VIERTE DIMENSION

4/1992

8. Jahrgang 1992 4. Quartal DM 7,50

Disco

mail boxen
dBase lesen
Beitrag zahlen!
Beiträge schicken!

eFORTH für 68HC11
EuroFORTH '92
TIMER-TICK
TSR-Uhr
NUN

FORTH MAGAZIN

Organ der FORTH-Gesellschaft e.V.

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

Die Beule

von Rolf Kretzschmar

Mir geht es wie Ihnen, ich mag Beulen nicht. Lassen Sie mich dennoch über eine Beule berichten, die nur Gutes erwarten läßt, wenn sie erst einmal geöffnet ist.

Erfahrung habe ich mit mehreren Beulenarten: Fast wäre mir im Urlaub ein Reifen am Wohnmobil geplatzt, hätte ich nicht rechtzeitig -wenn auch eher zufällig- die dicke Beule entdeckt. Beulen können also Signalwirkung haben und rechtzeitiges Handeln verhindert schlimme Folgen.

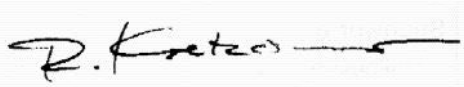
Die Beulen, die ich mir mehr als einmal am Kopf durch Unachtsamkeit zuzog, waren schmerzhaft und z.T. entstellend. Diese Art Beulen scheinen unvermeidlich; die Absichtserklärung, sich nie wieder das edelste der Körperteile an unverhofft vorstehenden Dingen zu stoßen, scheint müßig. Und wer sich noch nie eine Beule zuzog , war nicht neugierig genug. Neugierde ist aber der Humus, auf dem Kreativität gedeiht. In unbekannten Räumen wächst leider mit zunehmender Neugier die Wahrscheinlichkeit, sich eine Beule zu holen. Wer alle Widrigkeiten abwägen will, kommt nicht schnell voran. Also: Beulen können Ausdruck neugieriger Kreativität sein.

Forther holen sich ständig Beulen. Was turnen sie auch auf ungeschützten Seilen umher?! Es gibt doch Sprachen, die liefern mehr Halt (Mengen an Literatur)! Es gibt doch Sprachen, die haben Netze und doppelte Böden ("access denied")! Es gibt Sprachen, die sind derart verbreitet, daß man sich auf einschlägigen Partys in der Syntax dieser Sprachen verständigen, ja angeregt unterhalten kann! Forth? Nie gehört! Zeigt her Eure Forth-Beulen...

Da ist noch eine andere Forth-Beule. Sie wächst unaufhaltsam. Es drückt auf der einen, es zieht auf der anderen Seite. Ich hoffe, daß sie bald platzen wird! Wie sie aussieht? Nun, sie hat die Form von kleinen Kreuzchen und sie ist zu sehen im Stapel der eingegangenen Fragebögen (Leider bisher nur 32! Also bitte...!). Die Kreuzchen betreffen "Forth und Hardware-Anwendungen". Die Menge macht die Beule: 90% der Fragebogenzurücksender wünschen Artikel über Forth auf µP-Controllern, bzw. zur Hardwaresteuerung über den PC. 80% arbeiten schon mit einem oder mehreren (!) µP-Controller(n). Wo sind die Artikel dazu? Helft, diese Beule zum Platzen zu bringen!

Die Redaktion hat schon mal zum Operationsbesteck gegriffen:
In diesem Heft bieten wir - mit der Vorstellung von DisCo - eine attraktive Einstiegs-Droge für Programmierung in Forth, die sich auch noch sehen lassen kann, und es gibt Hinweise auf Testgeräte, die von FG-Mitgliedern kostenlos ausgeliehen werden können.

Euer



Mailboxing

von Rafael Deliano

Über Nutzen und Nutzer von Forth-Mailboxen

Das Problem

Wenn man sich die ca. 400 Mitglieder der FORTH e.V. gleichmäßig über die Fläche der Bundesrepublik verteilt vorstellt, wird offensichtlich, daß eine funktionsfähige lokale Gruppe eine statistische Anomalität sein muß. Es sind dafür einfach nicht genügend Leute auf einem Fleck beieinander. Man fragt sich auch, wie eine Fachgruppe effektiv arbeiten soll. Briefchen schreiben und beantworten ist mit der gelben Post so quälend langsam, daß jeder Elan daran schnell zugrunde geht.

Die Lösung

Es gibt ein Medium, das schnell und relativ billig ist, verschicken einer Nachricht an alle oder gezielt an einen Empfänger erlaubt. Wo man die Nachricht holt, wenn man Zeit für sie hat und nicht wie beim Telefon im falschen Augenblick gestört wird. Wo es keine Komplikationen beim Verschicken von Sourcecode mit unterschiedlichen Diskettenformaten gibt. Eine Medium, das ideal für Leute ist, die gerne mit Computern arbeiten. Und damit ideal für die Kommunikation innerhalb der FORTH e.V. ist: EMAIL.

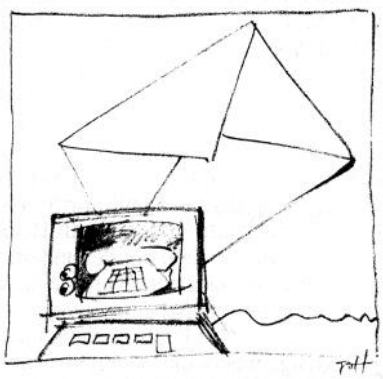
Der Nutzen ist nicht nur für den Verein gegeben, sondern auch für die einzelnen Benutzer. Eine in der Mailbox gestellte Frage wendet sich an alle Benutzer und hat damit beste Chancen auf eine qualifizierte Antwort. Sei es eine technische Frage, oder Sie brauchen die aktuelle Anschrift einer bestimmten Bezugsquelle, oder Sie wollen die Meinung zu einem Produkt hören, bevor sie es kaufen. Sie bekommen nicht nur eine informative Antwort, sondern auch eine aktuelle Antwort. Tatsachen veralten heute schnell. Bücher und Zeitschriften mit ihnen.

Stichworte

Mailboxen
Datentransfer
Kosten

Die Box

Nach einigen Anlaufschwierigkeiten ist die Mailbox der FORTH e.V. voll einsatzfähig. In `Comp.Lang.Forth` treffen laufend die neuesten Nachrichten aus den USA ein. Und auch die Diskussion im deutschen Forum ist in Gang gekommen. Trotzdem beteiligen sich noch



zuwenige Mitglieder. Die Redaktion der VD ist nur über Umwege erreichbar. Das Direktorium wird selten gesichted. Die Idee, daß Fachgruppen über EMAIL organsiert werden könnten, scheint noch weit entfernt.

Das Nachrichtenaufkommen in der Münchner Mailbox (`Comp.Lang.FORTH&FORUM`) rechfertigt derzeit ein bis zwei Anrufe pro Woche. Dann sind praktisch immer neue Nachrichten eingetroffen. `Download` ist im Klartext oder in gepackter Form möglich. Gepackt wird mit LZH, was für Texte sehr effizient ist. Leider ist nicht für jeden Computer der nötige Entpacker verfügbar. Für Amiga z.B. noch nicht und für Oldtimer wie C64 und CPC wird er wahrscheinlich nie verfügbar. `Download` im Klartext (ungepackt) bei 2400 Baud dauert dann für die Nachrichten einer Woche ca. 8 Minuten.

Womit wir bei den laufenden Kosten wären. Mit Ortsgesprächen kann man sich generell nicht ruinieren. Bei Ferngesprächen in Zone 3 klingelt die Kasse der Telekom jedoch. Aber nur vier- bis fünfmal im Monat.

Die Netze

Können die Kosten für die Benutzer vermindert werden, indem man den Inhalt der Münchner Box ins Z- und Mausnetz einspeist ? Eine Box dieser Netze könnte man eventuell zum Ortstarif erreichen. Im Gegensatz zur Münchner Box, die von der FG finanziert wird, erhebt die Mehrzahl der Boxen jedoch einen satten Mitgliedsbeitrag. Irgendwie muß die Grundgebühr der Post und die Gebühren, die durch den automatischen Fileaustausch entstehen, ja wieder hereingeholt werden. Das untergräbt den Kostenvorteil. Außerdem übernehmen die einzelnen Boxen ungern Files, die wenig Benutzer, aber laufenden Datentransfer haben. Nicht jede Box eines Netzes führt deshalb die FORTH-Nachrichten. Also muß lokal erst die kritische Masse an aktiven Mitgliedern erreicht werden, bevor eine externe Mailbox sinnvoll nutzbar ist.

Generell ist die Präsenz von FORTH in den Netzen natürlich wünschenswert, weil dies eine gute und billige Werbung für die FORTH e.V. darstellt. Und der Transfer von der Münchner Box in die Netze ist deshalb auch sinnvoll. Er ändert nur nichts an der Tatsache, daß in der Anfangsphase noch einge Zeit der Schwerpunkt der Aktivitäten in der zentralen Box der FORTH e.V. konzentriert werden muß.

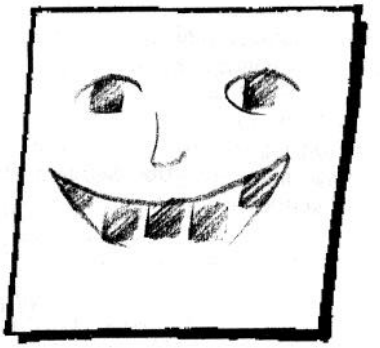
Erst wenn die Teilnehmerzahl wesentlich höher ist, werden die Netze wichtig.

❑

Inhalt

DisCo
Ein Lichtspiel

von Rolf Kretzschmar



"Dieses Ding hätten wir vor fünf Jahren schon haben müssen!" sagte mir mit vorwurfsvollem Unterton ein passionierter Forther aus Moers. Wer 1991 auf der Forth-Tagung in Frankfurt war, kennt es schon. Dem Rest der vd-lesenden Forther wird es nun vorgestellt: Das Lampenfeld, mit dem Programmierenlernen Spaß macht.

Ich weiß nicht, ob Sie sich vorstellen können, wieviele Programmierprobleme man an einem 8X8-Lampenfeld demonstrieren kann. Mir jedenfalls sind die Themen nicht ausgegangen, und ich unterrichte Forth und allgemeine Programmierung mit dem Ding schon seit einigen Jahren in der Berufs-, Berufsfach-, Fachober- und Technikerschule (siehe auch: *Wie das Licht aufging* S.7)

Die geplante Artikelserie über das Lampenfeld wendet sich am Anfang - also in diesem Heft- hauptsächlich an Leser, die Forth anderen vermitteln wollen. Zum zweiten sind die erfahrenen Programmierer angesprochen, denen es Spaß macht, eigene Ideen zu diesem Thema beizusteuern. Profitieren sollen später vor allem die Forth-Einsteiger und Forth-Anfänger.

Um Anregungen zu geben, stelle ich Ihnen zunächst eine Handvoll Aufgaben vor, indem ich mich am didaktischen Gerüst meiner Kurse orientiere. Der Text ist durchsetzt mit Hinweisen für Leute, die Forth unterrichten wollen. Im Anschluß daran stelle ich einige komplexere Anwendungen des Lampenfeldes vor, die auch für den realen Einsatz an Mikrocontrollern geeignet scheinen. Den Abschluß bildet ein Appell an alle interessierten Forther, sich am Projekt DisCo als einer systemunabhängigen (!) Plattform zu beteiligen. Könner haben Gelegenheit, interessante Aufgaben zu präsentieren und für Anfänger verständlich darzustellen. Wenn dieser Appell nicht ungehört verhallt, werden wir in den Folgeheften eine DisCo-Ecke einrichten.

Es geht los mit ganz einfachen Problemen der direkten, interaktiven Steuerung (Sorry, aber Pascal- und C-Programmierer steigen besser weiter unten wieder ein!). Mit Hilfe der Programmierumgebung DisCo (Display-Computer), einem Vokabular unter Forth, lernen die Schüler den ersten Umgang mit Forth-Worten und welche Wirkungen fehlende Leerzeichen und falsche Schreibweisen haben können. Wenn dann mit Sätzen wie

```
_weise ("zeilenweise")
      0 an 7 an
|weise ("spaltenweise")
      0 an 7 an
.weise ("punktweise")
2 5 an 5 5 an 2 2 an
3 2 an 4 2 an 5 3 an
```

ein Gesicht auf dem Lampenfeld erscheint, ist der Umgang mit den x y Koordinaten (kartesisch, x von 0 bis 7, y von 0 bis 7) schnell gelernt. Denn: Fehler werden sofort sichtbar und können z.B. mit `5 3 AUS 5 2 AN` korrigiert werden. Die ersten Klagen über wunde Finger beim zweiten oder dritten Malauftrag sollte Anlaß genug sein, den Begriff Programmierung einzuführen. Meine Definition lautet: *Programmieren heißt, dem Computer etwas beibringen, was er vorher nicht konnte!* Und wenn nach der Eingabe des Wortes `GESICHT` das Gesicht erscheint, während kurz vorher noch `GESICHT <- WHAT ?` oder ähnliches die Antwort war, dann glauben die Schüler Ihnen diese Definition auch. Aus didaktischen Gründen sollten Sie zum Schluß ein Ding programmieren lassen, dessen Einzelteile in späteren Aufgabenstellungen wieder auftauchen. Ich habe erlebt, daß Schüler in diesem Fall von sich aus auf die Idee kamen, Worte aus anderen Worten zusammenzusetzen. Das gehört dann zu den Sternstunden im Leben eines Lehrers. In jedem Fall sind Ihnen die Schüler dankbar, wenn sie Prinzip und Nutzen der Modularisierung derart anschaulich darstellen.[1)]

1) Mein Tip: Lassen Sie zunächst drei Dinger mit gemeinsamen Teilen unmodular programmieren, um nach Bekanntgabe der Modularisierung zeigen zu können, wie kurz und elegant danach der Quellcode aussieht!
Beispiel: `: Gesicht Kopf Augen Mund ;`
Zu welchem Zeitpunkt Sie den Editor einführen, kann individuell verschieden sein. Auf keinen Fall sollten sie die Einführung in die Programmierung und die Einführung in die Benutzung eines Editors gleichzeitig vornehmen. Sie konfrontieren die Schüler mit zwei Problemkreisen und vermindern mit Sicherheit den Lernerfolg! Meistens führe ich vorher schon den Forth-Editor als exemplarisches Beispiel für die Textverarbeitung ein, und erst wenn ich sicher sein kann, daß die Schüler den Umgang damit hinreichend beherrschen, gehe ich zur Lampenfeld-Programmierung über.

Für Schüler ist die Anwendung der Postfix-Notation bis hierher überhaupt kein Problem. Sie lernen spielend damit umzugehen, nehmen sie als Regel im Spiel ganz selbstverständlich an. Erst wenn plötzlich die arithmetischen Operatoren eingeführt werden müssen, kommt man als Lehrer in Erläuterungszwang. Bei der hier vorgestellten Wahl der Reihenfolge (erst Lampen-Operatoren, dann arithmetische und logische Operatoren) ist die Hürde aber deutlich geringer. Wer Forth-Unterricht mit der Erläuterung des Stapels beginnt, darf sich über ablehnende Haltungen nicht wundern.

Zur Einführung der Steuerstrukturen beginne ich mit dem DO...LOOP, weil sich dessen Einsatz bei der Lampenfeldprogrammierung sehr schnell durch kompakteren Code bezahlt macht. Für die Bilddiagonale gibt es kein Wort in der DisCo. Lassen Sie diese Programmieren, weil Sie z.B. ein großes X aus den beiden Diagonalen zeichnen lassen wollen, so fällt den Schülern sofort die typische Struktur der Parameter auf.

```
: DIA/
      .WEISE
      0 0 AN
      1 1 AN
      2 2 AN
      ...
      7 7 AN
      ;

: DIA\
      .WEISE
      0 7 AN
      1 6 AN
      2 5 AN
      ...
      7 0 AN
      ;
```

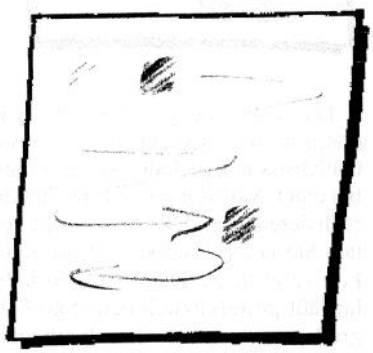
Bei der notwendigen Parameterberechnung bei `DIA\` wird erstmals die Postfix- mit der traditionellen Notation verglichen. Der Stapel wird aber immer noch nicht ins Spiel gebracht. Er ist für diese Problemstellung nicht von Belang und damit unnötiger Ballast. Das Prinzip ist: *Eine Spielregel sollte erst dann erklärt werden, wenn deren Verwendung notwendig wird.*

Je nachdem, welche Vorkenntnisse Ihre Schüler haben, können Sie an dieser Stelle die Themenbereiche: Daten und Information, Zahlensysteme oder logische Operatoren einführen. Ihr Lampenfeld hilft ihnen auch hier bei der Visualisierung: Bitmuster und deren logische Verknüpfungen können sichtbar gemacht werden, das Lampenfeld gar als Abbild eines 8-Bit Stapels genutzt werden. Lassen Sie die Schüler die Bitmuster der Zahlen von 1 bis 8 in das Lampenfeld programmieren und Sie werden mit Vergnügen feststellen, daß einige Schüler von der Regelmäßigkeit der Struktur überrascht sind. Die wenigsten Schüler hatten danach Probleme, in kürzester Zeit alle Kombinationen einer vorgegebenern Datenwortlänge zu notieren.

An dieser Stelle möchte ich den Sprung in anspruchsvollere Erlebnisräume der Programmierung machen. Vorher muß ich aber noch erwähnen, daß die bisher skizzierte Einführung in die Programmierung auf einem simulierten Lampenfeld auf dem Bildschirm durchgeführt wird. Das muß nicht sein. Die Motivation bei den Schülern kann noch gesteigert werden, wenn man ein *reales Lampenfeld aus Leuchtdioden* für jeden Schülerarbeitsplatz hat. Solch ein Lampenfeld existiert und es hat sich bewährt. Es trägt außer den LEDs lediglich ein IC und kann über die Centronix-Schnittstelle eines PC angesteuert werden. Ich werde darauf zurückkommen, wenn ich die Absichten der Redaktion bezüglich dieses Lampenfeldes erläutern werden.

Zufall

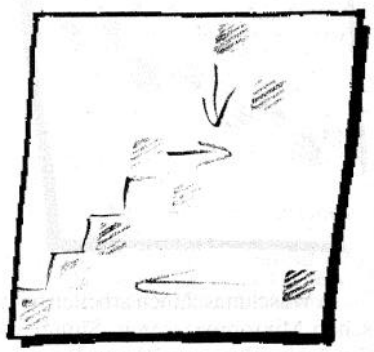
Einen Zufallszahlengenerator können sie für viele Aufgabenstellungen



mit dem Lampenfeld gebrauchen. Lassen Sie z.B. zufällig eine der 64 Lampen einschalten, und stellen Sie als Programmieraufgabe, daß ein Punkt solange auf einem vorgegebenen Weg vorwärts geht, bis er auf diese zufällig eingeschaltete Lampe trifft. Danach soll der Punkt den selben Weg bis zum Ausgangspunkt zurückgehen. (Bild1, Beispiel x) In DisCo können Sie die Zustände der Opjekte (Punkt, Zeile, Spalte oder Bild) vergleichen, um vom Ergebnis des Vergleiches Abbruchbedingungen usw. abzuleiten. Für einzelne Punkte gebe ich das Wort AN? (x y -- f), das im Fall der eingeschalteten Lampe das true-flag hinterläßt. Mit Aufgaben dieser Art läßt sich der Umgang mit Fallentscheidungen trainieren.

Sortieren

Zu den Aufgaben der Informatik an Schulen gehören die Sortieralgorith-

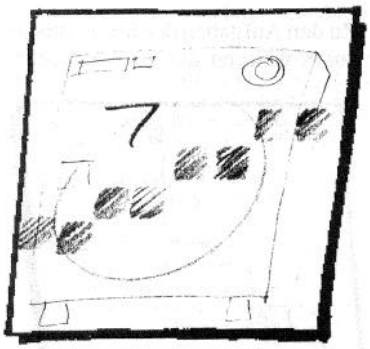


men. Ich glaube nicht, daß ich Ihnen schildern muß, wie schön Sie dazu das Lampenfeld verwenden können. Die Zeiten messen zu lassen, die die verschiedenen Algorithmen an unterschiedlichen Datensätzen verbrauchen und die Ergebnisse als Zahlen am Bildschirm auszugeben, ist eine Sache. Zu sehen, welche typischen Spuren die einzelnen Algorithmen auf dem Lampenfeld hinterlassen, zu sehen, wie die Blasen (Bubbles) bei Bubble-Sort aufsteigen, ist eine andere, eine einprägsame Sache. Schüler erinnern sich: "Das war doch der Sortierer, der...".

Labyrinth

Selbst ein Labyrinth läßt sich auf dem 8X8-Feld realisieren. Die zufällige Generierung unterschiedlicher Labyrinthe dürfte schon nicht mehr zu den trivialen Aufgaben zählen, wenn man sinnvolle Lösungen verlangt. Sinnvoll ist ein Labyrinth nur, wenn es alternative Wege gibt, die in Sackgassen enden. Andere Lösungen lassen sonst nur eine Art Hindernisrennen zu. Aber auch das könnte schon wieder zu einer neuen Aufgabe führen. Ich denke, als Aufgabe für Schüler reicht es schon, wenn sie versuchen einen Punkt möglichst schnell das vorgegebene Ziel (blinkender Punkt) zu erreichen.

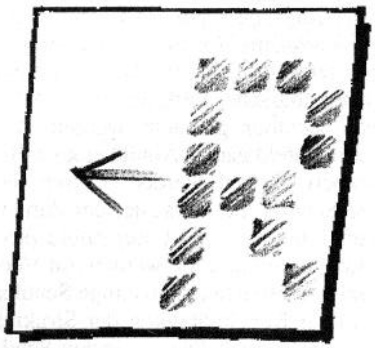
Waschmaschine



In Waschmaschinen arbeiten heute schon Mikroprozessoren. Simulieren Sie doch solch eine Waschmaschine, indem Sie für jeden Prozeßzustand ein typisches, bewegtes Bild programmieren lassen. Es gibt ja Waschmaschinen, die nicht das typische Fenster haben. In diesem Fall könnte man mit dem Lampenfeld die Illusion der Zustände (Füllen, Waschen, Spülen, Heizen und Schleudern) vermitteln.

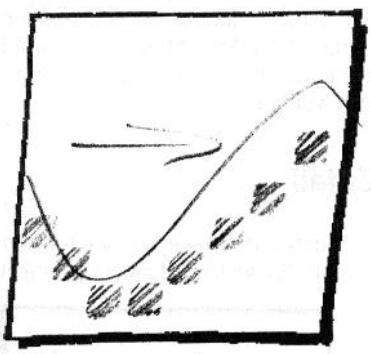
Text

Zwar kann man nicht mit den großen Anzeigetafeln konkurrieren, was die Lesbarkeit von Text angeht, aber es klappt besser als man denkt: Text läßt sich lesen, wenn man mit der richtigen Geschwindigkeit die Buchstaben von rechts nach links durch das



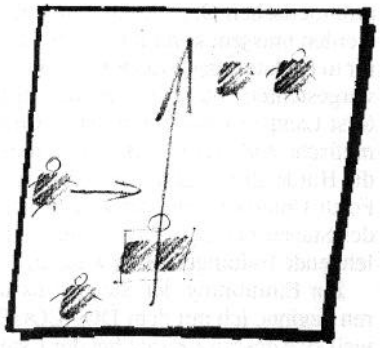
8X8-Lampenfeld wandern läßt. Zur Generierung fester Bilder (Bildkonstanten) sind in DisCo Worte vorgesehen. Ebenso für das Schiften und Rollen. Wie wäre es mit einer kleinen Wetterstation, die bei Bedarf die Windrichtung als Bild darstellt, die momentane Temperatur als Text ausgibt oder die letzten 8 Temperaturwerte in Form eines groben Histogramms anzeigt. Für die Darstellung von Tendenzen fällt Ihnen sicher auch noch etwas ein. Das Ganze betreiben Sie natürlich nicht mehr am PC, sondern an einem kleinen Mikrocontroller.

Oszilloskop



Die Darstellung typischer Spannungsarten (Rechteck-, Sinus-Gleichspannung) läßt sich grob auch bei einer Auflösung von 8X8 Punkten realisieren. Auf dem Bildschirm können Sie ja das Lampenfeld für diesen Fall vergrößern. Das DisCo-Vokabular läßt grundsätzlich beliebige Feldgrößen zu. Geplant ist auch ein reales Lampenfeld, das aneinandergereiht werden kann; so daß auch größere reale Lampenfelder benutzt werden können.

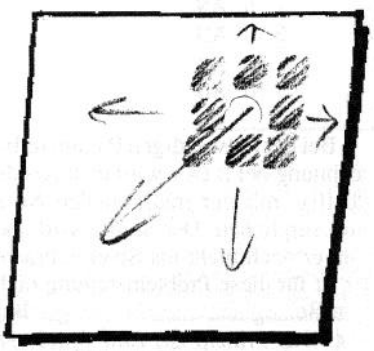
Aufzug



Lassen Sie einen Lampenpunkt als Aufzug fungieren. Die Anzahl der Lampen, die zufällig eingeschaltet sind, können die wartenden Personen auf vier Etagen symbolisieren. Wer schafft es, ein Programm zu schreiben, das den Aufzug möglichst effektiv steuert ("Alle Personen der Etagen 1, 2 und 3 wollen ins Parterre; die, die dort warten, wollen ins Obergeschoß. Der Aufzug trägt nur drei Personen.")

Einstellhilfe

Als Funkamateur ist mir noch die folgende Anwendung eingefallen: Zur Optimierung bestimmter technischer



Einrichtungen ist es oft nicht wichtig, zu wissen, welchen Wert eine Einstellgröße hat, sondern in welche Richtung man sie verändern muß, um optimale Bedingungen zu schaffen

(Antennenabstimmung). Dazu ist es notwendig, sich ein sinnvolles Bild für Einstellung ist optimal auszudenken. Wir wollen eine stromsparende Version wählen und bestimmen, daß in diesem Fall die vier Lampen in der Mitte angehen sollen. Nun können mindestens vier Parameter dieses Bild verändern. An der Art der Veränderung soll man intuitiv den Grad der Abweichung erkennen können, so daß man gezielt reagieren kann. Der erste Parameter verschiebt die Lage der leuchtenden Lampen horizontal. Der zweite vertikal. Ein dritter vergrößert das Lichtquadrat ohne Lageveränderung. Wenn ein vierter Parameter benötigt wird, kann das Quadrat in zwei Richtungen zu einem Rechteck verzogen werden.

Als Anwendung für eine solche Anzeige ist auch ein Autofeedback-Gerät denkbar: die vier Parameter könnten Pulsfrequenz, Blutdruck, Hautwiderstand, Atemfrequenz usw. sein. Als Referenz gibt man eine gewünschte Einstellung vor, und der Proband versucht seine autogenen Körperfunktionen so zu beeinflussen, daß das kleinste Quadrat im Zentrum des Lampenfeldes aufleuchtet.

Damit möchte ich meine Ideenbörse schließen und zum Abschluß noch einmal die Ziele des Projektes hervorheben:

Das Projekt Lampenfeldprogrammierung soll:

- Hilfen für den Programmiersprachenunterricht bieten
- ein einfach zu handhabendes Instrument zum Üben und Lernen einer Programmiersprache (in unserem Fall natürlich Forth) bieten,
- eine Plattform für interessante Programmierkonzepte (systemunabhängig!!) darstellen,
- Anregungen für eigene Experimente und deren Diskussion in der VD bieten,
- eine erste, attraktive Anwendung für Mikrocontroller zeigen und deren Programmierung unterstützen,
- einen Pool hübscher und vielleicht auch nützlicher Anwendungen sammeln.

Die Attraktivität des Projektes hängt von der Resonanz ab, die die

Wie das Licht aufging

Eine Rückschau auf die Entwicklung des DisCo

von R. Kretzschmar

Erfahrung.
Es hat mich als Schüler immer genervt, wenn ein Lehrer Beispielaufgaben zum aktuellen Stoff mit Sachen aus anderen Fächern verknüpfte, die gerade aktuell waren. In vielen dieser Situationen kämpfte ich mit einer Hydra: Die Sache, um die es ging, hatte ich noch nicht verstanden und nun sollte dieses nicht vorhandene Wissen Gegenstand einer neuen Problemstellung sein. Ich erinnere mich noch sehr gut, daß ich in solchen Momenten in eine Art Panik geraten konnte, die meine Aufnahmefähigkeit stark einschränkte. Wahrscheinlich hatten sich diese Lehrer sogar besondere Mühe gegeben. Sowas nannte und nennt man fächerübergreifenden Unterricht! Ich versichere Ihnen, die Theorien dazu sind zahlreich und prächtig.... Erschwerend konnte noch hinzukommen, daß mich das einbezogene Fach überhaupt nicht interessierte. Folge: die Motivation sank auf Null.

Seitenwechsel.
Mit diesen Erinnerungen im Hinterkopf wollte ich nun, da ich auf der anderen Seite des Lehrerpultes stand, die Grundlagen der Programmierung nicht an Beispielen der Mathematik unterrichten.
Welches Modell, welches Material war aber dann besser geeignet? Es mußte so einfach sein, daß jeder es kannte oder sofort begriff. Der Kenntnisstand aller Schüler über die Sache sollte gleich sein, egal welche Schule(n) vorher besucht worden war. Bei aller Einfachheit sollte die Sache doch variabel genug sein, um auch komplexere Aufgabenstellungen zuzulassen. Ach ja, und dann sollte die Beschäftigung damit auch noch Spaß machen. Unmöglich! werden Sie sagen.

Lichtblick.
Wie ich schließlich auf ein Feld von Lampen kam, weiß ich nicht mehr genau, aber damals waren bei den Schülern jene geschmackvollen Geräte für Musikanlagen in Mode, die in Abhängigkeit von der Tonhöhe und der Lautstärke die Helligkeit von drei farbigen Lampen steuerte. Ein Schüler fragte mich damals, wie man die Elektronik umbauen könnte, um das Verhalten etwas zu beeinflussen. Damit war das gesuchte Ding gefunden. Ich begann DisCo (DisplayComputer) zu entwickeln. Das war im Herbst 1985. Auf den Bildschirmen der 10 APPLE][unserer Schule gingen die Lampen an. Einfache, quadratische Blöcke konnten von den Schülern ein- und ausgeschaltet werden. Natürlich mit Forth. Damals noch mit dem holländischen FysForth, dessen Editor ich vollständig eingedeutscht hatte.

Experimente.
Was die Steuersprache (Forth-Worte) betraf, die ich den Schülern anbot, machte ich zu dieser Zeit zahlreiche Experimente: eine Klasse arbeitete mit einer mnemonischen Sprache, deren Worte nicht mehr als drei Zeichen hatten. Beispiel

```
1 2 LON (" Lampe 1 2 an")
3 ZOF ( "alle Lampen in Zeile 3 aus")
```

Eine zweite Klasse bekam eine sehr geschwätzige (redundante) Sprache vorgesetzt. In dieser Sprachumgebung konnte man Sätze bilden wie:

```
LAMPE DARUNTER EINSCHALTEN
ZUR LAMPE DARÜBER GEHEN
```

und man kam ohne Parameter aus. Doppelpunkt und Semikolon hießen übrigens `LERNE:` und `;LERNENDE`. Sie werden es kaum glauben, aber es gab Schüler, die diese Schreibweise beibehielten, als längst die Abkürzungen bekannt waren! Eine dritte Klasse durfte sich mit einer Syntax anzufreunden versuchen, wie sie in der CNC- Technik bis heute akzeptiert wird:

```
G10 ( 1,2) = "Lampe 1,2 an"
G11 (1,2) = "Lampe 1,2 aus"
```

Es zeigte sich, daß zwar mit zunehmender Redundanz, bzw. zunehmender Natürlichsprachigkeit die lernschwächeren Schüler profitierten, dagegen ließ die Lernleistung der "besseren" Schüler recht schnell nach. Vermutlich wurden sie durch die langen Eingaben gelangweilt.

Konsolidierung.
Seit etwa vier Jahren arbeite ich nun mit einem Wortschatz, der für mich ein Optimum an Verständlichkeit, Einfachheit und Nutzbarkeit für den Unterricht darstellt. Für die Vorstellung in der VD hat *Klaus-Peter Schleisiek* die Software noch einmal gründlich überarbeitet und mit zahlreichen Ideen versorgt. (Siehe Artikel *DisCo mit System* auf Seite 9) Um das Grundkonzept noch besser zu unterstützen, sollte eine Lehr- und Lernumgebung geschaffen werden, die für Anfänger und Fortgeschrittene gleichermaßen attraktiv ist. Eingesetzt wird DisCo in Berufsschulklassen, in der Fachoberschule und in Technikerklassen. Noch nie hat sich ein Schüler über einen Mangel an Seriösität bei den Aufgabenstellungen beklagt. Erst wenn ich sehe, daß die Schüler ein Programmierproblem in der Lampenfeldumgebung völlig beherrschen, wage auch ich den geforderten Praxisbezug. Oft genug muß ich dann mit Defiziten bei den Schülern (oder auch bei mir) in diesem Bereich kämpfen und meistens atmen die Schüler wieder frei durch, wenn wir zu unserem vertrauten Lampenfeld zurückkehren.

Hardware.
Doch zurück zur Entwicklungsgeschichte. Nun hatte ich zwar die Simulation eines Lampenfeldes auf dem Bildschirm, aber eine einzige reale rote Lampe, die durch den Computer ein und ausgeschaltet werden konnnte, würde ein vielfaches an Interesse bei den Schülern wecken, da war ich sicher. Welches Maß an Motivation würde dann erst ein reales 8X8-Lampenfeld bei Schülern hervorrufen?! Also baute ich den Prototyp eines Lampenfeldes mit 64 Leuchtdioden. Über die Centronix-Schnittstelle wurde dieses Lampenfeld wie ein Drucker angesprochen: jedes eintreffende Byte schob sich spaltenweise in die 8x8-Matrix und das Bitmuster wurde so sichtbar. Auf diese Weise konnten leicht (übri-

gens aus jeder Programmierumgebung) ganze Bilder und Laufschriften realisiert werden. Daß es funktionierte, hatte ich mit dem Prototypen bewiesen. Aber woher sollte ich mindestens zehn dieser Geräte bekommen? Der Aufwand für den Selbstbau war erheblich, so daß die Realisation mit Schülern nicht möglich war. Doch auch in diesem Fall kam mir Klaus-Peter -wie so oft- mit Rat und vor allem mit Tat zu Hilfe: Er entwarf ein Lampenfeld, das mit nur einem IC auskommt. Dieses konnte ich dann leicht mit Schülern nachbauen. Der Erfolg war: Schüler wollten solch ein Lampenfeld haben, um damit zuhause zu spielen und um Freunden endlich mal zu zeigen, daß Computer auch *was Nützliches* machen können....

Na KlaRo!
Von der Faszination, die dieses schlichte Instrument auf die Schüler ausübte, waren Klaus und ich dann doch überrascht. Wir beschlossen, dieses Konzept zu einer Art Lehrsystem (KlaRo: Klaus-Rolf) auszubauen. Von Anfang an hatten wir vorgesehen, daß das reale Lampenfeld über die VG-Leiste an einen Mikrocontroler (MiniBee, Firma D. Brühl) angeschlossen werden konnte. Damit war die Möglichkeit geschaffen, die Einführung in die Programmierung eines Mikroprozessors mit dem beliebten Lampenfeld als Zielmaschine zu realisieren. Muß noch erwähnt werden, daß die MiniBee (RSC-) Forth versteht? Aber selbst, wenn ich in entsprechenden Klassen Assembler unterrichten wollte, müßte ich mich nicht aus der vertrauten Umgebung entfernen.

Zukunftsmusik.
Inzwischen hat Klaus eine Variante zur *MiniBee* entworfen, den *Heinz65*, und gemeinsam wollen wir versuchen, das KlaRo-System (PC+Lampenfeld+DisCo+Teach/Software) zur Alltags- und Schultauglichkeit zu verhelfen.

❑

Redaktion von den VD-Lesern erfährt. An unserer Schule wurden von Schülern einige 8x8-LED-Felder in halber Europakarten-Größe gebaut. Der Entwurf und das Layout stammen von Klaus-P. Schleisiek (Aachen). Er legt gerade eine kleine Serie davon auf; wer Interesse hat, sollte sich bei ihm melden.

Das Software-Paket ist erhältlich:

- in der Mailbox der Forth-Gesellschaft , München
- bei den Autoren gegen SAFU-MFD (Selbst Adressierten Freiumschlag -Mit Formatierter Diskette).

❑

Buchbesprechung

F-PC 3.5 Technical Reference Manual

Dr. C. H. Ting
2nd Edition 1989 englisch,
269 Seiten
Offete Enterprises,
San Mateo, USA
(1306 South B Street, CA 94402)
no. 1008: $30(+$9 AirMail)

akg. F-PC 3.50 war das erste System, in das ich voll und tief einsteigen konnte, OHNE ein Buch dazu zu haben und ohne auch nur eine einzige Anleitung auszudrucken. Trotzdem, man/fra will im Bus oder im Liegetuhl nicht unbedingt laptoppen; da ist gedrucktes geeigneter. Zudem steht in diesem Buch gerade auch das, was *Tom Zimmer* nicht dokumentiert hat, und es steht das gemeinsam geordnet, was logisch zusammen gehört. Ein Buch muß VorwärtsReferenzen nicht vermeiden. *Ting* steht der Didaktik offensichtlich nicht fern. Vielleicht ist es auch natürlicher 'chinese wisdom' oder sein spezieller Stil, den ich gern mag und hinter dem sich sehr viel Background-Wissen verbirgt. (Er sitzt ja wohl auch mittendrin in der 4th-Szene).

v.3.50 ist zwar das "Offical Release F-PC", aber für manche schon der Schnee von gestern. Und obwohl Tom Zimmer mehr und mehr hinzuschafft, bleibt das Technical Reference Manual eine Referenz. Wer etwas tiefer einsteigen will, sollte es haben. Hier endlich fand ich Bestätigung und Zusatz-'aha'-Effekte, als ich z.B. AT begreifen wollte. Meine Fragen hier und da ergaben sonst nur Achselzukken.

Gravierende Änderungen, wie das Pointer-Memory-Konzept, fehlen hier. Auch wird man/fra weder in F-PC, selbst noch hier z.B. auf \CHARS gestuppst. Aber ich wurde mir z.B. im Umgang mit `PLACE` , `""` (besser: `X"`) und `; CODE` sicherer: drei interessante Wörter die *Tom Zimmer* nie selbst verwendet(e).

❑

Seminar

FORTH und FORTH-Prozessoren für Realtime-Aufgaben

Einen Lehrgang über FORTH und FORTH-Prozessoren mit praktischen Übungen veranstaltet die Technische Akademie Esslingen vom 15. bis 17. Februar 1993. Unter der Leitung von *Dr. Konrad Koller* von der Zentralabteilung Forschung und Entwicklung der SIEMENS AG bekommt der Teilnehmer einen Überblick über die Programmierumgebung FORTH und die heute auf dem Markt erhältlichen Realtime-Prozessoren auf der Basis von FORTH.

Die Veranstaltung richtet sich in erster Linie an Software-Entwickler, die sich grundlegend über Eigenschaften und Möglichkeiten von FORTH informieren wollen, an Mitarbeiter von Fachabteilungen, die sich mit Problemlösungen in der Steuer-, Regel- und Meßtechnik befassen, sowie an Anwender von Mikroprozessoren, die auch bei zeitkritischen bzw. maschinennahen Problemen nie wieder in Assembler programmieren wollen.

Interessenten richten ihre Anmeldung an die Technische Akademie Esslingen, Postfach 1265, 7302 Ostfildern, Tel. 0711/34008-23, FAX 0711/34008-27. Wegen der begrenzten Teilnehmerzahl wird eine frühzeitige Anmeldung empfohlen.

❑

DisCo
mit System

von Klaus-Peter Schleisiek

Die Unterrichts-Hilfe für Digital-Technik und Programmier-Spielwiese nach dem Motto: Am Anfang war das Bit, und das Bit machte Licht, und das Licht zeigte das Bit.

DisCo, der Display-Computer, ist die von Rolf Kretzschmar erdachte Kommando- und Programmier-Umgebung, mit der das Verständnis der Digitaltechnik vermittelt und geübt werden kann. Wie er seit Jahren in der Schule demonstriert, ist dabei nichts einleuchtender, als Bits durch Lampen abzubilden, nicht nur als Darstellung auf dem Bildschirm, sondern auch real mit LEDs als Lampen.

So schlicht und einfach wie die benutzte rechteckige Lampenmatrix soll natürlich das gesamte System sein, in Soft- und Hardware. Dafür habe ich gesorgt. Die folgende Darstellung richtet sich einerseits an die Anwender, die für Unterricht oder einfach zum eigenen Spaß mit den bereitgestellten Mitteln umgehen möchten, und zum anderen an Programmierer, die DisCo auf andere Rechner und Ausgabe-Geräte übertragen wollen. Die Quell-Files werden über die FG-Mailbox zur Verfügung gestellt.

Forderungen

Einfach muß das System sein, aber nicht primitiv. Wenige Methoden sollen - einmal verstanden - auf komplexere Objekte übertragbar sein. Aus verschiedenen Sichtweisen auf ein Feld die jeweils passende Methode zu wählen und anzuwenden, ist ein Bildungsziel, das sich an diesem einfachen Modell üben läßt. Konkret stelle ich die Objekte und Methoden hier kurz vor:

Stichworte

DisCo
Methoden
Lampenfeld
Vokabulare

Das Grundobjekt...

...ist die einzelne Lampe, mit der sich schon einiges anfangen läßt: trivial ist das Ein-, Aus- und Umschalten; es läßt sich aber auch der Zustand einer Lampe gewinnen und verwenden. Die nächstliegenden Verwendungen des Zustands sind: Darstellung an anderer Stelle, Vergleich mit anderem Lampenzustand und Verknüpfung mit anderem Lampenzustand durch die üblichen logischen Funktionen (und, oder, ..). Statt der üblichen logischen Funktionen sind ebenso beliebige selbstdefinierte Verquickungen in gleicher Weise verwendbar.

Das Geamtobjekt...

...ist das Bild, auf das sich die Methoden der lampenweisen Betrachtung sinngemäß anwenden lassen. Statt der identifizierenden Koordinatenpaare bei Lampen sind etwa die Namen verschiedener Bilder zu verwenden.

Auch Teile des Bildes sind als Objekte der Betrachtung geeignet; wir haben uns auf `ZEILENWEISE` und `SPALTENWEISE` beschränkt. Bei diesen genügt jeweils eine Koordinate zur Identifikation.

Außer den genannten Methoden sind zahllose weitere möglich, von denen die Erzeugung benannter Objekte als Konstanten oder Variablen besonders erwähnenswert ist. Um aber die Übersichtlichkeit nicht zu gefährden, haben wir uns auf ein Dutzend als "notwendig" bezeichnete Methoden beschränkt und aus diesen bis zu zehn weitere gebildet, die uns als besonders empfehlenswert erschienen. Das Glossar am Ende dieses Beitrags erklärt die Einzelheiten.

Einfach...

...und pflegeleicht wie die Anwendung soll auch die Software sein. Darum benutzen wir von Anfang an Forth. Seit dem Beginn mit A-Forth auf Apple II wurden einige Reifestadien durchlaufen. Das erste Konzept war mehr experimentell, das Bild war auf ein Paket aus 8 Linien zu je einem Byte beschränkt, und die zur Übertragung auf verschiedene Lampenfeld-Hardware notwendige Gliederung in Gerätetreiber und Handhabung fehlte zunächst. Inzwischen können Linien oder Bilder so viele Bytes enthalten, wie mit einer normalen Integerzahl ausgedrückt werden kann. Um die Objekte trotz ihrer Größe bequem auf dem Stack swappen und duppen zu können, wurde etwas in objektorientierte Programmierung und dynamische Speicherverwaltung investiert. Schließlich wurde durch geeignete Modularisierung die für die Verbreitung nötige Übertragbarkeit erreicht.

Die neueste Auflage des DisCo ist in Forthmacs geschrieben; für F-PC und Rockwells RSC-Forth (etwa 79er Standard) gibt es Einleitungen, die das DisCo File akzeptabel machen. Gerätetreiber für Bildschirm und Lampen-Hardware werden nach Bedarf zum Schluß geladen.

Beschreibung:

Da die Kommentare in den Quell-Files weder Einführung noch Überblick bieten, folgt hier die wünschenswerte Ergänzung in knapper Form.

Das Zeichen # verwenden wir allgemein als Kennzeichen für die Bildmatrix, | für Spalten, _ für Zeilen und . für Punkte bzw. einzelne Lampen.

Organisatorisches

Die Dimensionen des Lampenfeldes sind in den Values `#COLS` und `#ROWS` für Spalten und Zeilen enthalten. Ein Lampenfeld bildet den zusammenhängenden Speicherbereich beginnend mit Adresse `#SPACE` ab.

Jede Lampe entspricht einem Bit. Das 1. Byte liegt in der linken oberen Ecke des Lampenfeldes waagerecht. Die Lampe am linken Rand wird mit dem Bit-Offset 0 adressiert. Die folgenden Bytes schließen sich recht an. Nach dem letzten Byte einer Zeile folgt das linke Byte der darunter liegenden Zeile. Die Spalten und Zeilen werden also von der linken, oberen Ecke aus gezählt, mit 0 beginnend. Dies ist die intern benutzte "technische" Anordnung, die wohl auch den meisten Bildschirmen entspricht. Die äußere Darstellung ist davon unabhängig. In der Schule ist die kartesische Orientierung üblich, bei der die vertikale Achse nach oben zeigt. Gelegentlich wollen wir auch Binärzahlen darstellen, bei denen wir das höchstwertige Bit nicht rechts, sondern links außen brauchen.

Um alle Ansprüche erfüllen zu können, schalten wir den Eingabe-Modus um mit den Worten TECHNISCH, SCHULISCH oder WERTIG. Eingabe-Worte haben daher gegebene Koordinaten erst zu bearbeiten: Spalten werden mit | behandelt, das nach einem Bereichstest das Defer-Wort (|) anwendet, das je nach Modus |REVERS bzw. NOOP ausführt. Für Zeilen sind die Worte _ und (_) zuständig. Die Modus-Worte setzen (|) und (_) nach Bedarf; TECHNISCH z.B. installiert NOOPs.

Prinzipiell passen die Objekte der Betrachtung - von einzelnen Lampen abgesehen - nicht auf den Stack. Sie werden daher intern als Linien behandelt, die als Superstrings realisiert sind: Count-Wort (nicht -Byte) mit angehängten Datenbytes. Auf dem Stack ist als Repräsentant nur der Zeiger auf das Count-Wort. ALLOCATE und FREE verwalten den für diese Objekte reservierten Speicher von [SPACE bis SPACE] dynamisch. Diese Technik hat für die Benutzung nur eine Auswirkung: Sonderversionen für die Stack-Manipulation. *DUP, *DROP, *SWAP sowie *ROT und *OVER sind eingeführt zur einheitlichen Handhabung, obwohl manche nichts anderes tun als ihre *-losen Aliase.

Vokabulare...

...bieten die Möglichkeit, gleichlautende Worte in verschiedenem Kontext auf verschiedene Objekte spezifisch anzuwenden. Wir verwenden daher - innerhalb des Vokabulars DISCO - die folgenden:

```
.WEISE "punktweise" oder
       "lampenweise"
_WEISE "zeilenweise"
|WEISE "spaltenweise"
#WEISE "bildweise"
```

Diese vier Vokabulare sind IMMEDIATE, das hat sich bewährt.

Zur vorübergehenden Rettung und Restaurierung des CONTEXT bieten wir die Immediate-Worte ZEITWEISE und ALTWEISE. Dies ist besonders nützlich in einem Forth-System nach 79er Standard, das keine wahlfreie Suchordnung hat.

Manipulatoren

Jedes Vokabular enthält mindestens das Dutzend der im Glossar verzeichneten erforderlichen Worte, soweit sich die passende Version nicht bereits in der Suchordnung als erstes findet. Besondere Beachtung verdient das Wort BITS: , das den Schlüssel zur bitweisen Verknüpfung zweier Objekte liefert. Anstatt eines umfangreichen Satzes der üblichen AND, OR, etc. nebst ihren Invertierungen in jedem der Vokabulare bieten wir "nur" die spezfifischen Versionen des (state-)smarten BITS: als Werkzeug. Beispiel: (|WEISE)

```
\ Definition
: UND ( Obj1 Obj2--Obj )
      BITS: AND ;
\ interaktiv mit selbst-
\ gestrickter Verquick-
\ ung namens QUICK
1 GET 5 GET BITS: QUICK
5 PUT
```

Jede Bit-Verknüpfung, auch irgendein in DISCO definiertes QUICK, steht dadurch allen objekt-Weise Vokabularen zur Verfügung. Diese in bester Forth-Tradition stehende Technik hält nicht nur den Standard-Wortsatz schlank, sondern erlaubt die einheitlich elegante Verwendung beliebiger Paar-Verknüpfungen, die sogar von weiteren Bedingungen abhängig sein dürfen. (Puristen dürfen sich gerne die nicht-state-smarten BITS: und [BITS]: schreiben.)

Die empfohlenen Worte dienen vor allem dem bequemen Umgang: AN und AUS und UM ersparen im Standard-Fall Schreibarbeit, SHIFT und ROLL bieten etwas für das Auge des Anfängers bevor er böse Abstürze überstehen muß.

Der Treiber...

...hat sein eigenes Vokabular, das er ins Defer-Wort TREIBER schreibt, und darin das Wort PRINT, das über das Defer-Wort #PRINT zu benutzen ist. Zusätzlich kann er HELL und DUNKEL enthalten, um auszuwählen, wie 1-Bits dargestellt werden sollen. Falls er einen Bildschirm steuert, sind noch Rahmen mit Zeilen- und Spaltenzahlen nützlich. Weiten Spielraum bieten Variationen der Lampengröße und -Form. Die vom jeweiligen Treiber unterstützten Werte für #COLS und #ROWS, sowie eine Reihe davon abgeleiteter Values können mit >DEPENDANTS? belegt werden, das auch Speicher reserviert. Das gelieferte Flag zeigt an, ob sich eventuelle Voreinstellungen mit den geforderten Werten vertragen.

Nützliches

Unabhängig von spezieller Betrachtungsweise durch die vier obigen Vokabulare finden sich in DISCO noch andere schöne Worte:

NUR ist sehr praktisch, um nur das angezeigte Bild für ein neues zu löschen. (z.B: .WEISE NUR 0 1 AN)

CLEAR leert die Stacks und gibt eventuell reservierten dynamischen Speicher frei.

Mit SHOW und -SHOW kann man einstellen, ob die mit PUT (s.Glossar) erreichten Ergebnisse gleich angezeigt werden. Das ermöglicht unter anderem, durch Vergleich zu ermitteln, wieviel der verbrauchten Rechenzeit dem Bildaufbau zu verdanken ist.

Text-Darstellung ist in DISCO nicht enthalten. Diese schöne Anwendung stelle ich im nächsten Heft vor.

❑

Glossar der DisCo Standard-Worte

Zeichen-Erklärung:

(*)	erforderliche Worte
	empfohlene Wort-Einträge
Adr	0, 1 oder 2 Stack-Eintrag je nach CONTEXT
Obj	Repräsentation mit einem Stack-Eintrag
p	Pointer
col	Column, Spalte
row	Row, Zeile
#x	Adresse eines alternativen Bildspeichers, gleich groß wie bei #SPACE
<op>	Ein Wort zu Bitverknüpfung, z.B. AND

Speicherung: (alle Worte erforderlich)

#COLS	(-- n)	Value: Spalten im Speicher
#ROWS	(-- n)	Value: Zeilen im Speicher
(von diesen abhängig:)		
#B/COLS	(-- n)	Value: Bytes für eine Spalte im Speicher
#B/ROWS	(-- n)	Value: Bytes für eine Zeile im Speicher
#B/#	(-- n)	Value: Bytes für ein Bild im Speicher
#SPACE	(-- n)	Value: Anfang des Bildspeichers
>DEPENDANTS?	(cols rows -- f)	tut nichts, falls die Werte auf dem Stack übereinstimmen mit #COLS und #ROWS, (Erfolg) falls #COLS und #ROWS noch =0 sind: Initialisierung abhängiger Values, Speicher-Reservierung Meldet Fehler,wenn #COLS und #ROWS schon anders besetzt f = 0 zeigt Mißerfolg
CLEAR	(? --)	leert die Stacks und gibt eventuell reservierten dynamischen Speicher frei.

[SPACE und SPACE] sind die Grenzen des des durch ALLOCATE und FREE automatisch verwalteten dynamischen Speichers. Nicht zur direkten Anwendung vorgesehen.

Ausgabe:

TREIBER	(--)	Defer für Treiber-Vokabular (*)
#PRINT	(--)	DEFER für Treibers PRINT, (*) das den #SPACE abbildet
SHOW	(--)	veranlaßt, daß PUT auch #PRINT ausführt, evtl. mit schmückendem Beiwerk
-SHOW	(--)	veranlaßt das Gegenteil
(SHOW)	(-- f)	Flag-Variable. Wahr, wenn SHOW aktiv
HELL	(--)	Erscheinen eingeschalteter Punkte
DUNKEL	(--)	Erscheinen eingeschalteter Punkte

Eingabe:

(|)	(col - col')	Spalten-Eingabe - techn. Zählweise (*)
|	(col - col')	Bereichs-Prüfung, dann (|) (*)
(_)	(row - row')	Zeilen-Eingabe - techn. Zählweise (*)
_	(row - row')	Bereichs-Prüfung, dann (_) (*)
TECHNISCH	(--)	setzt (|) und (_) passend, dann SHOW (*)
SCHULISCH	(--)	setzt (|) und (_) passend, dann SHOW
WERTIG	(--)	setzt (|) und (_) passend, dann SHOW

Manipulation:

NUR	(--)	löscht ganzen Bildspeicher
.WEISE	(--)	Vok."punktweise" oder "lampenweise" (*)
_WEISE	(--)	Vokabular "zeilenweise" (*)
|WEISE	(--)	Vokabular "spaltenweise" (*)
#WEISE	(--)	Vokabular "bildweise" (*)
		Diese Vokabulare sind IMMEDIATE

Folgende Worte sind in jedem Vokabular definiert oder in passender Version in der Suchordnung zuoberst verfügbar:

*GET	(#x Adr -- Obj)	holt aus Bild-X das passende Objekt, (*) also Punkt,Zeile,Spalte oder ganzes Bild
*PUT	(Obj #x Adr --)	setzt Obj in Bild-X an Adr ein (*)
VARIABLE	(comp: --)	(run: -- p) (*)
@	(p -- Obj)	holt Obj aus der Var. in Arbeitsform (*)
!	(Obj p --)	speichert Obj in der Variablen (*)
INV	(Obj -- Obj')	invertiert alle Bits eines Objektes (*)
=	(Obj1 Obj2 -- f)	vergleicht zwei Objekte (*)
BITS: <op>	(Obj1 Obj2 -- Obj)	verknüpft 2 Obj., hinterläßt Ergebnis (*)

Der Name der Operation vom Input-Kanal wird benutzt.
BITS: ist state-smart, daher auch innerhalb einer :-Definition nutzbar

NULL	(-- Obj)	gibt Obj mit sämtlichen Bits = 0 (*)
*DUP	(Obj1 -- Obj1 Obj1)	DUP für Objekte (*)
*DROP	(Obj --)	DROP für Objekte (*)
*SWAP	(Obj1 Obj2 -- Obj2 Obj1)	SWAP für Objekte (*)
GET	(Adr -- Obj)	holt aus #SPACE das pass. Objekt
PUT	(Obj Adr --)	setzt Obj in #SPACE an Adr ein, s.o. SHOW
CONSTANT	(comp: Obj --)(run: -- Obj)	
AUS	(Adr --)	macht NULL PUT an gegebener Adresse
AN	(Adr --)	macht NULL INV PUT ..
UM	(Adr --)	macht das Gegenteil
SHIFT	(Obj n (m) -- Obj')	bewegt die Bits der Linie um n Plätze in die eingestellte Zählrichtung. n darf auch negativ sein. In die geräumten Plätze rücken 0-Bits nach. Auf Bilder angewendet wird um n Spalten und m Zeilen gesch. (Anzeige mit PUT)
ROLL	(Obj n (m) -- Obj')	arbeitet wie SHIFT, aber mit dem Unterschied, daß in die geräumten Plätze die hinausgeschobenen Bits nachrücken.

❑

Microcontroller zu verleihen !

Mitteilung von Rafael Deliano

Steinbergstr. 37, W-8034 Germering, Tel.: 089/8418317

Der Verleih der Microcontroller wird neu geregelt. Hier die Einzelheiten.

Bisher hat Jörg Staben das Handling der Boards durchgeführt. Aber da er mit seinem Direktorposten und seinem Engagement für F-PC bis über die Ohren zu ist, ist die Verwaltung einstweilen auf mich übergegangen.

Auch die Hardware hat sich etwas verändert. Die Controllerplatinen enthalten jetzt durchweg Steckernetzteil und 9-Pin-V24-Buchse. Deren Belegung enspricht einem Modem, es sind also handelsübliche Kabel verwendbar. Aber es liegt kein V24-Kabel bei.

Voraussetzungen

Wenn Ihr Tischcomputer einen 9-Pin V24-Stecker hat, brauchen Sie also ein 9poliges Kabel, das 1:1 durchverbindet. Wird oft als "Mauskabel" bezeichnet. Hat der PC einen 25-Pin Stecker, ist außerdem Kabel auch ein Adapter nötig. F65 ist nicht auf einen bestimmten Computertyp festgelegt. Man braucht zusätzlich nur noch ein beliebiges, einfaches Terminalprogramm. Beim Super8 liegen 3,5'' und 5''-Disketten für IBM-PCs bei. Grundfunktionen sind zwar auch mit anderen Tischcomputern und beliebigen Terminalprogrammen testbar, aber für ernsthaftes Entwickeln ist die Verwendung der mitgelieferten Software unumgänglich.

Stichworte

Microcontroller
Super8
65C02
Verleih

Zugriff

Wenn Sie Mitglied der FORTH e.V. sind und eines der Systeme zum Testen haben wollen, rufen sie mich an. Ich benachrichtige denjenigen, der das gewünschte System gerade hat. Er wird es direkt an Sie weiterschicken. Oder er wird Sie informieren, wie lange er es noch braucht. Wenn Sie das Gerät erhalten, öffnen Sie die Schachtel bitte vorsichtig. Sie brauchen sie nochmal beim Weiterschicken. Überprüfen Sie bitte den Inhalt auf Vollständigkeit und rufen Sie mich dann bitte an, damit ich nicht im Dunkeln tappe, wo die Boards der e.V. verblieben sind. Die Controller sind offene Leiterplatten. Da kann es durchaus vorkommen, daß man ein Board erledigt. Nicht weiter tragisch. Wenn Sie nicht selbst in der Lage sind es zu reparieren, senden Sie es bitte direkt an mich zurück, damit ich mich um die Instandsetzung kümmere.

	Super8	F65-KIT
Hardware		
CPU	8-Bit "Super8" Microcontroller von Zilog	8-Bit "65C02" Microprozessor von Rockwell
EPROM	8kByte (ROM im Controller)	28kByte (nur 14kByte belegt)
RAM	32kByte (batteriegepuffert)	32kByte
I/O	UART 3 Ports 2 Timer 8 IRQs	UART in Software
Software	FORTH in Microcode von CPU unterstützt FORTH-83-Standart	JSRthreaded-FORTH Assembler & Disassembler für 65x02

Einschränkungen

Wenn Sie nicht gerade zum Lötkolben greifen und Hardware anbauen, können Sie mit den Geräten einstweilen noch nicht sehr viel tun. Man kann natürlich Programme entwickeln. Aber Controller sind zum Ansteuern von Hardware gedacht, und die Boards haben derzeit noch kein passendes I/O. Es sind Demozusatzplatinen mit Displays, Laut- sprecher usw. in Vorbereitung. Aber bis die verfügbar sind, dauert es noch eine Weile.

❑

Anmerkung der Redaktion:
Sie können natürlich das DisCo-Lampenfeld damit ansteuern!

Anmerkung der Redaktion

Die Kolumne „Forth und der Rest der Welt" muß diesmal leider ausfallen, weil Andreas Goppold beruflich zu sehr beansprucht war. Er hat aber versprochen, sich im nächsten Heft wieder zu melden.
Die Rubrik „WANTED in Forth" mußte aus Platzmangel entfallen.

Vielen Dank an alle Autoren und Ideenlieferanten der VD. Damit das Heft in Zukunft noch vielseitiger wird, brauchen wir Ihre Mitarbeit auch im nächsten Jahr.

Die Redaktion wünscht sich daher viele Beiträge und seinen Lesern ein gutes und erfolgreiches Jahr 1993

Portierung: eForth ⇒ 68HC11

von Wolfgang Schemmert

StrahlenbergerStr.123 6050 Offenbach Tel:069-8001208 Fax:069-64825957

Das eForth für den 8051 Mikrocontroller wurde auf Assemblerbasis für den 68HC11 umgeschrieben. Neben wesentlichen Eigenschaften dieser Forth-Version wird die Vorgehensweise bei solch einer Portierung skizziert.

Der *68HC11* ist neben der 8051-Familie derzeit einer der gebräuchlichsten low-cost/low-end Mikrocontroller. Während der *8031/32* erstaunlich schnell und komfortabel arbeitet bei Aufgaben, die mit 128/256Byte Datenspeicher (und der im Controller eingebauten Peripherie) auskommen, ist der *68HC11* für Probleme, die intensiv 16-Bit Daten- bzw. externe I/O- Adressen benötigen, in mancher Hinsicht besser geeignet. Mit beiden Prozessoren habe ich des öfteren hardware- und assemblermäßig zu tun. Teils aus Neugierde, teils mit einer konkreten Anwendung im Hinterkopf kam irgendwann der Wunsch auf, auf Einplatinen-Minimalsystemen mit diesen Prozessoren auch in Forth zu arbeiten.

Als Hardwareplattform für das 68HC11 Forth habe ich den MOPS-Einplatinen-Controller gewählt, weil er preiswert und verfügbar ist (Beschreibung und Bauanleitung siehe Elrad Hefte 3 bis 5 /91). Besonders in seiner erweiterten Form (siehe Elrad Heft 8/92) wird er den verschiedensten I/O-Anforderungen gerecht.

Wie anfangen? Da ich mich eher zu den Forth-Laien rechne und auch keinen Metacompiler etc habe oder kaufen wollte, lag es nahe, das System vom Assembler her aufzubauen.

So habe ich mir zuerst ein Listing des 6800 fig.Forth besorgt, dann bin ich im Kleinanzeigenteil der 4.Dimension auf das von A.Klingelnberg vertriebene 8051 eForth gestoßen. Nach einigem Hin und Her habe ich mich entschieden, das *eForth* als Ausgangspunkt für meine eigenen Programmierversuche zu nehmen, vor allem, weil mir das einen Haufen Tipparbeit ersparte und zusätzlich zum primär angestrebten *68HC11-Forth* mich auch mit *8051-Forth* versorgte.

Im Unterschied zu anderen Forth-Versionen ist das *eForth* nicht "self sufficient" angelegt, seine Programmierumgebung und dazu notwendiger Massenspeicher wurde in einen externen Steuerrechner, z.B. IBM-PC, verlagert, mit dem der Targetcontroller während des interaktiven Betriebs via RS232 verbunden ist. Entsprechend programmierte stand-alone Applikationen laufen natürlich auch ohne die Nabelschnur.

Der erste Schritt bestand darin, das Programm von der Original-Diskette auf einer entsprechend konfigurierten 8051-Platine zum Laufen zu bringen. Das Originalprogramm ist mit dem Microsoft-Macroassembler codiert, wobei anzumerken ist, daß dabei der MASM eigentlich nur als komfortables Werkzeug zur strukturierten Belegung von Speicherbereichen eingesetzt wird. Der 8051- Maschinencode für die Primaries ist byte-weise von Hand à la KIM oder Junior-Computer eingetragen. Erfreulicherweise ließ sich der Code ohne größere Probleme "assemblieren" und in ein EPROM brennen. Dann waren allerdings noch einige Anstrengungen nötig, um einen in der Source versteckten Fehler zu finden (falsche RAM-Adresse). Obwohl im engeren Sinne "funktionierend" hat diese Version einen entscheidenden Pferdefuß, der sie für den praktischen Einsatz unbrauchbar macht: Es gibt keinerlei Möglichkeit, auf die "Special Function Registers" also auf die eingebaute Peripherie des 8051 von eForth aus zuzugreifen. Die dazu notwendigen Worte lassen sich zwar als Dictionary-Erweiterungen nachprogrammieren, aber nur mit ex-

Stichworte

e-Forth
68HC11 (MOPS)
8051
Portierung
Assembler

Der eForth Header

```
; Compile a code definition header.
.MACRO @CODE PLABEL,PLILABEL,LINK,LEX,NAME
       @ALIGN                    ;force HEADER to even address
       .DW ( * + 6+((LEX) &$1E)) ;points to subsequent code-addr
       .DW LINK                  ;Link to previous name-address
PLILABEL: .DB LEX,NAME           ;Lex byte and name string
       @ALIGN                    ;force CODE to even address
PLABEL:                          ;begin of primary code
.ENDM

; Compile a colon definition header.
.MACRO @COLON SLABEL,SLILABEL,LINK,LEX,NAME
       @ALIGN                    ;force HEADER to even address
       .DW (* + 6+((LEX) & $1E)) ;points to subsequent code-addr
       .DW LINK                  ;Link to previous name-address
SLILABEL: .DB LEX,NAME           ;Lex byte and name string
       @ALIGN                    ;force CODE to even address
SLABEL:                          ;begin of code
       nop                       ;align dummy
       lcall DOLST               ;Forth ENTER
.ENDM
```

Abb. 1

Auszug eines Listings

```
(.........)
; PRESET ( -- )
;        Reset data stack pointer and the terminal input buffer.
 @COLON PRESE,LPRESE,LEVAL,6,'PRESET'
 .DW SZERO,AT,SPSTO
 .DW DOLIT,TIBB,NTIB,CELLP,STORE,EXIT

; xio ( a a a -- )
;     Reset the I/O vectors 'EXPECT, 'TAP, 'ECHO and 'PROMPT.
 @COLON XIO,LXIO,LPRESE,COMPO+3,'xio'
 .DW DOLIT,ACCEP,TEXPE,DSTOR
 .DW TECHO,DSTOR,EXIT

; FILE ( -- )
;      Select I/O vectors for file download.
 @COLON FILE,LFILE,LXIO,4,'FILE'
 .DW DOLIT,PACE,DOLIT,DROP
 .DW DOLIT,KTAP,XIO,EXIT

; HAND ( -- )
;      Select I/O vectors for terminal interface.
 @COLON HAND,LHAND,LFILE,4,'HAND'
 .DW DOLIT,DOTOK,DOLIT,EMIT
 .DW DOLIT,KTAP,XIO,EXIT

; I/O ( -- a )
;     Array to store default I/O vectors.;
 @COLON ISLO,LISLO,LHAND,3,'I/O'
 .DW DOVAR       ;emulate CREATE
 .DW QRX,TXSTO   ;default I/O vectors

; CONSOLE ( -- )
; Initiate terminal interface.

 @COLON CONSO,LCONSO,LISLO,7,'CONSOLE'
 .DW ISLO,DAT,TQKEY,DSTOR     ;restore default I/O device
 .DW HAND,EXIT                ;keyboard input
(.........)
```

Abb. 2

tremer Bitfummelei. (In der nachfolgend beschriebenen 8051- Assembler-Version habe ich entsprechende Kernel-Erweiterungen SFR! und SFR@ hinzugefügt.) Der nächste Schritt bestand darin, die Source für das 8051-eForth in einen "richtigen" Assemblercode umzuschreiben, wozu ich den "UCASM"-Assembler verwendet habe.* Für "normale" Assembler-Programmierung empfinde ich den UCASM als recht komfortabel. Er unterstützt jedoch, anders als der MASM, weder Vorwärtsreferenzen bei der Belegung von Speicher noch Assemblierungsvariable.

Daher habe ich mich schließlich dazu durchgerungen, eine von *C.H.Ting*, dem Entwickler des eForth-Modells als wesentlich bezeichnete Eigenschaft des eForth - die getrennte Segmentierung von Code und Header - aufzugeben. Nachträglich stelle ich fest, daß ich damit keinen Nachteil, aber einige kleine Vorteile eingehandelt habe. Innerhalb des eForth mußten dazu nur zwei Worte ("TOKEN" und "$,n") modifiziert werden, auf die Portabilität von Programmen hat die Modifikation soweit ich es überblikke, keinen Einfluß. In Bild 1 ist die Konstruktion des eForth Headers, der in der Source als Macro realisert ist, wiedergegeben.

Durch die Macro-Erstellung bekommt die Assemblerfassung eine sehr übersichtliche modulare Gestalt, siehe Listing Auszug Bild 2. Jedes Forth-Wort wird durch den Namen seines Code-Address-Labels angesprochen, der max. 5 Zeichen lang ist und mnemonisch an den Forth-Namen des Wortes angelehnt ist

Ohne sich um Einzelheiten der Assemblernotierung kümmern zu müssen, kann man dergestalt sehr leicht modifizierte eForth Kerne herstellen. Das Ganze eignet sich also als - zugegeben äußerst primitiver - manueller Targetcompiler.

Der dritte und im Sinne dieses Artikels entscheidende Bearbeitungsschritt bestand in der Portierung der in Assembler geschriebenen Teile auf den 68HC11-Maschinencode. Schließlich, ein feiner aber wichtiger

Aufteilung der Speicherbereiche des 68HC11

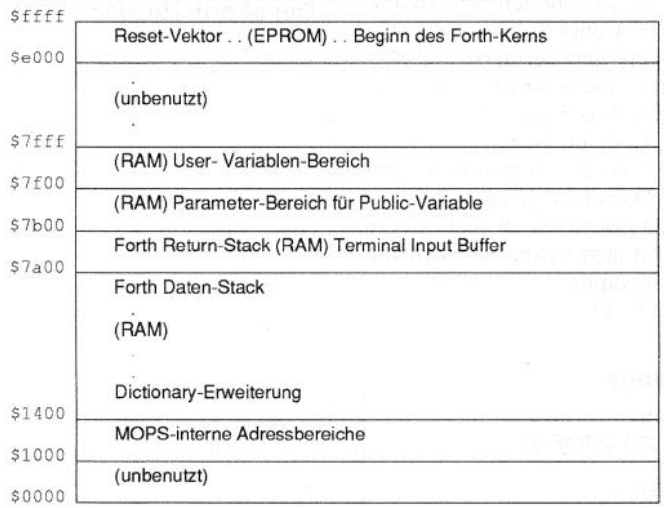


Abb. 3

*)
Der UCASM ist ein tabellenorientierter Macro-Crossassembler von F.Meersmann für alle gängigen 8Bit-CPU's. Schließlich habe ich jetzt als Nebenresultat den nicht prozessorabhängigen Teil als UCASM-kompatible Source und könnte mit sehr geringem Aufwand Versionen schreiben für nahezu alle 8-Bit- Prozessoren, die der UCASM unterstützt.

Schritt, ist die Umkehrung der 16-Bit Wortnotierung von der im Original verwendeten Intel-Konvention (Lowbyte vor Highbyte) auf die Motorola-Konvention (Highbyte vor Lowbyte). Bild 3 zeigt die Aufteilung der Speicherbereiche. Diese Konfiguration wurde speziell für das MOPS-Board erstellt, ist aber auch für jeden anderen Controller verwendbar, der zumindest von $1400 bis $7fff mit RAM und von $e000 bis $ffff mit EPROM bestückt ist.

Zuerst hatte ich gedacht, die im 6800 fig.Forth konstruierte virtuelle Forth-Maschine (unter Berücksichtigung des beim 68HC11 hinzugekommenen Y-Registers) auf das eForth zu übertragen. Dann habe ich mich aber treiben lassen und versucht, die Denkansätze des eForth möglichst direkt auf die Gegebenheiten des 68HC11 zu übertragen. Als wesentlicher Punkt ist dabei herausgekommen, daß (wie beim fig.Forth) der Hardwarestack als Forth-Datenstack dient, aber das oberste Element dieses Stacks im Akku D gehalten wird. Das ergibt relativ günstige Laufzeiten zwischen Parametertransfer und Parameterverknüpfung.

Das Index-Register Y arbeitet als Forth-Instruction-Pointer, das Index-Register X als temporäres 16-Bit-Hilfsregister. Ich bin mir nicht ganz sicher, ob eine Vertauschung beider Registerfunktionen einen Vorteil bringen würde, er wäre jedenfalls geringfügig.

Der Forth-Returnstack-Pointer und einige andere systeminterne Variable liegen im prozessorintenen Direkt-RAM.

Ohne mich jetzt auf irgendwelche Benchmark-Diskussionen einzulassen, bleibt festzustellen, daß das 68HC11 eForth mit 8MHz Quarz etwa 3 bis 4 mal schneller als die 8051-Version mit 11 MHz Quarz ist. Für die in Details interessierten Leser in Bild 4 das Listing einiger Routinen des inneren Interpreters und elementarer Primaries, um die Implementierung der virtuellen Forth Maschine auf dem 68HC11 zu verdeutlichen.

Nachdem das 68HC11 eForth endlich "funktionierte", war der Frust zuerst ziemlich groß:

Das nackte eForth kann mit den meisten üblichen Terminalprogrammen nur via Tastatur und das auch nur zum sofortigen Verbrauch programmiert werden, es bestand also keine Möglichkeit, kompilierten Code irgendwo langfristig aufzubewahren.

Der zweite sehr erweiterungsbedürftige Punkt war das Programmierinterface. Grundsätzlich ist es beispielhaft simpel und effektiv: es verarbeitet den Bytestrom der seriellen Schnittstelle.

Die Diskussion, ob Blockfiles oder nicht ist PC-seitig im Steuer-Terminal zu entscheiden. Für die Synchronisation längerer Programm-Eingaben gibt es einen einfachen aber brauchbaren Handshake- Mechanismus: Mit dem Wort "FILE" wird das Tastaturecho und das ".ok"-Prompt abgeschaltet. Statt des ".ok" wird nach Interpretation bzw. Kompilierung der Eingabezeile eine dezimale 11 über die serielle Schnittstelle als Pace-Zeichen an das Steuerterminal zurückgesendet. (Sicherlich wäre die HEX11, das bekannte XON- Kommando sinnvoller gewesen, vielleicht ist da im Laufe der verschiedenen Bearbeitungsprozesse ein Dollar verloren gegangen.) Das Steuer-Terminal muß lediglich nach jedem CRLF seine Transmission unterbrechen und auf das Pace-Zeichen warten. Zur Veranschaulichung wurde dieser Teil der eForth-Source in Bild 2 als Beispiel ausgedruckt.

Leider gibt es kaum Kommunikationsprogramme, die dieses Handshake- Verfahren unterstützen. "Procomm" ist das einzig mir bekannte; es wird folgendermaßen benutzt: Zuerst wird der nachzuladende Code auf einem normalen ASCII-Editor als Streamfile erstellt. In "Procomm" wird zunächst mit ALT-S das Menu "Setup Screen" gerufen. Dort ist im Untermenu 6, Punkt 3 (Pace Character) eine dezimale 11 einzutragen. Ferner unter Punkt 4 (Character Pacing) mindestens eine 10 und unter Punkt 5 (Line Pacing) sicherheitshalber eine 10 ein-

Das 68HC11 eForth mit 8 MHz Quarz ist etwa 3 bis 4 mal schneller als die 8051-Version mit 11 MHz

Listing einiger Routinen

```
.MACRO @NEXT_INSTR
;The Forth Inner Interpreter as a macro
 ldx 0,y
     iny
     iny
     jmp 0,x
 .ENDM
(.........)
; doLIT ( -- w )
;       Push an inline literal.
 @CODE DOLIT,LDOLIT,0,COMPO+5,'doLIT'
 pshb
 psha        ;preserve TOS, LO first
 ldd 0,y     ;get new TOS
 iny         ;forget the inline data
 iny
 @NEXT_INSTR
; doLIST ( a -- )
;        Process colon list.

 @CODE DOLST,LDOLST,LDOLIT,COMPO+6,
                                  'doLIST'
 ldx RP
 sty 0,x       ;save IP
 dex
 dex
 stx RP
 puly          ;new Instruction pointer.
               ;"jsr" in COLON header
               ;misused hardware stack
               ;as pointer latch
@NEXT_INSTR
( ............)
; EXIT ( -- )
;       Terminate a colon definition.
 @CODE EXIT,LEXIT,LEXECU,4,'EXIT'
 ldx RP
 inx           ;rewind RETURN stack poin-
               ;ter first because after
 inx           ;previous PUSH op it points
               ;to next empty cell.
 stx RP
 ldy 0,x      ;(RP)-->IP
 @NEXT_INSTR
 (.........)
; XOR ( w w -- w )
;     Bitwise exclusive OR.
 @CODE XORR,LXORR,LORR,3,'XOR'
 pulx
 stx TP        ;temporary storage in
               ;Direct RAM
 eora TP
 eorb (TP+1)
 @NEXT_INSTR
; + ( w w -- sum )
;   Add top two items.
 @CODE PLUS,LPLUS,LXORR,1,'+'
 pulx
 stx TP          ;temporary storage in
                 ;Direct RAM
 addd TP
 @NEXT_INSTR
 ( .......)
```

Abb. 4

tragen. Dann zurück in den Terminal-Modus, Wort `FILE` an den eForth-Interpreter senden. Dann mit dem Procomm-Kommando `PgUp- ASCII-Protokoll` den zu ladenden Code senden. Wenn fertig, über die Tastatur das Wort `HAND` an eForth senden, neue Worte im Terminal- Modus debuggen. (Natürlich wird man praktischerweise `FILE` und `HAND` in die zu ladende Source integrieren, so schien es aber leichter zu erklären).

Da mir dieser Tippaufwand schnell lästig wurde, habe ich als bisher letzten Bearbeitungsschritt ein spezielles eForth-Terminalprogramm (in C) geschrieben, bei dem folgende Arbeitsschritte per Funktionstaste aufgerufen werden:

F2: beliebigen Source Editor starten (ganz harte Forth-Freunde könnten also sogar mit dem ECHTEN WORDSTAR editieren !! Nix kompatibles oder so)
F3: Source zum Controller laden und kompilieren lassen
F4: SAVESYSTEM (Dictionary-Erweiterung und User-Daten in PC-Datei)
F5: LOADSYSTEM (Gegenstück dazu)
F6: SAVEBIN (beliebiges Binärfeld vom Controller in PC-Datei)
F7: LOADBIN (beliebige Binärdatei vom PC an frei wählbare Startaddresse im Controller)
F8: MS-DOS-Funktion oder Batch-Datei starten (hier starte ich häufig per Batch-Datei Editor und Assemblierung. Das Umschalten zwischen Forth, Editor und Assembler und zurück geht blitzschnell. Zusammen mit der LOADBIN-Funktion und dem Forth-Wort FUNCTION (s.u.) habe ich im eForth eine recht komfortable Kombination von EPROM-Emulator und Debug-Monitor für Assemblerprogrammierung. Mit etwas Zeigerfummelei lassen sich Assemblerprogramme auch speicherplatzmäßig in die inkrementelle Forth-Dictionary-Erweiterung integrieren.)
F9: Voreinstellungen (Namen und Pfade der in obigen Operationen verwendeten Dateien eingeben bzw. ändern)
F10: Voreinstellungen auf PC-Datei sichern

Eher beiläufig, zum teil als Nebenprodukt der Arbeit am Terminal wurde der eForth Kern um folgene Worte erweitert: `-1 0 1 1+ 1- CONSTANT EEP!` (Wort in internes `EEPROM` des 68HC11 schreiben) `FLIP FORGET FUNCTION` (defining word: beliebiges Maschinensprache-Unterprogramm mit eForth Header versehen und ins Dictionary einfädeln) `JOIN LOADBIN LOADSYSTEM MS SAVEBIN SAVESYSTEM SPLIT USA` (`USER` mit automatischer Offsetverwaltung). `VARIABLE` wurde so modifiziert, daß nachträglich programmierte Dictionary-Erweiterungen ebenfalls ROM-fähig sind. Work in progress: Kompilierung beschleunigen und kompilierte Dictionary-Erweiterungen im Speicher verschiebbar zu machen.

❑

ELRAD-Mops und eFORTH

Der passende 68HC11-Controller zum Schemmert-Artikel

akg. Die Zeitschrift ELRAD fällt zunehmend durch Forth und für Forther wichtige Beiträge auf. Immerhin bedeutet ELRAD: magazin für ELektronik und technische RechnerAnwenDung. In der VD sind Beiträge zu EinplatinenRechnern geplant; darunter solche, die sich mit eFORTH für 68HC11 (und 8051) beschäftigen (Wolfgang Schemmert). Die passende 68HC11-Hardwareplatform finden wir als MOPS in ELRAD 1991-03, -04, -05 (mit RTC und LCD-Port) und als Super-Mops in ELRAD 1992-08. Kurzsteckbrief: Parallel Port, 2 serielle Ports, 2 Relais-Ausgänge, A-D und D-A-Wandler.

❑

Zeitschriften

Elektronik Plus
Automatisierungspraxis 1,
Sonderheft der Elektronik,
'Feldbusse',
Franzis Verlag München, 1992,
130 Seiten, DM 28,-

Per Bus ins Feld

akg. Wir sehen uns einer Feldbus Inflation gegenüber, ein Überblick tut Not: von RS485-Multi-Drop über den Bus im Auto hin zu universellen seriellen Prozessen in StackTechnologie. Wer normt was, weshalb und WANN --- vielleicht ??? Aktoren, Sensoren, Datenverarbeitung verknüpfte Forth schon lange über Feldbusse, nur wußte damals noch keiner, daß sie mal so heißen würden: ein zu 'embedded controllern' eng verknüpftes Thema, also ein Muß für das Gross der Forth-Programmierer: verschiedene Autoren versprechen insgesamt eine objektivere Darstellung, Unterschiede in Firmen- / NormenPolitik werden offensichtlich, aber auch Unterschiede im EchtzeitVerhalten oder z.B. die Eichfähigkeit. Vielfältige Literaturhinweise, Kontaktadressen und ein Glossar runden die Sammlung ab. Das Heft bietet so einen guten Überblick und sinnvollen Einstieg in das Thema zu einem angemessenen Preis.

❑

Die Uhr (1)

von Friedrich 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.

Vorwort

In der VD 8/2 fragt Jörg Staben auf der Seite 26 unter 'WANTED', wer den Lesern der VD 'zeigt', wie eine residente Uhr via "Interrupts" links oben auf den Bildschirm zu zaubern wäre. Er wirft damit ein großes (sprich: umfangreiches) Thema gelassen auf. Die Programmierung von TSR Programmen ist unter 'normalen' Umständen (Compiler mit entsprechendem Interface, Assembler) schon nicht ganz trivial, setzt sie doch mehr als nur rudimentäre Kenntnisse der Hardware und des Betriebsystems voraus. Soll eine solche Uhr aus der 'laufenden' Oberfläche eines FORTH-Interpreters heraus initialisiert / deinitialisiert werden, dann wartet diese Programmieraufgabe mit zusätzlichen, eigenen Problemen auf.

FORTH-Systeme 'biegen' die Bedeutungen der physikalischen Register um, bzw. installieren in der virtuellen Maschine FORTH eigene Register, die physikalisch gar nicht vorhanden sind. Das bekannteste Beispiel ist die Redirektion des 8086/88 Registers BX nach W (Work). Wie aus dem nachfolgenden Aufsatz ersichtlich wird, machen diese Interna von FORTH es unabdinglich, daß der potentielle TSR Programmierer neben den bereits angesprochenen Voraussetzungen 'sein' FORTH "bis in's letzte Bit" beherrscht.

Daraus folgt aber wiederrum, daß das 'Zeigen' grundsätzlicher Techniken bei der Erstellung solcher Programme nur zu einem gewissen Prozentsatz von allgemeiner Relevanz sein kann. Die notwendiger Weise aufzuarbeitenden (oder zu vertiefenden) Grundlagen zur physikalischen Maschine und zum Betriebsystem sind dabei für die Programmierer aller FORTH-Systeme gleich (sofern sie mit Maschinen gleichen Typs arbeiten - z.B. PCs). Die letztliche Implementierung ist jedoch hochgradig von dem Assembler abhängig, den das jeweilige System zu Verfügung stellt.

Die nachfolgende Arbeit bedient sich bei der Lösung der gestellten Aufgabe des ZF-FORTH und des darin 'enthaltenen' Assemblers. ZF ist das von der Moerser Forthgruppe favorisierte System mit sequentiellem Editor, vom Dictionary separierten WORT-Bodies und einem für Forth-Systeme typischen ONE-Pass Assembler, der grundsätzlich nur das SMALL-Model bei der CODE-Erzeugung benutzt.

Der Aufsatz über die speicherresidente Uhr nimmt als Bestandteil eines Assemblerkurses der Moerser Forthgruppe Inhalte dieses Kurses vorweg. Die Syntax des ZF-Assemblers ist UPN-typisch und in den meisten Teilen für jeden FORTHer sicher leicht nachvollziehbar. Einige Besonderheiten, zum Beispiel in der Adressierung von Arrays, werden aber vermutlich nicht ganz so offensichtlich sein, so daß hier auf den besagten Kurs verwiesen werden muß.

TSR-Programmierungen kann man nicht 'zeigen', ohne daß ein solcher Aufsatz wenigstens im Ansatz die Zusammenhänge um Interruptaufrufe beschreibt. Andererseits kann ein Aufsatz für eine Fachzeitschrift das Thema Interruptsteuerungen nicht vollständig abhandeln, zumindest nicht in einer einzigen Ausgabe. Deshalb wird der Aufsatz mehrfach auf weiterführende Literatur verweisen müssen, die im Anhang aufgeführt ist. Bereits an dieser Stelle muß ausdrücklich darauf hingewiesen werden, daß ein vertiefendes Aufarbeiten der Thematik um TSR Programmierungen ohne diese (oder ähnliche) Literatur nicht möglich ist.

Trotzdem habe ich mich bemüht, mit der UHR gewissermaßen ein Gerüst für ähnliche Aufgaben zu liefern. Dieses Gerüst soll aufzeigen, wie 'man es generell machen kann' und worauf der FORTH-Programmierer achten muß, wenn er TSRs aus dem laufenden Interpreter heraus starten will. Das von der UHR bereit gestellte Gerüst läßt sich problemlos zum Beispiel auf Bildschirmschoner übertragen.

Selbstverständlich ist der Quelltext der UHR von allen Mitgliedern der FG frei nutzbar. Sie können damit anstellen, was immer Sie wollen. Verwenden Sie die Uhr in Ihren eigenen Applikationen, benutzen Sie die Uhr als selbstständige Applikation oder nehmen Sie den Source einfach nur als 'Tip'. Nur, bitte, verkaufen Sie mein Werk nicht.

In diesem Sinne wünsche ich den Lesern der VD viel Spaß bei der Lektüre und bei den sicher anlaufenden Experimenten.

Friederich Prinz
Moers, 15.Sept.'92

Stichworte

TSR
ZF-Forth

Grundsätzliches

Die Programmierung von TSR Programmen verlangt geradezu nach der Definition durch einen Assembler. Zwar bieten die allermeisten FORTH-Systeme durchaus Möglichkeiten, die notwendigen Interrupteinsprünge auf der Ebene des Compiler zu erzwingen, aber so definierte Routinen sind, so gering er im Einzelfall auch sein mag, mit einem zusätzlichen Laufzeitüberhang belastet.

Im Hintergrund des Systems tätige Routinen arbeiten in aller Regel ungesteuert und unkontrolliert. Das macht neben besonderer Sorgfalt bei der Programmierung auch besondere Kenntnisse notwendig. Alle Details um die Interna bei der Erstellung speicherresidenter Programme kann und will der vorliegende Aufsatz gar nicht erschöpfend behandeln. Dies würde den gesetzten Rahmen bei weitem sprengen und die Ausmaße eines ganzen Buches annehmen. Die Realisierung einer speicherresidenten Uhr aus einem laufenden, interpretierenden System heraus bringt, wie bereits angesprochen, ganz spezielle Probleme mit sich, die schwierig genug verständlich zu beschreiben sind. Dafür kann ein solcher Aufsatz, der FORTH als Basis nutzt, auf Dinge verzichten, die für den Programmierer eines herkömmlichen Compilers unabdingbar sind. Hier sei beispielhaft auf das komplexe Thema der Erstellung 'künstlicher' Programm Segment Präfixe verwiesen, oder auf mögliche Verfahren bei der Definition sich selbst initialisierender Programme. Wie noch zu zeigen sein wird, können FORTH-Systeme auf solche 'Fußangeln' verzichten. Trotzdem kommt der (Lern)Schweiß vor dem Erfolg - auch der FORTH-Programmierer kann auf einige, grundlegende Kenntnisse bei der Bewältigung der vorliegenden Aufgabe nicht verzichten. Wer diese Kenntnisse vertiefen möchte, findet im Anhang des Aufsatzes Hinweise auf weiterführende Literatur.

1	2	Segmentadresse
3	4	Offset
5	6	Segmentadresse
7	8	Offset
9	10	Segmentadresse
11	12	Offset
13	14	Segmentadresse
15	16	Offset
17	18	Segmentadresse
19	20	Offset

Abb. 1

Interrupts - was ist das ?

Die Realisierung eines TSR Programmes ist, zumindest unter DOS, ohne Nutzung von Interrupts gar nicht denkbar. Darum muß sich der vorliegende Aufsatz zunächst einmal grundsätzlich mit Interrupts und deren vielfältigen Möglichkeiten befassen. Interruptsteuerungen werden in der Literatur häufig angesprochen, selten verstanden, und von den Programmierern auf "NICHT PCs" als etwas absolut Undurchsichtiges abgelehnt.

Interrupts - Unterbrechungen - gehören zu den wesentlichen Faktoren bei dem Siegeszug der PCs in der modernen EDV. Sie kennzeichnen die häufig zitierte "offene Struktur" des Gesamtsystems PC und stellen eine beinahe perfekte Symbiose zwischen Hard- und Software dar. Diese Aussage allein trägt noch nicht viel zur allgemeinen Aufklärung bei und ist genau von der Art, mit der seit 20 Jahren Adepten der 'tieferen' Programmierkunst erfolgreich versuchen, den 'gewöhnlichen' Programmierer von ihren angestammten Pfründen fernzuhalten. Dabei ist alles so einfach ...

Voraussetzend, daß die Adressierungsarten der 8086/88 Prozessoren bekannt sind, stelle man sich vor, daß die ersten 1024 Byte im Arbeitsspeicher des PCs für ganz bestimmte Aufgaben reserviert sind. Diese 1024 Byte sind zu jeweils zwei 16-Bit Worten organisiert, entsprechend der Registerbreite in diesen Prozessoren. Abwechselnd kommt diesen Worten immer gleiche Bedeutung zu. Das erste Wort stellt eine Segmentadresse dar. das nächste Wort einen Segment-Offset. Das nächste Wort soll wieder eine Segmentadresse sein. Das wiederrum folgende Worte ist wieder ein Offset.

Dabei ergibt sich automatisch eine Tabelle, die man sich wie in der Abbildung 1 vorstellen kann. Diese Tabelle, die in der Folge INTERRUPT VEKTOREN TABELLE, oder einfach Vektoren Tabelle, genannt werden wird, existiert in genau dieser Form tatsächlich in jedem unter MSDOS/PCDOS arbeitenden PC. Was es mit dieser Tabelle im Einzelnen auf sich hat, läßt sich recht einfach erklären:

Das Mnemonic INT

Die CPUs der 80x86er Baureihe "kennen" allesamt einen Befehl der durch das Mnemonic INT gekennzeichnet wird. INT ist die Anweisung an den Prozessor 'einen Interrupt aufzurufen'. INT kann niemals ohne eine sogenannte korrespondierende Nummer, einen zugehörigen Parameter aufgerufen werden. Das heißt, daß solche Aufrufe immer in der Art "xx INT" (in PostFix) vorgenommen werden müssen.

Trifft die CPU auf einen solchen Befehl, dann führt sie automatisch (immer !) die folgenden Arbeiten durch:

Zuerst werden die Inhalte des Statusregisters (Flags), des CS Registers (CodeSegment) und des IP Registers (Instruction Pointer) auf den Systemstapel gerettet. Diese Inhalte werden später noch benötigt. Nach diesen Sicherungsarbeiten multipliziert die CPU (immer !) die korrespondierende Nummer des INT Aufrufes (z.B. 8 bei 8 INT) mit vier. Das Ergebnis der Multiplikation - hier 32 - interpretiert der Prozessor als das Byte innerhalb der Vektorentabelle, ab welchem er lesen soll. Die ersten beiden Byte ab dieser Adresse, die , wie in der Skizze beschrieben, eine Segmentadresse darstellen, werden automatisch in das Register CS geladen. Die nächsten beiden Byte, die den Offset auf das CodeSegment bilden, transportiert die CPU in das Register IP.

Bereits mit dem nächsten Prozessortakt kann die Maschine gar nicht anders, als an der durch IP spezifizierten Stelle im Segment CS mit seiner Arbeit fortzufahren - was immer dort definiert stehen mag !

Mit anderen Worten: die Vektorentabelle enthält definitiv keinen Code,

sondern wirklich nur Vektoren (Zeiger) im durch INTEL deklarierten 32-Bit Pointer Format. Diese Vektoren zeigen der CPU wo sie Codes finden kann und sorgen quasi in einer Zwangsführung dafür, daß die CPU auch dort ankommt ! Insgesamt enthält die Vektorentabelle, die 1024 Byte lang ist, 256 solcher Vektoren (1024 / 4). Zu jedem dieser Vektoren ist von der Firma MicroSoft exakt beschrieben, was der jeweils dahinter stehende Code machen soll. Allerdings wird nirgendwo definiert, wie die einzelnen Codes intern aussehen sollen. Das öffnet das System für viele 'Spielereien', für eine Reihe von Optimierungen (von denen eine ganze Reihe bekannter Software häuser lebt)

Die Programmierung von TSR Programmen verlangt nach dem Assembler.

und nicht zuletzt auch für die noch zu programmierende Uhr.

Ein Beispiel: der erste Vektor in der Tabelle, entsprechend der gerade beschriebenen Multiplikation der Vektor für den Interrupt NULL, zeigt auf die Adresse für eine kleine Routine, die auf Divisionen durch Null reagieren soll. Divisionen durch Null erzeugen undefinierte Ergebnisse und müssen grundsätzlich als Fehler aufgefasst werden. Das macht die CPU sogar automatisch, ohne weiters Eingreifen durch den Programmierer. Wenn die arithmetische Einheit auf eine solche Division trifft, löst die CPU ohne Zögern den Interrupt 0 aus und verzweigt in eben diese Routine. Das hat wiederrum in den meisten Systemen mindestens einen Abbruch des gerade laufenden Programms zur Folge. Wenn nun der Programmierer nicht will, daß sein Programm durch einen solchen Fehler 'abgeschossen' wird, ist eine der präventiven Möglichkeiten die, den Vektor auf eine eigene Routine 'umzubiegen'.

Mit anderen Worten: Die Auslösung dieses Interrupts, die von der Hardware vorgenommen wird, läßt sich nicht unterdrücken. Aber wenn dieser Interrupt wirklich ausgelöst wird, dann kann man ihm durchaus ein anderes Verhalten aufzwingen, als das Betriebssystem üblicherweise vorsieht.

Serviceroutinen MSDOS schreibt Dienstleistungen groß

Die wenigsten Interrupts sind "echte" Unterbrechungen des jeweils laufenden Programms. Vielmehr muß man den weitaus größten Teil der durch die Vektoren adressierbaren Codes als Serviceroutinen beschreiben. Das heißt, daß hier kleine, leistungsfähige Assemblerroutinen bereit liegen, über die sich gewisse Dienste vom Betriebssystem anfordern lassen. Zu diesen Diensten gehören Ausgaben auf den Bildschirm, Drucker-steuerungen, Zugriffe auf die Laufwerke, die Steuerung der Maus und anderer Geräte und Vieles mehr. Im Anhang dieses Aufsatzes wird auf Literatur verwiesen, die detailierte Auskunft über die Dienste der einzelnen Interrupts gibt.

"Echte" Unterbrechungen sind aber zum Beispiel der Interrupt 0, die Interrupts 1 und 3 (Einzelschritt und Breakpoint), der Interrupt 2 (NMI - Hardwarefehler), der Tastaturinterrupt - 9 - sowie die Interrupts 4 (arithmetischer Überlauf) und 5 (Hardcopy vom Bildschirm). Auch die Interrupts 8 und 1Ch, die beide zum Timer (Zeitgeber) der CPU gehören, bewirken echte Unterbrechungen des gerade aktuellen Programmlaufes.

Alle diese 'echten' Unterbrechungen werden von der Hardware ausgelöst und zum Teil sogar erzwungen.

Ansonsten ist der Vergleich der Serviceroutinen mit den PRIMITIVES in FORTH-Systemen vielleicht ein wenig weit hergeholt - er trifft aber den Kern des Ganzen recht genau. In den Serviceroutinen sind die CODEs definiert, auf denen große Teile eines späteren Programmes aufbauen, die von der Mehrzahl aller DOS Programme genutzt werden.

Zeitkritische Interrupts

Da dieses Thema nicht all zu weit vertieft werden, aber alle notwendigen Grundlagen anbieten soll, behandeln die folgenden Zeilen ein für die Programmierung via INT sehr wichtiges Thema - zeitkritische Interrupts.

Wenn in der Programmierwelt von 'zeitkritischen Abläufen' gesprochen wird, sind meist Routinen gemeint, die einfach mit der größtmöglichen Geschwindigkeit ein bestimmtes Ergebnis erbringen sollen. Ein Beispiel hierfür ist der 'Flaschenhals' der Bildschirmausgaben. Es ist kein Geheimnis, daß Bildschirmausgaben relativ viel Rechenzeit verbrauchen. Hier greifen viele Programmierer in ihren Programmen zu optimierten, interrupt-gesteuerten Routinen oder schreiben die auszugebenden Zeichen (unter Verlust der Kompatibilität zu anderen Systemen) direkt in den Bildschirmspeicher.

Das Betiebssystem meint mit 'zeitkritisch' aber etwas Anderes:

Zum Beispiel der Interrupt 13h, über den Disketten und Festplattenlaufwerke angesteuert werden, zählt zu den zeitkritischen Routinen. Eine Unterbrechung dieses Interrupts kann, zumindest wenn sie von etwas längerer Dauer ist, katastrophale Folgen für den jeweils gerade laufenden Prozeß (Schreib-Lesezugriff) haben. Mit einem längeren Zeitraum sind hier aber durchaus nicht Minuten oder auch nur Sekunden gemeint. Bei einem sich drehenden Medium, dessen gespeicherte Informationen sich mit relativ hoher Geschwindigkeit unter dem Schreib- Lesekopf des Laufwerkes hinweg bewegen, finden alle Zugriffe im Bereich von Millisekunden statt. Daß hier alle unterbechenden Routinen nur mit äußerster Sorgfalt (und mit unabdingbaren Hintergrundwissen um die Hardware) programmiert werden dürfen, sollte ohne zusätzliche Erklärung akzeptiert werden können.

Der Timer Interrupt 8h - sowie der anhängende Interrupt 1Ch - muß als besonders kritisch angesehen werden. Dieser Interrupt wird cirka 18,2 Mal pro Sekunde durch den Timerbaustein im PC ausgelöst und läßt sich nicht unterdrücken oder verzögern. Folgende Fehlersituation ist bei Experimen-

ten mit diesem Interrupt so etwas wie ein Standard bei Negativerfahrungen, die wohl jeder Programmierer auf dieser Ebene bereits gemacht hat:

- Der Interrupt wird ausgelöst - die Routine wird gestartet.....
- während die Routine abgearbeitet wird, wird durch Hard- oder Software ein weiterer Interrupt ausgelöst oder aufgerufen...
- der zusätzliche Interrupt arbeitet...

- 1/18,2 Sekunden sind vergangen, der Interrupt 8 wird wieder ausgelöst...

...ab hier, nach dem zweiten Auslösen des noch nicht zur Gänze abgearbeiteten Interrupts, begibt sich das System in eine rekursive Schleife, aus der es kein Entrinnen gibt - das System stürzt ab !

Das heißt, daß die durch die Vektoren 8 und 1Ch adressierten Routinen sehr schnell sein müssen, und daß diese Routinen nach Möglichkeit nicht durch andere Services unterbrochen werden sollen !

Ähnlich wie der Interrupt 13h ist auch der Interrupt 21h deshalb als zeitkritisch anzusehen, weil über ihn eine Vielzahl von Services aufgerufen werden. 21h ist 'der DOS-Interrupt'. Über ihn wird der gesamte Bereich des BDOS angesteuert. Wenn man sich vorstellt, daß eine laufende Schnittstellensteuerung über eine gewisse, zeitliche Grenze hinweg unterbrochen wird, kann man sich leicht die Folgen für die dabei transportierten Daten vorstellen.

Diese 'gewisse zeitliche Grenze' kann nun aber nicht exakt definiert werden. Sie ist hochgradig von der Hardware abhängig - vor allem vom Takt mit dem die CPU gesteuert wird. Das läßt sich ebenfalls relativ einfach erkären:

Nimmt man einen PC/XT der ältesten Baureihe zur Grundlage der folgenden Überlegungen, muß man mit einem CPU Takt von 4,77 MHz rechnen. 4,77 Millionen Takte in der Sekunde bedeueten, daß zum Beispiel der Timer Interrupt, der cirka 18,2 Mal pro Sekunde ausgelöst wird, ungefähr alle 262.087 Takte gestartet wird. Bei einer Befehlsfolge wie AX PUSH CX POP, die 24 CPU Takte zur Ausführung benötigt, können demnach zwischen zwei Aufrufen des Interrupts 8 cirka 10900 solcher Befehlsfolgen stehen. Nun nimmt die Routine hinter dem Interrupt 8 natürlich auch noch eigene Takte in Anspruch, so daß sich die Menge der möglichen Operationen zwischen zwei Aufrufen deutliche verringert.

Bei einer modernen CPU, die mit 50 MHz getaktet ist, sieht diese Rechnung schon ganz anders aus. Hier vergehen cirka 2,747 Million Takte zwischen zwei INT 8 Auslösungen. Das heißt, daß hier wesentlich mehr 'zwischen die INTs zu verpacken' ist.

Der INT 8 und sein Anhang

Der Interrupt 8, der bisher mehrfach als der 'Timer Interrupt' bezeichnet wurde, wird, wie bereits gesagt, cirka 18,2 Mal pro Sekunde durch den TIMER Baustein auf dem Motherboard ausgelöst. Nun kann der für den Takt verantwortliche, oszillierende Quarz entweder 18 Mal pro Sekunde schwingen, oder 19 Mal - aber auf keinen Fall 18,2 Mal. Tatsächlich ist dies ein rechnerischer Wert. Der Timer Baustein erzeugt auf einer seiner Ausgangsleitungen einen Takt, über den eine 16-Bit Speicherstelle in einer Stunde vollständig gefüllt wird. Das heißt, auf dieser Leitung liegt eine Frequenz von 65535 Schwingungen pro Stunde an ! Dvidiert man diese Zahl durch 3600 (Anzahl Sekunden/Stunde), ergibt sich der rechnersiche Wert von 18,20416667 Takten pro Sekunde.

(Vorsicht : diese Erklärung ist extrem vereinfachend - über den Timer - 8253 - lassen sich ganze Bücher füllen !)

Bei jedem Takt des Timers, eben cirka 18,2 Mal/Sekunde, wird eine Routine tätig, über welche das Betriebssystem eine Reihe von zeitabhängigen Regelarbeiten erledigt. Selbstverständlich ist hier an erster Stelle die Systemuhr selbst zu nennen. Aber auch z.B. die Nachlaufzeiten von Diskettenlaufwerken werden über diesen Interrupt gesteuert. Diese Arbeit überläßt das Betriebssystem nicht der Vergeßlichkeit des Programmieres !

Daß man die Routinen von INT 8 besser gar nicht erst antastet, geschweige denn den Vektor auf eine eigene Routine lenkt, sollte selbstverständlich sein. Die für INT 8 definierten Codes sind für das System unerläßlich. Experimente mit ihnen führen unweigerlich zum Systemabsturz.

Mikrosoft hat allerdings von Anfang an daran gedacht, sich selbst und seinen Programmierern ein Hintertürchen offen zu halten, über welches es möglich sein soll, eigene, zeitgesteuerte Routinen auf der Maschinenebene abarbeiten zu lassen. Dieses Hintertürchen heißt `INT 1Ch`. Der Interrupt 8 beendet seine Arbeit nicht wie sonst üblich mit einem `IRET` Aufruf, sondern ruft seinerseits am Ende seiner Routinen selbst einen Interrupt auf - 1Ch.

Der Interrupt 1Ch enthält aber wiederrum nichts anderes, als den notwendigen `IRET` Aufruf - die Anweisung, die Interruptroutine zu beenden. Damit präsentiert sich `INT 1CH` als Dummy. Diese Dummy-Funktion ermöglicht es dem Programmierer, an den Code von INT 8 eigene Routinen anzuhängen, ohne den eigentlichen Timer Interrupt zu stören. Voraussetzung ist allerdings, wie oben gesagt, daß die selbstdefinierten Routinen, die in den Vektor von 1Ch eingehängt werden, nicht zu den beschriebenen zeitkritischen Fehlern führen !

Fortsetzung folgt in VD 1/93

Im nächsten Heft:

KEYB8B (Arndt Klingelnberg)
Zufallsdaten? Nichts leichter als das (Jörg Staben)
Fis(c)hing Forth (Friederich Prinz)
Schweinekiste (Jörg Plewe, Ulrich Hoffmann)
Ein FILE DUMPen (Wolf-Helge Neumann) und und und...

Was tickt denn da...

von Thomas Beierlein

Was tun wenn man genaue Millisekunden braucht und F-PC keine zu bieten hat? Ein Blick in den PC bringt die Lösung.

Das Problem

In der VD 2/92 S.22. klagte *Arndt Klingelnberg* über die ungenaue Arbeit des Wortes MS im F-PC. (MS nimmt eine Zahl n entgegen und soll die entsprechende Zeit, in Millisekunden gemessen, warten.) Probiert man selbst einmal die Sequenz

```
TIMER 1000 MS
```

aus, ist man erstaunt wie weit die reellen Zeiten daneben liegen. (bei mir ergibt dies 1,37ohne aktiven Debugger und 4,... mit auf einem 8 MHz AT und 0,13 und 0,39 auf einem 33 MHz 386er).

Die Wurzeln allen Übels

Ein Wunder ist das ganze allerdings nicht. Wie schon im F83 und in anderen Forth-Systemen wird der MS-Befehl mit einer geschachtelten Schleife realisiert, in der die entsprechende Zeit totgeschlagen werden soll.

```
: MS  ( n -- )
      0 DO xxx
        0 DO LOOP
      LOOP
      ;
```

Mit `xxx` wird, je nach Geschwindigkeit des Computers, festgelegt, wie oft die innerer Schleife durchlaufen werden muß, um eine Millisekunde zu vertrödeln. Im F-PC wird diese Konstante beim Start bestimmt und in einer Variablen `FUDGE` festgehalten. Es gibt jedoch einige Probleme bei ihrer korrekten Bestimmung, vor allem auf unterschiedlichen Rechnertypen. Im Endeffekt erhält man stark differierende Zeiten. Dazu kommen noch weitere zwei Probleme. Auf eines davon hat Arndt Klingelnberg schon hingewiesen. Voraussetzung, um konstante Zeiten zu erreichen, ist, daß die FORTH-Worte stets die gleiche Zeit zur Abarbeitung benötigen. Dies ist nicht mehr der Fall, wenn der Debugger aktiviert wird. Dabei werden alle Code-Routinen so gepatcht, daß anstatt der im Wort integrierten `NEXT`-Funktion die zentrale `NEXT`-Routine angesprungen wird. Dadurch verlangsamt sich die Abarbeitung der Code-Worte und damit natürlich auch die Bearbeitung von `MS`. Das zweite Problem entsteht dadurch, daß, im Gegensatz zu meinem Beispiel, in der realen Version von `MS` in der innersten Schleife das Wort `PAUSE` aufgerufen wird. Dies gewährleistet, daß bei Verwendung des Multitaskers ein längerer MS-Befehl nicht die Bearbeitung der anderen Tasks blockiert. Denn eigentlich ist der Rechner ja während dieser Wartezeit gerade für andere Aufgaben frei. Die Dauer dieser anderen, in PAUSE ausgeführten, Tasks ist jedoch unbestimmt. So verlängert also jedes Task die Dauer einer mit `MS` bestimmten "Millisekunde" noch weiter.

Stichworte:

F-PC
MS
RTG (Echtzeit-Uhr)
BIOS INT 15

Die Lösung?

Was kann man nun gegen dieses ganze Dilemma tun? Die Lösung finden wir in unserem Alltag. Wir legen uns ja auch nicht abends ins Bett und zählen die Sekunden bis es Morgen ist. Stattdessen kaufen wir uns einen Wecker und verwenden diesen um uns wecken zu lassen. Es wäre also gut, wenn wir eine externe Baugruppe hätten, die unabhängig vom Programm den Ablauf der Zeit überwacht. Schaut man sich den PC einmal näher an, so findet man diese Baugruppe seit den ersten AT's standartmäßig eingebaut (nur XT-Besitzer haben diesmal Pech). Jeder dieser Rechner verfügt über eine Echtzeit-Uhr, die nicht nur die Zeit bei ausgeschaltetem Rechner weiterlaufen läßt, sondern auch 1024 mal pro Sekunde, also fast im gewünschten Millisekunden-Takt, einen Interrupt auslösen kann. Normalerweise ist dieser Interrupt in der Echtzeit-Uhr gesperrt. Es müßte also reichen, eine Interrupt-Service-Routine zu installieren und den Interrupt zu aktivieren. Gesagt, getan. Das Handbuch wurde hergenommen, die Register der Echtzeit-Uhr gesucht und erste Experimente angestellt. Nach ganz kurzer Zeit kam der erste Erfolg! Der Rechner piepte im Dauerton und nichts ging mehr... Nun gut, damit muß man rechnen.

Die Lösung!!!

Beim weiteren Suchen nach Details entdeckte ich dann, daß die Welt (des Rechners) doch nicht ganz so böse

eingerichtet war. Im BIOS gibt es eine Routine die genau auf unsere Zwecke zugeschneidert ist und uns den ganzen Ärger mit Interrupts, Vektoren, Abstürzen usw. erspart. Die Lösung heißt BIOS-Interrupt 15H, Unterfunktion 83H. Diese BIOS-Funktion, *AT Extended Services, Event Wait* genannt, nimmt eine Zeitangabe in Mikrosekunden und die Adresse einer Speicherzelle (mind. 1 Byte) für die Rückmeldung entgegen. Sie startet die Echtzeit-Uhr und kehrt zum F-PC zurück. Alles weitere ist Sache des BIOS. Nach jeweils 976 Mikrosekunden schlägt die Echtzeit-Uhr zu, löst den Interrupt 70H aus, der normalerweise im BIOS landet, und die passende Interrupt-Routine zieht von unseren gewünschten Mikrosekunden 976 ab - bis sie alle sind. Daraufhin setzt die Routine ein Bit in der angemeldeten Speicherstelle und schaltet den Interrupt wieder ab. Für unsere gewünschte neue MS-Variante brauchen wir also nur die Zeit in Mikrosekunden umzurechnen, die Speicherstelle zu löschen, den INTH aufzurufen und danach abzuwarten bis unsere Speicherstelle nicht mehr Null enthält. Währenddessen können wir wie im Original `PAUSE` machen und die anderen Tasks arbeiten lassen.

Das Programm

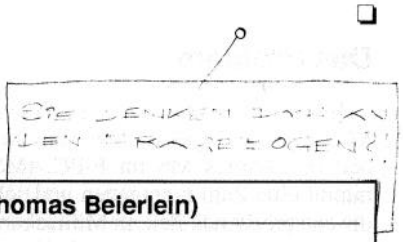
Das Programm ist im Listing abgedruckt. Wir brauchen eine Variable als Speicherstelle für die Fertigmeldung, eine Code-Routine, die den INTH passend aufruft und unser neues `MS`. In der Code-Routine werden die Register mit den geforderten Parametern geladen und der Interrupt aufgerufen. Da F-PC das ES-Register selbst nutzt, ist es erforderlich, dieses auf dem Stack zu retten. Wer mit Assembler-Programmierung nicht vertraut ist, sollte mir einfach glauben, daß es funktioniert.

Ein neues Problem?

So weit, so gut. Doch fern am Horizont taucht ein neues Problem auf - allerdings nur für die Unverbesserlichen, die mit F-PC "Multi-Tasken" wollen. Wir haben im PC nur einen Timer drin, ergo kann auch nur einer laufen und auf unsere Millisekunden aufpassen. Es darf also nur ein Task das neue MS aufrufen. Das ist in der Praxis eine ernsthafte Einschränkung. Es sind einige Lösungsvarianten für dieses Problem denkbar. Zum einen kann man in den anderen Tasks mit dem ursprünglichen MS arbeiten - das hat natürlich die oben genannten Nachteile. Die beste Lösung ist es aber, wenn man ein separates Task als zentralen Zeitverwalter einführt. Alle anderen Tasks, die gern ein wenige pausieren wollen, könnten sich dann bei ihm abmelden, beruhigt schlafen legen (`SLEEP`) und bei passender Gelegenheit wecken lassen.

Doch davon vielleicht in einem anderen Märchen...

❑



Listing zu: Was tickt denn da... (Thomas Beierlein)

```
\ Eine neue Variante fuer das MS-Problem

Der alte MS-Timer ist fuer genaue Zeitmessungen ungeeignet, da er in einer
DO-LOOP mittels PAUSE die Zeit vertroedelt.
Abgesehen von unterschiedlichen Maschinen schlagen auch noch Multitasking
und Interrupts in unterschiedlichster Haeufigkeit zu.
So ist MS ist Zeitmasstab nicht zu gebrauchen.

Die hier gezeigte Variante baisert auf der Nutzung der Echtzeit-Uhr im
Rechner. Sie wird ueber den INT 15H Unterfunktion 83H (Event Wait)
aktiviert.

!!!!!!!!!!!!!!!!!!! ACHTUNG !!!!!!!!!!!!!!!!!!!
Funktioniert erst ab Maschinen vom AT-Typ, nicht
in XT's
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

Decimal
ONLY FORTH ALSO DEFINITIONS
PREFIX


variable fertig                 \ Time-out-Flag

code start-RTC   ( d -- )       \ Startet den Echtzeittimer des
                                \ BIOS (RTC -- Real Time Clock)
        pop cx
        pop dx                  \ Zeitdauer in Mikrosekunden nach CX,DX
        push es
        mov ax, cs
        mov es, ax              \ Adresse des Flags, in dem das Ende
        mov bx, # fertig        \ mitgeteilt wird nach ES:BX
        mov ax, # $8300         \ Funktionsnummer nach AH und AL
        int $15                 \ und los geht es
        pop es
        next                    \ das war es schon
end-code


: ms   ( n -- )
        1000 *d                 \ ms in Mikrosekunden umrechnen
        fertig off              \ loesche Flag
        start-RTC               \ und starte Echtzeit-Timer
        begin
                pause           \ tue ev. etwas vernuenftiges
                fertig @        \ bis die Zeit um ist
        until ;

\S
Die Qualitaet kann man mit
                TIMER 10000 MS
oder aehnlichen Varianten testen.
```

❑

NUN endlich

Execute> TIB für die Light-Show!

von Arndt Klingelnberg

Straßburgerstr.12, 5110 Alsdorf, Tel.: 02404-61648, FAX: 02404-63039

Ein 'DOS-CommandTail-Option-Tool' für tCOM am Beispiel einer Zeit-Anzeige unter F-PC und DOS (tCOM : NUN.com)

Jörg Staben regte mich mit seiner Light-Show (VD92 II) dazu an, eine PulsdauerModulation der Keyboard LEDs zu versuchen. Aber mein Prozessor ist zu schlapp. Mein 286-16 schafft das zwar, aber die µPs in den Keyboards sind wohl ungünstig programmiert. Schade, gerade die sinnvollsten Programme werden durch unzureichende Hard- und FirmWare immer wieder vereitelt.

Das in diesem Artikel gewünschte `Execute> TIB` für tCOM ließ mich auch scheitern; das ist nämlich nicht sinnvoll zu realisieren. Das trifft auch für -- was Jörg Staben wohl meinte -- TIB-Interpret oder DOS-tail-INTERPRET zu. Die Funktionalität der von mir vorgestellten Alternative und die weiteren Lösungsvorschläge sollten jedoch voll ausreichen. Als Beispiel dient ein nützliches Programm zur Anzeige von Datum und Uhrzeit.

Nützlich? Das gibt's doch wie Sand am Meer !

Ja, aber ... eben nicht umschaltbar und mit Help-Screen, und auch nicht mit meinem geliebten, weil logischen und sortierbaren, international nicht mistverständlichen Datumsformat: 92-09-28 00:17 (das war übrigens jetzt NICHT NUN.com , sondern Alt-O P im F-PC-Editor, aber NUN.com hätte es sein können, wenn auch nicht als 'PASTE' in den Text hinein.)

Das Interesse gilt der universell nutzbaren Lösung, die die DOS-typischen "/"-Optionen aus der DOS-Kommandozeile extrahiert und eine tCOM-Anwendung entsprechend reagieren läßt.

F-PC dient zum interaktiven Testen des Programms. Weitere Hinweise zur gezielten Weiterentwicklung und/oder Nutzung für nicht F-PC-ler schließen sich an, wie auch Tips zu der im Quelltext genutzen halbautomatischen Möglichkeit der Versions-UpDate-Dokumentation.

Ich habe davon abgesehen, über F-PC ein Programm mit 250 KB zu erstellen (`Fsave`), das gerade mal das Datum anzeigt. Ich weiß, per `TURNKEY` geht es auch viel kleiner, aber kein Vergleich zu tCOM. Für F-PC wird gezeigt wie TIB (der Terminal Input Buffer) simuliert wird, um vergleichbare Verhältnisse zu tCOM zu

Stichworte

F-PC
Tcom
TIB
DOS-Command-Line
Country-Info
Optionen

Listing zu: NUN endlich, von Arndt Klingelnberg

```
\ NOW.seq       prints Day_of_week, date, time or ...          \ akg91aug25
}

 \ akg91dec02   added trailing CR for TCOM, added "
                \ and changed HUNDREDS? to MS?          \ see: tTIMER-X.seq

 \ akg91sep05   Please mind that in  Tcom  DS:  and  CS:  are NOT the same SEG


        nun eine variable Version:      deutsch -- englisch
                                        verschiedene Formate wählbar.


        this version of 'NOW' is now quite flexible:
                                        german -- english
                                        different time and date formats


        for the different possible formats see the help message  .help


        to compile this, execute:               TCOM   NOW  /OPT /NOINIT
{


\ ONLY
\ HOST ALSO  FORTH ALSO  COMPILER ALSO
\ TARGET DEFINITIONS ALSO
\ ASSEMBLER ALSO  FORTH                         \ bug in TCOM v.2.12

needs condc-tf.seq      \ CONDitional Compiling for TCOM and F-PC
        \ &&F  -lines are valid for F-PC
        \ &&T  -lines are valid for Tcom
        \ ||ak -lines are NOT valid for F-PC ak-version

&&T     needs ttimer-X.seq      \ with akg enhanced functions !!! auch deutsch
&&F ||ak    needs ttimer-X.seq  \ with akg enhanced functions !!! auch deutsch

&&F     ONLY FORTH ALSO DEFINITIONS

&&F ||ak        ' ?cs:          ALIAS #?ds:

FALSE   VALUE options?  \ false for NO option,
                        \ 1 for just a '/', n+1 for n option-letters

: ">"options    ( addr len -- addr' len' )
                \ AL-string holding  ALL  /option chars
```

⇨

erhalten. Das können allerdings nur Besitzer der AK-Version bequem nutzen. Die Originalversion gestattete es nicht, interaktiv einen String einzugeben, zumindest nicht ohne Verrenkungen: Das Laden von $TEST

```
" Dies ist ein STRING"
">$TEST
```

ist im Original nur INNERHALB einer Colon-Definition möglich.

Warum kein EXECUTE>TIB ?

Einer Target Anwendung das Wörterbuch und die Hilfsmittel hierzu mitzugeben, ist nicht möglich bzw. bläst das Programm unnötig auf. tCOM kann das (eher behelfsmäßig) mit der Option: /FORTH). INTERPRET sollte daher hier (!) nicht angestrebt werden. Ein EXECUTE , das auf eine Auswahl von Token (CFA == Code Field Address) zur Ausführung zugreift, ist schon eher möglich, EXECUTE> TIB jedoch nicht. Diese Namenswahl suggeriert, daß TIB auswechselbar ist z.B. EXECUTE> xy-$buffer , TIB ist aber kein 'counted'-string, die gültige Länge von TIB steht in der Variablen #TIB , und die steht irgendwo im RAM. TIB-EXECUTE bietet sich dagegen an, oder DOS-TAIL-EXEC , aber dazu später. Spitzfindkeiten? In der Anwendung ja! Forth-mäßig, also im Ablauf des Programms, sind es Welten!

Und 'NUN' die Lösung: Die möglichen Optionen aus der DOS-Kommando-Zeile werden der tCOM-Anwendung zuerst bekannt gemacht und dann z.B. über eine CASE-Anweisung ausgewertet.

tCOM stellt den Befehl DOS_TO_TIB zur Verfügung, damit steht die DOS-Kommandozeile, also alles das, was hinter dem Programmnamen in der DOS-EingabeZeile stand, im TIB . Das funktioniert ohne cMOVE , denn TIB zeigt nun einfach auf diese geänderte Adresse. TIB legt die Anfangsadresse auf den Stack und TIB# @ die String Länge, insgesamt also einen AL-string. (Zusatzinformationen: In F-PC gibt es den äquivalenten Befehl DOS>TIB , der aber hier nicht benötigt wird. Wird im

Fortsetzung des Listings zu: NUN endlich

```
            \ AL-string enclosed by '/' and  BL  or  END
        ?ds: !> sseg            \ we will be searching in the  DataSeg
        bl skip                 \ skip all leading blanks
        '/' scan                \ point to '/'
        2dup bl scan nip        \ point to first blank after options
        -                       \ we just need the remaining length
        dup !> options?         \ zero string ?  (no '/' found ?)
        '/' skip ;              \ point after '/'


\ bug in v.2.12  does not recognize following  ( DO ... ) LOOP

: ">ASCII       ( addr len -- nk ... n1 k | FALSE )
                \ convert  AL-string  to a list of ascii-values on stack
                \ FALSE for ZERO-string found
                \ last char == NextOfStack
        dup 0
        ?DO     drop  I 1+   over I + c@
                -rot
        LOOP nip ;                              \ drop addr

: .help         ( -- )          \ print help screen
        ."       v"
    cr ."  /? /h - Help                                 HILFE"
    cr
    cr ."     /n - No time                              Nur Datum"
    cr ."     /u - Usa            format  M/D/Y"
    cr ."     /g - German         format  D.M.Y"
    cr ."     /i - International format  Y-M-D"
    cr ."     /l + 3 Letter-month        yyMMMdd"
    cr ."     /b -                       jjMMMtt        3 Buchstaben monat"
    cr ."     /v -                       jjjj_MONAT_tt  Volle länge monat"
    cr ."     /y - full length Year      19xx"
    cr
    cr ."     /w - with Weekday                         mit Wochentag"
    cr ."     /e +      English "
    cr ."     /d -                                          Deutsch"
    cr
    cr ."     /t - Time only                            nur zeiT"
    cr ."     /s - with Seconds                         mit Sekunden"
    cr ."  /p /m - Precise time (with Ms)               Präzise zeitangabe"
    cr ."        ^"
    cr ." default  /   ==  /LE    ==  91sep03__20:45_"
    cr ." no option    ==  /DWYV  ==  Dienstag,_1991_September_03__20:45_"
    cr
 cr ." »NOW«    Arndt's  Datums Anzeige utility      TCOM/F-PC  91dec02v.1.5  "
    ;


false   VALUE HELP?             \ some flags set by options
false   VALUE DEUTSCH?
true    VALUE ENGLISH?
false   VALUE WEEKDAY?
true    VALUE TIME?
true    VALUE DATE?
false   VALUE ms?
false   VALUE seconds?

: set-default   ( -- )
                false   !> HELP?        \ some flags set by options
                false   !> DEUTSCH?
                true    !> ENGLISH?
                false   !> WEEKDAY?
                false   !> ms?
                false   !> seconds?
                true    !> time?
                true    !> date?
                false   !> century?
                yyMMMdd ;

set-default     \ set default values
                \ valid for 'empty' options:  '/' with NO trailing letter

: SET-options   ( char -- )
```

➪

'Command-Tail' REdirection eingegeben, also '>' '|' '<' so endet der transferrierte String vor diesem Zeichen.)

Die StringNamen in F-PC zeichnen sich meist, wenn auch nicht konsequent, bei AL-strings durch ein VORgestelltes >"< und bei counted strings (Adresse zeigt auf count-Byte) duch >$< aus.

">"options extrahiert aus einem AL-String einen String der nach einem / beginnt und mit einem Space als Delimiter oder END aufhört, und zwar wieder als AL-string.

Optionen sind fortlaufend nach dem "/" einzugeben, also ohne eingeschlossene Blanks. Das ist DOS-üblich. Einige DOS-Programme vertragen auch Blanks, andere wollen Blanks, wiederum andere wollen einen "/" je Option. Wer möchte, kann ja alles einbauen (und bitte in der VD vorstellen).

Es dürfen hier beliebig viele Optionen sein:

```
nun /dvwyp
```

wäre ein Beispiel (und zwar die ausführlichste Variante meines Lieblingsformats): Montag, 1992 September 28 01:34'45".610

Dieses wurde jetzt im DOS-shell (Ctrl-J) ablaufen gelassen und mit SNIPP ausgeschnitten und hierhin ge-'pasted'.

">ASCII wandelt einen AL-string in einzelne ASCII -Werte um. TOS (Top Of Stack) enthält den Zähler. Steht dort '0' , so ist nichts zu tun, steht dort 5, so liegen 5 weitere Werte tatendurstig auf dem Stack. Wichtig mag sein, daß die ERSTE Option zu unterst liegt, also typischerweise ZULETZT bearbeit wird und damit die HÖCHTE Priorität hat. Die in der DOS-Kommando-Zeile ZUERST angegebene Option dominiert, da sie alle vorher getätigten Einstellungen noch beinflussen kann. (Ob es eher andersherum sein sollte ?¿? Wer das letzte Wort hat ...).

Warum gibt sich der Test in F-PC so 'kompliziert' ?

Nun, ich hatte mit 'NUN' anfänglich (auch) Probleme mit vorlaufenden und folgenden Spaces in der EINgabe und AUSgabe. Mit der vorge-

Fortsetzung des Listings zu: NUN endlich

```
        CASE            \ use   BL OR   to convert to lower case
        'b'     OF      jjMMMtt         ENDOF           \ Buchstabe für monat
        'd'     OF      ON> DEUTSCH?
                        OFF> ENGLISH?   ENDOF           \ Deutsch
        'e'     OF      ON> ENGLISH?    ENDOF           \ English
        'g'     OF      D.M.Y           ENDOF           \ German        format
        'h'     OF      ON> help?       ENDOF           \ HELP
        'i'     OF      Y-M-D           ENDOF           \ International format
        'l'     OF      yyMMMdd         ENDOF           \ Letter for month
        'm'     OF      ON> seconds?
                        ON> ms?         ENDOF           \ Milli seconds == /p
        'n'     OF      OFF> time?      ENDOF           \ No time
        'p'     OF      ON> seconds?
                        ON> ms?         ENDOF           \ Precise time == /m
        's'     OF      ON> seconds?    ENDOF           \ Seconds
        't'     OF      OFF> date?      ENDOF           \ Time only
        'u'     OF      M/D/Y           ENDOF           \ Usa           format
                        \ I definitely hate this format, but ... here it is.
        'v'     OF      jjjj_MONAT_tt   ENDOF           \ Volle länge monat
        'w'     OF      ON> weekday?    ENDOF           \ Weekday  Wochentag
        'y'     OF      ON> century?    ENDOF           \ full-length Year
        '?'     OF      ON> help?       ENDOF           \ ?
        drop ( OTHERCASE )
        ENDCASE ;

: .date&time    ( -- )
        date?
        IF      weekday?
                IF      ENGLISH?
                        IF      .weekday
                        ELSE    DEUTSCH?        \ english? has priority
                                IF      .wochentag
                                THEN
                        THEN ',' emit space
                THEN
                .date space
        THEN
        TIME?
        IF      GETTIME FORM-TIME COUNT
                seconds?
                IF      ms? 0=
                        IF      4 -                                     \ akg91dec02
                        THEN
                ELSE    8 -                                             \ akg91dec02
                THEN type space
        THEN ;

\       F-PC: some words needed for easy debugging  ( faking TIB )

&&F     CREATE test-string   0 c, 150 allot     \ plenty of room

&&F     TIB     VALUE 'TIB-save
&&F     #TIB @  VALUE #TIB-save
&&F     >IN @   VALUE >IN-save

&&F     : TIB-save      TIB !> 'TIB-save        #TIB @  !> #TIB-save
&&F                     test-string count  #TIB !  'TIB !
&&F                     >IN @  !> >IN-save      >IN OFF ;

&&F     : TIB-restore   'TIB-save 'TIB !   #TIB-save #TIB !   >IN-save >IN ! ;

&&F     : ">test-STRING         dup 150 > ABORT" string too long    "
&&F             dup>R  test-string 1+ swap cmove    R> test-string c! ;

: now   ( -- )          \ the MAIN word
        DECIMAL                         \ always select decimal
        CAPS ON                         \ ignore cAsE
&&T     DOSIO_INIT                      \ init EMIT, TYPE & SPACES
&&T     DOS_TO_TIB                      \ move command tail to TIB
&&F     TIB-save                        \ set TIB to test-string (temporary)
        tib  #tib @  ">"options
        options?
        IF      ">ASCII         ( S   nk ... n1 k )
                \ convert to a list of ASCII values on stack
```

stellten Lösung simmuliere ich exakt DOS_TO_TIB des tCOM und arbeite nicht direkt vom normalen TIB , aber bitte:

```
nun /?
...
nun /ni
```

sollte auch direkt in F-PC funktionieren:

```
92-09-28
```

Das wurde nun wärend des Editierens in einem 4th-Shell (Alt-Enter, nur AK-Version) erzeugt. Dazu zuerst die Hilfe-Seite, dann noch etwas probiert, dann die endgültige Lösung über

```
>B nun /ni
```

in BROWSE.prn 'eingefangen' und später in diesen Text importiert.

Orientiert habe ich mich übrigens etwas an LOOK.seq . Das habe ich ausführlich kennengelernt, weil ich es umbaute. Dieser so nützliche String-Finder (kann ganze Platten durchscannen) mußte bei mir auf Viren(?)-Suche angesetzt werden. Einzelne Cluster der Festplatte waren mit NUL gefüllt. Im Directory sahen alle Datein normal aus, nur lesen und ausführen konnte man/fra sie nicht. NUN wiederum dient in solchen Fällen dazu, in ein *.LOG-file den Zeitpunkt einzugeben. Also im NUL-LOOK.bat steht dann irgendwo

```
nun >> ~%1.log
```

(bitte zweimal ein '>' sonst wird ~%1.log überschrieben). Mit den vorigen Sätzen wollte ich übrigens auch die Ventura Publisher-'Return-Stack' Probleme testen, bzw. die 'Setzerei'. Der VenturaPublisher lädt mit >R nicht den ReturnStack sondern macht etwas 'Mysteriöses'.

Fortsetzung des Listings zu: NUN endlich

```
                set-default
                0 ?DO   ( S  ... char )
                        bl or                   \ convert to lower case
                        set-options
                LOOP
                HELP?
                IF      .help
                ELSE    .date&time
                THEN
        ELSE    2drop                           \ spezielle Deutsche Version
                .wochentag ',' emit  space
                jjjj_MONAT_tt  .date  space
                GETTIME FORM-TIME COUNT 8 - type  space                 \ akg91dec02
        THEN
&&T     cr                                                              \ akg91dec02
&&F     TIB-restore
        ;

&&T \S          never load if  TCOM
\ \s ++++++++++++++++++ samples for F-PC ++++++++++++++++++++++++++++++++

" /h   " ">test-string   cr 30 spaces  '|' emit  now  '|' emit
cr  3 seconds
"   /h" ">test-string   cr 30 spaces  '|' emit  now  '|' emit
cr  3 seconds
" " ">test-string  cr 30 spaces  '|' emit  now  '|' emit
cr  1 seconds
"      " ">test-string  cr 30 spaces  '|' emit  now  '|' emit
cr  1 seconds
"   / " ">test-string    cr 30 spaces  '|' emit  now  '|' emit
cr
```

❑

Noch etwas zu den &&F und &&T -Compiler Directiven. Es ist ein an UNIX orientierter Versuch, mit dem UNlogischen Gebrauch von \u \if \+ \- \F \T \|u usw. klarzukommen. Sie entstammen einer ganzen Klasse von vereinfachten und sehr funktionellen Compiler Direktiven, auch für mehrzeilige Kommentare, zudem global schaltbar und auch insbesondere für Source-Samples geeignet. Interesse?

Was zeichnet den Quelltext noch aus ?

Lade-Direktiven, ob \F oder &FF , machen den Quelltest sicherlich nicht übersichtlicher, aber es ist hilfreich, EINEN Quelltest sowohl für F-PC als auch tCOM zu haben, oder für 286er und >=386er oder ... .

Zu Anfang stehen VersionsKommentare: Stichworte zu Änderungen. Der Sprung in die dritte Zeile, der Einschub einer neuen Zeile und das Namenskürzel mit Datum und mit '\'-KommentarZeichen erfolgen in F-PC-ak halbautomatisch. Ein 'Enter' nach dem Kommentar springt dann wieder an die EditierPosition.

Ebenso erfolgt das Stamping am ZeilenEnde auf Tastenbefehl. Ich habe in F-PC das automatische Stamping der VolksForth-Screens vermißt. Und unauffällige Veränderungen ohne Vermerk sind mir ein Graus.

Das TITLE: am QuelltextAnfang ist ein 'defining word' ähnlich ANEW , aber nun überall nutzbar. Es wird hier ein automatisches FORGET ermög-

now.seq (Tom Zimmers Kurzversion)

```
: main          ( -- )
                DECIMAL                         \ always select decimal
                INIT-CURSOR                     \ get intial cursor shape
                MARGIN_INIT                     \ initialize margins & TAB
                50 FUDGE !                      \ init MS timer, GUESS!!
                CAPS ON                         \ ignore cAsE
                ?DS: SSEG !                     \ init search segment
                DOSIO_INIT                      \ init EMIT, TYPE & SPACES
                $FFF0 SET_MEMORY                \ default to 64k code space
                DOS_TO_TIB                      \ move command tail to TIB
                COMSPEC_INIT                    \ init command specification
                ." Date: " .DATE cr
                ." Time: " .TIME ;
```

❑

licht.

OTHERCASE wurde geschaffen, um aufzuzeigen, daß CASE in F-PC kein exakter C.E.Eaker Case-Construct ist (sondern modifiziert). Das notwendige DROP vor ENDCASE wird aber sehr sehr gerne vergessen oder falsch plaziert: Oft fällt das nicht auf, da es oft gerade dort schlecht zu testen ist. Das ist gefährlich. Auch bei T. Zimmer gibt es Beispiele. Darum OTHERCASE. Damit OTHERCASE über SEE sinnvoll decompiliert werden kann, sollte es kein ALIAS sein, sondern einen eigenen 'body' haben.

Weitere Ausbau - Möglichkeiten:

Es kann aus DOS der eingestellte "Switch prefix character" ausgelesen werden: Er ist fast immer "/" , nur selten "-" und noch seltener irgenetwas anderes. Wollen wir es besser als 98% aller Programmierer machen, so gibt int $21 func $37 auf DL den switchar aus. Diesen sollte ">"options dann nutzen.

Der 'command-tail' steht im PSP (Programm Segment Prefix) an offset $80, und zwar forth-gemäß als Counted-String. In F-PC und tCOM gibt

```
?cs: $80 countL typeL
```

diesen String aus.

```
?cs: 0 250 Ldump
```

Zeigt den String ab ....:0080 (count-byte).

Da in F-PC (nicht aber tCOM) DatenSegment und CodeSegment dasselbe sind, und zudem das 'default' Segment, reichen in F-PC auch 'kurze' Operatoren, also:

```
$80 count type
```

bzw.

```
dos-line count type
```

und

```
0 250 DUMP
```

Sollte unter anderen Forth Systemen der Zugriff nicht funktionieren, bleibt immer der Weg über int $21 func $62 bzw. int $21 func $51, um DOS um die Adresse des PSP zu bitten.

DOS spuckt über int $21 func $38 subfunc $00 die 'country' Information aus. Ein Datenbuffer (mind. 32 Bytes) muß vorhanden sein, dessen Anfangsandresse wird vor dem Aufruf in die Register DS:DX geladen. (int $21 func $65 ist übrigens die hierzu passende extended function). An Offset 0 erscheint die wichtigste Information, und zwar die Codierung fürs Datumsformat: USA==0, Deutschland==1, International==2. Die Offsets der Seperator-Zeichen sind für Datum $B, Time $D und Dezimale 9. 12/24h-Welten werden an Offset $11 eröffnet. So könnte nun bzw. now im default Status ganz automatisch dasselbe tun wie DOS, nur eben besser. Es bleibt noch Arbeit zu tun.

ASSOCIATIVE: (das ist ein defining word und soetwas wie ein Anti-INDEX in ein ARRAY . Nicht mit einem Index, also einer Zahl aus einem lückenlosen, mit 0 beginnenden Zahlenfeld in ein ARRAY hineinschauen und 'irgendwelche' Zahlen herausbekommen, sondern eine 'beliebige' Zahl hineingeben und einen Index herausbekommen. Wer es in F-PC sucht: DECOM.seq . In tCOM fehlt es (noch?). Wie weit eine Umstellung von CREATE DOES> auf CREATE ;CODE auch in tCOM einen Geschwindigkeitspusch gibt?

Und die Ergebnisse von mit ASSOCIATIVE: gezeugten Kindern können mit EXEC: ausgewertet werden. Das wäre dann eine Alternative zum Case-Construct und könnte TIB-EXEC genannt werden. Aber so etwas wie ">ASCII wird weiterhin benötigt.

Sollen dann doch ganze Worte und nicht nur EinzelBuchstaben als Optionen auf einfache Weise erkannt werden, steht doch ein MINI-Wörterbuch und ein MINI-FIND hierzu als Tool aus. Also TIB-MINI-INTERPRET. Auch hierzu gibt es eine Lösung, ideal auch zur Paßwort Abfrage, da diese im Programm nicht (!) auftreten.

NUN ist aber Schluß.

Nachbemerkung:

NOW.seq versucht das file tTIMER-X.seq zu laden, das ist ein erweitertes und entbuggtes tTIMER.seq . Im Normalfall reicht das originale File aus, jedoch sind dann einige Optionen nicht im Zugriff (Lösung: Zeilen in HELP und im Case-Construct per '\ ' ausschalten !). Ähnliches gilt für F-PC. Dort gibt es ein TIMER.seq , was eigentlich dasselbe kann. Arbeitet Tom Zimmer in TIMER.seq mit DEFERred Words , so nutzt er in tTIMER ein Value--EXEC: Konzept, das muß wohl für tCOM verdaulicher sein.

Der Platz hier ist begrenzt. Wer tTIMER.seq haben möchte und NOW.seq und ein 3.355 Byte kleines (!) NUN.exe , bekommt das alles gegen SAFU-MFD (*) oder über die Forth-Mailbox.

❑

*) *SAFU selbst adressierter FreiUmschlag MFD mit formatierter Diskette .*

EuroFORTH '92

23. - 25. Oktober 1992 Southampton, UK

Ein Bericht von Dr. Wolf Wejngaard
Neuhöflirain 10, CH-6045 Meggen

Die jährliche europäische Forth-Konferenz fand dieses Jahr wieder im gemütlichen *Potters Heron* Inn in Ampfield bei Southampton statt mit 47 Teilnehmern aus 12 europäischen Ländern, den USA und Kanada.

Die neue Bezeichnung der Konferenz ('EuroFORTH' statt dem nur Eingeweihten verständlichen 'EuroFORML') wurde allgemein begrüßt. *John Hall*, Präsident der Forth Interest Group und diesjähriger "amerikanischer Gast", meinte sogar, daß auch die Asilomar'er FORML eine neue Bezeichnung vertragen könnte, was nun aber doch des Guten zuviel wäre.

Diesem Schritt zur Versachlichung entsprach der Geist der Veranstaltung. Es war ein wohltuender Zug zur Professionalität spürbar, der sich in einem auf kommerziellen Erfolgen gründenden Selbstbewußtsein zeigte. Nichts gegen geniale Hacker (das sind wir ja alle irgendwie), aber wir brauchen auch die Anwender, die mit überzeugenden (sprich professionellen) Lösungen das Potential von Forth an konkreten Problemen zur Geltung bringen. Ich gewann den Eindruck, daß sich Forth, zumindest auf den britischen Inseln, eine bleibende Stellung errungen hat.

Was nicht heißen will, daß Forth nun abgeschlossen ist, im Gegenteil. *Markus Dahm* zeigte eine Erweiterung zu seinem schönen objektorientierten Forth (natOOF), die ein Programmieren möglichst nahe an der natürlichen Sprache erlauben soll. *Andrija Maricic* stellte eine objektorientierte Entwurfssprache ETF1 vor und *Vassil Lolov* beschrieb ein Experten System für Echtzeit-Anwendungen. Für die Freunde des "Native Code" stellte *Anton Ertl* ein neues Compilerkonzept vor, das auf großes Interesse stieß. *Gordon Charlton* setzte seine umfassende Arbeit am Forth String Matcher (FOSM) fort, zeigte außerdem einen "hysteretischen" Heap (wann und wie soll man den Garbage collecten?) und erfreute die Konferenz mit dem neuen universellen Struktur-Wort `END`, das anstelle von `THEN`, `AGAIN`, `REPEAT`, `ENDOF` und `ENDCASE` tritt und vielfache WHILEs zuläßt.

Marcel Hendrix und *Erwin Dondorp* führten ihr Transputer-Forth vor (tForth), das sehr reif aussieht. Dagegen ist das freeForth von *Cristophe Lavarenne* momentan noch ein interessantes Konzept. Christoph verblüffte mit der Idee, den Interpreter immer als Compiler einzusetzen, also jede Eingabe zunächst zu compilieren und dann, wenn keine neue Definition vorliegt, auszuführen. Der Vorteil ist, daß die Strukturworte interaktiv verwendet werden können.

Prof. *Sergei Baranoff* aus St. Petersburg programmiert weiterhin unermüdlich und zeigte seine ausführbaren Beispiele zum bekannten Buch "Cellular Automata Machines" von *T.Toffoli* und *N.Margolus* (MIT Press 1987). Das (verfügbare) Programm realisiert sämtliche Beispiele des Buches. Außerdem zeigte Sergei ein Konzept zum Portieren von Forth Programmen auf fremde Rechner sowie eine Realisierung von LOGO in Forth.

An dieser Stelle muß ich den PIP von *Duncan Loutit* erwähnen. PIP ist eine geradezu geniale, sehr stabile, Box (9 mal 15 mal 22 cm), die sich mittels zweier zentralen Rollen wie eine LOGO Schildkröte bewegt und über eine Flachtastatur auf der Oberseite programmiert wird (nur Zahlen und Symbole wie Richtungspfeile u.a.). PIP wird in hohen Stückzahlen an englische Schulen verkauft und dient dort zur Einführung der Kids nicht nur ins Programmieren, das geschieht nebenbei, sondern in die natürlichen Zahlen und viele einfache mathematische und wissenschaftliche Konzepte. In den Schulen wird PIP mit LEGO-Bausteinen verkleidet und scheint ein Riesenerfolg zu sein. Das kann nur bestätigen, wer die EuroFORTH-Teilnehmer abends beim Spielen mit PIP beobachtet hat. An bemerkenswerten Lösungen sind mir der 'Walzertanz' und 'PIP als Sekundenzeiger' (jede Sekunde 6 Grad Drehung um die eigene Achse) in Erinnerung geblieben.

Um mit den Anwendungen von Forth fortzufahren: *Derya Demir* zeigte, wie man mittels FIFO und DMA in einem PC sehr schnell Daten mit einem DSP-Koprozessor austauscht (1024 mal 16Bit in 0.96 ms). *Alan Robertson* verpackte einen Forth Kern mit Multitasker in die 448 Byte eines HC11F1, inklusive seriellem Link für 'umbilical' Entwicklung. *Klaus Schleisiek's* FRP1600 Forth Prozessor wurde vom Hersteller, der ETA (*Robert Bleny*), vorgestellt, was darauf schließen läßt, daß es ihn wirklich gibt. *Mike Smart* beschrieb eine schöne Anwendung des RTX zur Erfassung von seismischen Signalen, wie sie seit 20 Jahren in Blacknest bei Reading zentral gesammelt werden. Die Signale (Hz) laufen frequenzmoduliert über Telefonleitungen und werden nun direkt über Multiplexer einem RTX vorgelegt. Durch raffinierte Periodenmessungen können 32 Kanäle simultan mit 15 Bit Auflösung verfolgt werden. *Nick Nelson* beschrieb die Möglichkeiten, SPS-Steuerungen mit Forth zu verbessern. Und *Stephen Pelc* berichtete über ein wahrhaft europäisches Forth-Werk, die Arbeiten am ESPRIT-Projekt PROCIC.

Zum Schluß möchte ich noch die Bemühungen um das Software Engineering in Forth erwähnen. *Bengt Grahn's* Sourcecodemanager 'SMAN' hat eine Versionenverwaltung bekommen, und ich konnte dieses Jahr endlich mein 'HOLON' als fertiges eigenständiges Entwicklungssystem für Forth-Programme vorstellen.

Fortsetzung auf Seite 35

Brief aus der Provinz

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

Rolf Kretzschmar's Traum von der 'Wende' träumen die Moerser FORTHer schon ziemlich lange. Und langsam scheint sich zumindest ein Teil unserer forthigen Träume zu erfüllen. Die VD wird besser! Ja, Rolf's Bemühungen zeigen erste Früchte! Das Layout ist in der letzten Ausgabe so 'gefällig' gewesen, daß ich mich erstmals getraut habe, die VD auf meinem Schreibtisch im Büro liegen zu lassen. Inhaltlich hat sich ebenfalls soviel getan, daß ich unsere Zeitschrift auch einigen Nichtforthern in die Hände gegeben habe. Die Kritik zu unserer ansprechenden 'Vereinszeitschrift' war durchweg positiv.

Was macht es da, daß die Ausgabe 3 (!) 92 erst am 13. November in Moers ankam ? :-)

Ganz besonders hat mir der Artikel zu der Automatisierung des Sägewerkes gefallen. Davon möchte ich persönlich gerne mehr lesen. Das ist FORTH 'ganz praktisch'. Das sind Dinge, die man auch skeptischen Nichtforthern 'verkaufen' kann, Dinge, die zeigen, daß Forth mehr als Hobby und überholte Spinnerei ist. Ein wenig merkwürdig finde ich die 'Kolummne' von *Andreas Goppold*. Vermutlich wird das in Zukunft eine wirkliche Bereicherung der VD werden (wenn's denn was wird), aber ich bin mir schon jetzt sehr sicher, daß ich bei Vielem von dem was da kommen wird, 'Bauchschmerzen' bekommen werde. (...) Aber wie dem auch sei, es kann nicht verkehrt sein, in der VD auch einmal das eine oder andere unforthige Thema aufzugreifen. Schließlich haben Forther weder Scheuklappen vor den Augen, noch Bretter vor den Köpfen (mit Ausnahme einiger Niederrheiner, versteht sich).

WANTED hab ich noch immer nicht verstanden. Was bezweckt denn diese Rubrik nun wirklich? *Jörg Staben* schreibt, daß Hilfesuchenden zu speziellen Themen bereits vorliegendes Material zu Verfügung gestellt werden soll. Gleichzeitig stehen unter WANTED Hilferufe nach Geräte-Treibern für TCOM, nach dBASE Dateizugriffen für Forth, nach einem SYSINFO unter Forth usw.. Ja was denn nun, Jörg ? Der eine oder andere Hilferuf interessiert mich selbst. Ich würde mich schon darein finden. Aber dann wird's wieder zu lang für Euch...

Die 'Boshafte Betrachtung' vom Forthkollegen *R.Freitag* hat mir ganz besonders gut gefallen. Endlich mal Jemand der 'meckert' und nicht aus Moers kommt :-) Weiter so ! Über ein regelrechtes Anfängerforum sollten wir (alle Forther) uns wirklich ernsthaft Gedanken machen. Heute muß Niemand mehr mit F83 auf die Nase fallen und Niemand muß sich mit F-PC erschlagen und alleingelassen fühlen :-) Wozu haben wir denn den Verein ?

Selbstverständlich findet auch die Umfrage meine vollste Zustimmung (...). Ich hoffe, daß die Redaktion im positivsten Sinne des Wortes bereits in Antworten erstickt ist. Die Auswertung erwarte ich schon heute mit großer Spannung.

So viel zur VD. Jetzt noch ein paar Zeilen aus dem Moerser 'Gruppenleben'. (...) Wir haben 'planmäßig' mit dem angekündigten Kurs zu den 'Innereien' unseres Forth-Systems (Innerer Aufbau in ZF) begonnen. Allerdings können wir noch nicht absehen, ob alle diesen 'drögen' Stoff bis zum Ende schlucken werden. Ein Teil der Gruppe hat gerade diese mehr theoretischen Informationen zu Wortaufbauten und zur Arbeit des Compilers benötigt und erwartet. Der andere Teil sieht für die 'graue Theorie' noch keinen praktischen Nutzen und läuft der eigenen Motivation hinterher. Aber das wird schon werden... Das war's für dieses Mal aus Moers.

Euer
Friederich Prinz

ZEIGE: XY.DBF

von Michael Major

In der VD 3/92 war der Hilferuf eines FG-Mitgliedes zu lesen, der wissen wollte, wie dBASE Dateien unter Forth anzufassen seien. Michael Major aus der Moerser Forthgruppe zeigt im folgenden Beitrag, wie man das prinzipiell macht. An 'Vorwissen' sind dazu lediglich die von der Moerser Gruppe im Frühjahr '92 vorgelegten Kursinhalte zum Dateiinterface in ZF, sowie das in der einschlägigen Literatur nachschlagbare Wissen um die Strukturen in dBASE's *.DBF Files, die in den Köpfen eben dieser Dateien enthalten sind. Der 'Rest' ist ein wenig Arbeit.

```
COMMENT:
Diese Demoversion für die Ausgabe von DBF Dateien dient lediglich zur Verdeutlichung
wie unter ZF- FORTH DBASE Dateien verarbeitet werden können. Dabei wird nur mit
HighLevel Forth gearbeitet. Es wird kein zusätzlicher Speicher angelegt, hier könnten
noch einmal über 8 Kb eingespart werden. Ich hoffe jedoch, daß so meine Mitstreiter
( Jeder Forth Anfänger usw. ) sich in vertrauter Code-Segmentumgebung eher zurechtfin-
det !.
Innerhalb der Demo können nur 32000 Sätze gezeigt werden, und Sätze mit mehr als 24
Feldern sollten nicht benutzt werden, da sonst die Ausgabe angepaßt werden muß.
Es erfolgt eine einfache Ausgabe der einzelnen Datensätze, die per Tastendruck gestoppt
werden kann. Die Ausgabe wir zur Zeit durch 500 MS gebremst und kann natürlich verän-
dert werden. Erweiterungen in Richtung anderer Speicherverwaltung und 32 Bit Pointer,
sowie einzelne Sätze auswählen usw. lassen dann meines Erachtens kaum noch Wünsche
offen und finden hoffentlich den Weg zurück in unsere Gruppe.

COMMENT;
\  ------------------------------------------------------------------------
  HANDLE   DATEI                                    \ Datei Handle
  CREATE   NEU-NAME   14 ALLOT                      \ Speicher fuer Handle
  CREATE   KOPF_SP  8192 ALLOT                      \ Speicher fuer Header
  CREATE   SATZ_SP   255 ALLOT                      \ Speicher I/O Daten
\ ------------------------------------------------------------------------
\ CONSTANTEN werden hier im Sinne von VALUES mißbraucht
\ ------------------------------------------------------------------------
  0 CONSTANT  ANZAHL                                \ ANZAHL LESEN
  0 CONSTANT  AUS                                   \ Einlesen u. ausgeben
  0 CONSTANT  POS-AUF                               \ Satzposition
  0 CONSTANT  NEUPOS
    VARIABLE  ZAHL1                                 \ ZAEHLER1
    VARIABLE  SATZ_ENDE                             \ wieviel Saetze maximal
\ ---------------------------- KOPF - WERTE ------------------------------
  0 CONSTANT  K_ADR                                 \ KOPF SPEICHER START ADR
  0 CONSTANT  MEMO                                  \ MEMO DATEI ?
  0 CONSTANT  LKOPF                                 \ KOPF LAENGE
  0 CONSTANT  LSATZ                                 \ SATZ LAENGE
  CREATE      G_SATZ 4 ALLOT                        \ GESAMTZAHL DER SAETZE
\ ---------------------------- DESKRIPTOR - WERTE ------------------------
  0 CONSTANT  D_ADR                                 \ DESKRIPTOR  akt.  ADR
  0 CONSTANT  D_ADR!                                \ DESKRIPTOR  START ADR
  0 CONSTANT  FNAME                                 \ START NAMENSFELD
  0 CONSTANT  FART                                  \ START FELD TYPE
  0 CONSTANT  FELANG                                \ FELD  LAENGE
  0 CONSTANT  FDECI                                 \ DECIMALSTELLEN
  CREATE      FDATEN 4 ALLOT                        \ START ADR DATEN
\ ------------------------------------------------------------------------
: INI-VARIABELN  ( -- )                             \ Setze alle Variabeln
    G_SATZ     4 ERASE                              \ fuer Programmstart
    FDATEN     4 ERASE
    SATZ_SP  255 ERASE
    0 ZAHL1 !
    0 SATZ_ENDE !
    0 =: NEUPOS
    0 =: ANZAHL   0 =: K_ADR   0 =: MEMO   0 =: LKOPF    0 =: LSATZ
    0 =: D_ADR    0 =: FNAME   0 =: FART   0 =: FELANG   0 =: FDECI
;
\ ------------------------------------------------------------------------
: LOESCHE-SPEICHER ( -- )               \ Loescht den Kopf
  KOPF_SP 8192 00 FILL                  \         und Deskriptoren Speicher
;
```

Fortsetzung zu: Zeige: XY.dbf von Michael Major

```
 COUNT                \ ADR1 COUNT
\ ------------------------------------------------------------------------
: I/O-FEHLER1 ( -- )
        CR  ." Datei konnte nicht gefunden werden !        "
        CR  ." Falscher Pfad, Eingabe oder nicht vorhanden"
;
\ ------------------------------------------------------------------------
: NICHT_OK? ( f - )
  DROP I/O-FEHLER1
;
\ ------------------------------------------------------------------------
: IN_IO  ( -- )       \ Hole Dateinamen vom TERMINAL INPUT PUFFER
  BL WORD              \ ADR1
  NEU-NAME 1+          \ ADR1 COUNT ADR2
  SWAP CMOVE           \ ADR1 ADR2 COUNT CMOVE
;
\ ------------------------------------------------------------------------
: DATEI-FELD  ( -- )                    \ Dateinamen einlesen
  NEU-NAME 13 ERASE                     \ Die Datei wird im Format
  12 NEU-NAME C!                        \ 8 Zeichen Name . 3 Zeichen Ext.
  IN_IO                                 \ erwartet. Die DBF Datei muss sich
   13 1 DO                              \ im aktuellen Verzeichnis befinden.
        BL NEU-NAME I + DUP
           C@  0  IF   DROP DROP
                  ELSE C!
                  THEN
        LOOP
   NEU-NAME DATEI $HANDLE
;
\ ------------------------------------------------------------------------
: DATEI-AUF ( - f )                     \ Datei Oeffnung
     DATEI-FELD                         \ s.o
     DATEI HOPEN DUP                    \ Datei auf alles klar dann
     0 = IF   DROP LOESCHE-SPEICHER TRUE \ Speicher sauebern sonst
        ELSE      NICHT_OK?       FALSE \ Fehlermeldung.
        THEN
;
\ ------------------------------------------------------------------------
: DATEI-ZU ( -- )
  DATEI HCLOSE DROP
;
\ ------------------------------------------------------------------------
: K_DATUM ( -- )
    K_ADR    3 + C@  5  4  AT  2 .R          \ DATUM TAG
    K_ADR    2 + C@  7  4  AT  2 .R          \ DATUM Monat
    K_ADR    1+  C@  9  4  AT  ."  19" 2 .R  \ DATUM Jahr
    40 4 AT ." Datum "
;
\ ------------------------------------------------------------------------
: K_RESERVIERT ( -- )
    20 0 DO
            K_ADR 12 + I + C@
            5 I + 8 AT HEX . DECIMAL        \ 20 Bytes reserviert
       LOOP
    40 8 AT ." Reservierte Bytes"
;
\ ------------------------------------------------------------------------
: DBFKOPF-JOB  ( -- )                   \ HEADER WERTE KONSTANTEN ZUORDNEN
    KOPF_SP =: K_ADR                    \ STARTADRESSE HEADER
     K_ADR C@ HEX =: MEMO DECIMAL       \ Memofeld vorhanden ?
     K_ADR  4 + @    G_SATZ     !       \ Gesamtzahl der vorhandenen Saetze
     K_ADR  6 + @    G_SATZ 2 + !       \
     K_ADR  8 + @  =: LKOPF             \ Laenge des Headers 2 Bytes
     K_ADR 10 + @  =: LSATZ             \ Laenge eines Satzes
;
\ ------------------------------------------------------------------------
: Des_Reserviert ( -- )
   14  0 DO
        D_ADR 18 + I + C@
            5 I + 17 AT  HEX . DECIMAL      \ Adresse
        LOOP
   40 17 AT ." Reservierte Bytes "
;
\ ------------------------------------------------------------------------
: DBFDESK-JOB ( -- )                    \    Werte zuweisen
```

Ein Saurier im Zimmer

Leserbrief von Arndt Klingelnberg
Straßburger Str. 12, 5110 Alsdorf,
Tel.: 02404-63039
Bezug: Brief aus der Provinz in 2/1992, S. 33

Welches Zimmer?

Wer verwendet ZF, F-PC v.2.25, F-TZ 3.3x, F-PC v.3.50, F-PC v.3.55, F-PC-ak0, F-PC-ak V.4.0 ? Und wie sieht es denn aus mit Volksforth, Turbo-Forth(!), WinForth usw. Wer verwendet neben embedded controllern noch etwas anderes als den 'IndustrieStandard'. Ich hoffe die FragebogenAktion schafft etwas Klarheit.

Welchen Saurier schlachten?

Die Ansicht, daß Saurier (nur) ein kleines Gehirn gehabt haben sollten, scheint überholt. Verteilte Intelligenz in Ganglien: aus technischer Sicht ein akzeptables Konzept! F-PC ist groß, aber gerade durch die Hilfen und den vorzüglichen Editor und das, was man/fra sonst immer erst noch hinzu schreiben MUSS.... Überhaupt, was soll's, zu Windows Zeiten ist F-PC ein Mini-Programm: F-PC und seine wichtigsten Hilfetexte sind doch im Cache. Den noch größeren Saurier newZ habe ich 'mal unter Jörg Stabens Fingern erleben können. Zugang zu zig MegaByte! Da wirbelten nur noch so die Pascal, C++ und Tcom Oberflächen vorbei. Schnell und komfortabel.

F-PC unhandlich?

Ja es fehlten ein paar Griffe und Knöpfe, und manchmal klemmt(e) das Hilfe System, schottete gewisse Bereiche ab. Nach einige Änderungen: ALLES ist ONLINE da, schnell und umfassend: ein Pronto-Saurus?!

Assemblerbeschreibung, wozu?

Wieso muß ich einen Assembler für ZF erst noch beschreiben? Für F-PC nehme ich eins der Bücher aus dem Regal. Änderungen gegenüber MASM sind gering. Mit einem zusätzlichen PASM.hlp ist eine Dokumentation ONLINE da, auch ohne Norton Guides. Ausdrücklich empfehle ich F-PC-ak (Was sonst? rk) zum Erlernen (testen) von Assembler Prozeduren und System-Routinen. Der Assembler des ZF ist dagegen meilenweit entfernt vom i8086 Standard. Für mich ein Unding. Auf einem PC muß ich doch nicht mehr die Forth Notlösungen, die ich auf meinen embedded Controllern gerne toleriere, akzeptieren. Ich entnehme eine ASM-Routine z.B. aus Tischer, ersetze Kommas durch Komma&Leerzeichen, ersetze mein DEFERred (NUMBER?), damit ich MASM übliche hex und decimal Notation erfassen kann und ab geht die Post. Aber eben NICHT postfix, sondern wie bei 99,9% anderen in PREfix. (Wer PREfix sagt, muß noch lange nicht auch INfix sagen.)

ZF- der große Unbekannte aus der 'Provinz' !

Insgesamt vermisse ich aber eine Vorstellung des ZF in der VD. Ich werde ZF nun auch installieren, aber die Meinung eines kompetenten Nutzers ist doch durch nichts aufzuwiegen. Es gibt Kurse zu ZF, Leserbriefe über ZF, aber keine Artikel! Ich weiß noch nicht einmal, in welches Schaffensjahr von Tom Zimmer denn diese Versionn fällt. Ist es pre F-PC v.2.25? Wenn ja, wieso soll sich Tom Zimmer und seine große Fan-Gemeinde verschlechtert haben? Ich brenne auf eine Antwort!

F-PC ist an verschiedenen Stellen in der VD beleuchtet worden, auch dessen 3er-Segmentierung, die aufgeräumten Header. Wo sind denn nun die Unterschiede? Was konkret geht besser? Wie anders sieht ein Quelltext aus? Wie anders das Compilat? Warum sollte ich auf die ausgefeilte HyperHilfe und

Fortsetzung zu: Zeige: XY.dbf von Michael Major

```
    K_ADR 32 + DUP =: D_ADR =: D_ADR!
    D_ADR =: FNAME                          \    Des_Feldname
    D_ADR 11 + C@ =: FART                   \    Des_Feldtype
    D_ADR 12 + @  FDATEN !
    D_ADR 14 + @  FDATEN 2 + !              \    Des_Adresse
    D_ADR 16 + C@ =: FELANG                 \    Des_Feldlaenge
    D_ADR 17 + C@ =: FDECI                  \    Des_DecStelle
\   Des_Reserviert
;
\ ------------------------------------------------------------------------
: FELD  ( -- )
    D_ADR 16 + C@ =: FELANG                 \    Des_Feldlaenge
;
\ ------------------------------------------------------------------------
: DBFKOPF-LESEN ( -- )                      \
    KOPF_SP 32 DATEI HREAD DROP
;
\ ------------------------------------------------------------------------
: DESKRIPT-LESEN ( -- )                     \
    LKOPF 32 - =: ANZAHL
    KOPF_SP 32 + ANZAHL DATEI HREAD DROP
;
\ ------------------------------------------------------------------------
: HEADER-EINLESEN  ( f - )
  -1 = IF                           \ DATEI OEFFNUNG IST OK
        DBFKOPF-LESEN               \ LESE DBF-HEADER ZUM SPEICHER
        DBFKOPF-JOB                 \ VERARBEITUNG HEADER
        DESKRIPT-LESEN              \ LESE DBF-DESKRIPTOREN ZUM SPEICHER
        DBFDESK-JOB     TRUE        \ VERARBEITE DESKRIPTORWERTE
      ELSE              FALSE       \ DATEI OEFFNUNG NICHT OK
      THEN
;
\ ------------------------------------------------------------------------
: HEADER-AUSGEBEN ( -- )                            \ Werkzeug Kontrollwort
   5 3 AT MEMO .   ." MEMODATEI vorhanden ?"        \ MEMO DATEI ?
   K_DATUM
   5 5 AT G_SATZ @ .
   9 5 AT G_SATZ 2 + @ . ." GESAMTZAHL DER SAETZE"  \ GESAMTZAHL DER SAETZE
   5 6 AT LKOPF .        ." KOPFLAENGE"             \ KOPFLAENGE
   5 7 AT LSATZ .        ." SATZLAENGE"             \ SATZ LAENGE
;
   K_RESERVIERT
\ ------------------------------------------------------------------------
: FELD_NAMEN_AUSG  ( -- )                   \ Alle Feldnamen ausgeben
   LKOPF 32 - 32 /
      0  DO
            3 1 I + AT FNAME 11 TYPE
                FNAME 32 + =: FNAME
         LOOP
;
\ ------------------------------------------------------------------
: D_AUS ( -- )
   SATZ_SP AUS TYPE
;
\ ------------------------------------------------------------------------
: POS_LESEN ( -- )                          \ Datei Zeiger setzen u. lesen
      SATZ_SP 255 32 FILL                   \ SATZ I/O PUFFER loeschen
      POS-AUF SD                            \ Positioniere Movepointer 1+
              DATEI MOVEPOINTER             \ da Anf. Adresse Del Abfrage
      SATZ_SP FELANG DATEI HREAD =: AUS     \ Satzspeicher Feldlang aus-
      AUS 65  IF 65 =: AUS THEN             \ lesen Wert nach AUS speich.
;                                           \ verkuerzte Ausgabe.
\ ------------------------------------------------------------------------
: SATZ-AUSGABE ( -- )                       \ Holt den jeweils geforderten
  NEUPOS 1 + =: POS-AUF                     \ Satz Grundposition ist 1 Byte
  D_ADR! =: D_ADR FELD                      \ Start Deskriptor ADR. 1. Feld
  LKOPF 32 - 32 /                           \ Kopflaenge 32 - Header 32 /
   0 DO                                     \ Deskriptoren SO OFT ausfuehren
      POS_LESEN
      14 1 I + AT D_AUS                     \ Ausgabe
      POS-AUF FELANG + =: POS-AUF           \ Pos. = Pos. + alte FELANG
      D_ADR 32 + =: D_ADR     FELD
      >FELD
   LOOP                                     \            BIS ENDE
;
```

den vortrefflichen Editor verzichten? Wie man eine Light-Version gegen die verbreitete SaurierPhobie bei Ein-, Um- und Zusteigern erzeugt, haben Ulrich Hoffmann (VD Vol7/1, März 1991) am Beispiel volksForth und am F-PC demonstriert. Könnte man damit nicht vorzüglich Forth83 lehren? Gibt es zu ZF so wichtige Utilities wie MULTASK.seq, RS232IB.seq ,INTERRUP.seq oder auch Hard- und (!) Software Floating Point? Welche Grafiktreiber gibt es ? Wie sieht es aus mit High-Level- und Code-Level-Debuggern?

Ansonsten installiere ich auch WINforth Explorer. Gewinnt ZF oder WINforth?

Noch ein Satz zu Goppold: ich werde verstärkt im großen Rest der Welt zu finden sein. Es bestehen gute Chancen, das C-FORTH - warum M. Bradly das selbst nicht brachte ? - auf dem PC unter DOS zum laufen zu kriegen. Das gibt viel ForCe, aber das wäre ja ein geschützter FirmenName.

- Well, that's it !!!

... und sie bewegt sich doch?

Leserbrief von Ulrich Paul
Erlenweg 18, 8901 Leitershofen

Hallo Freunde von der Redaktion,

ich kann es beinahe nicht glauben, daß die Forth-Gemeinde bereit ist, über ihren eigenen Tellerrand hinauszuschauen. Aber wie ich gesehen habe, daß *Andreas Goppold* "so etwas" schreiben darf (und das auch noch abgedruckt wird), keimte eine neue Hoffnung in mir auf. Sollte es möglich sein, daß die Veränderungen und Entwicklungen der anderen Programmiersprachen einen vergleichenden Denkprozeß in Gang gebracht haben? - Schlecht wäre das auf keinen Fall, sondern vielmehr eine Dekade überfällig. *(Soweit die Einleitung aus dem ersten Fax von* Ulrich Paul. *Kurze Zeit später "traute" er sich, der Redaktion die folgenden Sätze auf den Schreibtisch zu legen, nachdem er sich durch 200 Nachrichten der Forth-Mailbox gearbeitet hatte:)*

Dogma: Forth ist das System für embedded Controller.

?Warum kriegt man dann für praktisch jeden neuen Mikrocontroller einen C-Compiler (Cross-Compiler) und für praktisch keinen ein Forth-System?

Dogma: Forth ist nicht für die großen Systeme.

?Warum ist dann das hochgelobte F-PC so ein Monster, das an speicherhunger es mit jedem anderen System aufnehmen kann?

Dogma: Nur in Forth ist wirkliches Rapid Prototyping möglich.

?Warum sind dann aber die anderen so schnell und zahlreich am Markt - und Forthler praktisch überhaupt nicht?

Dogma: Nur Forth erlaubt die inkrementelle Programmierung.

?Wie nennt man dann das, was man mit dem Compiler und dem Debugger z.B. unter Turbo-Pascal alles machen kann: Während dem Testen Variablen verändern, gezielt Subroutines anspringen, etc.? (Wer mir etwas vom umständlichen Edit-Compile-Run-Zyklus erzählen will, der zeigt nur, daß er keine Ahnung hat, was derzeit Stand der Dinge ist. So schnell wie F-PC ist Turbo-Pascal beim Compilieren allemal!)

Dogma: In Forth kann ich jederzeit interaktiv Teile des Codes ersetzen.

?Und was soll das? Wenn ich ein Wort ändere, das von anderen aufgerufen wird, dann muß ich alle nachfolgenden neu compilieren - äußerst effektiv, wenn die Änderung ein Wort weit unten betrifft und die Applikation größer ist! Die "'anderen"' compilieren nur das fehlerhafte Modul und linken neu.

Dogma: Forth kann ich jederzeit modifizieren, um es für meine Anwendung brauchbar zu machen.

?Wo ist der Vorteil auf lange Sicht? Daß ich garantiert in Zukunft damit beschäftigt bin, Änderungen rückgängig zu machen, um an einem alten Projekt weiterzuarbeiten? Oder alle neuen Ideen in alle Kopien meiner modifizierten Systeme einzubauen?

Dogma: In Forth programmiert es sich einfacher und schneller.

?Was ist daran einfach, wenn ich alles, aber auch wirklich alles, per Hand selbst machen muß? Daß ich weiß, was ich tue? Oder vielleicht deswegen, weil ich "'eher das Taschentuch eines anderen Programmierers verwende als dessen Code"' (Kritik im OT von Hans Reilhofer)? Sind mir Libraries ein gar unheimliches Gebilde?

Dogma: Forth ist mächtig, weil es viele Worte hat.

?Seit wann ist die Größe des Wortschwalles ein Maß für die Mächtigkeit einer Sprache (Computer-, als auch natürliche)? Ist es nicht vielmehr so, daß die Prägnanz darin liegt, daß ich mit weniger Worten treffend formuliere?

Dogma: Die großen Firmen verschweigen, daß sie mit Forth arbeiten, weil es ihnen einen Wettbewerbsvorteil verschafft.

?Welchen Vorteil haben sie davon, es zu verschweigen, wenn doch alle wissen, daß sie es benützen? Oder ist die Aussage so zu verstehen, daß nicht die Verwendung, sondern das Verschweigen den Vorteil darstellt?

Dogma: Ich kann meinen Forth-Compiler jederzeit erweitern, die anderen nicht.

?Wie lange noch? Die anderen modifizieren nicht den Kern des Compilers, sondern erreichen das durch Direktiven für den Preprocessor - und dessen Fähigkeiten nehmen rapide zu. Es ist keine Frage von Jahren mehr, daß z.B. C++ in dieser Richtung Forth voll überholt, sondern nur noch von Monaten.

Lauter defätistische Fragen, die einem Mitglied des Forth-Vereins nicht anstehen? Droht mir jetzt der Ausschluß wegen mangelnder gemeinsamer Basis? Soll ich Zuflucht in einem Club der "'anderen"' suchen? Oder wäre es möglich, daß diese und ähnliche Fragen auch anderen Mitgliedern durch den Kopf gegangen sind? Daß es vielleicht ein nennenswertes Häufchen gibt, das über den Tellerrand hinausblickt und objektive Vergleiche anstellt? Sollte Forth doch nicht dem Alterstarrsinn verfallen sein (man war früher etwas, seiner Zeit voraus und will nun nicht einsehen, daß das Neue einen voll überholt hat)?

Wer hat wirklich Lust Forth auf einen modernen Stand zu bringen?

Mit freundlichem Gruß,
Ulrich Paul

Fortsetzung zu: Zeige: XY.dbf von Michael Major

```
\ ------------------------------------------------------------------------
: DATEN-AUSGABE
  LKOPF =: NEUPOS
    G_SATZ @                                          \ Anzahl der gesamt-
    DUP 32000  IF                                     \ Saetze holen hier
                  DROP EXIT                           \ groesser 32000 Ab-
               ELSE SATZ_ENDE !                       \ brechen sonst spei-
                 BEGIN                                \ chern. Schleife
                   SATZ-AUSGABE                       \ Satz Ausgabe
                   NEUPOS LSATZ + =: NEUPOS           \ Neuer Satzzeiger
                   500 MS                             \ VARIABLE DEMO BREMSE
                   ?KEYPAUSE                          \ Taste gedrueckt ?
                   ZAHL1 INCR                         \ Zaehler um 1 erhoehen
                   65 0 AT ." Satz:" ZAHL1 @ 8 .R     \ Satzzahl ausgeben
                   ZAHL1 @ SATZ_ENDE @ = IF EXIT THEN \ Ende aller Saetze
                 AGAIN                                \ erreicht ? Schleifen
               THEN                                   \ Ende.
;
\ ------------------------------------------------------------------------
: HPROG ( -- )
  DARK                                                \ Hauptprogramm / Wort
     INI-VARIABELN                                    \ Variabeln initialis.
     DATEI-AUF                                        \ Datei oeffenen
     HEADER-EINLESEN                                  \ Kopf einlesen True
     -1 = IF                                          \ dann Feldnamen ausg.
              HEADER-AUSGEBEN KEY DROP DARK           \ KANN entfernt werden
              FELD_NAMEN_AUSG                         \ Daten ausgeben
              DATEN-AUSGABE  DATEI-ZU                 \ Datei schliessen
              KEY DROP                                \ Flag droppen
          ELSE DATEI-ZU                               \ nicht True dann Da-
          THEN                                        \ sofort schliessen
;
\ ------------------------------------------------------------------------
: DBLES
   HPROG
;
\  Aufruf DBLES  AME.DBF
\ ------------------------------------------------- Michael Major -11/92---
\ ------------------------------------------------------------------------
\
\ Notiz:
\
\ Weiterfuehrende Informationen zur Dateiverwaltung unter ZF ueber die
\ Moerser Forthgruppe
\
\       Leitung:
\       Friederich Prinz, Forthgruppe Moers,
\       Hombergerstrasse 335, 4130 Moers 1
\       FG.Nr.: 2528
\
\       Kurs 1/92 Friederich Prinz
\        ZF - Dateischnittstelle
\
\ ------------------------------------------------------------------------
```

Reaktivation

Ein Forth-Wiedereinsteiger berichtet.

Leserbrief von Gerd Limbach
Ziegelstr. 5, 5620 Velbert 1; Tel 02051-255112

Als langjähriges Mitglied der FG, habe ich die VD immer interessiert gelesen (...wenn auch Inhalte nur teilweise nachvollziehen können). Doch nun ist ein erfrischender Wind in der VD erschienen (2.Quartal/92), so daß auch Anfänger und Wiedereinsteiger (wie ich einer bin) zu Interesse und zu Worte kommen.

Beginn 1983

Mit einem C 64 , einem Forth-Modul von Handic (...), einer englischen Minibroschüre und einer Datasette bin ich angefangen. (...) Einige kleinere für mich brauchbare Anwendungen sind dabei heraus gekommen (Maleraufmaß u.a. Anwendungs-Routinen). Auch im Bekanntenkreis habe ich versucht Forth zu aktivieren. Doch außer einem Aha-Effekt ist nicht mehr heraus gekommen, auch bei mehrmaligen Anläufen nicht (Vielleicht war ich zu der Zeit in diesem Gebiet noch nicht didaktisch gut genug drauf).

Wenig Forth

Danach hatte ich längere Zeit mehr mit dBase,Multiplan und Wordstar zutun, wenig mit Forth.

Ab 1988 mit PC und ab Anfang 91 mit AT 386 /33 (...)

Neuer Anlauf mit FORTH - Teil I

Der Artikel "FORTH - Kurs in Moers" in der VD März 92 hat mein Interesse wieder geweckt und ich habe Friedr. Prinz angerufen. Nach Rückruf habe ich von Friedr.Prinz einen zweiseitigen informativen Brief und das ZF-FORTH erhalten !!!kostenlos!!! Schon ein paar Jahre habe ich immer kleine Informationen zu der aktiven Moerser-Forth-Gruppe in der VD gelesen, registriert und nichts unternommen, obwohl der Wille mehrfach da war.

Was ich aber hier bekommen habe an Informationsvielfalt, übertraf alles was ich mir gedacht habe, ich war angenehm überrascht.

- Der Artikel in der VD 3/92 (als Anstoß).
- Das ZF - Forth Programm
- Die Forth-Kurse auf Diskette (Ausdruck bis zum 11. von 19 Kurstagen ca 100 Seiten !)

Der zusätzlichen erläuternden Dokumentationen auf der Diskette Die Ausdrucke der zwei Kurse und der Dokumentationen füllen schon einen DIN A 4 - Ordner (bisher fast 250 Seiten) sehr gut erläuterter deutsche Texte.

Ein Hoch auf die Moerser -FORTH - Gruppe für die Ausarbeitung und Übersetzung der Texte ! ! !

Forth-Anlauf Teil II

Parallel dazu habe ich Jörg Staben in Hilden angerufen, um mich über Forth allgemein und über F-PC zu informieren. (Der Mut hatte mich immer noch nicht verlassen.) Dabei habe ich F-PC bestellt und schnellstens bekommen. Erhalten habe ich ein 190 seitiges deutsches Forth-Buch (DIN A 4), das F-PC - Prg, einen Target - Compiler TCOM, ein Mathe- und Grafik- Prg von Mark Smily (Grafik, Fraktale, Floating Point - Prg u.a.), sowie einen englischen Forth-Kurs von Haskell und weiteren vielseitigen, größtenteils englischen Dokumentationen auf Disketten.

Neuer FORTH Start.

Meine Überraschung war noch größer, als ich die Programme (ZF- FORTH und F- PC) erst einmal gestartet hatte. Übersichtlich, mit einfachen und doch umfangreichen Hilfe-Funktionen, sichtbarer Stapeltiefe und vielen anderen optimalen Bedingungen. Es waren keine Screen- bzw. Block- Formate mehr und doch konnte ich eine ganze Reihe der noch nicht ganz vergessenen Forth-Befehlen aus meinen früheren Systemen sofort wieder verwenden. Später habe ich mir einige Programme aus den dort vorhanden, näher angesehen, teilweise geändert, Fehler eingebaut, geprüft und getestet zur Übung. Es hat mir bis jetzt wieder richtig Spaß gemacht, muß aber noch eine ganze Reihe wieder dazu lernen.

Wie gehts weiter mit Forth?

Die Kurse der Moerser-Forth-Gruppe eignen sich hervorragend für Kurse in anderen Regionen. Jeder, der mit Ausbildung oder Unterweisung zu tun hat, kann mit diesen Unterlagen hervorragend Forth-Wissen weitervermitteln. Aber auch zum Selbststudium sind diese Kurse sehr gut geeignet. Ebenso die englischen Forth-Kurse von *Haskell*.

Ich möchte mich auch ein wenig an der aktiven Forth-Gemeinschaft beteiligen. Auch an Forth-Literatur und - Artikeln habe ich einiges zur Verfügung. Darum bitte ich Forth-Anfänger, Wiedereinsteiger und sonstige Forth-Interessierte bei mir anzurufen, wenn sie ähnliche Probleme haben oder einen Forth-Gedanken-Austausch suchen.

Mo oder Die 20:00 - 22:00 Uhr
Tel.: 02051 - 25 51 12
mfg. Gerd Limbach (Jg. 1939)

Das F-PC und seine Stiefbrüder

Leserbrief von Jörg Staben

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

Zu meinem Erstaunen habe ich in der VD 3/92 bemerkt, daß mehrere Stiefbrüder des F-PC die Forth-Szene betreten. Diese unehelichen Geschwister zeichnen sich durch einen Namen wie F-PC-abc oder F-PC-YZ aus, aber was das nun genau ist, weiß man nicht.

Manche nennen sich bug-fixed, andere werben mit einem verbesserten human-interface - ob diese Systeme ihrem eigenen Anspruch gerecht werden, kann man nicht sagen - man kann aber sehr wohl sagen, daß diesen Systemen an allen möglichen Stellen verändert wurden, allerdings **nicht** von *Tom Zimmer*, dem Entwickler des F-PC.

Grundsätzlich ist man in der Software-Entwicklung froh, wenn ein Programm läuft und sich im Einsatz bewährt. Sie kennen ja aus der Industrie die I. Regel der Instandhaltung: Finger weg von der Maschine!

Sollte dagegen im Einsatz eines Programmes ein Fehlverhalten auffallen oder sich durch gewandelte Anforderungen die **Notwendigkeit** einer Programm- Änderung ergeben, wird der Systementwickler (zur Erinnerung: Beim F-PC ist das Tom Zimmer) oder der Projektleiter neu festlegen, was das Programm zu leisten hat. Führt der Projektleiter die vorgesehene Änderung nicht selbst aus, beauftragt er einen Mitarbeiter, diese Programm-Änderung streng nach der getroffenen Festlegung durchzuführen.

Der Mitarbeiter bearbeitet nun diesen Auftrag; ist die Änderung erfolgt, so beauftragt der Projektleiter einen anderen Mitarbeiter mit ausführlichen Tests oder führt diese Tests selbst durch. Abschließend werden die durchgeführten Änderungen dokumentiert und das verbesserte Programm kommt erneut zum Einsatz.

Stellen Sie sich dagegen dieses Szenario vor: Ein Mitarbeiter von Borland tritt strahlend vor *Philippe Kahn* und berichtet stolz, er habe einfach so die Quelltexte des TurboPASCAL 7.0 an den verschiedensten Stellen geändert; vor allem habe er writeln umbenannt in zeig's_mir. Da möchte ich gerne sehen, wie Philippe sein Saxophon aus der Hand legt!

Ich konnte damals nicht verstehen, warum Klaus Schleisiek-Kern nicht jedem den volksForth Meta-Compiler in die Hand drückte - nun weiß ich es!

Was kann Ihnen blühen, wenn Sie ein Nicht-F-PC-F-PC benutzen?

- Der Befehlsumfang weicht - Forth macht's möglich - von den Vorgaben ab; Sie können Ihr Programm nicht kompilieren und werden WHAT? gefragt. Das ärgert Sie zwar, die Fehlerquellen können Sie aber noch ganz gut finden.
- Manche Befehle zeigen ein falsches Verhalten; Ihr Programm stürzt ab. Das ärgert Sie schon mehr, aber dank der excellenten Unterstützung des F-PC bei der Fehlersuche...
- Manche Befehle zeigen ein anderes Verhalten als erwartet; Ihr Programm überrascht Sie in einigen Situationen mit ganz neuen Ideen. Viel Spaß!

Wann könnten Abkömmlinge eines Forth-Systems denkbar sein?

Selbstverständlich stützen sich auch Forthler auf bereits geleistete Arbeit und würdigen sie damit. So sind aus dem volksFORTH der Forth-Gesellschaft gleich mehrere Forth-Systeme entstanden: Das DeltaT-Forth als PC-Forth-System, das KK-Forth als Forth für µController und das bigForth als Atari-System. Sie sehen sofort den Unterschied: Jedes dieser volksFORTH-Derivate hat einen neuen Namen angenommen. Damit wird jedesmal deutlich, daß es sich nicht mehr um das originale volksForth handelt! Es ist keine Seltenheit, daß sich zu einem Sprachstandard Dialekte herausbilden, durch die die Sprache erweitert wird. Bei PASCAL hat sogar ein Dialekt den Sprachstandard ersetzt - aber daß sich nun zum F-PC, einem Forth-Dialekt weitere Unterdialekte bilden, innerhalb derer man noch nicht einmal die Quelltexte austauschen kann. Damit werden sinnlose neue Probleme in die Welt gesetzt.

F-PC 4.0

Wenn überhaupt Bedarf an einer Version 4.0 besteht, was könnte dieses F-PC 4.0 bringen?

Segmentüberschreitende Header

Tom Zimmer hat seine Sonderversion F-PC 4.0 mit segmentüberschreitenden Headern ausgestattet; dabei weist er daraufhin, daß dieses Konzept den Vorteil einer riesigen Anzahl an Headern mit dem Nachteil eines langsameren Compilierens und vor allem von Quelltextänderungen erkauft. Um das Headerkonzept des F-PC 4.0 zu nutzen, muß man also seine Quelltexte ändern, sofern sie auf Segmente zugreifen. Operationen innrhalb einer Segmentes bleiben unberührt. Tom Zimmer betont, daß F-PC 4.0 nur eine Sonderversion für einen bestimmten Zweck ist, das aktuelle F-PC ist nach wie vor die Version 3.56.

Externes speicherresidentes Editormodul

Tom Zimmer hat darüber nachgedacht, daß seine Editoren mehrfach in System vorhanden sind und vor allem nur so quälend langsam vorwärts kommen, Deshalb hat er ein neues Konzept in der Entwicklung: VED . Dieser Editor steht resident im Speicher und wird von der Applikation, die gerade einen Editor benötigt, einfach angesprungen. Zudem hat VED die maximale Dateigröße irgendwo im 16MegaByte Bereich.

Sind Namen Schall und Rauch?

Sie sehen, auch die Bezeichnung 4.0 ist schon vergeben. Wenn nun andere Forthe herkommen und sich gleich oder ähnlich nennen, stehen wir vor dieser Situation: Es gibt ein F-PC-xyz, das ist inkompatibel zum F-PC 3.56 und zum F-PC 4.0; dann gibt es ein F-PC- xyz 4.0, das ist kompatibel zum F-PC 3.56, aber natürlich nicht kompatibel zum F-PC 4.0. Ist es nicht schön? Das F-PC 4.0 wiederum ist nicht zu F-PC 3.56 kompatibel, dafür hat es aber wenigstens einen Grund: Die segmentüberschreitende Headerverwaltung für große Programmierprojekte.

Also: Wenn uns irgendwas noch gefehlt hat, ist das Zersplittern der Programmierumgebung F-PC in tausend kleine F-PC-abc's, die sich dann alle in der Editorbedienung unterscheiden! Dann lieber EIN F-PC und EINEN Editor NewZ, mit dem man alle möglichen Assembler und Compiler völlig identisch bedienen kann. Bitte beachten Sie: Nur wo F-PC draufsteht, ist auch F-PC drin!t

Wunschzettel

von Ulrike Schnitter, Forth-Büro
Postfach 1110, W-8044 Unterschleißheim, Tel.: 089-3173784

Bitte benachrichtigen Sie das Forth-Büro rechtzeitig bei Änderungen Ihrer Anschrift oder Bankverbindung, da fehlgeleitete Postsendungen und Überweisungen unnötige Kosten verursachen. Der VIERTEN DIMENSION Volume VIII, No. 4, (4/1992) liegt ein Mitgliedsantrag bei. Teilen Sie uns darauf Änderungen für Ihren Eintrag in der Mitgliederliste mit, oder benützen Sie diesen Antrag zum Werben neuer Mitglieder. Mit dem unteren Abschnitt des Antrags haben Sie die Möglichkeit, der Forth-Gesellschaft e.V. eine Einzugsermächtigung für den Mitgliedsbeitrag zu erteilen.

Mitgliedsbeiträge und Zahlungsmodus: Der Mitgliedsbeitrag gilt immer für das laufende Kalenderjahr. Beachten Sie bitte, daß die ermäßigten Beiträge nur gegen Nachweis gewährt werden können, also bitte für 1993 die Studienbescheinigungen usw. in Kopie beilegen. Wir benötigen nur *einen* Nachweis pro Kalenderjahr. Bei Auslandsadresse ist wegen der erhöhten Versandkosten keine Ermäßigung möglich. Unsere Mitglieder mit Auslandsadresse bitten wir, Ihren Mitgliedsbeitrag in DM mittels Auslandspostanweisung oder durch Postgiroüberweisung an uns zu entrichten. Ihren Mitgliedsbeitrag entrichten Sie bitte bis Februar 1993. Als Erinnerung liegt dieser VIERTEN DIMENSION ein Überweisungsformular für 1993 bei. Wurde der FG eine Einzugsermächtigung erteilt, werden wir den neuen Mitgliedsbeitrag Ende Januar abbuchen.

Hier die gültigen Mitgliedsbeiträge:

- Schüler, Studenten, Rentner u. Arbeitslose: DM 48,00
- Ordentliche Mitglieder, Auslandsadresse: DM 80,00
- Fördernde Mitglieder, Firmen u. Institutionen: DM 160,00

Für weitere Fragen stehen wir Ihnen gerne zur Verfügung. Telefonisch erreichen Sie uns unter

089-317 37 84.

Allen Mitgliedern und Lesern wünschen wir ein erfolgreiches, gesundes 1993.

❑

BITs vom BUS

von Arndt Klingelnberg

Ein Kurzbericht über den universellen I_X FeldbusProzessor von Delta t, realisiert als 12 bit-Stack-Maschine

Feldbusse sind keine allAchs-getriebenen ManschaftsTransportwagen, sondern erfüllen KommunikationsWünsche im Bereich von lokalen Bussen (ISA, VME, IC) bis LAN's (Ethernet-Novell, Arcnet-Kirnet, ...). Will die Hecklampen-SchalterEinheit der zentralen 'Intelligenz' rückmelden: "rechte BlinkLampe defekt"; der Längensensor in der Werkzeugmaschine der Motorsteuerung, daß nun bald die Endposition erreicht ist; der Scanner der nächsten Transportband-Weiche, daß der Koffer nach LON, DUS, JFK oder SIN geht, oder die EtagenÜberwachung in 55 dem Hausrechner, daß der Notausgang offen steht, dann sind Feldbusse gefragt.

Stichworte

BUS Feld
Bit
seriell
ForthProcessor

Die preiswerte Lösung bei Riesen-Stückzahlen ist eine 'dedicated' Hardware (mit maskenprogrammierter Firmware); bei 'nur' großen Stückzahlen mit PROM (oder OTP-EPROM), also einmalig programmierbarem Speicher. Für jede Abart unter der Feldbustypen-Inflation muß aber ein spezieller Silizium Chip da sein, der es - fest verdrahtet - auch schafft, der Bit-Flut Herr zu werden. So gibt es z.B. Derivate der 8051 oder 80166 Processoren für verschiedene Feldbusse.

Bei Klein-Serien, individuellen Sonderlösungen oder um kleine Außenläger zu realisieren, ist eine Lösung vorzuziehen, bei der ein Modul 'alles' kann. So kann entsprechend des FeldbusTypus dieser Werkzeugmaschine auf dem 'Ladentisch' das EPROM des WinkelSensors mit 'ProfiBus' geladen werden, dann wird noch der entsprechende AnschlußModul aufgesteckt und: einsatzbereit. Der nächste Kunde erhält das gleiche, aber mit 'FAIS' ausgestattet.

Forth Im Feldeinsatz

Abgestimmt auf dieses Einsatzgebiet hat die Firma *delta t* einen universell einsetzbaren FeldbusProcessor entwickelt: Interface X. Hierbei kam die Erfahrung bei dem Umgang mit ForthProzessoren, insbesondere der Entwicklung des FRP1600 (VD 1992-4 p.23) zu gute.

Der Bit-orientierte Processor ist für Bitraten bis 1Mbit/s ausgelegt. Bei einer Taktrate von 16 MHz muß der Prozessor ein Bit in weniger als 16 Zyklen verdauen. Daraufhin wurde der Prozessor optimiert: es ist eine Maschine mit 2 Stacks von je 12 bit Breite, 6 bit langen BefehlsToken (in klassischem Forth würden wir von CFAs sprechen), also 2 Kommandos je ReturnStack-CELL. Auf der einen Seite Bit-orientiert (Transmit, Receive, sowie Handshake), auf der anderen Seite umschaltbar für die Ankopp-

lung an Intel oder Motorola Memory-/AdressBusse. Je nach Auslastung durch das FeldbusProtokoll kann der I_X zusätzlich intelligente Sensor/Actor Aufgaben erledigen. Das könnte z.B. die Sammlung verschiedener Temperaturwerte sein (inkl. Filterung, Korrektur ...); verbunden mit sofortiger Weitergabe von AlarmZuständen.

Zum I_X gibt es einen Forth-Target-Compiler von MPE (GB), der das Busprotokoll und das zusätzliche Anwenderprogramm in die 64 nativen ForthWorte umsetzt.

Derzeitig (PrototypenEntwicklungBoard) wird das Programm im externen Speicher abgearbeitet. Mit einem internen EEPROM, dessen Inhalt beim Booten in eine schnelles internes RAM umgeladen wird, würde eine maximale Flexibiltät erreicht werden. So könnten die Feldbus-Stationen sogar nachträglich noch UPgedated werden. So etwas ist bei den Hardware/Firmware-Lösungen nur durch physikalischen Ersatz möglich.

p.s. Der I-X Processor war die TitelStory der Elektronik 1992 no.20 (Interkama). Ich möchte hier generell auf die 'Elektronik' hinweisen weil sich hier viele für Forthler interssante Themen finden, von Echtzeit Kernel über Feldbusse, 'embedded controller', Fuzzy, Meßdatenerfassung, und eben auch -- ab und zu -- Forth. Leider findet man/fra die Elektronik nur an sehr wenigen Stellen im freien Verkauf. (Beachten Sie bitte auch unsere getrennte Besprechung des Elektronik Feldbus-Sonderheftes).

❑

EuroFORTH '92

Fortsetzung von Seite 28

Die Arbeitssitzungen galten dieses Jahr den Themen: Netzwerke, Forth Vermarkten, Native Code, Standards und Schichten, Objekt-Orientierung und Software Engineering. Außer dem üblichen Wettbewerb und einigen 'Impromptu-Talks' blieben die Außerkonferenz-Aktivitäten erfreulich unorganisiert. Zum ersten Mal hatte ich an einer Forth-Konferenz das Gefühl, daß genügend Zeit vorhanden war, wozu allerdings auch das 'Fest der Überlebenden' beitrug, zu dem *Linda* und *Stephen Pelc* dankenswerterweise alle einluden, die am Sonntag nachmittag noch nicht genug hatten. Wir 'Schweizer' holten uns hier die ideale Bettschwere für das nächtliche Kreuzen des Kanals.

❑

TERMINE

Technische Messe Leipzig
1993märz9--13

AES Convention (Audio Engineering Society)
1993märz16--19 Berlin
- (professionelle Tontechnik nur ab und zu in Deutschland, auch ein AnwendungsGebiet für Forth: embedded controller, Digital Signal Processing, RealTime, LaborMeßtechnikg).

CeBIT
1993märz24--31 Hannover

Hannover (Industrie) Messe
1993april21--28

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

(Bitte geben Sie uns ihren Input zu für 'uns' sinnvollen Veranstaltungen: Forth-Anwendungen, Forth-Kunden, Forth-Konkurrenten, Hardware für Forther) akg

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 - 42 35 14 p
Treffen: jeden 1. Samstag im Monat im S-Bahnhof Derendorf
Münsterstr. 199, 4000 Düsseldorf

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

W-51XX Aachen
Arndt Klingelnberg,
Tel. 0+2404 - 6 16 48 agp
Treffen: jeden 1. Montag im Monat als Gruppe des Computer-Club der RWTH, Seminargebäude,
Raum 214, Nähe Wüllnerstraße
5100 Aachen

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.:089 - 8418317

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
E-Mail-Adresse:
plewe@mpi-dortmund.mpg.de
Tel. 0+208 - 42 35 14 p

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

Jörg Staben
Tel. 0+2103 - 24 06 09 p
Di. und Fr.

Frank Stüss
Tel. 0+6187 - 9 15 03 ap

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

Spezielle Fachgebiete

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
Klaus Kohl, Tel. 0+8233 - 3 05 24 p

Forth-Forum: 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

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

Für unsere Leser aus den neuen Bundesländern sind die Ortsnetzkennzahlen auch nach 92jun01 nicht eindeutig, bitte informieren Sie sich über örtliche Besonderheiten.

Appell

Es werden weitere Kontakte / Ansprechpartner gesucht, insbesondere aber auch im Osten. _Bitte wenden Sie sich an die Redaktion (A. Klingelnberg, Straßburger Straße 12, W-5110 Alsdorf,
Tel 0+24 04-6 16 48, Fax 0 24 04-6 30 39).

Redaktionadresse

Rolf Kretzschmar, Rote Gasse 7, W-5112 Baesweiler
Tel./Fax 0 24 01-8 88 91

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

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

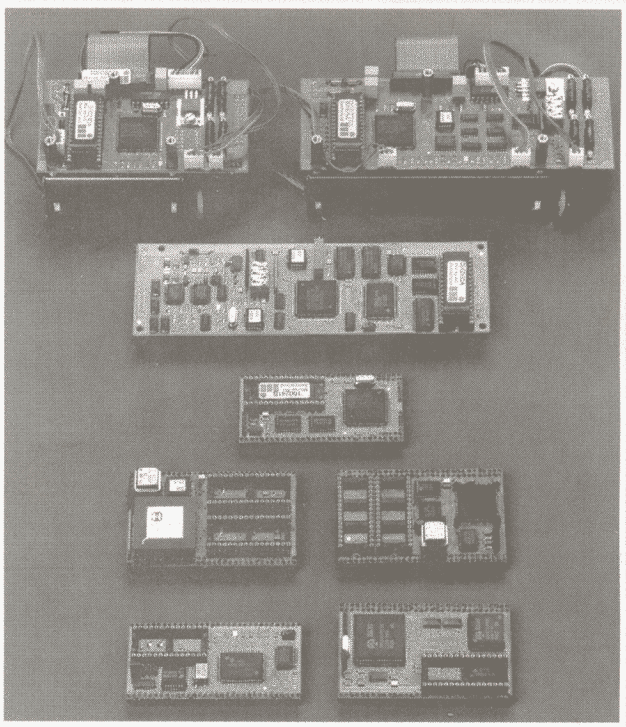
LMI FORTH-83 Metacompiler

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

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

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).