genannt. Die Typlisten sehen Sequenzen und Iterationen, und somit, ob der angegebene Typ unbestimmt oft auf dem Stack liegen kann.

Die Zulassung von Iterationen beinhaltet einen Nichtdeterminismus bei der Verifikation der Zusicherung, so daß der Programmierer die Verantwortung über die Kontrolle der Anzahl der Iterationsschritte trägt. Alternativen sind nicht zugelassen, da sie eine Verifikation unmöglich machen würden. Alternativen können notfalls über die Zuweisungskompatibilität von abgeleiteter Klasse zur Oberklasse formuliert werden.

Wörter können explizit als überladen definiert werden. Dadurch wird es möglich, äquivalente Wörter auf unterschiedlichen Typen unter dem gleichen Namen zu führen. Das System sucht zur Laufzeit anhand der Typinformationen das passende Wort aus.

```
Worddef ::=
      ":" Id [OVERLOAD]
      Assertion Seq ";"
```

Mit dem Dach "^" vor dem Typnamen in einer Zusicherung läßt sich der Pointertyp zu einem Typen angeben, ohne ihn explizit als Typ definieren zu müssen. Damit soll auch in Forth++ das Arbeiten mit Adressen unterstützt werden.

```
Assertion ::=
   ASSERT "(" Typlist
   "->" Typlist ")"
Typlist ::=
   { Type | Itertyp }
   | { Parameter }
Itertyp ::=
   "{"Type{Type}"}"
Parameter ::=
   Type ":" Id
Type ::=
   Basetype | Id_type
   | "^" Type
```

Der Wortrumpf Seq ist eine Folge von Befehlen (Instr), Kontrollstrukturen (Cntrlinstr) und Zusicherungen. Eine im Wortrumpf vorkommende Zusicherung dient dazu, den Stackzustand an dieser Stelle zu postulieren. Es dürfen bei diesem Typ von Zusicherung keine Parameter verwendet werden. Die Vorbedingung ist immer leer, da der Zustand der Stackspitze definiert wird. Eine sinnvolle Anwendung wäre beispielsweise, wenn in einer Schleife alle auf dem Stack vorkommenden iterierten Elemente verbraucht sind und man diesen Sachverhalt dem Typechecker mitteilen möchte.

Als Kontrollstrukturen gibt es die Fallunterscheidung und die Schleife. Bei Fallunterscheidungen müssen beide Fälle das gleiche Typverhalten haben. Bei Schleifen muß darauf geachtet werden, daß der Schleifenrumpf den Stack so hinterläßt, daß er noch einmal darauf zugreifen kann. Ansonsten liefert der Typechecker eine Fehlermeldung.

```
Seq ::= { Inst |
Cntrlinstr | Assertion }
Cntrlinstr ::=
   IF Seq THEN |
   IF Seq_true ELSE
     Seq_false THEN |
   BEGIN Seq UNTIL |
   BEGIN Seq_1
     WHILE Seq_2 REPEAT |
   DO Seq LOOP |
   DO Seq +LOOP |
```

Ein Befehl (Instr) kann ein Wortbezeichner, ein Konstantenbezeichner, ein Wörterbuchbezeichner, eine Typkonversion, die Angabe alternativer Wörterbücher in der Suchfolge, eine Variable, eine Konstante oder ein String zum Zwischenspeichern sein.

```
Instr ::=
   Id_word |
```

```
    Id_const |
    Id_voc |
    Typecast |
    ALSO Id_voc |
    Qualvar |
    Const |
    "." "{char}"
Typecast ::=
    "((" Type "))"
```

Der Typecast erzwingt eine Typkonversion des obersten Elementes auf dem Stack. Es liegt völlig in der Verantwortung des Programmierers, dieses sinnvoll zu machen. Der Typecast ist nur möglich, wenn beide Typen den gleichen Speicher beanspruchen. Variablen werden über ihren Typ definiert und verhalten sich analog zu Variablen in Forth83. Bei der Ausführung wird ihre Adresse auf den Stack gelegt, die Dereferenzierung erfolgt explizit mit FETCH. Aus Kompatibilitätsgründen wurden die Variablendefinition von Integer mit normaler und doppelter Genauigkeit beibehalten.

```
Vardef ::=
    VAR Typedescr Id |
        VARIABLE Id |
        2VARIABLE Id
```

Aufgrund der Erweiterungen gibt es bei Variablenausführungen viele Spezialfälle. Die Features einer Klasse werden angesprochen, indem man hinter dem Variablennamen mit Doppelpunkt den Featurenamen folgen läßt. Es ist dabei gleich, ob es sich um eine Objektvariable (Verbundkomponente) handelt oder um eine Methode der Klasse. Auch Klassenvariablen lassen sich auf diese Weise ansprechen. Feldelemente werden angesprochen, indem man dem Variablennamen die eckigen Klammern folgen läßt. Der Index wird auf dem Stack erwartet.

Eine Besonderheit ist NEW. Läßt man einer Pointervariable ":NEW" folgen, wird dynamisch Speicherplatz alloziert und in die Variable die Adresse eingetragen, ähnlich wie in Pascal. Läßt man einer Pointervariable "^" folgen, so wird die Adresse der von ihr referenzierten Variable auf den Stack gelegt. Eine spezielle Variable ist SELF. Sie darf nur innerhalb von Methodendefinitionen verwendet werden und bezieht sich auf die Klasse, die gerade definiert wird.

Wegen der vielfältigen Qualifizierungsmöglichkeiten einer Variable wird auch von einer qualifizierten Variable gesprochen (Qualvar).

```
Qualvar ::=
    Id_var |
    SELF |
    Qualvar "^" |
    Qualvar ":" Id |
    Qualvar "[]" |
    Qualvar ":" NEW
```

Wichtig ist, daß eine qualifizierte Variable keine Blanks enthält! Es ist Aufgabe des Forth++-Systems die Analyse durchzuführen. Benannte Konstanten werden analog zu Variablen definiert und legen bei ihrer Ausführung ihren Inhalt und nicht ihre Adresse auf den Stack. Von Klassen können keine Konstanten gebildet werden, von Strings ebenfalls nicht. Der Inhalt der Konstanten wird bei ihrer Definition auf dem Stack erwartet.

```
Constdef ::=
    CONST Type Id |
          CONSTANT Id |
          2CONSTANT Id
```

Da Forth++ nicht nur Zahlen kennt, sondern viele verschiedene Typen unterscheiden kann, muß es eine Möglichkeit geben, Konstanten verschiedener Typen auf den Stack zu legen.

```
Const ::=
    Numconst |
    Boolconst |
    Enumconst |
    Charconst |
    Stringconst |
    Typedconst
```

Zunächst einmal gibt es Konstanten verschiedener Standardtypen, die sich anhand ihrer Syntax unterscheiden. Integer16 sind einfache Ziffernfolgen, Integer32 werden mit einem "d" zum Schluß gekennzeichnet. Realkonstanten haben immer einen Dezimalpunkt. Der Unterschied zu Forth83 ist absichtlich gewählt, da es keinen Sinn macht, den Punkt an einer beliebigen Stelle in der Zahl zur Kennzeichnung von ganzzahligen Werten zu benutzen. Character werden durch einfache Hochkommata eingeschlossen, Strings durch doppelte Hochkommata. Für boolsche Konstanten sind die Schlüsselwörter TRUE und FALSE reserviert. Aufzählungskonstanten wird ein "$" zur Kennzeichnung vorangestellt. Dem "$" kann optional der Typbezeichner vorangestellt werden. Er ist zwingend voranzustellen, wenn der Aufzählungskonstantenbezeichner nicht eindeutig ist.

```
Enumconst ::=
    [Id_type] "$" Id
```

Um für Unterbereichstypen, Felder, Pointer und Adressen ebenfalls Konstanten auf den Stack legen zu können, gibt es getypte Konstanten (Typedconst). Sie beginnen mit dem Typbezeichner und in runden Klammern folgt eine Liste von Konstanten.

```
Typedconst ::=
    Id_type "(" Const
    { "," Const } ")"
```

## 2.3. Änderungen in Forth++ gegenüber Forth83

Das System des Forth++ ist in der Lage, sich einen Überblick über die auf dem Stack befindendlichen Typen zu machen. Insofern sind alle Stackmanipulationsbefehle generisch und Befehle wie SWAP, ROT, DUP, FETCH, STORE berücksichtigen die Anzahl der Bytes, die jedes Stackdatum beansprucht. Dadurch können alle Spezialstackbefehle entfallen, die sich auf andere Datengrößen beziehen, wie 2SWAP, usw.

Wie bereits erwähnt, erübrigen sich aus der größeren Mächtigkeit viele Wörter. Die Stackmanipulationsbefehle PICK und ROLL fallen ganz heraus, da sie mit dem Parameterkonzept von Forth++ ihre Hauptbedeutung verloren haben, nämlich sich Daten, die tiefer als drei Einträge auf dem Stack liegen, auf die Spitze zu laden. Ein anderer Grund für ihren Wegfall ist, daß der Typechecker nicht mehr deterministisch arbeiten könnte, da die Zugriffstiefe erst zur Laufzeit feststeht.

Mit Schrecken werden eingefleischte Forthprogrammierer vernehmen, daß CREATE DOES> herausgefallen ist. Durch die Möglichkeit, über Klassendefinitionen Objekte oder Datenstrukturen anzulegen und gleichzeitig die Methoden definieren zu können, ist das Haupteinsatzgebiet von CREATE DOES> durch andere sprachliche Konstrukte ersetzt. Das Konzept von Forth++, das Typverhal-

ten auf dem Stack von Wortneudefinitionen zuzusichern, hätte mit diesen Befehlen schwer realisiert werden können, da ihre Wirkung allgemein kaum nachprüfbar ist. Sollen sehr maschinennahe Sachen programmiert werden, wie z. B. der Zugriff auf Portadressen, so muß dies über klare Schnittstellen zur Maschine passieren, die man mit einem geeigneten Basiswortschatz zur Verfügung stellen kann. Aus Gründen der Sicherheit ist auch der Zugriff auf den Returnstack nicht mehr möglich.

## 3. Implementation von Forth++

Die am Lehrgebiet Prozeßdatenverarbeitung erfolgte Implementation von Forth++ ist noch als prototypmäßig anzusehen. Um Forth++ optimal implementieren zu können, müßte ein komplett neues System geschrieben werden, da die Funktionalität von innerem und äußerem Interpreter wesentlich umfangreicher sein muß, als bei einem herkömmlich Forthsystem. Der Schwerpunkt der bisherigen Arbeiten lag im Sprachdesign und die Umsetzung erfolgte durch einen Precompiler, der ein Forth++-Programm in Forth83 transformiert.

Als Entwicklungs- und Testumgebung diente ein UNIX-System mit C-Forth83, der Precompiler wurde in C implementiert. Aus Gründen des Rapid-Prototyping, sind Funktionen, die eigentlich dem Forth++-System zu überlassen sind, auf Compilerebene realisiert worden, wie z. B. das Mitführen einer Symboltabelle, die praktisch ein Abbild des Wörterbuchsystems darstellt. Im Forth++-System würden alle relevanten Informationen direkt im Wörterbuch gespeichert werden.

Im folgenden wird noch eine Auswahl der Transformationen vorgestellt, um zu zeigen, wie gut sich die Mechanismen mit den Mitteln des Forth umsetzen lassen. Weil der Compiler nicht wissen kann, wie der Systemzustand des Forthsystems sein kann, sind natürlich gewisse Einschränkungen hinzunehmen. Der Stackzustand wird im Compiler als Typliste parallel mitgeführt, basiert aber aufgrund des Nichtdeterminismus von Schleifen nur auf Annahmen. In einem implementierten Forth++-System würde jeder Stackeintrag die Typinformation mit enthalten und somit ein Nichtdeterminismus auf dieser Ebene nicht existieren.

Typdefinitionen bewirken einen Eintrag in die Symboltabelle. Hier werden Informationen wie Anzahl der Bytes, Bezugstypen, Typkonstruktor (Feld, Subtype,...) festgehalten. Wird eine Variable definiert, so wird mit CREATE die Anzahl der Bytes im Wörterbuch reserviert.

Bei jeder Definition einer Klasse wird ein eigenes Wörterbuch angelegt, in dem alle Features als Wortdefinitionen abgelegt sind. Für jede Objektvariable existiert ein Wort das lediglich die Offsetberechnung innerhalb des Objektes durchführt. Objekte sind Variablen, die neben dem Raum für die Objektvariablen einen Verweis auf das Klassenwörterbuch haben. Durch diese Technik haben nur sie Zugriff auf die Features ihrer Klasse.

Die Zusicherung eines Wortes wird vom Typechecker geprüft. Dabei wird getestet, ob ausgehend von der Vorbedingung durch sukzessives Abarbeiten, der im Wortrumpf verwendeten Wörter, die Nachbedingung herauskommt. Wenn ja, hat das Wort das Typechecking fehlerfrei überstanden und wird im Wörterbuch aufgenommen. Die Information der Zusicherung wird ebenfalls gespeichert. Werden in der Zusicherung Parameter verwendet, so muß ein Mechanismus die Datenzugriffe berechnen.

Dazu gibt es im System einen sogenannten Parameterstack, auf den die Parameter der Vorbedingung vom Stack auf den Parameterstack transferiert werden. Nach Abarbeitung des Wortes wird die Nachbedingung (sofern auch mit Parametern realisiert) vom Parameterstack auf den Arbeitsstack kopiert. Während der Ausführung des Wortrumpfes finden Zugriffe auf die Parameter über Offsetberechnung auf den Parameterstack statt. Durch Verzeigerung der Parameterbereiche eines Wortes können sich mehrere Worte gleichzeitig in der Ausführung befinden. Diese Funktionsweise ist analog der bei der Umsetzung von Unterprogrammaufrufen von Pascalprogrammen durch den Compiler.

## 4. Ausblick

Das hier veröffentlichte Forth++ ist nur ansatzweise als ausgereifte, praxistaugliche Sprache zu sehen. Es ist vielleicht eher ein bedeutsamer Schritt in die Entwicklung eines neuen Forthstandards. Prinzipiell werden gegenüber Forth83 stark erweiterte Sprachmittel geboten, aber auch viele Restriktionen. Eine weiterführende Aufgabe bestünde nun in einer Verfeinerung des Sprachdesigns in der Hinsicht, daß diese Einschränkungen nicht als Nachteil empfunden werden.

Die Implementation, die z. Zt. nur in Form eines Precompilers existiert, müßte durch ein Forth++-Laufzeitsystem erfolgen, in dem alle benötigten Informationen wie Zusicherungen, Typen, Stackelemente mit Typen usw. gespeichert sind und die für die nötigen Mechanismen bei der Umsetzung der Funktionalität sorgen. Als Beispiel ist die Behandlung von qualifizierten Variablen zu nennen, die ein Parsen des Variablenbezeichners mit seinen Spezifikatoren nötig macht.

❑

**Literatur:**
[Pleß92]
Pleßmann, K. W. (Hrsg),
Abschlußbericht 1991,
Sonderforschungsbereich 106,
RWTH Aachen
[Pout87]
Poutain, Dick, Objekt-Oriented Forth,
Academic Press, 1987
[Schö91]
Schönlau, Rolf, Eine Erweiterung der Programmiersprache Forth um Typen und Objekt-orientierte Strukturen,
Diplomarbeit am Lehrgebiet Prozeßdatenverarbeitung,
RWTH Aachen, 1991
[Zech87]
Zech, Ronald,
Forth 83: Eine gründliche Einführung in die Forth-Version,
Franzis'-Verlag, München 1987

# Hard-DisCo

## Die Hardware zum 8x8-LED-Feld

von Klaus-Peter Schleisiek
An den Finkenweiden 37, W-5100 Aachen, Tel.: 0241-873462

Die Vorstellung der Hardware in der letzten VD-Ausgabe (4-92) hätte den Rahmen gesprengt. Bei aller Begeisterung darf die Redaktion ja nicht die Pflicht zur Vielseitigkeit vergessen. Aber nun geht es zur Sache, zumal die Bausätze inzwischen verfügbar sind.

Mit DisCo haben wir uns das Lehren, Lernen und Spielen zum Ziel gesetzt: dementsprechend unkompliziertr und vielseitig muß die Hardware sein. Bei der Entwicklung gab es einiges zu bedenken:

### Wie schließe ich 64 Leucht-Dioden (LED) an?

Forther sind Menschen, glaube ich, die lieber zweimal selbst denken als dankbar die vom hohen Roß gefallene "perfekte" Lösung aufzusammeln. Darum will ich zuerst einmal, auch aus Rücksicht auf die Anfänger in Sachen Hardware, die nächstliegenden Möglichkeiten nennen und gegenüberstellen.

Es gilt, 64 Licht-Emittierende-Dioden (LEDs) mit ihren zusammen 128 Anschluß-Drähten auf einer nur etwa 5 cm x 5 cm kleinen Fläche anzubringen und einen Anschluß an eine Steuer-Elektronik herzustellen. Je universeller dieser Anschluß gelingt, desto besser ist er an die Bedürfnisse der recht unterschiedlich ausgerüsteten Leserschaft anzupassen.

Zwei prinzipiell verschiedene Möglichkeiten bieten sich an; jede hat Vor- und Nachteile:

#### Einzel-Anschluß

**– hoher Bauteil-Aufwand:**
64 Port-Leitungen sind schwerlich zu haben; sie müssen nachgebildet werden, z.B. aus acht 8-Bit-Schieberegistern mit nachgeschalteten acht 8-Bit-Verstärkern. Jede LED braucht einen Widerstand zur Strombegrenzung. Diese Schaltung verursacht kurzes Flimmern beim Bild-Aufbau.

**– hoher Verdrahtungs-Aufwand:**
64 LEDs brauchen 64 geschaltete Leitungen und eine gemeinsame Rückleitung. Mit einer 2-Lagen-Platine ist das nicht ohne große Abstände zwischen den LEDs zu machen. Hand-Verdrahtung ist daher angesagt. Der hohe Bauteil-Aufwand erfordert zusätzlich Platinen-Fläche.

**+ einfache Ansteuerung:**
Mit einer Takt- und minimal einer Daten-Leitung läßt sich das ganze Bild ausgeben. Mit 8 parallelen Daten-Leitungen, wie ein Drucker-Port sie hat, geht der Bildaufbau sehr schnell.

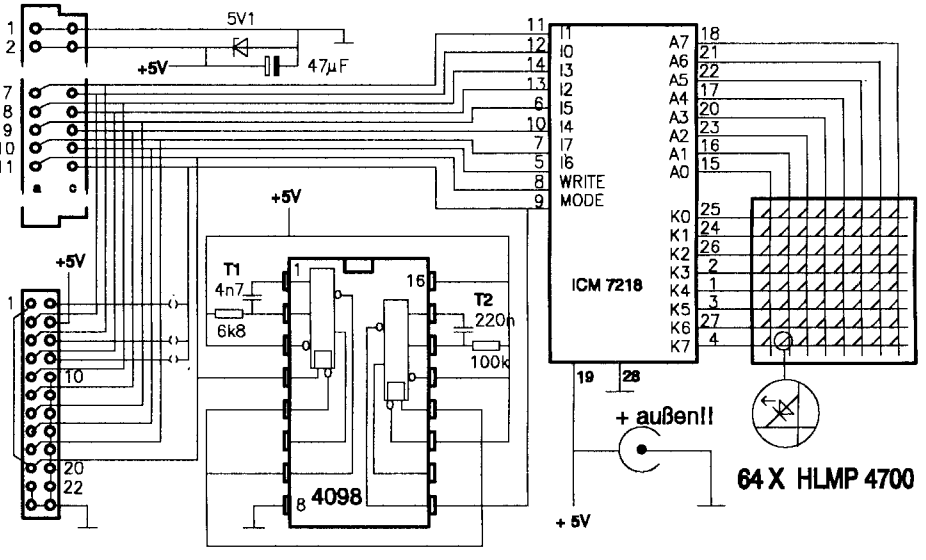
**+ statisches Bild:**
Nach dem Aufbau bleibt das Bild ohne weiteres bis zur nächsten Änderung bestehen. Der ausgebende Rechner hat Zeit für anderes.

#### Sammel-Anschluß in Matrix-Anordnung,

bei der je 8 LEDs an eine gemeinsame Leitung geschaltet werden, z.B. die Anoden (+Pol) an 8 Spalten-Leitungen und die Kathoden (-Pol) an 8 Zeilen-Leitungen. Wegen der Verknüpfung der LEDs untereinander muß diese Anordnung im sogenannten Multiplex-Betrieb erfolgen, bei dem in schneller Folge die Spalten einzeln eingeschaltet werden, jeweils zugleich mit der zugehörigen Zeilen-Kombination.

**– viel Rechenleistung**
ist nötig, um ein einigermaßen flimmerfreies Bild zu zeigen: mindestens 100 mal pro Sekunde muß die gesamte Ausgabe wiederholt werden. Bei jedem Wechsel müssen erst alle Spalten ausgeschaltet werden, bevor die nächste Zeilen-Kombination eingeschaltet wird, um "Geisterbilder" zu vermeiden. Professionelle Schaltungen liefern sogar 250 Bilder/Sekunde mit "wilder" Multiplex-Reihenfolge, um ein ruhiges und helles Bild zu erzeugen. Und jede Unterbrechung der Ausgabe-Routine läßt das Bild flakkern oder ausgehen.

**+ geringer Verdrahtungs-Aufwand:**
Leicht läßt sich die Matrix der Spalten- und Zeilen-Leitungen auf der




**Stichworte**
Disco
Lampenfeld
Hardware

Platine unterbringen.

**+ geringer Bauteil-Aufwand:**

Je ein 8-facher Anoden- und Kathoden-Treiber-Baustein reichen aus. Sogar auf Vorwiderstände kann unter Umständen verzichtet werden.

Im Hinblick auf vielfältige Anschluß-Möglichkeit und für ungeübte Löter zumutbar einfachen Aufbau wäre ein guter Kompromiß, einen integrierten Baustein (IC) mit Bildspeicher und Multiplexer gleich mit auf die Platine zu setzen, die die LEDs trägt. So etwas hat sich die Industrie auch gedacht, und deshalb gibt es solche ICs, die damit fast alle oben genannten Vorteile in sich vereinigen. (ICM-7218) Zwar mag sich mancher Controller zunächst noch langweilen, wenn er nur hin und wieder ein neues Bild rüberschieben darf, aber wir werden ihn mit Pixelarithmetik schon noch zum Schwitzen bringen.

## Wie steuere ich den Baustein an?

Die gewählte Variante des IC-7218 (AIJI) kann nicht gezielte Änderungen des Bildes vornehmen, sondern nur komplette neue Bilder übernehmen. Ob das Bild aus 8 Zeilen- oder 8 Spalten-Bytes besteht, hängt u.a. davon ab, wie man das LED-Feld aufstellen will. Aus verschiedenen Gründen haben wir uns für Spalten-Bytes entschieden; Spalte Nr.0 links, das höchstwertige Bit oben.

Eigentlich ist das IC nicht für 8*8 LED-Felder entworfen worden, sondern für achtstellige 7-Segment-Anzeigen, die ihren Namen daher haben, daß der zur Ziffer gehörige Dezimalpunkt nicht als Segment mitgezählt wird. Und weil diese Anzeigen gewöhnlich Ziffern darstellen sollen, wird das Segment-Muster angezeigt, das dem Binär- oder BCD-Wert der angelieferten Bytes entspricht. Für unseren Zweck ist das IC aber dennoch brauchbar, weil die Entwickler wohl ein Herz für Exoten hatten, die unbedingt ihre eigenen Muster zum Vorschein bringen wollen. Das richtige Code-Byte vor den eigentlichen Daten-Bytes, zusammen mit einem Signal an der "Modus"-Leitung, öffnet des ICs Ohr für ein Bild aus beliebigen Segment-Mustern. Jedoch ungestraft wird hier nicht gehext: Das höchstwertige Bit, sonst nicht bewertet, sondern als Dezimal-Punkt interpretiert, wird invertiert angezeigt! Ein Daten-Paket besteht also aus einem Steuer-Byte mit Modus-Signal und anschließenden acht Spalten-Bytes, beginnend mit Spalte Nr. 0.

Die Schaltung mit dem ICM-7218 ist recht einfach:

Die 8 Anoden-Anschlüsse sind mit den Spalten-Leitungen verbunden, die 8 Kathoden-Anschlüsse mit den Zeilen-Leitungen. Soweit der Output.

Der Input erfolgt über 10 Leitungen mit TTL-Pegeln. Neben einem 8-Bit-Datenport gibt es die Schreib- (strobe) und die Modus-Leitung. Übertragen werden ganze Bilder (8 Bytes) nach folgendem Protokoll:

- Bei anliegendem Datenbyte geht Strobe kurz auf 0 Volt (low) und dann wieder auf 5 Volt (high).

### Ist DisCo zu umständlich?

Wer in der letzten VD (4.92) meinen Beitrag "DisCo mit System" und das Glossar gelesen hat, wird sich vielleicht fragen: "Mein Gott, muß das so umständlich sein?" Zugegeben: es geht auch viel einfacher, die LEDs zum Leuchten zu bringen; schließlich hat *Rolf Kretzschmar* auch viel simpler angefangen.

Aber wir sind nicht dabei geblieben, weil wir genug davon haben, für jedes Forth von vorn zu beginnen, und jedes Programm in mehreren Versionen zu entwickeln. Wir wollen auch nicht warten, bis überall ANS-Forth verfügbar ist, sondern Kompatibilität JETZT. Dazu muß man schon etwas ausholen und sich manchen lieb gewordenen Trick verkneifen.

Und wo wir schon mal bei der Disziplin waren, haben wir auch gleich die Mengenlehre auf das Lampenfeld strenger angewendet. Dabei fiel uns auf, daß wir uns an die festen Schranken von 8 mal 8 schon so gewöhnt hatten, daß wir darüber hinaus kaum noch denken konnten; also weg mit den sturen Grenzen! Wenn das System etwas taugt, muß es auch z.B. mit 64 mal 16 zurechtkommen. Schließlich wollten wir doch mit DisCo das Denken anderer veredeln!

Um den Unterschied Vorher/Nacher zu demonstrieren, habe ich im File APPLY\CHEQUER.8X8 zwei einfache kleine Worte definiert, die ein Schachbrett-Muster erzeugen.

BANNER arbeitet brav auf jedem Forth, für das es schon DisCo gibt, auch mit großen Feldern und verschieden langen Rechteck-Seiten. Indem es mit Spalten- und Zeilen-Objekten arbeitet, hält es sich aus allen vermeidbaren Abhängigkeiten gebührend heraus. Es enthält keine Tricks und ist leicht zu lesen.

SCHACH dagegen ist auf einem 32-Bit-System wie Forthmacs, schon eim ersten Zug matt, weil es mit seinem ! zum Schluß über den Rand des reservierten Speichers schreibt. Im Speicher eines 680x0 stehen außerdem keine wechselnden Bitmuster, sondern nur die höherwertigen zwei Null-Bytes von jedem !. Auf größeren Feldern als 8 mal 8 gibt es ebenfalls Murx, keinesfalls ein vollständiges Schachbrett-Muster. Daß HEX AA55 unter bestimmten Umständen überhaupt geeignet sein kann, ist leider nur für die erfahrenen Bithacker offensichtlich, für die DisCo nicht so sehr gedacht ist. Zu loben ist aber, daß immerhin der Treiber über das defer-Wort #PRINT benutzt worden ist, anstatt obskure Bytes über irgendeine Schnittstelle zu blasen. Nun entscheiden Sie selbst: Ist DisCo zu umständlich?

PS Übrigens ist der eingangs genannte Artikel und vor allem das Glossar nicht mehr ganz aktuell. Die erste veröffentlichte Software auf Diskette und der FG-Mailbox enthält die gültige Version.

```
\ Vergleich: Hi-Level BANNER gegen Hacker's SCHACH (KPS)
\ CHEQUER.8X8
ONLY FORTH ALSO DISCO ALSO DEFINITIONS ZEITWEISE
: --chequer.8x8-- ;
BASE @ HEX
: SCHACH ( -- )           \ Nur für 8*8 geeignet, und
     #SPACE 8 BOUNDS      \ nur für 16-Bit-Forth
  DO AA55 I ! 2 +LOOP     \ Ob's links oben an ist oder aus,
  #PRINT                  \ hängt vom Speicherkonzept
 ;                        \ des Prozessors ab;
BASE !                    \ Aber schnell ist es.

: BANNER ( -- )           \ Unabhängig von der Bild-Größe,
     TECHNISCH NUR        \  nicht nur quadratisch
     #COLS 0              \ Eindeutig: links oben ist's aus
  DO |WEISE I UM 2 +LOOP \
     #ROWS 0              \ Leicht zu beschleunigen mit:
  DO _WEISE I UM 2 +LOOP  \ -SHOW BANNER SHOW +LOOP
 ; ALTWEISE
```

- Vor jedem 8-Byte-Bild wird ein Steuer-Byte übertragen mit Modus-Leitung high; bei den Bild-Bytes ist die Modus-Leitung low.

Aufgabe der Treiber-Software ist, das in Zeilen organisierte Bild aus dem Computer-Speicher (siehe VD 4/92, S.10) in ein spaltenweises zusetzen und mit vorangestelltem Steuer-Byte auf einem Port spaltenweise auszugeben. Wenn das Strobe-Signal nicht automatisch erzeugt wird, - die Standard-Printroutine des Betriebssystems für den parallelen Drucker-Port z.B. tut es - dann muß die Treiber-Software dieses Signal auf einer weiteren Ausgangs-Leitung erzeugen. Ähnliches gilt für die Modus-Leitung: Ein Controller kann das Signal an einer Ausgangs-Leitung erzeugen, aber der Drucker-Port bietet diese Möglichkeit nicht unbedingt, zumindest nicht über die Standard-Printroutine des Betriebs-Systems. Der Atari-ST hat nicht einmal die nötige Ausgangs-Leitung übrig.

Das LED-Feld an jedem Drucker-Port betreiben zu können ist aber sehr wünschenswert, denn nicht jeder kann und will sich einen Mikrokontroller zulegen, der noch dazu seinen eigenen Forth-Dialekt "spricht". Deshalb ist auf dem LED-Feld noch eine kleine Zeit-Schaltung vorgesehen, die mit einem Trick das fehlende Modus-Signal erzeugt. Ohne auf die Einzelheiten der monostabilen Kippstufen (Monoflops) einzugehen, fasse ich die Wirkungsweise gleich zusammen: Am Ende jedes Strobe-Signals stößt Stufe 1 die Stufe 2 an. Stufe 2 betreibt die Modus-Leitung, die beim ersten Stobe noch high war; beim Kippen fällt die Modus-Leitung auf low. Nach Ablauf der eingestellten Zeit kippt Stufe 2 zurück, wenn sie nicht vorher erneut angestoßen wurde. Wenn die Bytes in jedem Neuner-Paket schnell genug aufeinander folgen, dann ist beim 1. Byte die Modus-Leitung high und bei den folgenden Bytes low. Vor dem nächsten Neuner-Paket muß nur lange genug gewartet werden, damit Stufe 2 wieder in die Ruhelage zurück kippen kann. Diese Pause ist aber praktisch kein Problem, weil wir ja jedes Bild wenigstens ganz kurz sehen wollen, bevor es von einem anderen ersetzt wird.

Die Input-Leitungen sind der Vielseitigkeit wegen nach zwei Seiten herausgeführt: ein DIN-Kartenstecker schafft Anschluß an Mikro-Kontroller; die 24-polige Pfostenleiste kann direkten Anschluß zum Drucker-Port eines IBM oder Atari aufnehmen. Die Anordnung ist so gewählt, daß man die Stecker an eine Flachband-Leitung quetschen kann; das schließt Verwechselungen beinahe aus (auf Leitung 1 achten!) und erspart Löt-Arbeit. Drei verschiedene mögliche Ausgangsleitungen des Drucker-Ports können über Löt-Brücken an die Modus-Leitung angeschlossen werden. Die Versorgungs-Spannung kann über die genannten Stecker oder über die zusätzliche Power-Buchse zugeführt werden, denn kein Drucker-Port liefert die benötigten ca. 200 mA bei 5 Volt. Dafür sind Stecker-Netzteile geeignet.

Die verwendeten integrierten Schaltungen (IC) sind in CMOS-Technik hergestellt; darum vertragen sie keine hohen Spannungen durch elektrostatische Aufladung und auch keine Signal-Spannung an den Eingängen bei fehlender Betriebs-Spannung. Also z.B. nicht das Stecker-Netzteil rausziehen, während der Drucker-Port noch am laufenden Rechner hängt.

## Wer liefert was?

Die wichtigsten Bauteile sind nicht in jedem Elektronik-Laden vorrätig, und freifliegende Verdrahtung ist auch nicht jedermanns Sache. Daher werden die DisCo-LED-Felder als Teil des KlaRo-Systems fertig aufgebaut oder als Bausätze mit ausführlicher Bauanleitung angeboten. Eine entsprechende Anzeige mit Preisliste ist in diesem Heft.

Die Software und weitere Details sind zu finden auf der DisCo System-Diskette ( selbstadressierter Freiumschlag mit formatierter Diskette ) vom Autor ( bis 720 kB ) oder von *Rolf Kretzschmar* oder in der FG-Mailbox.

## Geht's nicht auch anders?

Natürlich kann man alles auch ganz anders machen; wir haben selbst noch Alternativ-Modelle, die auch gut funktionieren, und viele interessante Spezial-Lösungen in Hard- und Software sind mit kleinerem Umfang machbar. Zum Beispiel hat *Rafael Deliano* am 27.1.93 (in der FG-Mailbox) angekündigt, daß er für seine Verleih-Controller das direkt vom Prozessor betriebene Multiplexen benutzen wird. Und natürlich wird er nicht so eine "verkorkste" Software benutzen, wie er sich ausdrückt. Wir freuen uns, wenn wir auf dem Jahrestreffen in Nürnberg etwas konkretes zum Vergleichen sehen werden. DisCo wird dort natürlich vorgestellt!

❑

# Ein FILE DUMPen

von Wolf-Helge Neumann

Huttenstr. 27, 7000 Stuttgart 31, Tel: 0711-8872638

Es wird ein Werkzeug für bigFORTH beschrieben, mit dem ein Hexdump einer Datei in eine Datei auf der RAM-Disk geschrieben wird.

Manchmal braucht man von einer Datei einfach einen Hex-Dump. Das letzte Mal war das der Fall, als ich ein Konvertierungsprogramm für Bild-Dateien schrieb und meine Konvertierungsroutine offensichtlich ganz andere Vorstellungen vom Bild hatte.
Am einfachsten wäre natürlich gewesen, den Drucker in den Hex-Dump-Modus zu bringen und das Bild auszudrucken, aber ich wollte den Hex-Dump in einer Datei haben, um ihn gliedern und noch mit ein paar erklärenden Worten versehen zu können. Mit einem Ausdruck dieser Datei bewaffnet, wäre das Debuggen der Konvertierungsroutine wesentlich einfacher.
Deshalb habe ich mir mit bigFORTH ein kleines Werkzeug gebastelt, das den Hexdump nach meinen Vorstellungen in eine Datei auf der Ramdisk schreibt. Nach Aufruf von FILE-DUMP erscheint die gewohnte Fileselect-Box und man wählt die gewünschte Datei aus. Die Dump-Datei wird auf Laufwerk P erzeugt und hat den selben Dateinamen, nur mit der Endung "DMP". Drei Optionen können angegeben werden:

- `-Ascii` unterdrückt die Ausgabe der Spalte mit den Ascii-Zeichen.
- `-Control` wandelt alle Steuerzeichen in Punkte, so daß man die Datei auch drucken kann.
- `-Offset` unterdrückt die Ausgabe der Spalte mit den Offsets.
- `+Ascii, +Control, +Offset` bewirken das Gegenteil.

Die Optionen haben Schalterfunktion, werden also gespeichert und müssen vor Aufruf von FileDump nicht genannt werden.
Die Ergebnisse des Filedumps mit den jeweiligen Optionen sehen wie in Abbildung 1 aus.

❑

**Stichworte**

Dump
File-Handling
bigFORTH

## Beispiele zu: Ein FILE DUMPen

```
+ascii -control +offset filedump
abcdefghijklmnop  00000000 61 62 63 64 65 66 67 68 69 6A 6B 6C 6D 6E 6F 70
qrstuvwxyz..ABCD  00000010 71 72 73 74 75 76 77 78 79 7A 0D 0A 41 42 43 44
EFGHIJKLMNOPQRST  00000020 45 46 47 48 49 4A 4B 4C 4D 4E 4F 50 51 52 53 54
UVWXYZ..          00000028 55 56 57 58 59 5A 0D 0A

+ascii -offset filedump
abcdefghijklmnop           61 62 63 64 65 66 67 68 69 6A 6B 6C 6D 6E 6F 70
qrstuvwxyz..ABCD           71 72 73 74 75 76 77 78 79 7A 0D 0A 41 42 43 44
EFGHIJKLMNOPQRST           45 46 47 48 49 4A 4B 4C 4D 4E 4F 50 51 52 53 54
UVWXYZ..                   55 56 57 58 59 5A 0D 0A

-ascii +offset filedump
00000000 61 62 63 64 65 66 67 68 69 6A 6B 6C 6D 6E 6F 70
00000010 71 72 73 74 75 76 77 78 79 7A 0D 0A 41 42 43 44
00000020 45 46 47 48 49 4A 4B 4C 4D 4E 4F 50 51 52 53 54
00000028 55 56 57 58 59 5A 0D 0A

-offset filedump
61 62 63 64 65 66 67 68 69 6A 6B 6C 6D 6E 6F 70
71 72 73 74 75 76 77 78 79 7A 0D 0A 41 42 43 44
45 46 47 48 49 4A 4B 4C 4D 4E 4F 50 51 52 53 54
55 56 57 58 59 5A 0D 0A
```

Abb. 1

## Listing zu: Ein File DUMPen (Wolf-Helge Neumann)

### Screen #0

```
\ Info                                                    14apr92wn

    FileDump schreibt ein Filedump der Datei NAME.EXT
    in die Datei P:\NAME.DMP .
    Ein anderer Pfad für die Dump-Datei kann durch Anpassen
    von WriteInit (Screen 4) eingestellt werden.

    Aufbau der Dump-Datei:  ASCII  Offset  Hex-Dump

    Optionen:
    -Ascii, +Ascii: Aus-/Einschalten der ASCII-Ausgabe
    -Control, +Control: Bei +Ascii Aus-/Einschalten der
                        Steuerzeichen-Ausgabe
    -Offset, +Offset: Aus-/Einschalten der Offset-Ausgabe
```

### Screen #1

```
\ Loadscreen                                              14apr92wn

\needs fileselect       include fileselect.32b

gem dos also

Vocabulary FileDump     FileDump also definitions

1 ?head !               \ headerless
2 8 thru

?head off
9 load
```

Fortsetzung auf Seite 39 ➪

# Forth spinnt

## Fädeln von Wolle und Code

von Wolfgang Allinger
Brander Weg 6, 5650 Solingen 11, Tel./Fax 0212-66811

Beschreibungen kommerzieller Forth-Projekte, wie er hier vorgestellt wird, werden von vielen VD-Beziehern gerne gelesen. Wolfgang Allinger setzt Forth häufig für seinen Broterwerb ein und er zählt zu denen, die das Forthlicht auch auf den Tisch stellen wollen. Er teilt unsere Meinung, daß es unter dem Scheffel nichts zu suchen hat.

Anfang 1989 wollte ein Kunde von mir die Hard- und Software für einen "embedded controller" für die Steuerung einer Spinnmaschine entwickelt haben. Die Herstellung von Fäden ist eine interessante Aufgabe, und da ich ein Forth-Fan bin, war gefädelter Code natürlich die beste und selbstverständliche Wahl für die Herstellung von Fäden auf einer Direkt-Spinn-Maschine.

Der Controller sollte 3 Frequenzmess- und 1 Preset-Counter-Ein- gang jeweils für Frequenzen zwischen 0,1Hz und 1kHz haben. Jeder Frequenz-Kanal sollte zwei programmierbare Schwellen haben, deren Ansprechen auf jeweils einem Ausgangsbit angezeigt wird, so daß sie als 3-Punkt-Regler verwendet werden können. Der Preset-Counter sollte als programmierbarer Vorwärts/Rückwärts-Zähler mit frei programmierbarem Reload-Wert und zwei Schaltausgängen ausgeführt sein. Ein Ausgang als Vorwarnsignal, der andere als Fertigsignal.

Weitere Ein- und Ausgänge wurden für die Steuerung der Handshake Signale mit der Spinnmaschine benötigt.

Der Controller sollte außerdem als Datenlogger für die Produktionsmengen mit Los- und Schichtzählern, Qualitätsüberwachung, Betriebs- und Stillstandszeiten und Statistik, Störungsanzahl und -zeiten, Maschinen-, Schicht- und Bedienereffektivität usw. dienen.

8 Eingänge waren für die Fehleranalyse vorgesehen. Jeder Eingang stellte eine bestimmtem Fehlerart dar, z.B. zu niedriger Hydraulik-Druck, fehlender Luftdruck, Fadenbruch ( sowas wie zerstörte LFA's ?!), Materialmangel und sonstige Sch...spiele. Diese Eingänge müssen 2 mal pro Sekunde bearbeitet werden.

Alle Daten sollen durch den Bediener frei scalierbar sein, da die Frequenzeingänge von Magnetaufnehmern von Zahnrädern abgegriffen werden. Jede Maschine hat andere Zahnräder, Geschwindigkeiten, Durchmesser der Räder usw., so daß niemand vorhersagen kann, wieviel Pulse pro Meter oder anderen perversen Einheiten wie Fuß, Zoll, Yards oder Hanks kommen.

Zu allem Überfluß kann es noch passieren, daß der Bediener die Liefergeschwindigkeit in m/min sehen will, an einer Maschine mit 59 Pulsen pro Umdrehung an einem Rad mit 17,3 Inches Durchmesser ( */ ist das beste Wort für so einen Mist! ). Der Faden wird für den Kunden X auf Spulen aufgewickelt, die 517 Hanks (1 Hank ist in England 837m, in den USA 857m oder umgekehrt oder auch nicht und werweisnichtwieviel in Hinterkickiwauwau!!). Der will 36 Kisten mit je 24 Spulen roten Faden haben. Der nächste Kunde Y will 365 Spulen mit jeweils 1,5 km, zu jeweils 72 Spulen pro Karton und das alles in grün. Der Bediener wird für gefertigte Furlongs entlohnt und das Managment will die Produktion in mils/Woche ( Grrr ).

Nobody knows the trouble I have seen, aber ich hab viel über abartige Maßeinheiten gelernt. Die dazugehörige Mathematik werde ich in einem andern Artikel über lange Mathematik erläutern. ( Zur Abschreckung nur ein Hinweis: Floatingpoint mit 24 bit Mantisse macht Rundungsfehler, die sich über Tage und Wochen unerhört summieren, verbraucht viel vom internen Memory. Ich habe zur Skalierung Routinen benötigt, die */ für 32 bit Zahlen mit 64!!! bit Zwischenergebnis brauchten!)

Alle Daten mußten auch bei Netzausfall erhalten bleiben. Da der Kunde keinerlei Batterie in seinen Geräten haben wollte, mußten die Daten in einem EEPROM gespeichert werden. Mit 1 oder 2 neuen Datensätzen pro Sekunde kann das EEPROM zwar beschrieben werden, erreicht aber bei einer durchschnittlichen Lebensdauer von einigen 10 Tausend Schreibzyklen nach wenigen Stunden das Ende seines Lebens ( That's the end my friend! ).

**Stichworte**

Produktbeschreibung
embedded Controller
80C535 (8051)

*Zum Autor:*
*Dipl.-Ing. Wolfgang Allinger, Jg. 46, studierte Nachrichtentechnik und entwickelt seit 1970 Hard- und Software ua. für Anwendungen im Straßenverkehr und im Industriebereich, speziell: Walzwerke und Kernkraftwerke. Viele Jahre Abteilungsleiter für System- und Softwareentwicklung; seit 1987 selbständig (Ingenieurbüro). Erfolgreiche Entwicklung großer Echtzeitanwendung mit 60 µC's im gleichzeitigen Einsatz und kleiner µC-gesteuerter Meßgeräte.*
*Hobbys: Ehefrau, Siamkater Dango, RC-Modellsegelflug, Beruf, Sportwagen gut fahren, Computer, Forth Metacompiler (Reihenfolge rein zufällig!)*

Es wurde also noch eine Powerdown Logik, eine Interruptservice-Routine und ein großer Kondensator als Energiespeicher benötigt, um genug Zeit zu haben, daß etwa 1kByte Daten beim Netzausfall weggeschrieben werden können. Beim Wiederkehr der Spannung müssen die geretten Daten auf Korrektheit geprüft und der Prozessor damit wieder in den Zustand bei Netzausfall gebracht werden, damit das Programm hoffentlich wieder richtig weiter läuft, wenn der Bediener den Netzausfall quittiert hat.

An der Frontplatte sollte eine Folientastatur (Bild 1) *( Bildunterschrift: Die Frontplatte ist eine Folientastatur)* mit 18 Tasten für Dateneingaben und Programmierung, 6 LEDs für die Anzeige der aktuellen Schicht und 16 7 Segment LED Anzeigen in vier Gruppen sein. Sie sollten die Meßwerte, den Controllerstatus, die Fehlerzustände und den Dialog mit dem Bediener anzeigen.

Der Controller sollte keinerlei anwendungspezifische Programmierung haben, sondern sollte erst vor Ort vom Anwender frei über die Tastatur oder die serielle Schnittstelle programmiert werden.

Die 2 großen Anzeigen sollten gleichzeitig noch Fenster in zwei Felder mit bis zu 10 Meßwerten oder Variablen sein. Die Reihenfolge, Anzahl und Inhalt sollten durch den Bediener frei festlegbar sein. Vielleicht wollte er an 1. Stelle die Liefergeschwindigkeit, an der 2. die Betriebszeit an der 3. den Schichtzähler usw. sehen.

Die serielle Schnittstelle sollte konfigurierbar sein für singlepoint RS232, RS422 und multidrop RS485 mit 31 slave Controllern an einem seriellen Bus.

Ein SIEMENS 80C535 Mikrocontroller wurde als CPU benutzt. Es ist ein erweiterter 8031 in low power C-MOS ( die power-down Problematik läßt grüßen!), mit sehr nützlichen Ergänzungen in einem PLCC-68 Gehäuse: 6 8bit Ports, 256 byte internes Memory, A/D Wandler mit 8 Multiplex Eingängen, 3 16bit Zählern mit Reload und Capture Registern, 12 IR-Quellen, 4 Prioritätsebenen...

Das interessanteste für diese Applikation sind die Capture Register. Timer 2 kann als kontinuierlicher 1Sec Zähler laufen und die vier Puls-Eingänge können das Laden der dazugehörigen Capture-Register und einen Interrupt auslösen. Wenn die IR-Service-Rou- tine den letzten benutzten Wert abspeichert, kann sie die Zeit seit der letzten Flanke, also die Periodendauer und damit die Frequenz berechnen. Timer 2 läuft alle 65mSec über. Damit wird ein weitere Interrupt ausgelöst, der eine 8bit SW-Zähler inkrementiert. Wir haben also einen 24bit Wert als Ergebnis. Die IR-Service-Routine verwirft den Wert, falls er unterhalb einer durch den Anwender programmierbaren Schwelle liegt. So können sehr einfach Störstacheln ausgeblendet werden.

Die Software habe ich auf zwei Wegen in Angriff genommen:

Alle IR-Service-Routinen ( außer Power-Fail) habe ich mit einem Standard 8051 Assembler geschrieben, alles andere ist in Forth programmiert. Ich benutzte PC-Forth und Metacompiler von LMI. Der Grund für diesen Mischung war, daß ich das Beste beider Welten wollte.

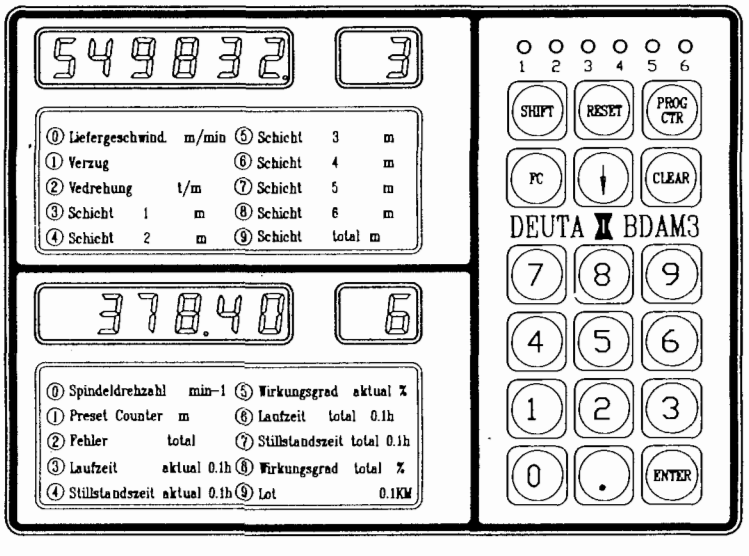
Die IR Routinen für diese Applikation sind recht komplex und falls ich zum debuggen und zur timing Messung einen Logik-Analysator benötige, ist es fast unmöglich, die echten Adressen und Inhalte des Forth Assemblers zu finden. Beim Listing eines normalen Assemblers ist das sehr einfach, und die starke Verwendung von zT. selbstmodifizierenden Makros macht die Programmierung ebenfalls leichter.

Das Systen kommt in Assembler hoch, initialisert die Harware, das IR-System und springt dann in COLD des Forth Kerns. Forth selber initialisiert den Kern, gibt dann das IR-System frei und gibt die Kontrolle an das oberste Wort MAIN der Applikation.

Um die großen Zahlen für den Bediener darzustellen, die bis zu 6 Stellen mit bis zu 3 Nachkommastellen hatten, dachte ich zuerst an den Einsatz einer käuflichen Floating-Point-Library. Die war jedoch sehr schlecht entworfen, verbrauchte praktisch das gesamte Interne Memory und war viel zu langsam für diese Applikation! So blieb mir nichts anders übrig, ich mußte die lange Festkomma-Mathematik selber machen.

Das Interne Memory wird fast vollständig durch die IR-Service-Routinen gebraucht. Die festen Adressen waren mit EQU im Forth und Assembler zugeordnet, so daß ich keinen Linker brauchte. WennForth auf mehr als ein byte im internen Memory zugreift, wird das durch primitives geregelt, die den IR schließen und regenerieren ( critical regions ).

Ein bitadressierbares Byte im internen Memory wird als Flagbyte für einen einfachen FORTH-Multitasker benutzt. Wenn der Tasker ein gesetztes bit findet, wird es gelöscht und die




Die Frontplatte ist eine Folientastatur

dazugehörige TASK0..TASK7 ausgeführt. Bit 0..3 werden durch die 20mSec IR-Routine bearbeitet. Bit0 wird alle 20mSec, bit1 alle 100mSec, bit2 alle 500mSec und bit3 alle !Sek gesetzt. Bit 4..7 können beliebig von dem gesetzt werden, der TASK4..TASK7 starten will.

Es gibt weiter noch interne Timer, die bis 0 runterlaufen mit unterschiedlichen Zeitrastern. Falls eine Task einen Timer benötigt, setzt es diesen mit der Wartezeit und wartet darauf, daß er wieder Null wird.

Das Wort MAIN sieht etwa wie folgt aus:

```
: MAIN
  ...
        \SOME INITS
  ...
  BEGIN
    TASKS \should be
          \stack neutral!
    DEPTH HOST? OR
  UNTIL
  .HOST   \display 'HHHH'
          \to operator
  7 EMIT CR \some noise
  DEPTH 0 IF .S THEN
  ...
          \SOME
          \HOUSEKEEPING
  ...
;
```

HOST? liefert t= falls das Weckzeichen im UART Receiver Buffer ist, andernfalls liefert es f=.

Falls irgendein Wort den Stack verändert, wird MAIN ebenfall verlassen. Dies ist sehr nützlich beim Debuggen.

Die Fernsteuerung ist ebenfalls sehr einfach: sende das Weckzeichen, warte auf <CR>, mache egal was in Forth und rufe etwas, was wieder MAIN aufruft.

❑

*Literatur:*
*W. Allinger,*
*Threading Code and Wool,*
*Proceedings of the 1990 Rochester Forth Conference*

## Dank...

...an meine damaligen Partner *Bernd-J. Werner*, der die Hardware entwikkelt hat, an *Tesuya Nakata*, der viel von der Long Mathmatik machte, *Peter Güttler* und meine Frau *Gaby* für deren Geduld und an meinen damals 15 Jahre (inzwischen 18 Jahre) alten Siam-Kater *DANGO*, der ständig über meine Tastatur lief oder auf dem Monitor saß, während ich zu Hause war und nachts programmierte, weil ich nicht schlafen konnte wegen der vielen Probleme bei dieser Entwicklung, und an *Chuck Moore*, ohne dessen Erfindung von FORTH dieses Projekt vermutlich nie zu einem Ende gekommen wäre.

## Frage, Antwort und dann?

rk. Die Auswertung der Fragebogen-Rückläufer (immerhin ca. 12% der VD-Bezieher) ist noch nicht beendet. Wir hoffen, auf der Forth-Tagung in Nürnberg, die wichtigsten Daten und Interpretationen vorstellen zu können. Sehen Sie sich mal die Mitgliedsnummern der fünf Gewinner (Seite 28) an! Sie werden bemerken, daß dort die niedrigen Nummern unterrepräsentiert sind. Eine Kontrolle ergab: lediglich vier "alte Hasen" haben den Fragebogen zurückgeschickt. Wie sollen wir das interpretieren?

## Rudimente eines einfachen Multitaskers

```
( ?tasks                                        ALL 13:00 12JAN90)
0 .IF

: ?TASKS
  $FTASK 0  IBIT?  IF   $FTASK 0 0IBIT   TASK0   EXIT   THEN
  $FTASK 1  IBIT?  IF   $FTASK 1 0IBIT   TASK1   EXIT   THEN
  $FTASK 2  IBIT?  IF   $FTASK 2 0IBIT   TASK2   EXIT   THEN
  $FTASK 3  IBIT?  IF   $FTASK 3 0IBIT   TASK3   EXIT   THEN
  $FTASK 4  IBIT?  IF   $FTASK 4 0IBIT   TASK4   EXIT   THEN
  $FTASK 5  IBIT?  IF   $FTASK 5 0IBIT   TASK5   EXIT   THEN
  $FTASK 6  IBIT?  IF   $FTASK 6 0IBIT   TASK6   EXIT   THEN
  $FTASK 7  IBIT?  IF   $FTASK 7 0IBIT   TASK7   EXIT   THEN
 ;

1 .IF
          : NOTASK ;   \ SO GEHT'S ??? MIT MC2.2
CODE ?TASKS
  DPTR,# ' TASK0 MOV    $FTASK.0 , 9$ JBC
  DPTR,# ' TASK1 MOV    $FTASK.1 , 9$ JBC
  DPTR,# ' TASK2 MOV    $FTASK.2 , 9$ JBC
  DPTR,# ' TASK3 MOV    $FTASK.3 , 9$ JBC
  DPTR,# ' TASK4 MOV    $FTASK.4 , 9$ JBC
  DPTR,# ' TASK5 MOV    $FTASK.5 , 9$ JBC
  DPTR,# ' TASK6 MOV    $FTASK.6 , 9$ JBC
  DPTR,# ' TASK7 MOV    $FTASK.7 , 9$ JBC
  DPTR,# ' NOTASK MOV             9$:  A CLR   @A+DPTR JMP
END-CODE    .THEN

( --- n ; 0=FALSE proceed; -1=TRUE=<ESC> ; n= RS485 adr HIT   )
: HOST?
        ?TERMINAL IF
            %GADR C@ ?DUP IF        \ RS485 MODE
                DUP KEY = IF   EXIT ( -- nADR ) \ RS485 adr HIT
                           ELSE DROP THEN
            ELSE                    \ RS232|422 MODE
                KEY 27 = ?DUP IF EXIT THEN \ ( -- t=<ESC> )
            THEN
        THEN 0  ;    ( -- f= NOTTING )


( main                                          ALL 16:37 28JAN90)

: MAIN      ASCII \ EMIT    ADDRframe
    BEGIN
        ?TASKS
        DEPTH
        HOST?  OR
    UNTIL
    DATAframe       .HOST   7 EMIT
    DEPTH 0 > IF  .S THEN
    %STENO @  IF  .N THEN   QUIT ;
```

❑

# Einladung zur ordentlichen Mitgliederversammlung

am

## Sonntag, den 25.April 1993

im

Jugend- und Economy-Hotel
Gostenhofer Hauptstraße 47-49,
8500 Nürnberg 70

**Tagungsodnungspunkte**

1. Rechenschaftsbericht des Direktoriums
2. Kassenbericht und Bilanz 1992
3. Satzungsänderung zur Klärung der Selbstlosigkeit und des Auflöseverfahrens
4. Aussprache, Entlastung, Wahl des Direktoriums
5. Neue Ziele der Gesellschaft
6. Verschiedenes

**! Tom Zimmer kommt !**

Wie wir kurz vor Drucklegung aus dem Tagungsbüro erfahren haben , wird Tom Zimmer, der Vater von ZF, F-PC in Nürnberg über den TCOM-Forth-Compiler sprechen.

**Ergänzende Tagungsordnungspunkte** sind bis Ende März über das Forth-Büro dem Direktorium einzureichen.
Die Forth-Tagung '93, zu der bereits gesondert eingeladen wurde, findet im Zeitraum vom 23. - 25.4.1993 am selben Ort statt. Sie endet nach der ordentlichen Mitgliederversammlung am Sonntag Mittag. Die Teilnahme an der Mitgliederversammlung ist kostenfrei. Bitte kommen Sie zahlreich und mit ausreichender Zeit, damit die Mitgliederversammlung beschlußfähig wird. Auf der Tagesordnung stehen wichtige Themen, die dieZukunft der Forth-Gesellschaft e.V. maßgeblich beeinflussen.

# KEYB8B

von Arndt Klingelnberg

Straßburger Str. 12, 5100 Alsdorf, Tel.: 02404-61648, Fax.: 02404-63039

Als Alternative zu der Lösung von Claus Vogt wird eine Forth83 konforme, einfache Lösung des 8bit-enhanced-ASCII-Problems für F-PC vorgestellt.

Umlaute im F-PC; Schnee von gestern? Für mich ja und für andere auch -- so dachte ich -- . Zu deutschen Umlauten in F-PC gab es immerhin schon vier Artikel/Kurznotizen in der VD, und zweimal habe ich auf den Konferenzen der Forth-Gesellschaft darüber berichtet; Tom Zimmer wurden verschiedene Lösungen zugeschickt. Aber Rolf (rk) meinte, meine Lösung müßte getrennt von der ak-Version, eben nicht zugedeckt von den vielen anderen Veränderungen wie auch der sinnvoller strukturierten Funktionstastenbelegung, vorgestellt werden. Nun ja, hier ist sie, und zwar ohne (!) den Forth-üblichen Header-Standard zu verlassen [Vogt-VD92-3].

Bei der folgenden Lösung wird Forth-üblich reiner ASCII ( ISO-USA , 7bit ) in die Symbole ( Header ) compiliert. Das Forth-übliche TRAVERSE bleibt also nutzbar. Die Lösung ist sehr einfach, dazu sogar schneller und gestattet den normalen Umgang mit beliebigen Tastaturen. Die 'enhanced codes' der MF2 Tastatur kann meine Lösung auch nicht; auch nicht die östlichen DoubleByteCharcterSets (DBCS). Aber welches Programm kann das schon.

Konsequenz: drei Funktionen verändert, dazu zwei Hilfsfunktionen. Alles funtioniert wie gewünscht inkl. von Alt-### zur Key Eingabe und die Alt-Ctrl Sonderfunktionen von k3 [Gerdes-k3].

**Stichworte**

F-PC
IBM-8bit-ASCII
key
keyboard
mouse
patch

## Blö(!)cke ODER Fliess(!)text ? Fließ(!)text !

Kurz nach Jahreswechsel 1989/1990 drückte mir Rolf ein F-PC 3.5000 in die Hand. Da für mich Sourcen in Blocks ein stricktes 'NoNo' sind und ich in TurboForth (Jedi, Frankreich) noch nicht voll eingestiegen war, erschien F-PC als DIE Lösung. Wenn ..., ja wenn ich nicht dauernd mit 'ä' den Cursor hoch gescroll-t hätte, mit 'ö' den Tabulator versetzt und -- seltener -- mit 'Ö' das Print Menü aufgerufen hätte. Mir ungeneigte Editoren bringen mich zur Weißglut, aber ich war Optimist, und einfacher, als die Blöcke im Volksforth zum fließen zu bringen, erschien es mir allemal, dem schoenen F-PC die deutsche Tastatur anzugewoehnen.

Primaer ging es mir darum, meine Tastur ohne Ueberraschungen nutzen

**Listing zu: KEYB8B (A. Klingelnberg)**

```
\ KEYB-8b.seq           8bit IBM enhanced keys for F-PC                 \ akg92jun21

 \ akg92nov15  final tests and modifications for VD 92-4

ONLY FORTH ALSO DEFINITIONS  decimal

comment: vvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvv\ akg92jun21

                        Welcome on KEYboard

    modifies the original F-PC to use NON-ASCII keyboards!

    tested successfully with F-PC 3.50 and 3.5610                       \ akg92nov15
    disables keyboard-macros

    This is a somewhat simplified solution compared to  F-PC-ak5 ...
                                                    ... F-PC-ak ver.4.0.

    see:  ANS  eKEY   E.2.4  p.185f. !!!!!  DOS-key  DOS-key?

    Wolfgang Allinger  Forth style
    Ulrich Hoffmann   'Konfiguration DEUTSCH'   VD 91mar p.12

  Problems with German Keyboards:
     ^U    paragraph-sign  (thanks to  mother-blue  for this big problem)
    ß 225   Ctrl-F4          (the combination of  'beta'  and the german  'sz')
    ä 132   scroll-UP
    ö 148   Alt-T
    ü 129   Alt-0      ( Alt-zero ! )
    Ö 153   Alt-P

        This is PublicDomain.

        Arndt Klingelnberg, Strassburgerstr. 12 , D-W 5110 Alsdorf, Germany
        ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
comment;

VOCABULARY PATCH
ONLY FORTH ALSO DEFINITIONS   decimal   EDITOR ALSO  HIDDEN ALSO  PATCH ALSO

defined \|u nip 0=      \ F-PC original version of  ||  == \if
#IF                     \ needed for v.3.50  opposite action of \u
: \|u   ( <name> -- ).
        defined nip
        IF      [COMPILE] \
        THEN ; IMMEDIATE
#THEN
```

zu koennen, weniger darum, sehr bequem Umlaute in den Text hineinzubringen. Die Lösungsansätze für die vereinfachte Eingabe von Umlauten im heutigen original F-PC [F-PC3.5606] sind daher für mich Gewinne an der falschen Front.

## Funktionstasten verdeckt

So entstand die ak-Version. Anfangs wurden nur die kritischen Codes 'verdeckt', man/fra konnte nun 'normal' tippen. Spezielle Tasten waren immer noch nicht erlaubt, noch konnte der 'k3' Tastaturtreiber seine ganze Herrlichkeit unter F-PC entfalten. Meine Lösung [ak-Version] [akvd90-3] bedingte damals, einige Funktionstasten 'abzuklemmen' und als Folge davon einige Funktionstasten anders zu belegen. Dabei wurden die Funktionen strukturiert. WordStar, Borland (SideKick oder Turbo...) und der Norton Commander, wie auch die Vielfalt wichtiger PublicDomain Programme waren die Vorbilder. Die notwendigen Änderungen waren zahlreich, und für eine Patch-Datei ungeeignet. Daher entstand eine spezielle F-PC-Version mit getrenntem Namen. Immerhin waren nun Menü- und FunktionstastenSätze umschaltbar. Die OriginalBelegung ist zusätzlich weitgehend erhalten worden. Menü- und TastenSätze sind aber nun beliebig erweiterbar, z.B: auch in Deutsch oder/und Anfänger oder ...).

Ein Anwender wollte gerne auch den '§' nutzen. Das bedingt zwingend, daß Ctrl-U eben NICHT mehr die Datei sichert. Dieses Ctrl-U Problem hat nichts mit ASCII zu tun und nur bedingt mit F-PC, wir haben es einem großen schreIBMaschinen Hersteller zu verdanken. [F-PC-ak3] [akvd90-4]

Meine Lösung in F-PC-ak v.3.50 ( ak0 bis ak3 ) -- damals hatte ich wenig praktische Erfahrung -- war umständlich, holperig. Nun, ich konnte es nicht besser, aber es funktionierte und deshalb habe ich mich getraut, es zu verteilen.

### Fortsetzungs des Listings zu: KEYB8B (A. Klingelnberg)

```
\ ================================== part 0          key-utils
\ ----  UTILS2.seq ( ak version )

\ separating enhanced (IBM) ASCII and FunctionKeys                    \ akg91mar18

: IBMASCII?     ( -- flg )    \ false if last key read byKEY was functionkey
        @> bioskeyval   split drop ;



\ ================================== part 1          FORTH LINEeditor
\ ----- Ledit.seq

PATCH DEFINITIONS
: ?char         ( -- )  \ handle normal AND ENHANCED ASCII keys, insert them
                \ now able to decide between function keys and enhanced ASCII
        IBMascii? 0=                    \ is it NOT a char              \ akg91mar18
      ?EXIT
        lchar bl <                      \ is it a  ^char
      ?EXIT                                                             \ akg92nov14
        lchar <ichar>
 \       2R>drop                         \ EXIT from CALLING word !!!   \ akg92nov14
        2R> 2drop               \ F-PC orginal does not know 2R>drop    \ akg92nov15
      ;

PATCH @> ?char          HIDDEN !> ?char                 \ see: DOkey


\ : ledit_itself  ( -- )  \ FORTH       ( Ulrich Hoffmann )
\        Lchar <ichar> ;                \ used for ^U=='                \ akg91apr06
\ may be installed in  ?control    position  21 ( $15 )



\ ================================== part 2          the EDITOR
\ ----- sEDIT2.seq

PATCH DEFINITIONS
: ?functions    ( -- )  \ handle function codes, but ONLY functions
                        \ now allowing enhanced IBMascii                \ akg91mar18
                    \ stack command corrected                           \ akg91mar18
        IBMASCII?
      ?EXIT
        keychar $80 <   \ functions codes have values >127              \ akg91mar25
      ?EXIT             \ EXIT if 8th bit NOT set                       \ akg92nov15
 \       keychar 127 -  ?.sfuntbl sfuntbl        \ original
\u  ?.sfuntbl  keychar 127 -  dup sfuntbl  ?.sfuntbl drop                \ akg92oct04
                .               \ FIRST do something THEN show it to us \ akg92oct04
\|u ?.sfuntbl  keychar 127 -  sfuntbl                                   \ akg92nov15
 \       2R>drop                         \ EXIT from CALLING word !!!   \ akg92nov14
        2R> 2drop               \ F-PC orginal does not know 2R>drop    \ akg92nov15
      ;

PATCH @> ?functions     EDITOR !> ?functions            \ see:  DOaCHAR

        .
control U   ctlset ?schr        \ '' reserved !!!                       \ akg91jan02
                \ updtX         \ save changes and continue  now  Alt-U
\ ( Alt-U )     150 fnset updtX           \ save changes and continue
                \ wUNdel is now available at Alt-F5 and Alt-T           \ akg91mar20
        \ this all would requires a lot of changes in the original version



\ ================================== part 3          the MOUSE
\ ----- mouse.seq

HIDDEN DEFINITIONS
: fake-bioskey  ( char -- char )        \ sets BIOSkeyval               \ akg91apr09
        dup 127 >       \ must be FUNCTIONkey   \ see:  DOaCHAR  DOkey
        IF      dup 128 +  255 and
                flip dup  !> bioskeyval   !> bioschar
        THEN ;                                                          ➪
```

## DOitYOURself

*Ulrich Hoffman* [UH-vd91-1] zeigte dann eine Lösung in Forth-typischer Einfachheit: In denjenigen Funktionstasten, deren F-PC-internen Scancode einem deutschen Sonderzeichen entsprachen, wurde eine 'itself' Funktion installiert, die elegant den _char als ASCII Normal-Code weiterverarbeitete:

```
: ledit_itself   ( -- )
  \ FORTH LINEeditor
  \ insert (even function)
  \ characters themselves
   lchar <ichar> ;

: sed_itself     ( -- )
  \ EDITOR
  \ insert (even function)
  \ characters themselves
   keychar schr ;
```

Schon damals gab Forth dem VenturaPublisher Probleme auf, die erste Definition war in der VD mysteriös verunstaltet. F-PC erzeugte nur ein ungläubiges 'What?'.

Hiernach suchte ich ganz anders und woanders eine Lösung, es ließ mir keine Ruhe. Auch Wolfgang Allinger verdanke ich wichtige Anregungen zu meinem damaligen Programier-'Stil'. Zum Forth Kongress 1991März hatte ich dann die nächste Variante fertig [ak-91München].

## Looking Ahead

Hinter KEY KEY? gibt es in F-PC einige sehr interessante Interna. Nicht, daß mir jemand vorwerfe, ich behauptete, sie seien nicht dokumentiert, aber zumindest ich habe lange gesucht, bzw. habe lange gebraucht, zu begreifen, was sie wirklich machten und daß hier die Lösung vor sich hin schlummerte. Damals lag mit das 'Technical Reference Manual' zu F-PC v. 3.50 noch nicht vor [Ting-techman] und den Satz im Hilfetext:
" This can be useful when writing a key interpreter for an application." übersah ich. Das bezieht sich auf BIOSKEY BIOSKEY? , darunter steht dann noch etwas zu SCANCODES . Meine Application war: deutsche Tastatur. Ich grub tiefer im Quelltext und gelangte endlich zu den Wunderworten BIOSCHAR und BIOSKEYVAL , die ein 'Looking-Ahead' erlauben. Neben dieser Vorausschau erlauben sie aber eben auch die nachträgliche (!) Ermittlung des ScanCodes und da steckt die eigentliche Lösung: BIOSkeyVAL erlaubt im Nachhinein festzustellen, ob es nun 8bit-IBM-enhanced ASCII oder eine Funktionstaste war.

Die oben [UH-vd91-1] aufgeführten KEYchar und Lchar hatten mich weiter zu doKEY (forth) und DOaCHAR (edit) geleitet, dort werden die Tasten analysiert als Funktionstaste, Steuerungstaste ( Ctrl ) oder Buchstabe und dann ausgeführt. Zu beachten ist, daß in Forth (direkt interaktiv) und im Editor eine andersartige Auswertung genutzt wird. Das erscheint mir eher Platzverschwendung und Verwirrung als eine Notwendigkeit. Es ist wohl historisch aus zwei Quellen heraus so gewachsen. Immerhin erlaubt es den interessanten Vergleich zweier gleichwertiger (?!) Lösungen.

Zusammen mit KEY? wird der Wert in die Variable BiosChar geschrieben, bei KEY in die Variable BiosKeyVal . In Forth (LineEditor) landet der Wert dann im Value lCHAR bzw. im Editor im Value KeyChar . Hieran wurde nichts geändert.

Meine Modifikation besteht nun darin, daß im LineEditor (FORTH) das Schreiben eines Characters nur dann verhindert wird, wenn es eine Funktion(staste) ist, also in Bios-KeyVal das HighByte gesetzt ist. (Im Original wird das Screiben unterbunden sobald das bit 8 gesetzt ist, genauer: sobald Lchar nicht BL '~' BETWEEN liegt.) In CHAR? wird ein EXIT erzwungen.

Im Editor muß alles genau anders herum laufen, dort erfolgt der Ablauf intern umgekehrt. So wird dort die Ausführung einer Funktion unterbunden, wenn das HighByte gelöscht ist, es also keine Funktionstaste war. ?FUNCTIONS wird zum EXIT gezwungen. (Im Original wird dagegen jeder Code mit gesetztem bit 8 ausgeführt).

Leider beschränkt F-PC alle FunktionstastenCodes auf 0 bis 127, sie werden in der Tabelle zudem noch auf 128 bis 255 umgemappt. Damit sind einige Funktionen letztlich doch nicht zugänglich bzw. verdecken einander. Alt-Esc entpricht so Alt-0 (Alt-NULL), Alt-PgUp oder F11 sind nicht zugänglich. Aber damit konnte ich leben, ... bis ich mal wieder das Nagetier über den Tisch rollte.

### Fortsetzungs des Listings zu: KEYB8B (A. Klingelnberg)

```
PATCH DEFINITIONS
: mousekey      ( -- char )             \ allow mouse press to return key
\u  show.ms   show.ms
        BEGIN   pause
                key?
                mousechar or
        UNTIL mousechar ?dup
\u  hide.ms   hide.ms
        IF      off> mousechar
                on> mousewasdown
\u  ?fix-cursor ?fix-cursor
                fake-bioskey                                    \ akg91apr09
        ELSE    bioskey dup 127 and 0=
                IF      flip dup 3 =
                        IF      drop 0
                        ELSE    127 and 128 or
                        THEN
                ELSE    255 and
                THEN
        THEN keyfilter ;

PATCH ' mousekey   IS key


ONLY FORTH ALSO DEFINITIONS   decimal
```

## Die Maus als Affe

Zu spät bemerkte ich, daß die Maus nur noch Y_es, N_o, $1B (ESC) oder andere 7bit Zeichen ausgab. Sollte sie Funktionstasten auslösen, so spuckte sie wiederwillig 8-bitige Irgendwas-Zeichen heraus. Der Mouse-Treiber wurde verändert, nun äfft er per `FAKE-BIOSKEY` einen normalen Tatendruck auch bis ins 'Innere' hinein nach, eben bis hinein in `BIOSkeyVAL` und ist damit voll kompatibel zur Tastatur ( to fake == nachäffen, vortäuschen, türken ). Zu beachten ist allerdings, daß die Mouse KEINE 8-bit Zeichen ausgeben kann. Alle Werte >127 werden wie bisher (also voll-KOMPATIBEL) in Funktionen umgesetzt. [ak-92Rostock] Bis zur neuesten völlig überarbeiteten Version F-PC-ak v.4.0 [F-PC-ak-v.4.0] ist an diesen 8bit-Routinen nichts wesentliches verändert worden.

Wehmutstropfen gibt es aber noch: Keyboard-Macros können noch keine 8bit-chars. Das erscheint mir in der Praxis jedoch recht unbedeutend. Ich vermisse es nicht, obwohl ich laufend Macros nutze, da ich sehr faul bin. Allerdings sind im Original F-PC die Keyboard-Macros nicht mehr zugänglich. Ich überlasse dieses Problem gerne anderen, da in meiner Version ja alles läuft.

Weniger ein Bug sondern ein Quirk: z.B: ergibt Ctrl-Ü ein ESC. Es wären also noch einige weitere Worte in der Original Version zu modifizieren. Aber auch das sprengt hier den Rahmen.

Noch abzuändern wäre der Zeilen-Umbruch und die Wort-Links/Rechts-Sprung-Funktionen. Ich habe das sinnvoll ausfaktoriert, es ist nun kürzer als im Original, aber ... .

## msDOS kann's

Zu der von *Klaus Vogt* angesprochenen GROSSschreibe-Konvertierung betreff `aTBL` vertrete ich eine völlig konträre Meinung. Ich benötige//will keine Nicht-ASCII im Header, meine `aTBL` soll also auf UPERcase 7bit übersetzen. Dabei vertrete ich die Meinung, daß âäàåÄÅÆ alle auf 'A' umgesetzt werden müssen. Das Betriebssystem msDOS bietet Funktionen, die soetwas können: int $21 func $38 bzw. int $21 func $65 subfunc 01 spuckt einen Pointer zu einer Tabelle aus bzw. füllt eine bereitgestellte Tabelle, in der wiederum findet man/fra einen Pointer einer CaseMapCallAddress. Dagegen spuckt int $21 func $65 subfunc 02 den TabellenAnfang einer UPPERcase Tabelle aus. ( ab DOS v.3.3 in Verbindung mit country.sys oder codepage )

Die `aTBL` patche ich durch temporäres Umschalten des DirectoryPointers (`DP` ) auf den Body ( die PFA ) von `aTBL` und lade dann eine modifizierte Kopie des Original-Quelltextes. Diese Art des Patches macht am wenigsten Arbeit und ist als Quelltext 'normal' lesbar.

## Patch as Patch can

Gar nicht unwichtig sollte der Hinweis sein, daß meine Patch Methode über das VOCABULARY PATCH auch nach dem Patchen einen recht einfachen Umgang mit Sourcen und HilfeTexten in F-PC erlaubt:

```
PATCH WORDS
PATCH THESE WORDS
      <string1> <string2>
PATCH SEE <word>
PATCH b <word>
h <word>
```

Frühere Versuche waren allzu kompliziert oder führten für `BROWSE` und `HELP` zu trügerischen Ergebnissen. Zudem sind so ALLE Patches auf EINEN Blick zu übersehen. Eine erweitertes Wort `PATCH>` ( <name> -- ) patch-t das Wort im `CONTEXT` mit dem Wort aus `PATCH` , legt aber vorher eine Colon-Definition an mit dem gleichen Namen, und zwar im `VOCABULARY OLD` und mit der alten Funktion. Letzeres erfolgt nur, wenn <name> noch nicht in `OLD DEFINED` ist. Wie gesagt, ich bin tippfaul und bei meinen vielen Patches, wenn ich in den Eingeweiden von F-PC wühle... .

Welcome on KEYboard !

*p.s.*

Dieser Artikel wurde mit dem umgebauten F-PC erstellt. (Der modifizierte Editor des F-PC ist in meinem NortonCommander als StandardEditor (F4) installiert.)

*2.p.s.*

Ich habe *Tom Zimmer* mein `?CHAR` und `?FUNCTIONS` 1991März zugeschickt und auch mit ihm 1991Mai darüber mehrmals gesprochen. Bisher mochte er dieses nicht integrieren (ist für Amerikaner natürlich von untergeordneter Bedeutung). Trotzdem gebe ich nicht auf und werde es nochmal versuchen. Ein F-PC-HyperText-Demo-Action-Brief an ihn soll endlich fertig werden.

❑

### Literaturnachweise


- [ak-Version]
  Programm F-PC-ak
  veröffentlicht 1990aug21 (Version: ak0)
- [ak-vd90-3]
  VD 90sep p.9 Zimmmer forth V.3.50a.k.
- [F-PC-ak3]
  Programm F-PC-ak (ak3) ab 1990nov20
- [ak-vd90-4]
  VD 90dec p.6 Kurzmitteilung
- [UH-vd91-1]
  VD 91mar p.12 Ulrich Hoffmann
  'Konfiguration DEUTSCH' ???
- [ak-91München]
  F-PC-ak (ak5) vorgestellt bei der
  Forth-Konferenz München 91apr
- [Ting-tech-man]
  Technical Reference Manual, F-PC 3.5,
  2nd Edition, Dr. C. H.Ting, 1989nov
- [ak-92Rostock]
  F-PC-ak (ak8) wiederum vorgestellt
  Forth-Konferenz Rostock 92mar
- [F-PC3.5606]
  Tom Zimmer's Umlaute auf Anregung
  v. Andreas Goyke
  160. 12/31/91 ***** F-PC version 1.5606 *****
- [Vogt-VD92-3]
  VD 1992-3 p.7 Klaus Vogt:
  Internationaler Zeichensatz im F-PC
- [F-PC-ak-v.4.0]
  VD 1992 3 p.32
  Kutznotiz zu F-PC-ak ver.4.0
- [Gerdes-k3]
  Gerdes: Keyboard-Driver c't 88/07p.178
  modifiziert und entbuggt: akg


### Anmerkungen zu F-PC

`@>  ( <name> --n )`
wurde hier auch für eine Variable genutzt, da sie schneller ist als ein `@` .

`2R> 2drop ( -- ) ( R n1 n2 -- )`
F-PC sichert einen FAR Instruction Pointer auf dem Return Stack. Hiermit wird also um EINEN zusätzlichen Level UNnested .

# Die Uhr (2)

von Friederich Prinz

Am Beispiel einer im Hintergrund arbeitenden Uhr wird die TSR Programmierung unter ZF-Forth ausführlich behandelt. Es wird gezeigt, wie man TSRs via Interpreter startet.

## Vektoren 'verbiegen'

Bisher wurde mehrfach davon gesprochen, daß es möglich ist, die Vektoren in der oben beschriebenen Tabelle zu 'verbiegen'. Daß dies tatsächlich sehr einfach ist, wollen die folgenden Zeilen vermitteln.

Der FORTH Programmierer kennt in seiner Sprache einige 'Mechanismen', die im Prinzip mit der gleichen Technik arbeiten - die sogenannten Deferred Definitions. Auch die Konstruktion des 'puren' Umleitens via Sequenzen der Art - ' IrgendWas IS WasAnderes - bewirken technisch nichts Anderes als das, was bei der Umeitung eines Interruptvektors geschieht. Bild 1 soll das noch einmal verdeutlichen:

Der Vektor ist eine 32-Bit Adresse, bestehend aus Segmentangabe und Offset innerhalb dieses Segmentes, an welcher der vorgesehene Code zu finden ist.

Wird diese Adresse geändert, kann der Vektor auf einen anderen Code zeigen.

Durch die "Zwangsführung" bei der INT-Ausführung wird der mit dem Vektor adressierte Code zuverlässig ausgeführt.

Weil nun aber die Adresse der Vektorentabelle bekannt ist (die ersten 1024 Byte im RAM), läßt sich die Adresse jedes einzelnen Vektors 'von Hand' ermitteln (Nummer Mal 4) und sein Inhalt entsprechend den eigenen Vorstellungen umschreiben. Von dieser Möglichkeit, die im Übrigen problemlos funktioniert, sollte man von vorneherein Abstand nehmen. MSDOS/PCDOS's jüngere Varianten, sowie eine Reihe von speicheroptimierenden Programmen, lagern große Teile des Betriebssystems in Speicherbereiche aus, die bisher von DOS nicht konsequent genutzt wurden. Kompatibilitätsprobleme bei 'händischen' Manipulationen sind damit gewissermaßen vorprogrammiert. Deshalb bieten die Interrupts, neben vielen anderen Diensten, eine eigene Serviceroutine, um die Inhalte der Vektorentabelle auszulesen und neu zu beschreiben. Auf das Funktionieren dieser Routinen darf sich der Programmierer auch in allen zukünftigen DOS Versionen verlassen.

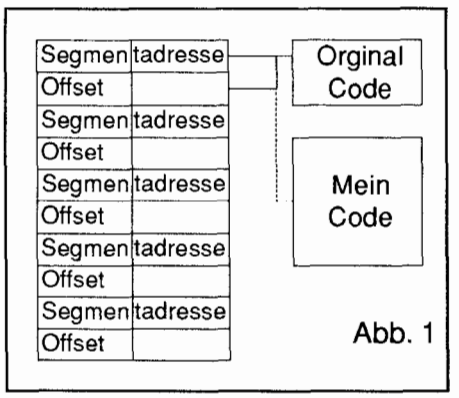


Abb. 1

## Der Assembler

(Fast) alle Forth-Systeme beinhalten einen Assembler, mit dessen Hilfe sich weit mehr als nur Primitives wie `DUP` und `SWAP` definieren lassen. Voraussetzung ist natürlich auch hier, daß der FORTHer 'seinen' Assembler bis in's Detail kennt und weiß, wie er welche Programmieraufgabe angehen muß.

Selbstverständlich lassen sich die Assembler der verschiedenen FORTH-Systeme nicht mit der Leistungsfähigkeit eines MASM 6.0 messen. Typischer Weise müssen sie ja auch nicht die gleichen Aufgaben bewältigen wie der MASM (Micro-Soft Macro Assembler). Das heißt nun aber nicht, daß MASM Programme erzeugen könnte, die einem FORTH-Assembler nicht möglich wären. Im Gegenteil - zwar sind FORTH-Assembler weniger komfortabel und beschränken sich in aller Regel auf das, was 'der Forther' mit seinem Assembler machen möchte, aber "möglich" ist trotzdem "Alles".

ZF's Assembler ist für die FORTH-Welt recht typisch. ZF's Assembler ist ein typischer ONE-Pass Assembler, das heißt, ein Assembler der seine Übersetzungsarbeit komplett in einem 'Durchlauf' erledigt. Das bringt einige Restriktionen in der 'Anlage' eines Programmes mit sich.

Bei der Arbeit eines ONE-PAss Assemblers (oder Compilers) können, ohne diverse Tricks, grundsätzlich nur Sprünge "nach rückwärts" erzeugt werden. Das heißt, wie bei FORTH generell üblich, daß alle in eine neue Definition einfließenden Routinen bereits bekannt sein müssen. Es ist nicht möglich eine Subroutine mit dem Namen "IrgendWas" aufzurufen, wenn diese Subroutine nicht vorher definiert wurde.

Ebenso können ONE-Pass Assembler in der Regel keinen relozierbaren Code erzeugen. Das heißt, daß solche 'einfachen' Assembler nur Adressbezüge errechnen können, die sich auf ein einziges Speichersegment beziehen. ZF's Assembler berechnet alle Adressen für das CODE-Segment des ZF-Systems. Soll ein solcher Code, wie es bei der UHR notwendig ist, in eine separates Segment ausgelagert werden, dann muß der Programmierer sich etwas einfallen lassen...

Abbildung 2 soll helfen, diese Thematik verständlich zu machen:

Die während der Übersetz-ung des CODEs erzeugten Adressen sind allesamt 'Abstände' zum Start des Code-Segmentes des je-weiligen FORTH-

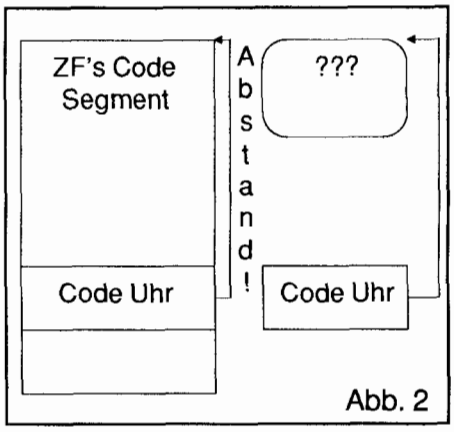


Abb. 2

Systems. Wenn der so erzeugte Code in eigenes, separates Seg-ment ausgelagert wird, bleiben die Adreßbezüge, die als "Abstände" er-rechnet wurden, gleich. Diese Abstände 'zeigen' dann aber in einen nicht definierten Bereich! Dieser Code wäre unbrauchbar!

Selbstverständlich gibt es Mittel und Wege, diese Klippe zu umschiffen. Auch die Codes von ONE-Pass Systemen lassen sich durchaus so gestalten, daß sie in anderen Segmenten als dem Ursprünglichen arbeiten können. Der Programmierer muß halt dafür sorgen, daß sein Programm die notwendigen Adressrechnungen während der Übersetzung vornimmt. MASM macht es nicht anders - nur eben 'von selbst'. Wie das bei einem Assembler herkömmlicher FORTH-Systeme funktioniert, wird letztlich der Quelltext der UHR zeigen.

Ein weiteres Problem bei der Arbeit mit FORTH-Assemblern ergibt sich aus der 'Redirektion' physikalischer Register auf Registernamen der Forth-Maschine. Ein Beispiel wurde bereits eingangs genannt. Das 'typische' Arbeitsregister in FORTH-Maschinen heißt W (Work). PCs nutzen hier 'von Haus aus' das Register AX. W ist aber in allen FORTH-Systemen identisch mit dem physikalischen Register BX. Das kann den Programmierer doch schon durcheinander bringen, um so mehr, als das Betriebssystem bei der Arbeit mit Interrupts seine Parameter in ganz bestimmten Registern erwartet. Hier sind die bereits angesprochenen Kenntnisse aller drei Elemente - PC-Hardware, Betriebssystem, FORTH-Maschine - unabdingbar.

Neben dieser 'Umbenennung' von BX nach W und der damit unterschiedlichen 'Wertigkeit' des physikalischen Arbeitsregister AX, müssen zumindest in ZF noch zwei weitere Register besonders erwähnt werden. Das physikalisch vorhandene IP (Instruction Pointer - Befehlszeiger) ist nicht identisch mit dem FORTH IP (Interpreter Pointer). Das FORTH IP ist im Grunde eine Systemvariable. Über diese Variable steuert der Interpreter den Ablauf/Fluß von gerade in Arbeit befindlichen Compile-Definitions !

Um hier Kollisionen zu vermeiden, hat *Tom Zimmer* auf die Implementierung des physikalischen IP in seinen Assembler verzichtet. Es gibt, zunächst, keine Möglichkeit IP 'von Hand' zu laden, weil der Assembler IP gar nicht 'kennt'. Statt dessen 'zeigt' das physikalische SI (SourceIndex - Quellen Index) auf FORTH's IP. SI wird aber bei vielen 'Datentransporten' intern benötigt und ist bei Transporten zwischen unterschiedlichen Speichersegmenten unabdingbar. Das macht bei Zugriffen auf SI umfangreiche Sicherungsarbeiten notwendig !

Ein ähnliches Problem wirft die Redirektion des physikalischen BP (Base Pointer) auf. BP 'zeigt' auf das 'Forthregister' RP (Returnstack Pointer). Und spätestens hier kann der sorglose Umgang mit den physikalischen Registern wirklich 'haarig' werden. Der Returnstack wird nicht nur zur Laufzeit von Applikationen genutzt, wenn der Interpreter auf ihm die Rückkehradressen für das interne Nesting sichert. Auch während der Übersetzung von Compiler und/oder Assemblerdefinitionen arbeiten schließlich FORTH-Worte die ebenfalls auf dem Returnstack diverse Adressen sichern. Mit anderen Worten: BP sollte nur sehr vorsichtig in Assemblerdefinitionen genutzt werden - und auch nur nach entsprechenden Sicherungen seiner Inhalte !

## Debugging - Fehler werden immer gemacht.

Die Fehlersuche ist, zumindest bei umfangreicheren oder komplexeren Assemblerdefinitionen, ein Kapitel für sich. Die Angebote der unterschiedlichen FORTH-Systeme an entsprechenden Werkzeugen reicht hier von 'ganz brauchbar' bis zu 'nicht vorhanden'. Ein wirklich gutes, direkt als Bestandteil eines FORTH's mitgeliefertes Werkzeug zur Fehlersuche in CODES, ist mir bisher nicht untergekommen. Dafür boten die entsprechenden Werkzeuge anderer Entwicklungsumgebungen, die sich über ZF's 'SYS' Kommando problemlos aufrufen lassen, allen notwendigen Komfort und alle notwendigen Optionen. Als ganz besonders 'Entwicklerfreundlich' hat sich in den letzten beiden Jahren das Programm BXDEBUG.COM erwiesen. BXDEBUG wird als Bestandteil des Entwicklerpaketes zu TCOM ausgeliefert und bietet dem Assemblerprogrammierer alles, was dieser zur völligen Kontrolle in der Testphase benötigt. Dabei ist BXDEBUG klein genug um die Festplatte nicht unnötig zu belasten, nimmt dem FORTH-System keinen zusätzlichen Speicherplatz weg (weil es als externes Programm gar nicht zu Forth gehört) und ist unkompliziert genug, um auch dem Einsteiger vom Start weg ein angenehmes Arbeiten zu ermöglichen.

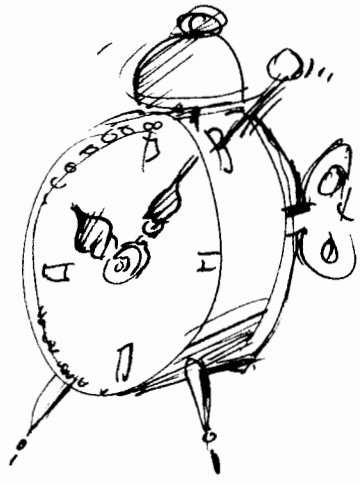
Nun ist die Arbeit mit solchen Werkzeugen, wie auch die Arbeit mit einem spezifischen FORTH selbst, mittlerweile, bedauerlicher Weise, zu

Wichtige Meldung in letzter Minute:

**Tom Zimmer** kommt zur Forth-Tagung nach Nürnberg! Er wird über seinen TCOM berichten.

"Ein wirklich gutes, direkt als Bestandteil eines Forth's mitgeliefertes Werkzeug zur Fehlersuche in CODES ist mir bisher noch nicht untergekommen."

einer Art Glaubenskrieg ausgeartet. Gewohnheit spielt auch bei der Auswahl der Werkzeuge eine große Rolle. Am Niederrhein heißt es sehr treffend : "Was der Bauer nicht kennt - das frißt er auch nicht". Darum soll an dieser Stelle nicht weiter auf ein spezielles DEBUG-Programm eingegangen werden. Vielmehr sollen die wichtigsten Kriterien aufgezählt werden, die von einem solchen Programm erfüllt werden müssen.



Einzelschritte müssen durchgeführt werden können. Das heißt, daß ein CODE Schritt für Schritt abgearbeitet werden muß. Dabei sollte die Option, die Anzahl der einzelnen Schritte vorzugeben, nicht fehlen. Nach jedem Einzelschritt müssen die Registerinhalte angezeigt werden. Farbliche Kennzeichnung sollte sofort auf veränderte Registerinhalte hinweisen. Die Flags im Statusregister müssen einzeln angezeigt werden. Auch hier sollten Veränderungen sichtbar sein.

Die Registerinhalte und die einzelnen Flags müssen vor jedem Schritt 'von Hand' veränderbar sein.

Der Stack sollte 'sichtbar' sein und mindestens die jüngsten 10 Einträge anzeigen. Auch die Inhalte dieser Einträge sollten während der Arbeit mit dem Debugger veränderbar sein.

Als ganz unverzichtbar hat sich zum Beispiel bei der Arbeit an der UHR die Option gezeigt, auf beliebige Speicherinhalte zugreifen zu können. Das heißt, ein 'ordentlicher' Debugger sollte jeden beliebigen Speicherbereich anzeigen können und Manipulationen darin zulassen.

Auf die Möglichkeit Quelltexte während des Debuggens explizit anzeigen zu lassen, kann fraglos verzichtet werden. Um im Einzelschrittmodus arbeiten zu können, muß das auszuwählende DEBUG-Programm ohnehin den zu betrachtenden CODE disassemblieren, so daß die Möglichkeit die diversen MOVs und POPs anzusehen, immer gegeben sein wird. Die im Quelltext (hoffentlich) vorhandenen Kommentare sind während des DEBUG-Laufes weniger wichtig.

Für welches Werkzeug sich der Einzelne letztlich auch entscheiden mag - auf den Einsatz eines Solchen kann bei der Assemblerprogrammierung kaum verzichtet werden. Und ebenso sicher wie der Programmierer 'seinen' Assembler beherrschen sollte, muß er auch mit seinen zusätzlichen Werkzeugen umzugehen verstehen.

## Endlich - die UHR

Selbstverständlich sind die bisherigen Teil-Themen - Interrupts - Assembler - Debugging - in keiner Weise erschöpfend behandelt worden. Das kann der vorliegende Aufsatz nicht leisten. Statt dessen sollten die vorhergehenden Seiten grundsätzliche Probleme und Fragen ansprechen, die sich dem Einsteiger in das Gebiet speicherresidenter Programmierungen aufwerfen. Wie bereits mehrfach gesagt, müssen notwendige, vertiefende Informationen aus der im Anhang vorgeschlagenen Literatur entnommen werden. Bei der Entwicklung zum 'Profi' (ich bin selbst keiner !) kommt dann noch der notwendigste aller Faktoren - Schweiß - dazu. Ohne Erfahrungen und Fehler geht's einfach nicht. Was die Implementierung residenter Hintergrundtasks angeht, nimmt 'die Uhr' zukünftigen, ähnlichen Projekten hoffentlich eine Menge dieser Erfahrungen vorweg. Fangen wir also an:

Zunächst sollte einmal grob beschrieben werden, was der Code zur Uhr eigentlich leisten muß:

Die Uhr soll in bestimmten, regelmäßigen Zeitabständen aktualisiert und auf den Bildschirm ausgegeben werden. Diese regelmäßige Aktualisierung soll 'immer' funktionieren, unabhängig davon, was das FORTH-System, oder die darin gerade laufende Applikation aktuell 'macht'. Das heißt - die Uhr muß schnell sein ! Sie darf weder zeitlich, noch durch Veränderung der Registerinhalte eine laufende Applikation stören. Letztlich muß der Code auch dann noch 'laufsicher' sein, wenn ein Programm zum Beispiel über ZF's 'SYS' Kommando

zeitweilig in die DOS-Ebene wechselt!

Die regelmäßige, zeitliche Steuerung läßt sich im Prinzip nur über den INT 8, bzw. über dessen Anhängsel INT 1Ch realisieren. Der INT 8 wird, wie oben beschrieben, in regelmäßigen, zeitlichen Abständen, nämlich alle 1/18,20416667 Sekunde, durch den Timer Baustein 8253 ausgelöst. Die Hardware sorgt zuverlässig dafür, daß dieses Signal geliefert wird. Weil der INT 8 aus den bereits beschriebenen Gründen besser unangetastet bleibt, muß ein Programm wie die Uhr sich an das Anhängsel, also an den INT 1Ch halten.

Nun muß die Uhr aber nicht wirklich jedesmal aktualisiert werden, wenn das Signal des Timers 'hereinkommt'. In der Regel wird eine solche Bildschirmanzeige in eine der oberen Ecken plaziert. Nachfolgende Applikationen werden ihre eigene Bildschirmanzeigen dann so einrichten, daß sie mit der Ausgabe der Uhr nicht kollidieren, weil die Uhr alle anderen Ausgaben rigoros überschreibt. Es würde also theoretisch ausreichen, wenn die Uhr einmal in jeder Sekunde aktualisiert würde. Für manche Applikation mag das zuwenig sein. Insbesondere dann, wenn die Uhr nicht am Bildschirmrand plaziert wird und trotzdem auch bei 'scrollenden' Bildschirminhalten ruhig stehen bleiben soll, wäre es wieder sinnvoll, sie bei jedem Signaleinlauf aufzufrischen. Also sollte die Uhr, entsprechend solchen Überlegungen, variabel oft pro Sekunde aufrufbar sein.

Der Timer liefert ein Signal mit einer bestimmten Frequenz, mehr nicht. Das heißt, daß dieses Signal zunächst einmal in eine gültige Uhrzeit umgerechnet werden muß. Rechenergebnisse zeitigen Zahlen - keine Ziffern ! Folglich wird eine weitere Aufgabe der Uhr sein, das Signal zu 'zählen', umzurechnen und die dabei entstehenden Ergebnisse in eine auf dem Bildschirm darstellbare Zeichenkette umzuwandeln. Natürlich kommt dann noch die Ausgabe selbst hinzu, die so trivial auch wieder nicht ist. Schließlich wollen laufende Applikationen ihre Bildschirmausgaben an genau der Stelle fortsetzen, an der sie von der im Hintergrund arbeitenden Uhr unterbrochen worden sind. Das heißt aber, daß die Uhr sich bei der Ausgabe die aktuelle Position des Cursors auf dem Bildschirm merken muß, ihre eigene Ausgabe vornehmen und danach den Cursor wieder an seine vorherige Position stellen muß.

'Außerhalb' der eigentlichen Uhr werden dann noch Routinen benötigt, mit denen sich die Uhr ein- bzw. ausschalten läßt. Ein- und Ausschalten bedeutet hier aber, daß der entsprechende Vektor 'verbogen' werden muß, daß bei DOS Speicherplatz für den CODE angefordert und die Uhr dorthin transportiert werden muß, und daß, beim Ausschalten, der nicht mehr benötigte Speicherplatz an DOS zurück gegeben werden muß.

*Fortsetzung folgt in VD 2/93*

# Fis(c)hing Forth

von Friederich Prinz
Homberger Str. 335, 4130 Moers 1, Tel.: 02841-58398

Es wird gezeigt, wie man das FISCHER-Technik-Interface mit Forth anspricht und so kleine Steueraufgaben löst.

Das PC Interface von FISCHER Technik (c) wird gewöhnlich ohne jegliche Steuersoftware ausgeliefert. Diese Steuersoftware kann zusätzlich bei der Firma FISCHER bezogen werden. An Steuerungsprogrammen bietet FISCHER ein TSR Modul (speicherresidentes Treiberprogramm) als Schnittstelle zu den gängigsten BASIC Interpretern, wobei in Verbindung mit GWBASIC Probleme bei der interaktiven Arbeit auftauchen. Zusätzlich zu dem TSR Modul wird von FISCHER ein Assembler Modul vertrieben, welches sich direkt in das BASIC RamTop einladen läßt und eine wesentlich direktere Steuerung des Interfaces erlaubt.

Neuere Versionen der FISCHER SoftWare bieten Schnittstellen zu Turbo Pascal und C, wobei auch hier ein TSR Modul die eigentliche Ansteuerung der HardWare übernimmt.

Via FORTH läßt sich das FISCHER Interface mit den nachfolgenden Worten direkt und unkompliziert steuern!

**Stichworte**

ZF
Steuerung
Messen
Interface
Assembler 8086
PRN-Port

## Ansteuerung des FISCHER Interfaces

- 4 Motorleitungen mit jeweils 4 Aktionen:
  Ein / Rechtslauf / Linkslauf / Aus
- 2 Analoge Eingänge
  EX / EY
- 8 Digitale Eingänge
  E1 bis E8
- Entwicklungssystem ist ZF-Forth, F83 kompatibles PD-System auf Compaq 80386SX LapTop unter MSDOS 5.0

❑

## Listing zu: Fis(c)hing Forth (Friederich Prinz)

```
HEX

 ( ------------------------------------------------------------------- )
 ( -allgemeine Konstanten für die Parameterübergabe an das Interface-  )
 ( ------------------------------------------------------------------- )

55 CONSTANT Ein              01 CONSTANT E1          90 CONSTANT EY
FF CONSTANT Aus              02 CONSTANT E2          A0 CONSTANT EX
55 CONSTANT Rechtslauf       04 CONSTANT E3
AA CONSTANT Linkslauf        08 CONSTANT E4
                             10 CONSTANT E5
03 CONSTANT Motor.1          20 CONSTANT E6
0C CONSTANT Motor.2          40 CONSTANT E7
30 CONSTANT Motor.3          80 CONSTANT E8
C0 CONSTANT Motor.4
```

Dem parallelen Druckerport stehen zwei Portadressen zur Verfügung, die beide exakt gleich arbeiten. Der 'originale' Druckerport liegt auf der Adresse 0378h (Ausgänge) und 0379h (Eingänge).
Viele Rechnersysteme benutzen aber statt dieser Adressen die Druckeradressen auf dem MDA (Monochrome Display Adapter), 03BCh (Ausgänge) und 03BDh (Eingänge). Welche Adressen tatsächlich von der einzelnen Maschine, bzw. vom Betriebssystem benutzt werden, muß im Zweifelsfall ausprobiert werden...

```
CREATE PortOut  03BC ,          \ Ausgabeport - altern. 0378h
CREATE PortIn   03BD ,          \ Eingabeport - altern. 0379h
CREATE BitMaske 0000 ,          \ 'Steuerbyte' enthält als Bitmaske die
                                \ Kennungen für die einzelnen Motoren
                                \ Links, Rechts, Aus ...

   ( ------------------------------ )
   ( --- Ansprechen der Hardware --- )
   ( ------------------------------ )

LABEL IF.INIT ( -- )            \ INIT-Grundroutine, wird von mehreren
  BL  BitMaske #)    MOV        \ CODEs genutzt...
        0008 #  CX MOV
      PortOut #) DX MOV
HERE
            30 #  AL MOV      \ 'Ruhepegel' X & Y 'getriggert'
                  BL RCL      \ Alle Bitkennungen werden durchgereicht...
     M <=  IF 04 #  AL OR     \ wenn CARRAY gesetzt, dann 'DATA.OUT'...
       THEN
             0    AL OUT      \ 'Ruhepegel' an Port senden
            08 #  AL OR       \ 'Clock'- Interface 'triggern'
             0    AL OUT
LOOP
            39 #  AL MOV      \ 'LOAD.OUT' an Port senden ...
             0    AL OUT
                  STI         \ externe Interrupts freigeben.
NEXT C;

CODE Interface.Init ( -- )    \ Initialisierung des Fischer-Interface
          0000 #  BX MOV      \ Bitmaskierung der Motoren wird auf '00'
                  CLI         \ gesetzt, externe Interrupts werden ge-
sperrt
       IF.INIT #)     JMP     \ und die Subroutine IF.INIT aufgerufen...
END-CODE

CODE MOTOR ( n -- )           \ Motoren ansprechen
                  AX POP      \ AX = AH,AL / AH = Aktion,AL = MotorNummer
      BitMaske #) BL MOV      \
                AL BL OR      \ MotorNummer nach BL
                AL AH AND     \ Aktion/Motor 'ausmaskieren'
                AH BL XOR     \ und nach BL übertragen
      BL BitMaske #) MOV      \ neue BitMaske speichern
          If.Init #) JMP      \ Übergaberoutine aufrufen
END-CODE

LABEL AnalogOut ( -- )        \ Ausgabe der analogen Werte
      03FF # AX MOV           \ Ergebnis aus der Abfrage des AD Wandlers
          CX AX SUB           \ ist 1023 minus Anzahl der Loops, bzw.
             AX PUSH          \ minus des 'Restes' in CX...
                STI
NEXT END-CODE

LABEL Analog ( -- )           \ Analoge Eingänge auslesen...
      03FF #  CX MOV          \ Schleifenzähler
         0    AL OUT          \ Bitmuster 90h (EY) oder A0h (EX)  Interf
```

## Fortsetzung des Listing zu: Fis(c)hing Forth von Friederich Prinz

```
       38 #  AL MOV          \ Eingabe 'triggern'
        0    AL OUT
  PortIn #)   DX MOV          \ Adresse des Eingangsports nach DX
HERE                          \ Schleife : Abfragen des AD Ports, bis das
        0    AL IN           \ 'RCL' das CarryFlag setzt. Als Ergebnis
             AL RCL          \ wird in AnalagOut 1023 minus Anzahl der
      AnalogOut JAE          \ der Schleifendurchläufe ausgegeben.
LOOP
      AnalogOut #) JMP
END-CODE

CODE Lesen ( b -- n | f )
             CLI             \ ext. Intrps. sperren
           AX POP            \ Eingangsnummer aufnehmen
 PortOut #) DX MOV
     90 #  AL CMP            \ Eingang EY ???
       Analog JE             \ dann Analogroutine aufrufen
     A0 #  AL CMP            \ Eingang EX ?
       Analog JE             \ dann Analogroutine aufrufen
           AX PUSH           \ Eingangsnummer sichern
   0008 #  CX MOV            \ Interface initialisieren
     32 #  AL MOV
      0    AL OUT
     08 #  AL OR
      0    AL OUT
HERE                          \ Bits auf den Eingangsleitungen einlesen
           AH RCL
  PortIn #) DX MOV
      0    AL IN
     80 #  AL AND              \ Bits ausmaskieren mit 10000000
     80 #  AL CMP
0= IF 01 #  AH OR
 THEN
             CLC
 PortOut #) DX MOV             \ Interface zurücksetzen
     30 #  AL MOV
      0    AL OUT
     38 #  AL MOV
      0    AL OUT
LOOP
        AH BL MOV             \ Eingangsleitung einlesen
           AX POP
        BL AH MOV
        AH AL AND
        AH AH XOR             \ 'Lesen' liefert die Nummer der
     00 #  AL CMP             \ Eingangsleitung zurück, wenn diese
          0= IF       AL PUSH \ 'gesetzt' ist, ansonsten liefert
           ELSE  1 #  AL MOV  \ 'Lesen' eine NULL !
                      AL PUSH \ Das wird hier korrigiert - Eine
           THEN               \ 'gesetzte' Leitung liefert nur
             STI              \ noch '1'
NEXT END-CODE

DECIMAL
```

HighLevel Steuerung des Interfaces durch interaktive Aufrufe oder durch programmierte Applikationen

Das Wort `Lesen` erwartet auf dem Stack die Nummer der auszulesenden Leitung. Da die Leitungsnummern in versprachlichen den Konstanten abgelegt sind, wird zunächst die Konstante mit dem gewünschten Leitungsnamen aufgerufen, und dann `Lesen`. `Lesen` liefert als Ergebnis bei digitalen Eingangsleitungen eine '1', wenn auf der jeweiligen Leitung ein Signal anliegt, und eine '0' wenn kein Signal anliegt. Bei den analogen Eingangsleitungen liefert Lesen einen Wert zwischen 1023 und 0 auf den Stack.
Beispiele :

```
E1 Lesen .   --   0
EX Lesen .   -- 570
```

Das Wort "Schalten" wird mit der MotorNummer und der jeweiligen Aktion aufgerufen.
Beispiele:

```
Motor.1 RechtsLauf Schalten
Motor.2 Ein        Schalten
Motor.3 LinksLauf  Schalten
Motor.4 Aus        Schalten
```

Da während des 'Durchreichens' der 'BitMaske' im Wort `IF.INIT` alle Stati aller vier Motorleitungen an das Interface gegeben werden, werden bei jedem Aufruf von `Schalten` alle bereits 'gesetzten' Motoren angesprochen. Das heißt, daß zum Beispiel der Aufruf `Motor.1 Linkslauf Schalten` auch alle anderen Motoren mitschaltet, wenn diese vorher schon einmal angesprochen wurden. Wenn ein Motor nicht mitlaufen soll, dann muß die entsprechende Leitung ausdrücklich abgeschaltet werden !!!

```
: Schalten ( Motor.n Aktion -- )
    256 *          \ Aktion nach 'AH' schieben
      +            \ MotorNummer aufaddieren
    Motor
;
  ( Ende der Software Schnittstelle )
```

---

Das Testprogramm zum FISCHER Interface soll dabei helfen, die HardWare auszutesten. Insbesondere muß auf jedem System aus FISCHER - PC - und Betriebssystem ausprobiert werden, ob die Portadressen PortOut und PortIn korrekt gewählt wurden. Wenn das Interface von dem jeweiligen PC nicht angesprochen wird, müssen diese Portadressen auf die oben angegebenen Alternativen geändert werden. Wenn das Interface angesprochen wird und die Eingangssignale abgefragt werden können, soll die 'MotorRoutine' des Testprogramms zeigen, ob die jeweilgen Drehrichtungen der einzelnen Motoren korrekt angegeben werden.

```
        ( ---------------------------------------- )
        ( --- FISCHER Interface Testprogramm --- )
        ( ---------------------------------------- )

: .Rahmen ( -- )
    18  5 AT ."      FISCHER Interface Testprogramm     "
    18  6 AT ."                                         "
    18  7 AT ."  E1 E2 E3 E4 E5 E6 E7 E8    EX    EY    "
    18  8 AT ."                                         "
    18  9 AT ."                                         "
    18 10 AT ."  Motor.1   Motor.2   Motor.3   Motor.4 "
    18 11 AT ."                                         "
    18 12 AT ."                                         "
    18 13 AT ."   fep 12/92                  ZF-Forth"
;
: .Leitungen ( -- )              \ Digitale Signale ausgeben
     20  8 AT
      E1 Lesen 2 .R  E2 Lesen 3 .R  E3 Lesen 3 .R  E4 Lesen 3 .R
      E5 Lesen 3 .R  E6 Lesen 3 .R  E7 Lesen 3 .R  E8 Lesen 3 .R
      EX LESEN 7 .R  EY LESEN 6 .R
;
: .MotorStati ( -- )             \ Auslesen der Motorzustände und
     BitMaske @
     DUP   1 AND   1 = IF 20 11 AT ." links "  THEN   \ Status Motor.1
     DUP   2 AND   2 = IF 20 11 AT ." rechts"  THEN
     DUP   3 AND   0 = IF 20 11 AT ." aus   "  THEN
     DUP   4 AND   4 = IF 30 11 AT ." links "  THEN   \ Status Motor.2
     DUP   8 AND   8 = IF 30 11 AT ." rechts"  THEN
     DUP  12 AND   0 = IF 30 11 AT ." aus   "  THEN
     DUP  16 AND  16 = IF 40 11 AT ." links "  THEN   \ Status Motor.3
     DUP  32 AND  32 = IF 40 11 AT ." rechts"  THEN
     DUP  48 AND   0 = IF 40 11 AT ." aus   "  THEN
     DUP  64 AND  64 = IF 50 11 AT ." links "  THEN   \ Status Motor.4
     DUP 128 AND 128 = IF 50 11 AT ." rechts"  THEN
         192 AND   0 = IF 50 11 AT ." aus   "  THEN
;
: Interface.Test ( -- )
    DARK
    .Rahmen
    BEGIN
    .Leitungen                    \ 200 MilliSekunden Pause bis zum
                                  \ nächsten Aufruf --- AUSPROBIEREN
     .MotorStati        200 MS    \ dieser Wert ist maschinenabhängig !
      13 15 AT ." Ein beliebiger Tastendruck beendet das Testprogramm"
      13 17 AT ." Zur Motorenkontrolle bitte MotorNummer und 'Aktion'"
      13 18 AT ." angeben, und das Testprogramm erneut aufrufen....."
      KEY? IF  EXIT THEN
    AGAIN
;
```

# ANS Forth

## Der letzte Stand

von Bernd Paysan
Stockmannstr. 14, 8000 München 71, Tel. 089-798557

Der langerwartete ANSI-Standard für Forth nähert sich nach 5 Jahren Arbeit der Verabschiedung.

In der ersten Phase trug ein Experten-Komitee (das X3J14 Technical Committee) aus vorhergehenden Standards und aus der allgemeinen Praxis Vorschläge zusammen, bewertete sie und suchte Kompromisse, wo es in der Praxis unverträgliche Unterschiede gab. Insgesamt nahmen 45 Personen teil, darunter Prominenz wie *Charles Moore*, *Bill Ragsdale* oder *Tom Zimmer* (die alle nicht mehr dabei sind). 18 arbeiteten am dpANS 3. Den Vorsitz führt *Elizabeth Rather*.Während dieser Zeit waren der vorgeschlagene Standard über FTP oder E-Mail zu beziehen, die BASIS-Dokumente. Danach schlossen sich zwei 4-monatige "Public Review Periods" an. Seither unterliegen die Dokumente dem Copyright des *American National Standard Institute*, sind also nicht mehr über elektronische Netze zu haben.

Bis zur endgültigen Annahme des Standards folgen nun zweimonatige Public Review Periods, solange, bis der Standard offensichtlich akzeptiert ist. Die erste solche Review Period endete letztes Jahr am 25. August; das Echo war so positiv, daß wahrscheinlich keine weitere Review Period kommt. Dann kann der vorgeschlagene Standard einer unabhängigen Kommision vorgelegt werden (also Leuten, die nicht aus der Forth-Ecke kommen). Wird er hier verabschiedet, ist er offiziell gültig; man kann also seine Systeme und Programme mit dem Attribut "ANS FORTH" versehen. Nach dem derzeitigen Terminplan soll das im Juni sein (aber lassen wir uns überraschen).

**Stichworte**

ANS-Forth
dpANS 4

## Der Aufbau des Standards

Der Standard beschreibt Forth als Sprache, die aus einzelnen semantischen Einheiten (den Wörtern) besteht. Die einzelnen Wörter sind zu Gruppen geordnet, den sogenannten "word sets". Damit ist der Standard modular. Zwingend mu nur das CORE word set implementiert werden; auch hier reicht es, wenn Teile nur als Source vorliegen und auf einfache Weise geladen werden können. Jedes word set ist in einen Stamm und eine Erweiterung (mit EXT markiert) aufgeteilt. Sobald man ein word set anbietet, müssen alle Wörter des Stamms implementiert sein, aus dem EXT word set kann man sich die "Rosinen" herauspicken. Allerdings darf man dann keine gleichnamigen Wörter anbieten, die etwas anderes leisten.

## Änderungen seit dpANS 2

- SHIFT ist jetzt in LSHIFT und RSHIFT aufgeteilt, die beide vorzeichenlos bitweise schieben.
- Mit EKEY>CHAR isoliert man die Information, welches Zeichen gedrückt wurde, aus einem Tastaturevent.

### dpANS 4 Word Sets

**BLOCK**

```
BLK BLOCK BUFFER EVALUATE
FLUSH LOAD SAVE-BUFFERS UPDATE
```

**BLOCK EXT**

```
EMPTY-BUFFERS LIST REFILL SCR
THRU \
```

**CORE**

```
 ! # #> #S ' ( * */ */MOD + +!
+LOOP , - . .\dq{} / /MOD 0<
0= 1+ 1- 2! 2* 2/ 2@ 2DROP 2DUP
2OVER 2SWAP : ; < <# 0 > >BODY
>IN >NUMBER >R ?DUP @ ABORT
ABORT\dq{} ABS ACCEPT ALIGN
ALIGNED ALLOT AND BASE BEGIN
BL C! C; C@ CELL+ CELLS CHAR
CHAR+ CHARS CONSTANT COUNT CR
CREATE DECIMAL DEPTH DO DOES
DROP DUP ELSE EMIT
ENVIRONMENT? EVALUATE EXECUTE
EXIT FILL FIND FM/MOD HERE HOLD
I IF IMMEDIATE INVERT J KEY
LEAVE LITERAL LOOP LSHIFT M*
MAX MIN MOD MOVE NEGATE OR OVER
POSTPONE QUIT R> R@ RECURSE
REPEAT ROT RSHIFT S" S>D SIGN
SM/REM SOURCE SPACE SPACES
STATE SWAP THEN TYPE U. U< UM*
UM/MOD UNLOOP UNTIL VARIABLE
WHILE WORD XOR [ ['] [CHAR] ]
```

**CORE EXT**

```
#TIB .( .R 0<> 0> 2>R 2R> 2R@
:NONAME <> ?DO AGAIN C" CASE
COMPILE, CONVERT ENDCASE ENDOF
ERASE EXPECT FALSE HEX MARKER
NIP OF PAD PARSE PICK QUERY
REFILL RESTORE-INPUT ROLL
SAVE-INPUT SOURCE-ID SPAN TIB
TO TRUE TUCK U.R U> UNUSED
VALUE WITHIN [COMPILE] \
```

**DOUBLE**

```
2CONSTANT 2LITERAL 2VARIABLE
D+ D- D. D.R D0< D0= D2* D2/
D< D= D>S DABS DMAX DMIN
DNEGATE M*/ M+
```

**DOUBLE EXT**

```
2ROT DU<
```

**EXCEPTION**

```
CATCH THROW
```

**EXCEPTION EXT**

```
ABORT ABORT"
```

**FACILITY**

```
AT-XY KEY? PAGE
```

## dpANS 4 Word Sets (Fortsetzung)

### FACILITY EXT

```
EKEY EKEY>CHAR EKEY? EMIT? MS TIME&DATE
```

### FILE

```
( BIN BLOCK BUFFER CLOSE-FILE CREATE-FILE DELETE-FILE
FILE-POSITION FILE-SIZE FLUSH INCLUDE-FILE INCLUDED
OPEN-FILE R/O R/W READ-FILE READ-LINE REPOSITION-FILE
RESIZE-FILE S" SAVE-BUFFERS SOURCE-ID W/O WRITE-FILE
WRITE-LINE
```

### FILE EXT

```
FILE-STATUS FLUSH-FILE REFILL RENAME-FILE
```

### FLOAT

```
>FLOAT D>F F! F* F+ F- F/ F0< F0= F< F<D F@ FALIGN FALIGNED
FCONSTANT FDEPTH FDROP FDUP FLITERAL FLOAT+ FLOATS FLOOR
FMAX FMIN FNEGATE FOVER FROT FROUND FSWAP FVARIABLE
REPRESENT
```

### FLOAT EXT

```
DF! DF@ DFALIGN DFALIGNED DFLOAT+ DFLOATS F** F. FABS
FACOS FACOSH FALOG FASIN FASINH FATAN FATAN2 FATANH
FCOS FCOSH FE. FEXP FEXPM1 FLN FLNP1 FLOG FS. FSIND FSINCOS
FSINH FSQRT FTAN FTANH F PRECISION SET-PRECISION SF!
SF@ SFALIGN SFALIGNED SFLOAT+ SFLOATS
```

### LOCAL

```
(LOCAL) TO
```

### LOCAL EXT

```
LOCALS|
```

### MEMORY

```
ALLOCATE FREE RESIZE
```

### SEARCH

```
DEFINITIONS FIND FORTH-WORDLIST GET-CURRENT GET-ORDER SEARCH-WORDLIST
SET-CURRENT SET-ORDER WORDLIST
```

### SEARCH EXT

```
ALSO FORTH ONLY ORDER PREVIOUS
```

### STRING

```
-TRAILING /STRING BLANK CMOVE CMOVE> COMPARE SEARCH SLITERAL
```

### TOOLKIT

```
.S ? DUMP SEE WORDS
```

### TOOLKIT EXT

```
:NONAME ;CODE AHEAD ASSEMBLER BYE CODE CS-PICK CS-ROLL
EDITOR FORGET STATE [ELSE] [IF] [THEN]
```

- Es gibt `THROW`-Codes für eine groe Anzahl Fehler.
- `BIN` erlaubt Dateizugriff auf Binär-Dateien; damit können Text- und Binär-Dateien unterschieden werden.
- Das Blockfile-Interface wurde gestrichen. Es gibt jetzt keien Standardmethode mehr, um Blockfiles auszuwählen.
- Zum Wandeln von Fließkommazahlen in Texte wurde das Primitiv `REPRESENT` eingeführt, welches die Mantisse in einen Puffer ablegt und Exponent/Vorzeichen auf dem Stack zurückgibt. Damit lassen sich verschiedene Ausgabeformate (Fixpunkt, wissenschaftliche und Ingenier-Ausgabe) einfach erzeugen.
- Zur direkten Benutzung von Locals wurde `LOCALS|` eingeführt.
- `TIB` und `#TIB` wurden durch `SOURCE` ersetzt, das auch beim Interpretieren von Blöcken den Inputstring zurückgibt.
- `SLITERAL` compiliert ein String-Literal.
- Ein paar Wörter wurden gestrichen (wie `LEX`), umbenannt und zwischen den Gruppen umhergeschoben.

❑

## Verschlucktes

rk Traditionell hat unser DTP-Prcgramm wieder Sonderzeichen verschluckt. *Michael Majors* Artikel (ZEIGE: XY.DBF) wurde besonders liebevoll bedacht: nicht weniger als 12 mal wurden ">" oder "<" unter den Ventura-Teppich gefegt. Dort haben wir sie hervorgeholt. Hier sind sie: <<<>>>>>>>>>. Ach so, Sie wollen auch noch wissen, wo sie hingehören? Nun gut.

```
S.30/Z.12   : >IN_IO ( -- ) ...
S.30/Z.21   >IN_IO
S.30/Z.24   C@ 0> IF DROP DROP
S.30/Z.28   ... $>HANDLE
S.31/Z.11   : >FELD ( -- )
S.31/Z.15   ... \ <--
S.31/Z.19   ... \ <--
S.31/Z.58   POS_AUF S>D
S.31/Z.61   AUS 65 > IF ....
S.31/Z.66   ... D_ADR >FELD
S.32/Z.05   DUP 32000 > IF
S.32/Z.37   ... DBLES <NAME>.DBF
```

## Glück gehabt

rk Aus den 42 Einsendern (von insges. 49 ), die pünktlich bis zum 20.01. ihren Fragebogen an die Redaktion geschickt hatten, wurden die folgenden fünf Gewinner gezogen:

1. Bernock J., 2857
2. Freitag R., 2930
3. Hohl H., 2666
4. Feltgen M., 2674
5. Kotulla L., 2559

Herzlichen Glückwunsch. Als Glücksfeh fungierte *Dorothea Knopf.* Die Gewinner haben durch *Jörg Staben* Gewinne (Disketten + Boxen) erhalten, die von *Friederich Prinz* gestiftet wurden.

## dBase wanted:

(siehe WANTED in 3/92)

akg Zu diesem Problem verweise ich auf folgende Files in TurboForth ("Jedi-Forth"): dbCOOK.FTH und DEMOcook.FTH. Der Quelltext ist Fließtext mit französischen Kommentaren, F83-ähnlich.

## Buch gesucht

Gesucht wird das Buch «Designing And Programming Personal Expert Systems» von *C. Townsend & D. Feucht.* Titel ist vergriffen. Eventuell leihweise.

K. Giese, Tel. 0+2423 - 52 92 a.

# Lust statt Frust

## Ein Einstieg in die DFÜ

von Jens Wilke

Grundbegriffe der Datenfernübertragung werden erläutert. Dem DFÜ-Neuling soll der Artikel als praktischer Ratgeber dienen.

Das interessante an Forth-Büchern ist immer, daß, bevor das Buch richtig anfängt, man erst mal mit den unzähligen Vorteilen überschüttet wird, die diese Sprache hat. Dieses Vorgehen kritisiere ich zwar, da eigentlich die Sprache für sich selbst "sprechen" sollte. Ich werde es in diesem Einführungstext aber genauso halten, um den Leser, der noch gar nichts von DFÜ weiß und/oder nichts damit zu tun haben möchte, etwas neugierig zu stimmen. Aber noch eine einführende, kurze Erklärung vorweg: DFÜ benutzt zur Übertragung normale Telefonverbindungen, wie z.B. auch Telefax. Um den Computer an das Telefon anzuschließen, braucht man ein sogenanntes Modem (i.A. DAS Modem, obwohl es von der MODulator/DEModulator kommt) oder einen Akkustikkoppler. Das Modem wird direkt an die Telefonleitung angeschlossen; es kann veranlaßt werden, selbstständig zu wählen, während bei einem Akkustikkoppler der Telefonhörer in extra dafür vorgesehene Muffen gesteckt wird und von Hand gewählt werden muß. Da beim Akkustikkoppler der Bedienungskomfort und die Übertragungssicherheit/-geschwindigkeit (wegen der Muffen) nicht sehr groß ist, wird er heutzutage nicht mehr sehr oft benutzt (höchstens in Telefonzellen...).

- pro DFÜ: (im Gegensatz zu Zeitungen oder gelber Post)
  Billiger und schneller als Briefe zu verschicken -Mit DFÜ werden Diskussionen über Grenzen hinweg erst möglich, da eine Nachricht viele Leute erreicht -Billiger als miteinander zu Telefonieren und man kann überlegt seine Antworten schreiben -Ersetzt den Anrufbeantworter -Man ist sehr gut über Neuigkeiten informiert -Bei dringenden Problemen ist oft jemand mit Erfahrung zur Hand.
- contra DFÜ:
  Um längere Programme/Texte zu verschicken ist DFÜ in der Regel ungeeignet (Ferntarif!) -Man muß sich ein Modem kaufen und in ein sog. Terminalprogramm einarbeiten.

Für Forthler sind die Vorteile wohl besonders gewichtig, da sie so weit in alle Windrichtungen verstreut sind, daß man sich nie an einen Tisch zusammensetzen kann.

**Stichworte**

Mailbox
BBS
DFÜ
MODEM

## Was braucht man zum Einstieg?... Modems

Das wichtigste ist ein Modem. Beim Modemkauf ist zunächst die Übertragungsgeschwindigkeit/rate wichtig. Sie wird in Baud bzw. Bits pro Sekunde (BPS) gemessen (das ist übrigens nicht genau das gleiche!). Die momentan gängige Geschwindigkeit ist 2400 Baud, das entspricht (geteilt durch zehn) etwa 240 Zeichen in der Sekunde. Es gibt auch schnellere Modems, die sich für den privaten Gebrauch kaum eignen; es sei denn, man möchte tiefer einsteigen und erreicht überhaupt keine Mailbox zum Ortstarif. Diese Modems sind heute schon für 200-300 DM zu haben. Der Kauf eines postzugelassenen Modems rentiert sich normalerweise nicht. Obwohl der Betrieb nicht gestattet ist, wird von der Post praktisch nichts dagegen unternommen. Billige Modems aus Fernost oder Amerika sind sogar zu bevorzugen, da diese erfahrungsgemäß den wenigsten Ärger beim Betrieb verursachen. Die deutsche Modemindustrie steckt teilweise noch in den Kinderschuhen; der einzige Marktvorteil ist die ZZF-Nummer, die aber dem Anwender nichts bringt; denn ein einwandfreies Funktionieren ist mit ZZF-Nummer nicht garantiert. Achten sollte man hier auf ein Modem das Hayes bzw. AT-Befehlssatz hat. Damit kann man dem Modem Kommandos geben wie `Wähle`, `Auflegen` usw. Dieser Befehlssatz wird eigentlich von jedem Modem unterstützt; wenn nicht, muß man einen Exoten erwischt haben.

## Terminalprogramm

Das Terminalprogramm ist sozusagen das Handwerkszeug das man zur DFÜ benötigt. Ein Terminal ist ein Bildschirm mit serieller Schnittstelle, das im Deutschen auch ausdrucksvoll als "Datensichtgerät" bezeichnet wird. Dabei wird oft der Begriff ANSI-Emulation erwähnt. Das bedeutet, daß ein ANSI-Terminal - per Programm emuliert - nachgbildet wird. Mit dieser Emulation versteht der Computer dann "Escape-Sequenzen", mit denen die Farbe umgeschaltet oder der Cursor bewegt werden kann. Im Prinzip ist das Schnickschnack, aber recht beliebt. Ein TTY-Terminal stellt nur ASCII-Zeichen dar und versteht vielleicht gerade noch Return und einen Tabulator. In unserer Mailbox wird eigentlich nur ein TTY- Terminal richtig unterstützt, damit es keine Probleme bei unterschiedlichen Computern/Terminalprogrammen gibt. Ein Terminalprogramm sollte aber noch etwas mehr können. In einem Terminalprogramm hat man ein oder mehrere Protokolle. Ein Protokoll benutzt man wenn man Daten wie z.B. Binärdaten oder gepackte Files fehlerfrei übertragen möchte. Dazu zerstückelt das Protokoll die Daten in Blöcke und gibt jedem Block noch eine Prüfsumme mit auf den Weg. Das gängigste Protokoll ist ZMODEM, ein Terminalprogramm, das ZMODEM nicht

unterstützt, ist nicht praxistauglich. Zusätzlich bietet ein Terminalprogramm noch andere nützliche Features, z.B. die Verwaltung eines "Telefonbuches" mit den Mailboxnummern und vieles mehr. Für den PC ist das Programm *Telemate* zu empfehlen, da es sehr umfangreich und durch seine Menüs bzw. Oberfläche (SAA-Like) sehr gut zu bedienen ist.

## Ein wenig Geduld

In der DFÜ ist es selten, daß alles auf Anhieb klappt. Es gibt dafür zu viele Ecken und Enden an denen man drehen kann oder sogar drehen muß. Zunächst muß man auf die richtige Geschwindigkeit achten. Stimmt sie nicht, bekommt man nur Zeichensalat. Das nächste wären die Parameter. Hier ließt man häufig in Mailboxlisten "8N1". Das steht für 8 Datenbits, No Parity und 1 Stoppbit. Um die armen Anwender nicht allzusehr zu verwirren verwendet man 8 Datenbits auch da, wo eigentlich nur 7 Bit ASCII-Zeichen übertragen werden müssen, obwohl natürlich nur 7 Datenbits ein Bit einsparen würde und somit schneller wäre. Der Grund liegt auch darin, daß man früher oder später auch Binärdaten übertragen möchte.

## EMail

Wenn man den Einstieg in die 1DFÜ geschafft hat, tut sich einem sofort die wunderbare Welt der EMail (electronic mail) auf. Bei den meisten Boxen kann man nicht nur Nachrichten an Benutzer derselben Mailbox abschicken, sondern fast mit dem gleichen Aufwand und Handgriffen eine Mail an einen Freund in Californien oder einen Kollegen in Australien verschicken. Dazu gibt es verschiedene Netzwerke, das sind Zusammenschlüsse von Computersystemen über die ganze Welt hinweg. Im Gegensatz zu den Netzwerken, die wir kennen, muß es aber nicht nötig sein, daß zwischen den Computern eine feste Verbindung besteht; teilweise wird hier auch wieder auf die gute alte Telefonleitung zuruückgegriffen. Universitäten und große Firmen sind meist direkt miteinander verbunden, während Mailboxen über die Telefonleitung ihre Nachrichten austauschen. Bei gut organisierten Netzwerken, die über die Telefonleitung arbeiten, kann man davon ausgehen, daß ein Brief am nächsten Tag seinen Empfänger erreicht hat.

## Foren, Newsgroups, Bretter

Newsgroups sind Bereiche in denen bezüglich eines bestimmten Themas Nachrichten ausgetauscht werden. Hier können Fragen gestellt werden oder Diskussionen geführt werden. Foren und Bretter sind Synonyme für das englische Wort Newsgroup. Jede Nachricht hat eine Betreff-Zeile (genannt SUBJECT), damit jeder andere sofort weiß, um was es ungefähr in dem Text geht. Steht ein "RE:" (heißt soviel wie: zurück) in der Subject-Zeile, kann man daran sehen, daß dieser Text auf einen vorhergehenden Bezug nimmt. Das ist dann meistens die Antwort auf eine Frage oder eine zusätzliche Stellungnahme zu einem Thema. Auch Newsgroups werden Vernetzt und somit in andere Mailboxen übertragen. Es gibt auch internationale Newsgroups mit der Diskussionssprache Englisch, die um die ganze Welt gehen. Zwischen den Netzen gibt es übrigens noch sogenannte "Gateways", die Knotenpunkte zwischen zwei verschiedenen Netzen darstellen und EMails und Newsgroups untereinander austauschen.

## Files

Aber auch Daten wie Programme und interessante Texte, kann man sich von den Mailboxen holen. Ein solcher Vorgang wird im DFÜ- (Fach)Jargon als "Saugen" oder "Download" bezeichnet (das Gegenteil ist dann "Upload"). Hier findet man auch eine ähnliche Unterteilung, wie bei den Newsgroups. In manchen Mailboxen werden die Dateibereiche einfach nach Thema (linear) geordnet. Manchmal findet man auch eine hierarchische Ordnung, ganz einfach über einen Directory-Baum. Um diese Files nun auf seinem Rechner zu übertragen, braucht man ein Protokollprogramm (siehe Termianlprogramm). Man sagt dem Mailboxrechner den Filenamen, dann startet man das Protokoll, sagt diesem gegebenenfalls auch noch mal, wie das File in unserem Rechner heißen soll und schon geht die Übertragung los.

## Accounts, User, Fächer, Briefkästen, Privilegien

Bei den Mailboxen (auch BBS "Buletin Board System" genannt) kann man sich als Benutzer registrieren lassen. Für die einzelnen Dienstleistungen wird dann u.U. Geld verlangt. Es ist aber auf der untersten Stufe zumeist kostenlos. Wenn man sich als Benutzer einträgt, muß man dem Sysop (Systembetreuer) Name und Adresse hinterlassen, damit er weiß, mit wem er es zu tun hat. Dafür erhält man dann einen persönlichen Briefkasten (Fach) in der Mailbox, in dem dann private Mails landen. Damit niemand unerlaubt Zugriff auf den Briefkasten erhält, ist er durch ein Passwort geschützt, das sie der Mailbox einmal angeben und dann jedesmal wenn Sie das System betreten (einloggen) nach Ihrem Benutzernamen eingeben müssen. Zusätzlich erhalten sie noch verschiedene andere Berechtigungen, mit denen sie z.B. in den Filebereich schauen oder internationale Mails abschicken dürfen (oder auch nicht...). Das ganze nennt man Account.

## Anrufen / Datenaustausch

Eine Mailbox kann im Normalfall nur ein User zu selben Zeit benutzen. Für andere, die es in der Zeit probieren, ist der Telefonanschluß belegt. Es gibt auch Multiline-Mailboxen, bei denen mehrere User anrufen können. Dann kann man auch mit anderen Useren über die Mailbox "chatten" (ratschen). Hat man eine Stammmailbox, schaut man dort ein paarmal im Monat oder vielleicht sogar in der

Woche vorbei und schaut was es für neue Sachen gibt, z.B. prersönliche Briefe und News. Auf diese Weise geschieht also der Datenaustausch mit der Mailbox. Indem nämlich die unterschiedlichen User abwechselnd die Mailbox anrufen, sich das neuste holen und dann ihrerseits wieder Antworten und Mitteilungen abschicken. Im Umgang mit der Forthmailbox sei hier bemerkt, daß man mindestens zweimal in der Woche (besser jeden Tag) anrufen sollte, da man sonst der Diskussion im Forum oder in COMP.LANG.FORTH nicht folgen kann.

## Für den täglichen Gebrauch

Generell sollte man sich eine gewisse Arbeitsweise im Umgang mit Mailboxen angewöhnen. Bei den ersten paar Versuchen mit Mailboxen sollte man sich die Anleitung durchlesen. Die gibt es immer! Sie wird bloß nie beachtet. Wenn man schon Erfahrung mit Mailboxen hat ist dies dann auch nicht mehr so wichtig. Es gibt allerdings einen grundlegenden Unterschied: Kommando oder Menüorientiert. Um Telefonkosten zu sparen sollte man sich nur kurz einloggen, das neuste mitspeichern und sich dann offline (ohne Verbindung zur Mailbox, also im eigenen Rechner) anschauen. Dann schreibt man Antworttexte und bereitet alles vor um diese dann schnell wieder in der Mailbox abzusetzen. Erkundigen Sie sich am besten bei Ihrem Sysop, da die meisten Mailboxen eine solche Vorgehensweise schon unterstützen und sowas auch automatisch ablaufen kann.

## Praxis? Kein Problem!

Wie das in der Praxis aussieht, können Sie gerne bei einem (elektronischen) Besuch in der FORTH-eV Mailbox in München erleben. Die Nummer ist 089/8714548, 2400 Baud 8N1. Wenn Sie einen Account haben möchten, gibt man beim Login (beim einloggen) NEW ein. Das System fragt jetzt nach Ihren persönlichen Daten. Wenn Sie FORTH-eV Mitglied sind, geben Sie bitte dannach ANTRAG als Kommando ein, damit Sie vollen Zugriff erhalten. Unter INFO liegt eine Einführung ins Mailboxprogramm (vgl. VD 2/91) und auch wie man den Datenaustausch mit der Mailbox automatisieren kann, um die Telefonrechnung klein zu halten. Im Filebereich ist unter /DOCS eine komplette Anleitung (ins Deutsche übersetzt) von Waffle, dem Mailboxprogramm, zu finden.

❑

## Neu: Fachgruppe "Roboter"

Rafael Deliano

**Was haben Roboter in der FORTH-Gesellschaft zu suchen ?**

Man kann Roboter mit FORTH programmieren, und manche tun das sogar. Man sollte sich etwas von der vorherrschenden Tunnelsicht - FORTH, FORTH und nichts als FORTH - lösen und mehr auf die Bedürfnisse des Anwenders eingehen. Im Bereich Steuerungen ist Software nur ein Teilproblem neben Hardware und Mechanik. Und Software ist nicht nur Hochsprache, sondern auch Assembler und Multitaskingkerne. Nur wer alle Teilbereiche beherrscht, kann hier Probleme lösen. Anwender müssen Probleme lösen! Roboter sind ein Thema, das über embedded Controller hinausgeht und auch Bereiche wie Sensoren und Motorsteuerungen abdeckt. Funktionen also, die immer enger mit Software verzahnt sind. Hier kann FORTH also in der Anwendung zeigen, wozu es gut ist. Und nur erfolgreiche Anwendungen überzeugen.

**Warum sollte diese Fachgruppe besser funktionieren als die bisherigen ?**

Erstens: Voraussetzung für die wirksame Zusammenarbeit räumlich verteilter Gruppen ist Kommunukation. Ein regelmäßiger Newsletter löst dieses Problem. Selbst wenn er auf 10 bis 20 fotokopierte A4-Seiten beschränkt bleibt, erfüllt er seine Funktion ohne großen Aufwand an Zeit und Geld zu erfordern.

Zweitens: die Gruppe hat ein konkretes Ziel vor Augen: auf der übernächsten Echtzeit eine neue Version des klassischen Micro-Mouse-Labyrinths vorzustellen.

**Was ist das für ein Labyrinth ?**

Autarke, kleine, selbstlernende Roboter versuchen den Weg durch ein Labyrinth zu finden. Derartige Wettbewerbe haben von ca. 1978 bis 1983 öfters stattgefunden.

Autark: der Roboter wird nicht ferngsteuert oder fremdgespeist. Er ist also auf seinen Bordcomputer und Akku angewiesen.

Klein: ca. 15x15cm

Selbstlernend: der Bordcomputer muß seine Fehlversuche analysieren um aus ihnen zu lernen. Automatische Labyrinthlöseverfahren waren in den 50er und 60er Jahren ein Thema der AI-Forschung. Aber heute harmlos genug, um realisierbar zu sein.

Ein besonderes Bonbon: das Labyrinth soll vertikal werden, d.h. an der Wand hängen. Wo es besonders gut sichtbar ist und man mit ihm keine Platzprobleme hat. Das erfordert natürlich sorgfältige Optimierung von Gewicht und Wirkungsgrad der Roboter.

**Was tun Aktive ?**

Sie bauen einen Roboter! Möglichst den, der am schnellsten vom Eingang zum Ausgang findet.

**Aber ist es nicht furchtbar schwierig so einen Roboter zu bauen ?**

Die Mechanik ist relativ einfach zu realisieren, wenn man die Komplexität in Elektronik und Software verlagert. Grundkenntnisse in Sachen Schaltungstechnik und Controller sind allerdings unerläßlich. Wissenslücken zu den speziellen Problemen werden durch den Newsletter geschlossen. Wer keinen Hang zur Hardware hat, kann sich am Labyrinthlösealgorithmus versuchen. Das ist ein Teilproblem, das man voll auf Tischcomputern simulieren kann.

**Interesse ?**

Ausgabe 1 des Newsletters anfordern bei:
Rafael Deliano, Steinbergstr. 37
8034 Germering, Tel. 089/8418317
oder in der Mailbox der FORTH-e.V.:
089/8714548
eine Nachricht an "jrd" schreiben.

# Sprachenstreit in der Bücherecke

von Jörg Staben

Hagelkreuzstr.23, 4010 Hilden, Tel.:02103-240609

Eine etwas andere Besprechung dreier Bücher, die sich mit Programiersprachen befassen.

Nichts macht den Programmierern aller Schattierungen mehr Freude, als der - längst entschiedene - Sprachenstreit. Zeitschriften von C'T an abwärts haben damit ihre Seiten gefüllt, unsinnig war es eh' immer, aber schön war's. *Ulrich Paul* hat in der VD 4/92 mit einer Reihe von "Dogmen" diese Diskussion wieder aufleben lassen und Fragen aufgeworfen, die doch immer wieder faszinieren. Ganz kurz das Ende des Sprachenstreits:

*Die Sprache ist die beste, die SIE! am besten beherrschen!*

## C

Ich hatte irgendwo schon mal das Buch erwähnt, das so richtig Lust auf Programmieren mit Borland C++ macht:

**SECRETS OF THE BORLAND C++ MASTERS**

SAMS Publishing
ISBN 0-672-30137 ca. 90 DM
etwa 700 Seiten, incl. 2 HD-Disk mit allen Quelltexten und etlichen Produkten von Drittanbietern wie Editor, Versions-Kontrollsystem oder einem Zeichenprogramm, das C-Quelltext generiert.

Dieses Buch möchte ich hier im Detail vorstellen; dabei soll die Glossarform zeigen, wo zukünftige Forth-Literatur ansetzen kann. Nach einer praxisnahen Einführung in die optimale Installation und Bedienung des BC++ und seiner integrierten Bedienoberfläche IDE widmet sich das Buch den drei Sprachkonzepten C, C++ und Assembler sowie den Hilfsprogrammen des BC++, dem Präprozessor CPP, der Projektverwaltung MAKE und des Dateisuchers GREP.

Hat man das Kapitel "Power Features of the IDE" verinnerlicht, so wird sofort klar: Es ist **keine** (!) Frage von Monaten, bis C++ Forth voll links überholt! Allein die Projektverwaltung, das MAKE-Programm des BC, wird mit zweiundzwanzig (22!) Befehlswörtern und zwanzig (20!) Compilerdirektiven gesteuert, wozu sich noch weitere zwanzig (20!) Kommandozeilen-Optionen gesellen. Diese mehr als 60 Befehle nicht für den C-Compiler, sondern nur für das MAKE beherrscht man nicht von heute auf morgen.

Daran schließt sich das Einrichten der IDE für externe Programme an; hier müssen etwa dreißig (30!) Makros sinnvoll kombiniert werden. Das Erlernen der Turbo Editor Makro Language TEML wäre dann der nächste Schritt. All' das in ein paar Monaten? - nie!!

Lachen Sie nicht, aber als ich ein bissle Grafik in C machen wollte, dauerte es bald 10 Minuten, bis ich die entsprechende Stelle in der Dokumentation gefunden hatte. Unter den 18 Handbüchern mit ihren 7000 Seiten war mir klar, OBJECT VISION muß die Grafik sein; war's aber nicht; dann bestimmt TURBO VISION. War's auch nicht!? Aber ein kurzer Blick in Band 18, die Dokumentation zur Dokumentation, verriet: BORLAND C++ PROGRAMMIER-HANDBUCH enthüllt die mehr als 70 Grafikfunktionen unter DOS, WINDOWS geht natürlich extra.

Wenn überhaupt, wird Forth frühestens Ende der 90er Jahre auf seinem Weg zur Vollkommenheit von C überholt, wobei noch gar nicht klar ist, wie dieser Weg überhaupt aussieht.

**Versions-Kontrollsysteme (Revision Control Systems)** Kurz gefaßt hält ein RCS alle Änderungen an einem Projekt nach, stellt die jeweils aktuelle Version zur Verfügung, bietet den Zugriff auf beliebige frühere Versionen und gibt eine Übersicht über die einzelnen Entwicklungsschritte des Projektes. Zudem dient ein RCS der Dokumentation, hält den kritischen Fragen während eines Kunden-Auditing zur Qualitätssicherung stand und verhindert, daß Herr Müller die aktuellen Änderungen von Frau Maier überschreibt. Ganz rudimentär gab's sowas mal in Forth, das stamping. Dabei schrieb das System das Namenskürzel des jeweiligen Programmierers und das Datum der letzten Änderung in den Quelltext, aber das war's dann auch. Meiner Meinung nach sollte wir es hier in Forth damit gut sein lassen und den tapferen F-PC-Editor nicht überfrachten. Diejenigen, die z.B. das GNU-RCS über Jahre hinaus entwickelt haben, haben sich bestimmt schon einiges an Gedanken gemacht. Dabei sind dann so tolle Konzepte wie 'reverse deltas' herausgekommen, die in der Dokumentation zum GNU-RCS näher erläutert werden. Das Buch liefert direkt ein ShareWare RCS namens ATTIC mit, das für einzelne Programmierer gut geeignet scheint. Darüberhinaus wird das RCS-System PVCS, eines der bekanntesten Produkte dieser Art, detailliert beschrieben.

**Klassenbibliotheken** Hier wird erstmal ein klares Statement abgegeben: "Der Wechsel ... nach C++ macht aus Ihnen keinen produktiveren Programmierer." Ganz im Gegenteil, die Lernphase und die Projektdauer unter C++ ist deutlich länger - die Vorteile von C++ zeigen sich erst später. Aber zurück zu den Klassenbibliotheken. Da stehen zwei völlig getrennte zur Auswahl: Eine erweiterte Klassenbibliothek auf der Basis von Objekten zur Kompatibilität mit Turbo/BorlandC 2.0 und ein Container-Bibliothek auf der Basis von Templates, BIDS genannt. Zudem findet sich noch irgendwo eine DLL- Bibliothek zur WINDOWS-Unterstützung. Genug gehört? Sie sehen, in den oben er-

wähnten Monaten tut sich in einem C++ so gut wie garnichts, später vielleicht dann mehr, aber das werden wir sehen...

**Grafik und Sound** sind zwei weitere Schwerpunkte, die in aller Ausführlichkeit erläutert werden; und da liegt auch der wirkliche Vorteil eines BorlandC: Wenn die - bestimmt gut gemachte - Dokumentation Fragen offen läßt, gibt es bereits ein, zwei Monate nach Erscheinen der Software gute begleitende Literatur in reichhaltiger Auswahl.

**Sorgfältige Dokumentation, ausführliche Tests und Fehlersuche** nehmen breiten Raum ein, ein Zeichen für die Qualität des Buches. Zugleich gehen die Autoren der Frage nach, wodurch die **Wiederverwendbarkeit** von Programmen eingeschränkt wird. Ursachen für mangelnde **Wiederverwendbarkeit** können eine schlechte Dokumentation, komplizierter Code, unzureichende Lesitungsfähigkeit oder ganz einfach Fehler sein. In der Analyse für das Erstellen von stabilen, zuverlässigen Programmen kommt sogar ein altes Forth-Dogma zu Ehren: "Den Wörtern die richtigen Namen geben", ausführlich erläutert in "Die Kunst der Namensgebung", L.Brodie, IN FORTH DENKEN, S.159ff. Sie sehen schon, ob C oder Forth - die Programmiersprache selbst tritt angesichts der Problemstellungen und ihren Lösungsmöglichkeiten weit in den Hintergrund. Die Autoren propagieren für das Erstellen von guter Software: Von Anfang an testen, oft testen. Zudem geben sie Tips, wie man aussagekräftige Tests schreibt, um die Qualität von Programmen sicherzustellen. An den hier angelegten Maßstäben sieht man deutlich, wie wenig brauchbar die vagen Aussagen über vermeintliche oder wirkliche Fehler im F-PC sind, die immer mal wieder in der VIERTEN DIMENSION erwähnt werden. Nur sorgfältige und ausführliche Tests stellen - beim modularen Programmierkonzept Forth wichtiger denn je - sicher, daß grundlegende Programmfunktionen fehlerfrei sind.

**Optimierung** gehört eng mit dem Testen zusammen; man kann nichts optimieren, was man nicht vorher ausführlich getestet hat. Beides wird auf fast 100 Seiten gezeigt; alles Themen, von denen man in der VIERTEN DIMENSION viel zu selten liest. Aber auch hier sind Werkzeuge wie Profiler und Debugger sowie Techniken wie das Remote Debugging, bei dem zwei Rechner über Kabel verbunden sind, so komplex, daß zur souveränen Handhabung eher Jahre als Monate vergehen. Das mag auch für die Fehlersuche in TSR-Programmen gelten, auch ein Thema in diesen beiden Kapiteln.

**#OUT --- a1**

(Fig) not defined.
(79) not defined.
(83S) not defined.
(F83) numbers of characters sent since last CR.
(F-PC) A variable that holds the column number of the most recent type or emit to the display.

**Implementations:**

MVP: 8088/86: not used.
Fig: 8088/86: not used.
F83: 8088/86: VARIABLE #OUT
F-PC: Same as F83.

**Example: #OUT @ .**

Examine the present value in the variable #OUT and print it.

**Comment:**

In Fig-Forth and MVP the value is referred to with the ideogram OUT and is a User Varaible. In eitehr case, OUT or #OUT, is useful for checking space remaining on a line of output.

Bild 1: So werden hunderte der gebräuchlichsten Forth-Wörter beschrieben.

**Das Zusammenspiel mit Fremdprodukten** wie dem T-ASM, der sich ab v3.1 mit dem Klebe-Etikett "objektorientiert" schmückt, wird ebenso ausführlich besprochen wie auch die **TSR-Programmierung**. Zu beiden Themen gibt es nicht nur eine Einführung in die Grundlagen, sondern auch Verweise auf Drittanbieter wie das TesSeRact Development Team, die ein Standardverfahren für TSR's über den `Int2F` entwickelt haben.

**Internationales Vermarkten von Software** ist ein Thema, das man wirklich selten findet. Seitdem weiß ich, daß WINDOWS die Codepage 1004 verwendet.

**Hochgeschwindigkeitsübertragung über die Serielle Schnittstelle** stellt - oh Neid! - eine Klassenbibliothek zur Verfügung, die mit 115.200 baud Daten über die RS232-Schnittstelle jagt. Gut, daß wenigstens die F-PC-Anwender ein "RS232IB.SEQ" haben...

**Fazit:** Dieses Buch macht deutlich, wo die Vorteile von Konzepten liegen, die am Markt erfolgreich sind: Begleitende und weiterführende Informationen gibt's an jeder Straßenecke - dagegen können wir innerhalb derer, die Forth machen, nur unsere eigene Schaffenskraft setzen! Wir sollten nicht nur modular programmieren, sondern auch modular denken. Indem wir bei unseren Arbeiten immer wieder aufführen, wo andere Mitstreiter am gleichen Thema arbeiten und wie Vorhandenes genutzt werden kann, vergrößern wir unsere Basis und kommen - endlich - weiter. Denn die 90er Jahre im Umfeld der riesigen Microsoft- und Borland-C Compiler wie auch dem Borland-PASCAL mit seinen 6 Compilern bieten den anderen, homogeneren Konzepten eine neue Chance.

## Forth

Merkwürdig - dieses Buch ist erst durch eine Diskussion innerhalb der Forth Gesellschaft wieder in mein Blickfeld gerutscht. Gekauft hatte ich es schon im letzten Winter für ca.150 DM, aber über die Funktion der Wärmedämmung war es in meinem Arbeitszimmer nicht hinausgekommen.

Ich hatte mich der Frage zu stellen, ob es möglich ist, im F-PC ein DOS-Programm aufzurufen. Dies ist in F-

PC sehr einfach, da es dafür ein Wort "SYSCOMMAND gibt. Dieses Wort ist im Vokabular HIDDEN definiert, das während des Compilierens mit ALSO HIDDEN zugänglich gemacht

**All about Forth**

An Annotated Glossary
Glen B. Haydon
Third Edition June 1990;
revised and updated, includes COMMON USAGE, STANDARDS DOCUMENTATION, FOUR IMPLEMENTATIONS

werden muß. Diese erforderliche Vokabularumschaltung ist übrigens keine Eigenheit des F-PC, sondern das durch den Standard Forth83 festgelegte Verhalten von ":". Wenn jemand sowohl die Systemdokumentation als auch den gültigen Sprachstandard kennt, wirft diese Programmieraufgabe keine Schwierigkeiten auf.

In einundeinhalb Kilogramm Papier werden solchen Sachverhalte dokumentiert. Diese Fleißarbeit, auf über 500 Seiten vier verschiedene Forth-Implementationen Wort für Wort nebeneinander zu stellen, ist den tausenden von MVP-Forthlern gewidmet. Hier bemerkt *G.B.Haydon* mit Stolz, daß MVP-Forth seit über 10 Jahren stabil läuft. Ein Zeitraum, von dem Produkte wie MS-Word für Windows nur träumen. IN All About FORTH werden hunderte der gebräuchlichsten Forth-Wörter mit ihrer Definition nebeneinandergestellt, mit einem kommentierten Beispiel und weiteren Hinweisen versehen. Siehe Bild 1 (Seite 33). Und so finden Sie dann jedes einzelne bekanntere Forth-Wort besprochen. Bei BYE wird darauf hingewiesen, daß dieses Wort kein notwendiges Wort in einem Forth ist. Es gibt ja auch Fälle, in denen kein Betriebssystem da ist, zu dem man zurückkehren kann! Bei F-PC wird bemerkt, daß F-PC zusätzlich BYEFUNC ausführt, worin beiliebige Funktionen ausgeführt werden können, vielleicht noch ein abschließendes Löschen des Bildschirms oder Zurückgeben von DOS-Speicher.

Abschließend enthält dieses stark an den großen Brockhaus erinnernde Buch noch alle bisherigen Sprachstandards fig, Forth79 und Forth83. Hier steht also alles drin, was man über Forth als Sprache wissen sollte.

Hinweis: Um der Frage "Wo gibt es das?" nicht nur mit "Bei OFFETE ENTERPRISES" entgegenzutreten, bin ich gerne bereit, mit diesem Buch einen inner-FG Buch-Verleih zu beginnen. Also: Wer's haben will, ruft mich an; dann geht's als Büchersendung 'raus und der nächste meldet sich dann direkt beim aktuellen Leser.

## dB

**Objektorientierte Datenbanken**

John G.Hughes
HANSER Verlag;
ISBN 3-446-16583-5

Wie bitte? Was? Ja, der kleine Mikrokosmos der Programmiersprachen ist schon längst um Sprachen ergänzt worden, die in der kommerziellen EDV bereits an Bedeutung gewonnen haben. Diese werden von einem Werk präsentiert, das mit ca.60 DM noch verhältnismäßig günstig ist, und sehr souverän die verschiedenen aktuellen Konzepte exakt darstellt.

Übrigens, PASCAL und Forth gehören nicht zu diesen aktuellen Konzepten, sondern SMALLTALK, C++, EIFFEL und SQL werden nebeneinander gestellt. Vorher wird noch das gern eingesetzte relationale Datenbankmodell besprochen, bevor an praxisnahen Beispiele objektorientierte Sprachkonzepte mit ihren Stärken und Schwächen dargestellt werden. Plötzlich spielen Begriffe wie 'Integrität', 'Normalform' als anderes Wort für die altbekannte 'schrittweise Verfeinerung' eine Rolle und vor die Einsparung von Taktzyklen hat der Autor die 'Wiederverwendbarkeit' von Problemlösungen und Programmen gestellt. 'Korrektheit' ist ein weiterer Begriff, der dann auch die Nähe zu dem oben besprochen C++ Buch zeigt. Hier sieht man, daß da eine sehr homogene Welt aus C++, objektorientierten Datenbankkonzepten und modernem Software-Engineering am Enstehen ist, die dann wirklich nicht nur Forth links und rechts überholt.

❑

## Buch + Hardware

von Jörg Staben

Da Forth sich in erster Linie im Bereich Messen-Steuern-Regeln zu Hause fühlt, bietet sich der Hinweis auf Literatur zu diesem Thema an. Alle nachfolgenden Bücher enthalten als "letzte Seite" eine Platine, die mit fortschreitenden Kenntnissen immer weiter bestückt wird:

**PC-Bastelbuch**
Kai Hamman
Markt&Technik 1990,
ISBN 3-89090-331-2, ca.98 DM

**Der PC als intelligente Schaltzentrale**
P.Wratil/R.Schmidt
Praxisbuch für zeitorientierte Interfacetechnik.
Markt&Technik 1990,
ISBN 3-89090-651-6, ca. 119 DM

**PC/XT/AT - Messen, Steuern, Regeln**
P.Wratil/R.Schmidt
Markt&Technik 1987,
ISBN 3-89090-477-7, ca. 99 DM

**PC-gestützte Meß- und Regeltechnik**
D.Schulz
Grundlagen und praktische Anwendung
Franzis 1991,
ISBN 3-7723-6675-9, ca. 68 DM

**PC-gesteuerte Meßtechnik**
K.Dembrowski
Entwicklung von Meßsystemen mit Hilfe von Einsteckkarten, der RS232 und der IEC-Schnittstelle.
Markt&Technik 1991,
ISBN 3-89090-958-2, ca. 119 DM

**Mikrocontroller-Entwicklung auf dem PC**
H.-J. Blank
16Bit-Controller selbst programmiert am Beispiel des NEC µPD 78312.
Markt&Technik 1990,
ISBN 3-89090-899-3, ca. 119 DM

**Steuern mit PCs - Messen, Steuern, Regeln**
Nach einem SPS-Programmierkurs mit einer SPS-Simulationssoftware soll man ein Prozeßsimulationsprogramm handhaben können.
INTEREST-Verlag
Best.-Nr.: 6500, ca. 98 DM.

Die vorstehende Liste stellt nur in einigen Fällen eine Beurteilung dar, sondern soll lediglich eine erste Orientierung ermöglichen. Dennoch wurde versucht, die Bücher nach aufsteigendem Schwierigkeitsgrad zu ordnen.

# Zufallsdaten?

## Nichts leichter als das! oder: F-PC in den Händen der Magier

von Jörg Plewe und Jörg Staben

Jörg Plewe: Haarzopfstr. 32, 4330 Mülheim an der Ruhr - Tel.: 0208-497068

Jörg Staben: Am Hagelkreuz 23, 4010 Hilden 10, Tel.: 02103-240609

Es wird eine Möglichkeit vorgestellt, wie man mit F-PC sehr einfach und schnell große Mengen an Zufallsdaten erzeugen kann, um z.B. Datenbankprogramme besser testen zu können.

Irgendwann im Dasein eines jeden Menschen kommt der Punkt, wo sie oder er sich entschließt, das bisherige Leben völlig zu ändern. So reifte auch bei mir der Entschluß, ab heute (wann ist heute?) Ordnung zu halten; Disketten sollten katalogisiert, Texte sortiert - überhaupt sollte alles erfaßt und verwaltet werden!

Was braucht man dafür, wenn man schon einen Rechner besitzt? Forth? Vielleicht - aber eher noch fällt der Blick auf die reißerischen Anzeigen der Firmen Borland und Microsoft, die mit ihren Datenbanken Paradox und FoxPro alle Seligkeiten dieser Programmspezies versprechen! Um sich vor einer eventuellen Enttäuschung nach der Registrierung zu schützen, muß man beide Produkte geprüft haben. Leider kann man eine Datenbank nur mit vielen vielen Datensätzen - so um die 10.000 Stück - testen, die vielleicht jeder noch mehrere Felder enthalten. Nur ist es äußerst mühsam, z.B. 10.000 Datensätze mit je drei Feldern von Hand einzugeben; hier würde ein Datei, in der diese 30.000 Felder nach den Regeln der DBASE-Kunst durch Kommata getrennt drinstehen, ein ganzes Stück weiterhelfen. Eine solche Datei schnell zu erzeugen, damit prahlen immer die Anwender von QuickBASIC, unsere TurboPASCAL-Freunde kämpfen da so niedlich mit ihrem `REWRITE` und `ASSIGN`. Dummerweise beherrsche ich nur mein F-PC-Forth und muß halt damit zurecht kommen.

Tja, nun würde ich hier gerne eine lange Entwicklungsgeschichte beschreiben, aber alles, was der F-PC-Anwender braucht, ist `>B` und das Wissen, daß `1 .R` im F-PC das Ausgabefeld einer Zahl richtig auf den Bildschirm bringt. Geben Sie nach dem Compilieren ein:

```
>b werte
```

Die Datei `BROWSE.PRN`, die Ihnen anschließend im Editor angezeigt wird, ist schon das, was Sie wollen.

Wollen Sie sich zuerst PARADOX zur Brust nehmen, so denken Sie bitte daran, PARADOX für alle Eventualitäten auf den Punkt als Kennung für Fließkommazahlen einzustellen und als Feldtrenner das Komma zu vereinbaren. Nach dem Einlesen in eine systemeigene Tabellenform freut sich PARADOX darauf, Ihnen 10.000 Datensätze mit je drei Feldern zum Spielen anbieten zu können. Dieses Spielen mit den beiden o.g. Datenbanken ergab, daß PARADOX gegenüber FoxProx zwar die gewöhnungsbedürftigere und noch zu verbessernde Bedienoberfläche hat, dafür die eigentlichen Datenbankfunktionen aber leistungsfähiger zu sein scheinen.

❑

P.S. Im F-PC finden Sie statt des hier verwendeten ››RANDOM.SEQ‹‹ Zufallsgenerator ››RAND3.SEQ‹‹ und eine ähnliche Funktion in ››SPLAT.SEQ‹‹.

**Stichworte:**

F-PC
Datenbanken
Paradox / FoxPro
Testdaten

**Auf die schnelle 30.000 Werte**

```
needs pb\rnd\random.seq
\ Zufallsgenerator

1000 Constant op.id
10   Constant typ.id

: werte      ( -- )
             op.id 0 DO
                      typ-id 0 DO
                               j 1 .r ." ,"
                               j 1 .r ." ,"
                               10 rnd .
                               cr
                               LOOP
                      LOOP
                      ;
```

❑

## Brief aus der Provinz
### Ein Streifzug durch die VD

Leserbrief von Friederich Prinz
Homberger Str. 335, 4130 Moers 1,
Tel.: 02841-58398

*Rafaels Deliano*'s Argumentation zu funktionierenden Forthgruppen kann ich nur sehr bedingt nachvollziehen. Schließlich sind die Mitglieder der FG zumindest in einigen Städten durchaus relativ stark konzentriert. Da sollte es wenigstens so viele 'funktionierende' Gruppen geben, wie die Anzahl dieser Konzentrationen ist. Allerdings weiß ich, daß das Funktionieren einer Gruppe von einigen wesentlichen Faktoren abhängig ist, die durch die Mitgliederkonzentration allein nicht gegeben sind. Vielleicht sollte zu diesem Thema einmal etwas in der VD stehen... Dafür stehe ich zu 150% hinter Herrn *Deliano*'s Anregung bezüglich der DFÜ. Das wäre doch was, wenn sich die Forther der FG untereinander 'vernetzten'....

*Rolf Kretzschmar*'s Vorstellung des DISCO habe ich sehnsüchtig erwartet. Das ist einmal mehr 'Forth ganz praktisch'. Ich freue mich auf die hoffentlich reichlich folgenden Beiträge zum DICO, umso mehr, als ich intensiv mit der Moerser Gruppe an und mit dem DISCO arbeiten möchte.

*Ulrich Paul*'s Leserbrief zu den forthigen Dogmen hat mir sehr gefallen. Ich neige eigentlich selbst dazu, mich in meinen Teller zu 'vergraben' und den Rand desselben höchsten als Option zu betrachten. Damit bringen wir FORTH natürlich ebenso wenig weiter, wie mit den immer wieder von mir kritisierten Sauriern. Oft ist Weniger ein wenig MEHR, manchmal kann selbst ZF nicht 'genug' und immer wieder sollten alle Forther bereit sein, sich auf Neues einzulassen. NEU war zum Beispiel im letzten Jahr der 'Pascal-Compiler mit forthigem Vokabular', zu dem ich einen sicherlich nicht uninteressanten Bericht verfaßt hatte, der leider für die VD entschieden zu umfangreich ausgefallen war...

*Gerd Limbach*'s Leserbrief hat mir, selbstverständlich, ganz besonders gur gefallen. Laßt es mich an dieser Stelle noch einmal ausdrücklich sagen: Das ZF (*Tom Zimmer*'s erstes System) gibt's in Moers immer noch !!!umsonst!!! Dazu verschicke ich gerne alles an Kurstexten und Aufsätzen, was in den letzten Jahren hier entstanden ist. (...) Für exakt 0,- DM gibt die Moerser Gruppe ein ForthSystem von *Tom Zimmer* weiter das einfach (fast) allen Ansprüchen gerecht wird.

Mit freundlichen Grüßen vom linken Niederrhein verbleibt

Friederich Prinz

## ANS Forth kommt - Interresse geht?

Leserbrief von Michael Kalus
Plönstr. 24a, 2427 Malente/Gremsmühlen

Dieses ANS Forth scheint einfach immer zu spät zu kommen. Forthentwickler geben wohl schneller neue Dinge in die Welt, als Kommissionen aufarbeiten können. Da müßte man schon die Welt anhalten.

LMI ist inzwischen solider entwickelt, erprobter und reichhaltiger als alles sonst; viele sind damit ungebremst kreativ tätig. In der public domaine hat mit F-PC ein reichhaltiges Forthsystem die Nachfolge des F83 angetreten. Und die Forthchips oder in Forth programmierte Controler gehen munter eigene aufgabengerechte Wege. Und es ist schwerlich möglich, solch divergente Kreise unter einen Hut zu bringen. Standard ist, was zur rechten Zeit zur Verfügung steht - siehe MSDOS.

Was sich in Rostock 1992 beim Forthtreffen ohne großes Aufsehen ergab, scheint sich nicht nur bei uns abzuspielen - Forth ist. Aber nicht ANS, sondern das, was es in der jeweiligen Aufgabe eben sein muß. Mit jeweils eigenen Werkzeugen. Und wer kein passendes Forth hat, macht sich eben eins. Und wer das nicht kann, kann sowieso mit Forth nichts anfangen. Und während noch am ANS Forth gefeilt wird (also dem Compiler), gehen die Forthprogrammierer lustig dazu über, den Bau eines jeweils neuen Compilers bereits als Teil der Applikation zu begreifen. Ja, nicht mal ihr Metacompiler ist ihnen heilig. Forth ist faszinierend vielseitig und es ist eben nicht möglich, etwas sich Wandeldes endgültig zu beschreiben.

Forth tut also einfach, was es soll und verändert sich mit seinen Aufgaben - oder es ist nicht mehr! ANS ist wohl schon nicht mehr. Die Bericht aus

## Schweinekiste

von Jörg Plewe
Haarzopfer Str. 32
4330 Mülheim an der Ruhr - Tel. 0208 / 497068

Das kann dem besten Forthfreak passieren: er muß in C programmieren. Dabei scheint es, dass auch in dieser unglaublich professionellen Programmiersprache selbst Kleinigkeiten zu ungeheuren Sauereien entarten können.

Im Zuge einer wissenschaftlichen Anwendung galt es, eine Folge von sich nicht wiederholenden 16 Bit Indizes zu produzieren, die die niedrigeren Werte anfänglich stärker berücksichtigt. Dazu habe ich die folgende kleine Routinen verfasst.

```
#define ROL(x) ((x & 0x8000) ?   (((x<<1) & 0xffff)|1):((x<<1) & 0xffff))
#define BSHIFT(x,i) ((x & bit[i+1]) ?   (x):((x & ~bit[i]) | bit[i+1]))

int dasschwein()
{
static int bit[16] =
{ 0x1, 0x2, 0x4, 0x8, 0x10, 0x20, 0x40, 0x80,
   0x100, 0x200, 0x400, 0x800,
   0x1000, 0x2000, 0x4000, 0x8000 };

static int lowerbits[16] =
{ 0, 0x1, 0x3, 0x7, 0xf, 0x1f, 0x3f, 0x7f, 0xff,
   0x1ff, 0x3ff, 0x7ff, 0xfff,
   0x1fff, 0x3fff, 0x7fff };

static usigned int schwein=0;
int i, j=0, ret;
   ret = schwein;
   for( i=0; i<15; i++ )
         if( schwein & bit[i] )
               {
               if( ret != (schwein = BSHIFT( schwein, i )) )
                     {
                     schwein &= ~lowerbits[i];
                     schwein |= lowerbits[j];
                     return ret;
                     }
               j++;
               }

   schwein = ROL( schwein );
   while( schwein & bit[0] );
         schwein = ROL( schwein );
   schwein |= bit[0];
   return ret;
}
```

Als ich mir mein Werk danach so ansah, dachte ich prompt an die Schweinekiste. In meinem Kollegenkreis fand ich niemanden, der aus dem Listing die Funktionsweise sofort ersehen konnte, also ging es an die Redaktion.

Der geneigte Leser möge sich nun folgende Fragen stellen:

- ☐ Was macht die Routine?
- ☐ Wie macht sie es?
- ☐ Wie sähe sie wohl in Forth aus?

Ich kann nur hoffen, dass jede mögliche Forth-Variante ein bedeutend schöneres Ergebnis liefert. Oder liegt es am Ende gar nicht an der Programmiersprache, sondern am Problem?

❑

USA jedenfalls sind versiegt. Das Thema ist aus den Forth-Medien verschwunden, nichts kommt mehr über die Boxen zum Thema ANS (Stand 12/92). Obschon die Kommission wohl weiter arbeitet und auch weiter hofft, dereinst einmal ein ANS Forth herauszubringen. Nur - was, wenn sich niemand drum schert? mk.
PS: Letzter Stand ist dpANS-4 (1/93).
(siehe Seite XXX ANS-Forth, der letzte Stand)

## Mögliche Wirkungen des ANS-Forth

Mitch Bradley

P.O.Box 4444 Mountain View,
CA 94040, Feb.93
(Übersetzung: KPS)

*(Auf eine entsprechende Frage von KPS zum ANS-Forth-Stand erhielten wir folgende Antwort:)*

Ob ANS-Forth zu spät kommt, um sich durchzusetzen? Vielleicht.

Ein positiver Effekt könnte sein, daß die Anstengungen für Forth intensiviert werden, weil Anbieter einen Anstoß bekommen, ihre Produkte zu überarbeiten und zu vertreiben. Außerdem wird der neue Standard quasi das Spielfeld ebnen, indem es existierenden Anbietern etwas von ihrem Vorteil gegenüber Neulingen nimmt und so zu einigen neuen Anbietern führen kann.

Dies wiederum kann positive und negative Wirkungen haben. Auf der Plus-Seite steht, daß neue Anbieter oft sehr innovativ sind, weil sie keine Tradition zu bewahren haben, oft recht energisch sind, und bei den Anwendern neue Begeisterung wecken können. Negativ daran ist, daß zu viele neue Anbieter den Markt verunsichern und in so kleine Stücke spalten können, daß niemand von Forth-Produkten leben kann. Außerdem verschwinden die meisten neuen Anbieter erfahrungsgemäß nach drei bis vier Jahren wieder vom Markt, weil so ein Geschäft zu führen doch schwerer ist, als viele glauben.

Wenn ein Anbieter vom Markt verschwindet, schadet das Forth, weil dessen Kunden mit ihren Anwendungen betroffen sind. Solche Erfahrung kann Kunden auf C umschwenken lassen. ANS-Forth kann in dieser Hinsicht nützen, indem es die Varianz zwischen den Forth-Systemen zwar nicht beseitigt, aber verringert. Mindestens so wesentliche Dinge wie File-Zugriff, Fließkomma, Speicher-Verwaltung und Exceptions werden übertragbar sein.

Die existierenden erfolgreichen Anbieter werden zweifellos Bauchweh bekommen wegen der Neulinge - ich auch -, aber ich denke, daß die Old-Timer überleben werden, weil sie auf dem Weg zum Erfolg gelernt haben, sich anzupassen und ihr Geschäft über Wasser zu halten.

Was wir hoffentlich nicht erleben werden, ist eine Flut neuer Public-Domain Implementationen. Es gibt schon zu viele PD Implementationen, und sie ermutigen nicht gerade zu Langzeit-Investitionen. PD-Befürworter sagen, daß PD Implementationen Neulingen den Einstieg erleichtern. Das stimmt zum Teil, aber nach meiner Erfahrung ist für den erfolgreichen Gebrauch einer Programmier-Sprache der Support und die Begleitung wesentlich wichtiger, als mal jemandem eine Implementation zum Spielen zu besorgen. PD-Systeme verderben dermaßen die Preise, daß sich seriöse Anbieter die nötige Programm-Pflege und Anwender-Unterstützung nicht leisten können. Für das Thema Lernen und Lehren können speziellen Low-Cost Versionen und Schul-Discounts angeboten werden, und viele Anbieter tun das auch.

Ein Public-Domain System ist im Grunde ein Ego-Trip für einen Hacker, der zwar seine Arbeit unter's Volk bringen will, aber die harte Arbeit scheut, die Vermarktung und Support erfordern. Ein PD-System ist ein Aufruf der Art: "Schaut mal, wie toll ich programmieren kann, aber verschont mich mit Geschäfts-Überlegungen!" (Manche Benutzer neigen dazu, PD-Systeme wegen des erhältlichen Quellcodes zu bevorzugen, aber der Quellcode ist auch von einigen kommerziellen Systemen zu haben, meinen zum Beispiel, und das nicht unbedingt teuer.)

Mitch Bradley

## Freelancer WANTED

akg Wer möchte unserer US-Schwester, die «4th Dimensions» auswerten und kurze deutsche Abstracts/InhaltsÜbersichten der VD liefern? Mitteilung über eine ständige Mitarbeit bitte an die Redaktion senden.

## Guter Aufnehmer gesucht

von Rolf Kretzschmar

Wer entwickelt einen Eingabe-Recorder, mit dem typische Anfängerfehler dingfest gemacht werden können? Könner können zeigen was, sie können!

Wir wollen eine Zeitschrift für den Forth-Anwender sein. Ausdrücklich betrachte ich dabei den Forther, der lediglich in seiner Freizeit und nur für den Hausgebrauch (also: just for fun) Programmierung betreibt, als Forth-Anwender. Nun lernt man bekanntlich den Umgang mit einer Programmiersprache am Besten, indem man schaut, wie ein anderer bei der Lösung des eigenen Problems vorgegangen ist. Dabei ist wirklich wichtig, daß das Problem des anderen auch das eigene ist.

Ich will nun den Versuch machen, ein Problem zu konstruieren, von dem ich hoffe, daß zahlreiche VD-Leser sagen: "Ja, das interessiert mich!" Und Forth-geschrittene sollen sich herausgefordert sehen, der Redaktion zu diesem Problem eine wirklich saubere Lösung zuzusenden (mit für Anfänger verständlichen Anmerkungen in dem ihnen genehmen Forth-System oder in der von ihnen bevorzugten Programmiersprache!).

Es **soll** gar kein Artikel im üblichen Sinn sein; der ausführlich kommentierte Quellcode reicht. Den Artikel dazu liefere ich ja schon! Eingehende Lösungen werden getestet (hoffentlich finde ich zu jedem System einen Tester....) und - wenn sie funktionieren und ordentlich kommentiert sind - in der VD veröffentlicht.

Anfänger sind aufgerufen, einen Lösungsweg zu suchen und die leidvollen Erlebnisse - etwa bei der Suche nach brauchbaren Worten im F-PC - zu skizzieren, um uns zu schreiben, in welchem Stadium man die Klamotten in die Ecke geworfen hat! (Woher mein Pessimismus kommt? Na, lesen Sie denn nicht die Leserbriefe der VD?!)

Natürlich wird wieder ein "Sieger" ermittelt und mit Lob überschüttet werden! Jury sind die Redaktion und die Direktoren der FG. Der Kriterienkatalog liegt noch nicht fest. Auf alle Fälle gehen die Verständlichkeit von Code und Kommentaren und das Eingangsdatum bei der Redaktion in die Bewertung ein.

So, nun das Problem:

Anfänger machen beim Programmieren Anfängerfehler. Könner können oft garnicht verstehen, wo man überhaupt Fehler machen kann. Anfänger können Könnern nicht erklären, welche Fehler sie machen. Wenn sie das können, sind sie auch Könner. Wie kann Könner Anfänger verstehen? Nun, indem er sich neben ihn setzt, und ihm bei jedem Fehler sagt, was er falsch macht. Wohl dem Anfänger, der solch einen Lehrer findet! Wohl dem Könner, der dazu die Nerven und/oder die Zeit hat. Dabei könnte es so einfach sein! Die Eingaben des Anfängers brauchten nur in einem Log-File festgehalten zu werden. Diesen Log-File kann man ausdrucken, um darüber zu reden, oder auch wieder in Forth laden. Soviel zum Thema: Sittlicher Nährwert.

Die Wettbewerbsaufgabe lautet:

**Entwickeln Sie in einem Ihnen genehmen Programmiersystem ein Softwaremodul, das es gestattet, die Eingaben (ohne zugehörige Ausgaben oder Systemmeldungen) beim Programmieren im Direktmodus in einem File aufzuzeichnen. Dazu soll nach jeder Betätigung der ENTER-Taste dieses Log-File aktualisiert werden. Das Log-file soll als Textdatei für den Menschen lesbar sein. Zudem soll das File wieder - wie z.B. eine Batch-Datei - geladen werden können. Am Anfang und am Ende des Log-Files soll das aktuelle Datum und die Uhrzeit eingetragen werden.**

Wir sind - mit unseren newcomers - sehr gespannt auf die Resonanz. Endlich lernen wir auf diese Weise den Umgang mit dem File-System. Endlich sehen wir, wie einfach das doch im System xyz geht. Endlich wird uns in den Kommentaren erläutert, warum ein bestimmter Weg beschritten, eine bestimmte Funktion genutzt wurde. Endlich!

## Multi Drop (RS485 ...)
### eine Bitte an unsere Profis

akg Feldbusse waren in Forth realisiert, als es den Begriff 'Feldbus' noch gar nicht gab (Beispiel: Flughafen Gepäckverteilung). 'Tausend' Lösungen wurden realisiert, keine veröffentlicht, wenn ich mal W. Schnitter's ARCnet Lösung in Forth ausnehme. Liegt nicht irgendwo eine 'verstaubte' Lösung, die etwas überarbeitet/dokumentiert in der VD erscheinen könnte und (zumindest privat) allgemein genutzt werden dürfte? Das wäre schön.

## Bug, Prompt, EC-Geldautomat und Whiskey:

akg Stellen Sie sich vor, da ist ein Eingabe-Menü. Es wird eine Zahleneingabe verlangt. Der Dezimalpunkt wird ausgewertet, der Benutzer kann aber Zahlen auch ohne Punkt eingeben. Er gibt 123.45 ein (DPL enthält 2). Beim nächsten Prompt gibt er 4 ein. (DPL enthält immer noch 2). Was ist geschehn? -1=$FFFF wäre richtig.

Die Eingabe wurde mit `QUIT | QUERY | INTERPRET | FIND` und `NUMBER` ausgewertet. Sind z.B.: -2 bis 4 als Literale in dem ForthSystem definiert, weil das 'Speed' bringen kann, so schlägt `FIND` anstelle von `NUMBER` zu und der Inhalt von DPL bleibt bestehen. So werden dann aus der 4 z.B. 2631.44. Warum wird ein Forth mit solchen Literalen eigendlich nicht in EC-Bargeld-Automaten eingebaut, zumindest dort wo ich abhebe?

Aber..., warum erscheinen denn da nun 2631.44? Und warum würde das höchstwahrscheinlich doch nicht so reibungslos funktionieren?

Wir verlosen weder Whiskey noch Schampus, aber vielleicht (be)schreiben Sie uns die noch fehlenden Erklärungen !?!?

Lösung

Vor der Abfrage DPL auf -1 setzen.

## Inserentenverzeichnis



## Der INT-Spion — Buchbesprechung von Arndt Klingelnberg

**Undocumented DOS**
**Schulman, u.a.**
Addison-Wesley-Verlag, Bonn
engl., Softcover, 2*1,2MB, 694 Seiten
ISBN 0-201-57064-5, DM86...99

Wer all die hilfreichen Betriebssystem Service Funktionen nutzen will, die MSdos nun mal hat, und wer eben nicht gegenüber den großen Firmen wie beispielsweise Borland, Lotus oder MicroSoft selbst völlig hinterherhinken will, muß auch die von MicroSoft verschwiegenen oder sagen wir besser 'offiziell nicht dokumentierten' Funktionen kennen. Da gibt es eine 'List of Lists', die den Zugang zu wichtigen Informationen ermöglicht, oder wir lernen logische Laufwerke anzulegen. Stabile und funktionsreiche TSRs sind ein wichtiges Thema. Vielleicht wollen wir einfachst mit int $2E eine Funktion über Command.com starten, z.B. ein Batchfile direkt aus Forth heraus über diese 'backdoor'. Das ist zwar ein Klassiker (Ray Duncan, Dr. Dobbs, 1986 dec), aber wir finden es weder in 'Tischer 3' noch in Ray Duncans Büchern von MicroSoft Press.

Allein die HyperTextDatenbank ist wertvoll, zumindest vom Inhalt her, ansonsten verwöhnt Norton Guides oder !Tech-Help .

Locker interessant geschrieben, in der Darstellung nicht so gut wie z.B. Tischer PCintern 3.0 oder Ray Duncan 'Advanced MSDOS'. Ein Muß für PC-Systemprogrammierung, wo das unter Forth doch Spaß macht.

Mit Beispielprogrammen, einem Interrupt-Spion und einer Hypertext Datenbank auf Diskette.

## Das ist das Letzte — von Rolf Kretzschmar

### Schamlos...

...hängt sich der *Darius Glasig Verlag, Geilenkirchen-München* an den Erfolgskurs der VD an und bringt zum 1.04.1993 das Forth-Magazin *Easy FORTHen* heraus. Zwar sind wir sicher, daß der VD nicht so leicht das Wasser abgegraben werden kann, doch dazu werden wir zusätzliche Anstrengungen machen müssen. So wirbt der Verlag in der Vorankündigung mit einigen Fietschers, die uns doch zu denken geben: Den Autoren wird z.B. garantiert, daß die eingesandten Texte nicht von spitzen Klammern (<,>) befreit werden! Angeblich soll sogar die drucktechnische Meisterleistung gelungen sein, Photos so widerzugeben, daß Personen erkannt werden können!! Und -das ist der Gipfel der Dreistigkeit- das Heft soll *innerhalb* des aufgedruckten Quartals die Leser erreichen. In der VD-Redaktion wird z.Z. eine Strategie gegen derartige unlautere Methoden erarbeitet.

### Schluß...

...macht *Tom Zimmer* nun endlich mit der steigenden Flut an F-PC Derivaten. *Jörg Staben* ist autorisiert, die *F-PC Version 5.40356js,42* am 04/01/1993 offiziell für den deutschsprachigen Raum zu vermarkten. Neben einigen lange erhofften Änderungen (z.B. können deutsche Umlaute nun sehr schnell über die Tastenkombination: ESC+rechte Pfeiltaste+O+E für Ö eingegeben werden; durch Eingabe des Wortes PRINZ werden automatisch alle Hilfe- und Mustercodedateien gelöscht, danach meldet sich das System mit ZMF: Zugeschnittenes Moers Forth) hat dieses F-PC einen AKG-Schutzmechanismus (Any Korrektion Goof) in Form eines gebändigten Virus: Mit jeder Betätigung der Enter-Taste geht dieses 111-byte-kleine Stück Code auf die Suche nach möglichen Veränderungen im Quellcode des F-PC. Wird AKG fündig, ertönt die Melodie "Üb' immer Treu und Redlichkeit.." und danach ist der alte Zustand wieder hergestellt. Genial! Mit Eintreffen dieser Nachricht verabschiedete sich ein Redakteur der VD mit den Worten: "Eigentlich wollte ich immer schon am großen C spielen.." Gegen SAFU-MFD kann man/fra ab 93apr01 über das Schicksal dieses armen Menschen mehr erfahren. Halten Sie auf alle Fälle auch Ausschau nach Inseraten für ein C++akg!

### Bedenken...

...hat *GreenPeace* gegen die Vermarktung eines Hardwareproduktes der Computerindustrie: PLCC-Fassungen mit eingebauten Pull-Up-Widerständen für spezielle Mikrocontroller. Um die Fassung den unterschiedlichen µP's und Bedürfnissen anzupassen, können die nicht benötigten Widerstände durch Anlegen einer konstanten Wechselspannung von 230V verdampft werden. Ein Spezialkabel mit zwei blanken Spitzen und einem Schuko-Stecker wird vom Hersteller (nur gegen Vorkasse) geliefert. Probleme für die Umwelt sieht *GreenPeace* in den giftigen Dämpfen, die beim Entfernen der Widerstände entweichen. Die Firma APERILO-Plast hat sich darauf hin bereit erklärt, jede Packung (zusätzlich zum obligatorischen Grünen Punkt) mit folgendem Aufdruck zu versehen: "Achtung! Das Einatmen dieser Fassung kann Ihrer Gesundheit abträglich sein. Bezüglich der Nebenwirkungen lesen Sie bitte die Packungsbeilage!"

# Fortsetzung des Listings auf Seite 12: Ein File DUMPen (Wolf-Helge Neumann)

## Screen #2

```
\ Variablen, Konstanten...                                  14apr92wn

Variable LeseHandle                      \ Handle der LeseDatei
Variable SchreibHandle                   \ Handle der SchreibDatei
Variable Länge                           \ Länge der LeseDatei
Variable Zähler
Variable <AsciiPlace>
Variable <OffsetPlace>
Variable <Control>

&16  2+  8 +  1 +  1 2 + &16 *  +  2 +   Constant Pufferlänge
 \ 16 Stellen ASCII, 2 Leerstellen, 8 Stellen Offset, 1 Leer-
 \ stelle, 16 mal 1 Leerstelle + 2 Stellen HEX, + CR LF

Create ZeilenPuffer   Pufferlänge allot
```

## Screen #3

```
\ Fehlermeldungen                                           14apr92wn

: ?Fehler:Öffnen   ( n -- )
  0< IF  1 0" [1][Fehler beim Öffnen der Datei][Abbruch]"
         form_alert abort THEN ;

: Fehler:Platz
  1 0" [1][Uff dr Diskett isch|net gnug Platz!][Scheißele!]"
  form_alert abort ;
```

## Screen #4

```
\ Pfade initialisieren, Lese-Datei auswählen                14apr92wn

: PfadInit   ( Pfad$ -- )
  count dup inpath c! inpath 1+ 0place ;

: DateiInit   ( Datei$ -- )
  count dup insel c! insel 1+ 0place ;

: ReadInit    ( -- )   " P:\*.*" PfadInit  " " DateiInit ;
: WriteInit   ( -- )   " P:\*.*" PfadInit ;

: @LeseDatei   ( -- )   ReadInit fileselect ?dup
  IF openfile dup ?Fehler:Öffnen
  0 over 0 fseek drop ( Lesezeiger wieder auf Anfang setzen)
  LeseHandle ! Länge ! THEN ;
```

## Screen #5

```
\ DUMP-Datei anlegen                                        14apr92wn

: -endung   ( addr -- addr )
  dup count ascii . -scan nip ?dup IF 1- over c! THEN ;
  \ schneidet die Datei-Endung .* ab, sofern vorhanden

: +endung   ( Datei Endung -- addr )   0 here c! here $sum !
  swap -endung count $add  " ." count $add  count $add   here ;
  \ alle Strings sind counted strings
  \ ersetzt Datei-Endung bzw. fügt sie an -> Ergebnis in addr

: ÖffneSchreibDatei   ( -- )   WriteInit insel " DMP" +endung
  dup c@ insel c!  count insel 1+ 0place
  path+file fcreate dup SchreibHandle ! ?Fehler:Öffnen ;
  \ ersetzt die Endung der in insel übergebenen Datei durch DMP
  \ und erzeugt die neue Datei
```

## Screen #6

```
\ Dateien schließen, Schreiben in ZeilenPuffer              14apr92wn
: schließeDateien   ( -- )
  LeseHandle @ fclose SchreibHandle @ fclose  2drop ;
: ?ascii   ( ascii -- ascii' )   <Control> @
  IF dup bl < IF drop ascii . THEN THEN ;
  \ <Control> = true: Steuerzeichen werden durch Punkte ersetzt
: AsciiPlace   ( ascii -- )   <AsciiPlace> @
  IF   ?ascii Zähler @ ZeilenPuffer + c!
       ( <AsciiPlace> = true: ASCII-Werte werden an die
       entsprechenden Stellen in den Zeilenpuffer geschrieben)
  ELSE drop THEN ;
: OffsetPlace   ( -- )   <OffsetPlace> @
  IF base push hex   LeseHandle @ position? &16 -
     extend <# # # # # # # # # #>
     [ ZeilenPuffer &18 + ] literal swap move THEN ;
  \ <OffsetPlace> = true: Offset mit 8 Stellen in Zeilenpuffer
```

## Screen #7

```
\ Schreiben in ZeilenPuffer                                 14apr92wn

: HexPlace   ( Wert -- Wert )   base push hex
  dup extend <# # # #>
  [ Zeilenpuffer &28 + ] literal Zähler @ 3 * + swap move ;
  \ schreibt den Hex-Dump in den Zeilenpuffer

: 16lesen   ( -- addr n )   pad dup &16 LeseHandle @ fread ;

: ZeilenPufferInit   ( -- )   ZeilenPuffer PufferLänge bl fill ;

: >ZeilenPuffer   ( addr n -- )
  Zähler off   ZeilenPufferInit   OffsetPlace
  bounds DO  i c@ HexPlace AsciiPlace  1 Zähler +!  LOOP ;
  \ baut den Zeilenpuffer für eine Zeile auf
```

## Screen #8

```
\ Zeile schreiben, lesen+schreiben                          14apr92wn

: +0D0A   ( addr count -- addr count+2 )
  2dup + $0D over c! $0A swap 1+ c!   2+ ;

: schreibeZeile   ( -- )
  Zeilenpuffer Pufferlänge -trailing  ( Leerzeichen am Ende weg)
  <AsciiPlace> @ not IF bl skip THEN ( Leerzeichen am Anfang nur
                  dann abschneiden, wenn kein ASCII gewünscht!)
  +0D0A dup >r  SchreibHandle @ fwrite r> -
  IF schließeDateien Fehler:Platz THEN ;

: lesen+schreiben   ( -- )   16lesen >ZeilenPuffer schreibeZeile ;

: LeseEOF?   ( -- f )   0 LeseHandle @ 1 fseek Länge @ = ;
```

## Screen #9

```
\ FileDump + Optionen                                       14apr92wn

: -ascii     ( -- )   <AsciiPlace> off ;          \ ohne ASCII
: +ascii     ( -- )   <AsciiPlace> on ;            \ mit ASCII

: -Offset    ( -- )   <OffsetPlace> off ;        \ ohne Offset
: +Offset    ( -- )   <OffsetPlace> on ;          \ mit Offset

: -Control   ( -- )   <Control> on ;       \ ohne Steuerzeichen
: +Control   ( -- )   <Control> off ;       \ mit Steuerzeichen

+Ascii  +Offset  -Control                          \ Voreinstellung

: filedump   ( -- )   @LeseDatei ÖffneSchreibDatei
  BEGIN lesen+schreiben  LeseEOF? UNTIL  schließeDateien ;
  \ liest und schreibt, bis Dateiende erreicht ist
```

❑

## Forth-Gruppen regional

**W-1000 Berlin**
Claus Vogt
Tel. 0+30 - 2 16 89 38 p
Treffen: nach Absprache

**W-4XXX Rhein-Ruhr**
Jörg Plewe
Tel. 0+208 - 49 70 68 p
Treffen: jeden 1. Samstag im Monat im S-Bahnhof Derendorf
Münsterstr. 199, 4000 Düsseldorf

**W-4130 Moers**
Friederich Prinz
Tel. 0+2841 - 5 83 98 p
Treffen: jeden Samstag 14:00
Arbeitslosenzentrum, Donaustr. 1
4130 Moers

**W-51XX Aachen**
Klaus Schleisiek,
Tel. 0+241 - 87 34 62 gaf
Treffen: jeden 1. Montag im Monat als Gruppe des Computer-Club der RWTH, Seminargebäude,
Raum 214, Nähe Wüllnerstraße
5100 Aachen

**W-6100 Darmstadt**
Andreas Soeder
Tel. 0+6257 - 27 44

**W-6800 Mannheim**
Thomas Prinz
Tel. 0+6271 - 28 30 p
Ewald Rieger
Tel. 0+6239 - 86 32 p
Treffen: jeden 1. Mittwoch im Monat, Vereinslokal Segelverein Mannheim e.V., Flugplatz
6800 Mannheim-Neuostheim

## µP - Controller Verleih

Rafael Deliano
Steinbergstr. 37,
W-8034 Germering
Tel.:0+89 - 8 41 83 17

## Gruppengründungen, Kontakte

### Regional

**W-7000 Stuttgart**
Wolf-Helge Neumann
Tel. 0+711- 8 87 26 38 p

### Fachbezogen

**8051 ... (Forth statt Basic,e-FORTH)**
Thomas Prinz
Tel. 0+6271 - 28 30 p

## Forth-Hilfe für Ratsuchende:

### Forth allgemein

Jörg Plewe
Tel. 0+208 - 42 35 14 p
plewe@mpi-dortmund.mpg.de

Karl Schroer
Tel. 0+2845 - 2 89 51 p

Jörg Staben
Tel. 0+2103 - 24 06 09 p

### Spezielle Fachgebiete

**Anfänger und Wiedereinsteiger**
Gerd Limbach
Tel. 0+2051 - 25 51 12 p
Mo. + Di. 20:00 - 22:00

**32FORTH (Atari)**
Rainer Aumiller
Tel. 0+89 - 6 70 83 55 gp

**FORTHchips (FRP1600, RTX, Novix ...)**
Klaus Schleisiek-Kern
Tel. 0+40 - 2 20 25 39 gp

**F-PC & tCOM, ASYST (Meßtechnik), embedded controller (H8/5xx//TDS2020, 8051 ... eFORTH...)**
Arndt Klingelnberg
Tel. 0+2404 - 6 16 48 agp

**Gleitkomma-Arithmetik**
Andreas Döring
Tel. 0+721- 59 39 35 p

**HS/Forth (Harvard Softworks)**
Wigand Gawenda
Tel. 0+30- 44 69 41 p

**KI (Künstliche Intelligenz), OOF (Object Oriented Forth)**
Ulrich Hoffmann
Tel. 0+431 - 80 12 14 p

**Unterricht mit FORTH**
Rolf Kretzschmar
Tel./Fax 0+2401 - 8 88 91 ap

**UUCP (FORTH ... per eMAIL)**
Andreas Jennen
Tel. 0+30 - 3 96 52 27 ap

**volksFORTH/ultraFORTH/RTX-FG-Forth/Super8Forth**
Klaus Kohl,
Tel. 0+8233 - 3 05 24 p
Fax. 0+8233 - 99 71 f

## Forth-Vertrieb

**volksFORTH (PC, ST, C64) / F68K (68000) / ...**
Johannes Teich
Ettaler-Mandl-Weg 19
W-8110 Murnau
Tel. 0+8841 - 29 43
jgt@bbs.forth-ev.de

## Forum: Forth-Mailbox

**Forth-Mailbox**
Jens Wilke (SysOp)
Tel. 0+89 - 8 71 43 52 p
Mailbox 0+89 - 8 71 45 48,
300-2400 baud (8N1)

## Hinweise

### Zu den Telefonnummern

f == FAX
a == Anrufbeantworter, hier können Sie Ihren Ansprechpartner eventuell vorinformieren, erwarten Sie bitte keinen (kostspieligen) Rückruf
g == geschäftlich, zu erreichen innerhalb typischer Arbeitszeiten
p == privat, zu erreichen außerhalb typischer Arbeitszeiten

Die Adressen des Forth e.V. (Forth Büro) und der Redaktion / Anzeigenverwaltung finden Sie im Impressum.

# FORTH-SYSTEME GMBH

Postfach 1103,
7814 Breisach

Telefon (0 76 67) 5 51
Telefax (0 76 67) 5 55

**Telefon Schweiz:**
**(055) 53 65 55**


## UR/FORTH

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

## LMI FORTH-83 Metacompiler

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

- 8086/8088
- Z80/HD64180
- 8080/8085
- 68000
- Z8
- 1802
- 6809
- 8096/97
- 78310
- 8031/32/535
- 6303
- 6502
- V25
- 68HC11
- RTX 2000
- 80C166

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

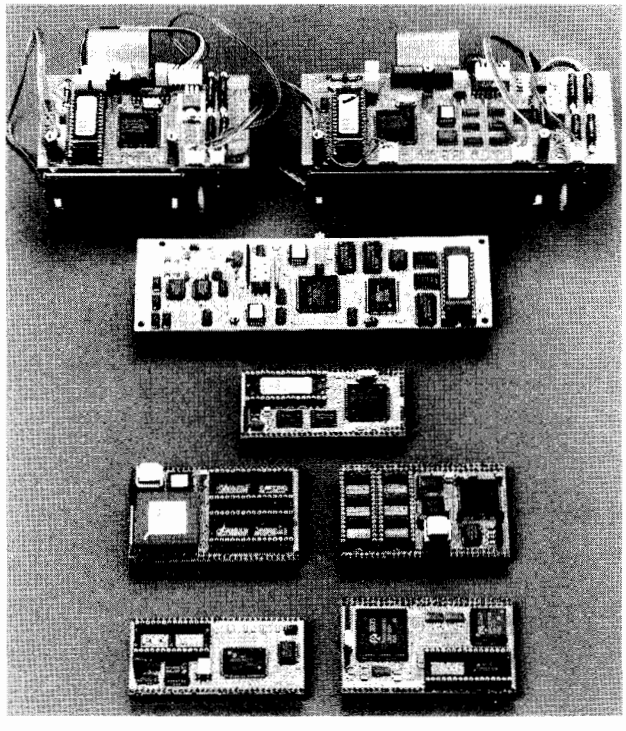
## WinFORTH

- UR/FORTH kompatibel
- Windows Funktionen werden voll unterstützt
- Erweiterte Debugging-Hilfsmittel
- Online Windows Hilfe
- Coprozessor Unterstützung möglich
- Software-Gleitkomma-Paket
- Viele Beispielprogramme
- Upgrades von UR/FORTH Systemen auf WinFORTH sind preisgünstig zu erhalten

## SRS II

- Serieller ROM Emulator
- Unterstützung folgender Bausteine:
  27256, 27512, 271000, 27010, 27020, 27040
- Minimale Zugriffszeit 100 ns
- Maximale Baudrate 115.200 bits/s
- Highspeed Interface als Option
- Gleichzeitiger Zugriff von Host und Zielprozessor
- Zusätzliche serielle Schnittstelle über den ROM-Sockel
- Intel-Hex, Motorola-S oder ASCII/binär Formate werden unterstützt
- Der SRS II ist nur 157 x 94 x 36 mm groß
- SRS63 kompatibel

## ModuNORM



CPU-Steck-Module im Scheckkartenformat:

- 8 Bit z.B. 6303
- 16 Bit z.B. V25
- Highspeed RTX-2000/1
- 80C166
- Softwareunterstützung durch SwissFORTH
- Thermodrucker und Controller
- LCD Grafik-Controller

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