Vierte Dimension
Volume II/Nr. 2

VIERTE DIMENSION

STYLE
TREE
CONDITIONALS
TERMINAL PROGRAMM

FORTH MAGAZIN

5,00 DM

Impressum

Titel: VIERTE DIMENSION, Magazin für die Mitglieder der Forth Gesellschaft eV.
Herausgeber: Forth Gesellschaft eV, Friedensallee 92, 2000 Hamburg 50.
Forth: Klaus Schleisiek und die Mitglieder des Review-Boards sowie alle namentlich genannten Autoren.

Auflage: 500 Stück europaweit. Druck: Offsetdruckerei Saborowski
Erscheinungsweise: In jedem Quartal.

Redaktion & Anzeigenleitung: Michael Kalus, Präsidentenstraße 40, 5830 Schwelm, Telefon 02336-82204.
Redaktionsschluß: Der mittlere Quartalsmonat.

ARTIKEL BITTE NUR AN DIE REDAKTIONSANSCHRIFT EINSENDEN!

MISTER X

Vereinsmitglieder
haben mit der letzten
Vierten Dimension, April 1986, die
Mitglieder-Liste erhalten.
Waren Sie ein Mister-X ?
Ehemalige Vereinsmitglieder sind an
einem "X" statt einem "M" in der
ersten Spalte der Liste
zu erkennen.
'Ehemalige' haben den diesjährigen
Beitrag (noch) nicht bezahlt.
Sollten Sie weiterhin ein
'Ehemaliger' bleiben
wollen, so brauchen Sie auch jetzt
nichts weiter zu tun.
Wollen Sie hingegen Mitglied
bleiben, überweisen Sie Ihren
Beitrag bitte noch Heute.

BEITRAG NOCH HEUTE

EDITORIAL

"In seinem ganzen Leben lernt der Mensch nie wieder so viel in so kurzer Zeit."
(Rene Spitz über die ersten Lebensmonate des Kindes.)

Diese Ausgabe enthält Beiträge zur Entwicklung der Kommunikation zwischen Computern und deren Benutzern. Die Mailbox der Forth Gesellschaft wird vorgestellt und es wird gezeigt, wie man sie benutzen kann. Und es gibt ein Terminalprogramm dazu, geschrieben in Volksforth.

Ich habe den Eindruck, daß die Forth Gesellschaft in ihrem zweiten Geschäftsjahr begonnen hat, mit Erfolg die Projekte zu verwirklichen, die im ersten ausgedacht worden sind. Einer der drei Direktoren der FG, Klaus Schleisiek in Hamburg, hatte Anfang des Jahres da ehr 'schwarz' gesehen und von einer Kriese der FG gesprochen (Vergl. VD86 II/1).

Für Ernest Bornemann hingegen sind Neugeborenen-Kriesen ehr etwas natürliches. Die erste Krise ist eine Zeit der Anpassung an die extrauterine Welt, das Erleben der eigenen Existenz, die fortan Kreislauf, Atmung, Verdauung und Temperatur selbst regeln muß. Kälte, Erschütterung, Geräusche, Licht und andere Störungen wollen bewältigt sein. Die Wahrnehmung des Neugeborenen, in der ersten Zeit noch auf das eigene System gerichtet im Zustand von Selbstliebe und Selbsttäuschung, wird auf die Wirklichkeit gelenkt. Die Phase des Narzismus wird für gewöhnlich überwunden.

In der Forth Gesellschaft lassen die Klagen über die ach so böse Welt, die von Forth nichts wissen will, langsam nach. Die Jammerer verstummen und es kommen die zu Wort, die ganz selbstverständlich mit Forth arbeiten. Während noch 1985 auf dem Jahrestreffen der FG ein Teil der Programmierer Minderwertigkeitskomplexe hatte und nach Argumentationshilfen suchte, um ihren Arbeitgebern Forth schmackhaft machen zu können, ist jetzt zu erkennen, daß Projekte in Forth einfach gemacht werden. Diese Leute sollten sich in der Forth Gesellschaft versammeln, finde ich. Und diese Projekte würde ich gerne im FORTH MAGAZIN vorstellen.

Jetzt im Sommer werden wir erleben, wer dabei bleibt. In diesem Heft rufen wir zum letzten Mal im Jahre 1986 die bisherigen Mitglieder auf, ihren Beitrag zu entrichten. Denn wie Sie wissen, endet eine Mitgliedschaft in der Forth Gesellschaft automatisch, wenn kein neuer Beitrag geleistet wird. Es ist nicht nötig zu kündigen. Wir wollen mit der Forth Gesellschaft die ernsthaft Interessierten verbinden. Die FG bleibt die 'gute Adresse' des Forth. Sie können mit ihrem Beitrag und ihrer Stimme daran mitarbeiten, die FG in diesem Jahr weiterzubringen und die Aufgabengebiete für 1987 abzustecken.

Ich habe mit Forth und der Forth Gesellschaft den Sprung vom "Gnöbsche-Drigger" hin zu tieferreichendem Wissen um die 'Neue Technologie' erlangt, sehr zu meinem Vorteil, beruflich und privat. Von Forth als gutem Werkzeug für eine ganze Reihe von Aufgaben in dieser Technologie bin ich überzeugt. Vielen sehen das ebenso. Und wenn ich daran denke, was in diesem Jahr in der FG bereits realisiert worden ist, dann blicke ich guten Mutes in die Zukunft der Forth Geselleschaft e.V.

Michael Kalus

LESERBRIEFE

Zum universellen Stackwort von D.Hoekman

Aha, das lag also sozusagen in der Luft - allerdings kann ich vor der vorgeschlagenen Syntax aus eigener Erfahrung nur warnen, sie versetzt der Lesbarkeit von FORTH den endgueltigen Todesstoss. Nach etwas Reifezeit ist bei uns folgendes herausgekommen: Man schreibt einfach einen Kommentar, der nicht durch die 'runde klammer offen', sondern durch S(eroeffnet wird. Der Compiler erkennt dies und sorgt fuer das Eincompilieren der entsprechenden Laufzeitanweisungen. Beispiel :

```
S( n adr len - adr n adr len n ) - TYPE  ( adr n ) ....
```

Wie man sieht, bekommt man jetzt sogar noch eine zusaetzliche Dokumentation. Das neue Wort wird hauptsaechlich zum Handhaben grosser Parameter, wie Longstrings oder komplexen Zahlen verwendet: ist das erste Zeichen eines Parameter-Bezeichners eine Ziffer, so gibt dies an, wieviel Platz er auf dem Stack beansprucht. Beispiele :

```
: ROT   S( n 2k - 2k n ) ;      : 8DROP S( 8 - ) ;
: 5ROLL S( n 4k - 4k n ) ;      : 4over S( 4n 4p - 4n 4p 4n ) ;
```

Leider macht's wenig Sinn, hier meine Source-Screens abzuliefern (Selbstgemacht, Non-Standard 6502 Forth, die leidige Kompatibilitaet), aber fuer Volksforth-Kenner duerfte die Realisierung kein unloesbares Problem darstellen. Wer die Listings trotzdem haben möchte, kann mich gerne unter 089-2283531 anrufen. DFÜ möglich.
Robert Schoerghuber, Muenchen

32bit-Erweiterungen für Z80 fig-Forth

Für meine fig-Forth-Version habe ich 32bit-Operatoren erstellt. Die Wörter sind mit CREATE und nachfolgendem Hex-Code erzeugt. Die Multiplikation behandelt 32bit-Wörter als 16bit-Doppelwörter und benützt die vorhandenen 16Bit-Routinen. Die Division folgt dem 'wiederherstellenden Verfahren'. Ich stelle das gern zur Verfügung (formatierte Diskette 5.25" nebst Diskparametern). Auch kann ich Forth83 auf mc-Fortmat bieten.
In 'FORML Proceedings 1982' wurde auf S.67 der Artikel: "Design Considerations for a 32 Bit Forth" veröffentlicht. Falls er Ihnen zugänglich ist, bitte ich um eine Kopie.
Günter Petrikowski, Füll 10, 8500 Nürnberg 1

(Mit dem Artikel kann ich leider nicht dienen. Aber vielleicht einer unsere Leser? mk)

32bit-Worte des Z80-fig-Forth
D. D.R DABS D+- 2@ 2! 2DUP DMINUS D+
Meine Erweiterung
D- D2* D2/ 0. 1. 2. 3. 2DROP 2SWAP 2OVER 2ROT 4DUP
D0= D= DU< D< D> D0> D0< DMIN DMAX 2VARIABLE 2CONSTANT
2>R 2R> DU* D* D/ D/MOD DMOD

6809 fig-Forth: Ein Fehler in (+LOOP)

In decrementierenden Schleifen arbeitet +LOOP nicht richtig (Abb.1). Der Fehler ist in (+LOOP) versteckt: In XLOP2 muß es ORCC statt ANDCC heißen (Abb.2). Dann ist es richtig. Wenn Sie die Version patchen wollen:
HEX 1A 244 C!

```
SCR # 77 ANDREAS SOEDER  25.2.86
  0 (    ... +LOOP SCHLEIFEN    AS.860204
  1
  2 : VER0  0  3 DO I . -1 +LOOP ;
  3 : VER1 -1  3 DO I . -1 +LOOP ;
  4 : VER2 -4 -1 DO I . -1 +LOOP ;
  5 : VER3 -3  0 DO I . -1 +LOOP ;
  6 : VER4 -1  1 DO I . -1 +LOOP ;


VER0 3 2 1  OK
VER1 3 2 1 0 -1  OK      nicht OK
VER2 -1 -2 -3  OK
VER3 0 -1 -2  OK
VER4 1 0 -1  OK          nicht OK
```

Abb.1

Falls Sie 6809-fig-Forth noch nicht haben, können Sie es von mir bekommen. Die Bedingungen der FIG sind ja sicher jedem bekannt, mit der Weitergabe sollen keine großen Geschäfte gemacht werden.
Andreas Soeder, Seeheim-Jugendheim, Tel: 06257-2744

ZX-81 Forth

...hatte ich im Zusammenhang mit meiner Tätigkeit in der Abteilung Pathologie der RWTH Aachen Gelegenheit, mich mit Forth zu beschäftigen. Erste Anwendungen in einem Temperaturerfassungs-System waren erfolgreich. Für diese Aufgabe wurde ein ZX-81 mit Forth-ROM eingesetzt, der durch dieses Betriebs-System für den Einsatz als Controller auf Grund seiner Geschwindigkeit und seiner Multitask-Fähigkeit hervorragend geeignet ist.

```
          FCB    $87
          FCC    '(+LOOP'
          FCB    $80+')
          FDB    LASTNM
LASTNM    SET    NEXTNM
XPLOOP    FDB    *+2
          PULU   D
XLOP2     TSTA
          BPL    XPLOF
          ADDD   ,S
          STD    ,S
       ►  ANDCC  #$1       SET C-BIT
          SBCB   3,S
          SBCA   2,S
          BPL    ZBYES
          BRA    XPLONO
XPLOF     ADDD   ,S
          STD    ,S
          SUBD   2,S
          BMI    ZBYES
XPLONO    LEAS   4,S
          BRA    ZBNO
NEXTNM    SET    *

► statt ANDCC muß es heißen: ORCC
```

Abb.2

Durch diesen ersten Kontakt wurde mein Interesse geweckt und ich forderte eine kostenlose fig-Forth Version für meinen CP/M-Rechner an. Die erste Literatur, die ich zu dem Thema fand, war L. Brodie, "Starting Forth" und Ronald Zech, "Die Programmiersprache Forth", womit ich zunächst beschäftigt und zufrieden war. Später kam Andreas Goppold/Roger Bouteiller, "Forth: Ein Programmiersystem ohne Grenzen" dazu und gab mir den Anstoß, den umständlichen fig-line-editor durch einen Screen-Editor zu ersetzten, obwohl die vorgestellte Fassung nicht ohne Änderungen auf meinem Rechner zu installieren war (Ein Listing davon gebe ich gerne weiter).
Mitlerweile sind mir die Grundfunktionen des Forth geläufig, jedoch fällt es mir schwer weiterführende Literatur oder die

```
VER0 3 2 1  OK
VER1 3 2 1 0  OK
VER2 -1 -2 -3  OK
VER3 0 -1 -2  OK
VER4 1 0  OK

Jetzt ist es richtig !
```

'Forth Dimensions' zu erhalten. Daher freue ich mich über das Erscheinen der VIERTEN DIMENSION ganz besonders. Interessiert bin ich an Artikeln und Kontakt zu den Themen Z80-Assembler, Floating-Point-Arithmetik und Trigonometrische Funktionen.
Dipl.Ing. Rainer Owczorz, Suermondtplatz 11, 51 Aachen

* * *

AUS DEM VEREIN

Forth Review Board

Seit Anfang dieses Jahres bemüht sich die Forth Gesellschaft darum, neben der guten oder nützlichen Programmidee auch auf die nötige Eleganz und eine saubere Dokumentation der Programme hinzuwirken. Für die Qualitätssicherung in den Artikeln der Vierte Dimension wurde der Ausschuß zur kritischen Besprechung von Forth Code gebildet. Derzeit sind acht Personen im Ausschuß tätig: Ulrich Hoffmann in Kiel; Thomas Asche in Paderborn; Bernd Pennemann, Marco Pauck, Klaus Schleisiek und Georg Rehfeld in Hamburg; Siegfried Hirsch in Karlsruhe und - last but not least - Max Hugelshofer in der Schweiz (Reihenfolge rein zufällig). Über die Erfahrungen im Review Board wollen wir bald berichten. mk

Neues Forth Büro Hamburg

H.G. Lynsche hat die Verwaltungsarbeit der FG in diesen Tagen an Axel R. Mertins abgegeben, der sich bereits an die Arbeit gemacht hat, um aus den Kartons und Kisten das neue Forthbüro in den Räumen von Marco Pauck entstehen zu lassen. Die neue Anschrift finden Sie im Impressum.

Die Mailbox der Forth Gesellschaft ist im neuen Forthbüro bereits in Betrieb. Sie heißt "CommuniTree", hat die Telefonnummer 040-3904204 und steht bei Marco Pauck.

Unter der gleichen Telefonnummer erreichen Sie den Support des Forthbüros, der einmal pro Woche geboten werden soll. Dann ist der Tree für einige Stunden nicht angeschlossen, sondern ein 'Homo Forthius' nimmt Ihre Anrufe entgegen. Und zwar zunächst jeden Dienstag in der Zeit von 18-20 Uhr.

* * *

VERSCHIEDENES

Bitte Telefonnummer nicht vergessen!

Lieber Leser, bitte geben Sie bei Zuschriften an die Redaktion Ihre Telefonnummer an. Das erleichtert uns die Aufnahme von Beziehungen zu Ihnen sehr. Ganz besonders wichtig ist Ihre Telefonnummer für uns dann, wenn Sie uns einen Beitrag zur Veröffentlichung schicken. Beiträge müssen oft noch überarbeitet werden, und dann gibt es etliches zu bereden. Vergessen Sie also Ihre Telefonnummer nicht!

Ihre VD.

Kurz notiert aus der Forth Dimensions, Vol.VII, No.6

String Package

Auf S.25ff präsentierte Clifford Kent sein 'string package' aufgebaut auf dem F83 public-domain Forth Model. Er benutzte dabei die Vorarbeiten zum String Stack von Cassady. Im letzten Heft der VIERTEN DIMENSION hatten wir bereits den Vorschlag von Klaus Schleisiek, Hamburg, einen String Stack zu bauen, vorgestellt. Da das Thema aktuell ist, hier die WORDS des 'string package' von Kent.

WORDS
$P0 $P $P-INIT $P0$P! $P! $P@ $P2@ ?1$P@ ?2$P@ $DROP $@ TOP$ SEC$ $! $. $DUP $+ $OVER $SWAP
$POS $CHRS (CHAR-TEST) $DELETE $COPY $IN $VARIABLE $VAR-ARRAY $VARFILL $VAR@ $VAR! ["] $"
$CONSTANT $CONST-ARRAY $D.L $D.R $D. $N.L $N.R $U. $N. (DECIMAL$D.R) $DOLLARS $LC->UC $UC->LC $TRIM
$STRIP $VAL $= CLR$S .$S .$V

EuroFORML

Auf S.15f findet sich ein Bereicht über die euroFORML-Konferenz auf der Burg Stettenfels, Oktober 1985, B.R.D., veranstaltet von der Forth Gesellschaft eV. Robert Reiling, Präsident der Forth Interest Group U.S.A., berichtet darin kurz über die Arbeiten der Konferenz. Diese Beiträge sind mit der Sammlung der Assilomar-FORML papers USA in den '1985 FORML Proceedings' enthalten. Das Buch kann bei der Forth Interest Group in USA oder hiesigen Forth-Händlern bestellt werden.

Titelliste der euroFORML papers:
English as a Second Language for Forth Programmers, Wil Baden.
Interpretive Logic, Wil Baden.
Data Collection in Elementary Particle Physiks with 32-Bit VAX/68K Forth, R. Haglund.
In-Situ-Development: The Ideal Complement to Cross-Target-Compiling, Haley/Oakford/Stephens.
Forth and Artificial Intelligence, Robert LaQuey.
A Forth-Driven Network System for Applied Automation, D.C.Long.
Performance Analysis in Threaded-Code Systems, M.A.Perry.
Generic Operators, T.Rayburn.
Control Simulation for a Tape Deck, L.Richter-Abraham.
Preliminary Report on the Novix 4000, C.L.Stephens/W.P.Watson.
A Set of Forth Words for Electrical Network Analysis, J.Storjohann.
Forth Language Extension for Controlling Interaktive Jobs on Other Machines, D.K.Walker.
RTDF: A Real-Time Forth System Including Multi-tasking, Wijnands/Bruijn.
Event-Driven Multi-tasking: A Syntax, J.Zander

Das 'IF THEN Konstrukt' wurde gestürzt!

Ein altes, immer virulentes Thema wurde auf S.21 aufgegriffen: Die Tradition der Syntax von Conditionals in Forth, genannt das 'IF THEN Konstrukt'. Es ist immer wieder Quelle von Verwirrung und Frustration für den Forth-Studenten beim Einstieg in die Sprache. Der Artikel von R.E.Apra, "Lets keep it simple" befasst sich damit. Auch auf die Gefahr hin, Mitglieder der FG mit Kopien

zu langweilen, weil sie parallel die Forth Dimmensions lesen, wurde dieser Artikel ins Heft aufgenommen. Sie finden Ihn weiter hinten in Heft.

Denn nicht zum ersten Mal wurde die Redaktion auf dieses Thema aufmerksam gemacht. Nur bisher wollten wir nicht an der guten (?) alten Tradition dieses Konstruktes rütteln, um nicht noch mehr Verwirrung zu schaffen. Doch wie es scheint, formiert sich global eine 'IFTRUE IFFALSE Liga', und schiebt uns einen hässlichen alten Bekannten erneut unter, das ENDIT. Früher hieß er noch ENDIF.

Das können wir nicht einfach so hinnehmen, Freunde des Forth. Her mit Eurer Phantasie! Das 'IF THEN Konstrukt' ist kein Dogma mehr! Es darf munter vom Sockel geschubst werden. Lasst Euch was Zündendes einfallen und schickt es an die Redaktion. Damit könnt Ihr in die Geschichte des Forth eingehen! mk

* * *

FORTHQUELLEN

Style Conventions

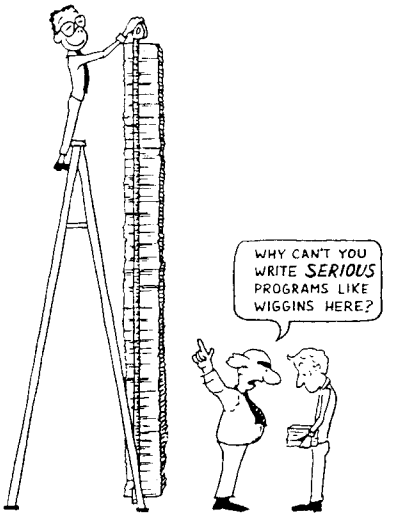
nach Leo Brodie

Auf mehrfachen Wunsch einiger Autoren und Programmierer hin gebe ich hier die 'stilistischen Konventionen des Forth' wieder, wie sie im Buch 'Thinking Forth' von Leo Brodie gesammelt worden sind.

Über die Sammlung von Brodie läßt sich streiten. Die Abkürzungen sind aus der englischen Sprache gewonnen und für's deutsche 'un-mnemonisch'. Und über die 'kryptographischen Elemente' der Namensgebung (z.B. /LINE oder SLIDE<) ließt man heute besser hinweg. Der Trend geht zu langen Klartext-Namen (also UP-LINE oder SLIDE-BACK), die quasi selbstdokumentierenden Code erzeugen sollen.

Aber ein Teil der Syntax-Regeln hat sich eingebürgert, ist wegen starker Aussagekraft zur 'Konvention' geworden. So das nachgestellte Fragezeichen, um anzuzeigen, das ein Flag auf dem Stack hinterlegt wird (z.B. FERTIG?).

Die Sammlung von Brodie ist ein Extrakt der amerikanischen Forth-Praxis. Sie können natürlich auch eigene Syntax erfinden. Was Sie auch tuen, seine Regelsammlung anwenden, erweitern oder umwerfen, sorgen Sie für LESBAREN Code.



Die folgenden Seiten wurden nicht ins deutsche übersetzt sondern aus dem Original kopiert. Die Abkürzungen ergeben nämlich nur einen Sinn im Zusammenhang mit den englischen Begriffen. mk

Spacing and Indentation Guidelines

1 space between the colon and the name
2 spaces between the name and the comment
2 spaces, or a carriage return, after the comment and before the definition
3 spaces between the name and definition if no comment is used
3 spaces indentation on each subsequent line (or multiples of 3 for nested indentation)
1 space between words/numbers within a phrase
2 or 3 spaces between phrases
1 space between the last word and the semicolon
1 space between semicolon and IMMEDIATE (if invoked)

No blank lines between definitions, except to separate distinct groups of definitions

Stack-Comment Abbreviations

n	single-length signed number
d	double-length signed number
u	single-length unsigned number
ud	double-length unsigned number
t	triple-length
q	quadruple-length
c	7-bit character value
b	8-bit byte
?	boolean flag; or:
t=	true
f=	false
a or adr	address
acf	address of code field
apf	address of parameter field
'	(as prefix) address of
s d	(as a pair) source destination
lo hi	lower-limit upper-limit (inclusive)
#	count
o	offset
i	index
m	mask
x	don't care (data structure notation)

An "offset" is a difference expressed in absolute units, such as bytes.

An "index" is a difference expressed in logical units, such as elements or records.

Input-Stream Comment Designations

c	single character, blank-delimited
name	sequence of characters, blank delimited
text	sequence of characters, delimited by non-blank

Follow "text" with the actual delimiter required; e.g., text" or text).

Samples of Good Commenting Style

Here are two sample screens to illustrate good commenting style.

```
Screen # 126
  0 \ Formatter          Data Structures -- p.2              06/06/83
  1  6 CONSTANT TMARGIN \ line# where body of text begins)
  2 55 CONSTANT BMARGIN \ line# where body of text ends)
  3
  4 CREATE HEADER  82 ALLOT
  5   { 1left-cnt ¦ 1right-cnt ¦ 80header }
  6 CREATE FOOTER  82 ALLOT
  7   { 1left-cnt ¦ 1right-cnt ¦ 80footer }
  8
  9 VARIABLE ACROSS    \ formatter's current horizontal position
 10 VARIABLE DOWNWARD  \ formatter's current vertical position
 11 VARIABLE LEFT      \ current primary left margin
 12 VARIABLE WALL      \ current primary right margin
 13 VARIABLE WALL-WAS  \ WALL when curr. line started being formt'd
 14
 15


Screen # 127
  0 \ Formatter           positioning -- p.1                 06/06/83
  1 : SKIP  ( n)  ACROSS +! ;
  2 : NEWLEFT  \ reset left margin
  3    LEFT @  PERMANENT @ +  TEMPORARY @ +  ACROSS ! ;
  4 : \LINE  \ begin new line
  5    DOOR  CR'  1 DOWNWARD +!  NEWLEFT  WALL @  WALL-WAS ! ;
  6 : AT-TOP?  ( -- t=at-top)  TMARGIN  DOWNWARD @ = ;
  7 : >TMARGIN  \ move from crease to TMARGIN
  8    0 DOWNWARD !  BEGIN  \LINE  AT-TOP? UNTIL ;
  9
 10
 11
 12
 13
 14
 15
```

Naming Conventions

Meaning	*Form*	*Example*
Arithmetic		
integer 1	1name	1+
integer 2	2name	2*
takes relative input parameters	+name	+DRAW
takes scaled input parameters	*name	*DRAW
Compilation		
start of "high-level" code	name:	CASE:
end of "high-level" code	;name	;CODE
put something into dictionary	name,	C,
executes at compile time	[name]	[COMPILE]
slightly different	name' (prime)	CR'
internal form or primitive	(name)	(TYPE)
	or < name>	<TYPE>
compiling word run-time part:		
systems with no folding	lower-case	if
systems with folding	(NAME)	(IF)
defining word	:name	:COLOR
block-number where overlay begins	namING	DISKING

Meaning	*Form*	*Example*
Data Structures		
table or array	names	EMPLOYEES
total number of elements	#name	#EMPLOYEES
current item number (variable)	name#	EMPLOYEE#
sets current item	(n) name	13 EMPLOYEE
advance to next element	+name	+EMPLOYEE
size of offset to item from beginning of structure	name+	DATE+
size of (bytes per) (short for BYTES/name)	/name	/EMPLOYEE
index pointer	> name	>IN
convert address of structure to address of item	> name	>BODY
file index	(name)	(PEOPLE)
file pointer	−name	−JOB
initialize structure	0name	0RECORD
Direction, Conversion		
backwards	name<	SLIDE<
forwards	name >	CMOVE >
from	< name	< TAPE
to	> name	> TAPE
convert to	name >name	FEET > METERS
downward	\name	\LINE
upward	/name	/LINE
open	{name	{FILE
close	}name	}FILE
Logic, Control		
return boolean value	name?	SHORT?
returns reversed boolean	−name?	−SHORT?
address of boolean	'name?	'SHORT?
operates conditionally	?name	?DUP (maybe DUP)
enable	+name	+CLOCK
or, absence of symbol	name	BLINKING
disable	−name	−CLOCK −BLINKING
Memory		
save value of	@name	@CURSOR
restore value of	!name	!CURSOR
store into	name!	SECONDS!
fetch from	name@	INDEX@
name of buffer	:name	:INSERT
address of name	'name	'S
address of pointer to name	'name	'TYPE
exchange, especially bytes	>name<	>MOVE<
Numeric Types		
byte length	Cname	C@
2 cell size, 2's complement integer encoding	Dname	D+
mixed 16 and 32-bit operator	Mname	M*
3 cell size	Tname	T*
4 cell size	Qname	Q*
unsigned encoding	Uname	U.

Meaning	*Form*	*Example*
Output, Printing		
print item	.name	.S
print numeric (name denotes type)	name.	D. , U.
print right justified	name.R	U.R
Quantity		
"per"	/name	/SIDE
Sequencing		
start	< name	< #
end	name >	# >
Text		
string follows delimited by "	name"	ABORT" text"
text or string operator	"name	"COMPARE
(similar to $ prefix in BASIC		
superstring array	"name"	"COLORS"

How to Pronounce the Symbols

!	store
@	fetch
#	sharp (or "number," as in #RECORDS)
$	dollar
%	percent
^	caret
&	ampersand
*	star
(	left paren; paren
)	right paren; paren
–	dash; not
+	plus
=	equals
{ }	faces (traditionally called "curly brackets")
[ ]	square brackets
"	quote
'	as prefix: tick; as suffix: prime
~	tilde
|	bar
\	backslash. (also "under," "down," and "skip")
/	slash. (also "up")
<	less-than
	left dart
>	greater-than
	right dart
?	question (some prefer "query")
,	comma
.	dot

* * *

Der ForthTREE

Informationssystem der Forth Gesellschaft eV
Vorgestellt von Marco Pauck

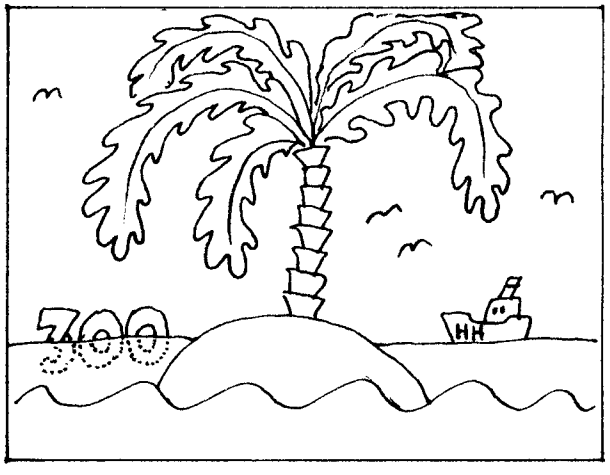
Das Konzept des TREE's und Unterschiede zu üblichen Mailboxen

Seit dem 28.April 86 ist der ForthTREE der Forth Gesellschaft eV online. Es handelt sich dabei nicht um eine der üblichen Mailboxen, sondern um ein neuartiges Informations- und Kommunikationssystem. Damit ist 'Computer Conferencing' nun auch auf relativ kleinen Rechnern möglich.

Auf größeren Systemen kennt man diese Möglichkeit schon seit einiger Zeit, doch die kleinen Systeme auf Mikrocomputer-Basis ähneln in ihrem Aufbau eher konventionellen Zeitungen. Sie enthalten zum einen verschiedene redaktionelle Rubriken, unter denen Informationen abgerufen werden können, und zum anderen die schwarzen Bretter (= Kleinanzeigen), wo Gesuche, Gebote u.ä. Platz finden. Insgesamt sind die Kommunikationsmöglichkeiten, ähnlich wie bei einer Zeitung, recht eingeschränkt. Speziell die offene Kommunikation zwischen den Usern ist aufgrund der linearen Brettstruktur kaum möglich. Die Nachrichten sind einfach in Form einer Liste chronologisch aneinander gereiht, und oftmals müssen sie sogar zeitlich rückwaerts gelesen werden, so daß man Antworten auf andere Nachrichten zuerst gezeigt bekommt. Wenn ein User nun auf eine ältere Nachricht antworten will, so wird seine Nachricht an das Ende der Liste gehängt, wo sie eigentlich nicht hingehört, da sie sich auf eine ganz andere Stelle bezieht.

Der Ausweg aus diesem Dilemma ist die wohl vielseitigste Möglichkeit Informationen zu strukturieren: der n-äre Baum (tree). Mit ihm ist es möglich, eine Nachricht an jede beliebige andere zu hängen. Die Nachrichten bilden also die Blätter (Knoten) eines Baumes, der entlang seiner Äste durchwandert werden kann. Dadurch ist es nun möglich Diskussionen zu führen, da jederzeit an einen bestehenden Zweig durch das Anfügen von Nachrichten an den gewünschten Punkten neue Aspekte, in Form von neuen Unterzweigen, angebracht werden können.

Um effektiv im TREE arbeiten zu können gibt es eine Reihe von Befehlen und Optionen, die hier jedoch nicht näher aufgeführt werden sollen. Sie sind im HILFE Zweig des TREE's abrufbar.



Die Hardware des TREE's ist ein PC-Kompatibler aus Fernost mit Harddisk und Modem. Die Software des TREE's ist in Forth geschrieben, könnte aber natürlich auch in den meisten anderen Vielzweckspra-chen realisiert werden. Warum ich Forth jedoch für besonders angemessen halte, versuche ich im folgenden näher zu begründen.

Erste Erfahrungen (und etwas ForthTREE Philosophie)

Der TREE ist während des Schreibens dieses Artikels etwas über einen Monat in Betrieb. Die Erfahrungen dieser Zeit zeigten für mich einige verblüffende Parallelen zu Erfahrungen im Umgang mit Forth:

1. - Polarisierendes Konzept
Die meisten User reagierten eindeutig - entweder positiv oder negativ (letzteres gottseidank weniger). Ähnlich wie es fast nur fanatische Forth-Gegner oder Forth-Apostel zu geben scheint, sind auch im Falle des TREE's die Ansichten recht geteilt.

2. - Gewöhnungsbedürftiges Prinzip
Speziell wenn ein User bereits andere Boxen kennt, benötigt er einige Zeit, um das Prinzip zu verstehen und anwenden zu können. Diese Eigenschaft muß zweifelsohne auch Forth zugesprochen werden.

3. - 'Aha-Erlebnis' notwendig
Viele User sind zwar relativ schnell in der Lage den TREE zu benutzen, benötigen aber wesentich länger und/oder ein konkretes 'Aha-Erlebnis', um den TREE wirklich in seinen entscheidenden Möglichkeiten auszuschöpfen. Erfahrungsgemäß bleiben ebenso viele Forth Einsteiger leicht auf einem Stand stehen, der das Schreiben von Forth Programmen erlaubt, jedoch die speziellen Eigenheiten und Möglichkeiten von Forth nicht genügend berücksichtigt. Derartigen Programmen sieht man sehr leicht die Basic-Pascal-Fortran-Denkstruktur an, die hinter ihnen steckt.

4. - Einfachheit und Vielseitigkeit
Ein Grund für die Eleganz 'richtiger' Forth Programme ist oft die Einfachheit ihrer inneren Struktur. Ähnliches zeigt sich auch beim TREE, da er, im Gegensatz zu vielen anderen Systemen, praktisch nur eine, auf allen Ebenen einheitliche Struktur, die des Baumes, aufweist. Auf diese einheitliche Datenstruktur werden Befehle und Optionen mit (leider nicht immer ganz) klar definierter orthogonaler Bedeutung angewendet. Eine solche Beschaffenheit ist z.B. sicher auch ein Grund für den Erfolg von UNIX (siehe Dateistruktur, Pipes/Filter, Befehlssatz).

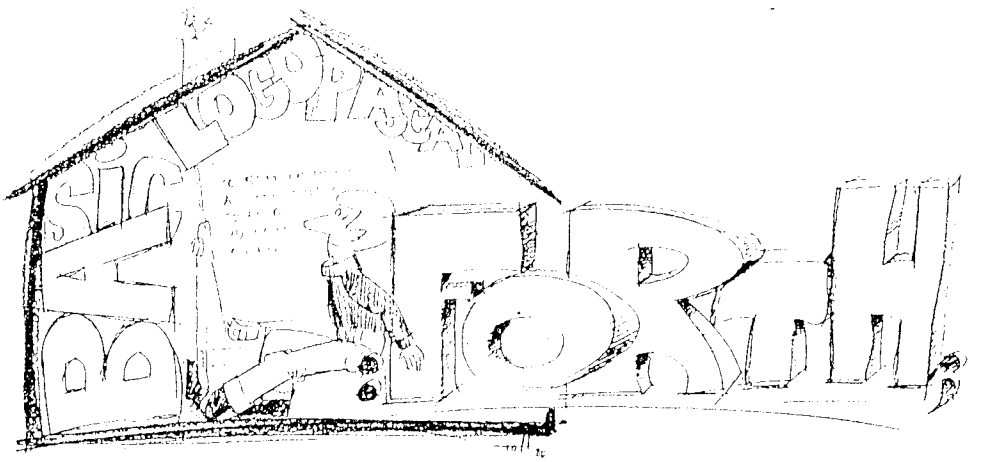
5. - 'Organisches Wachsen'
Die für den SysOp anfallende Wartungsarbeit ist relativ gering. Nachdem das Bäumchen gepflanzt ist, ist schon die meiste Arbeit getan. In der dann folgenden Zeit des Anwachsens muß noch gelegentlich ein Ast etwas beschnitten oder besonders kunstvoll geführt (Bonsai) werden, alles weitere geschieht aber 'von selbst'. Um wieder etwas mehr zum ForthTREE zurückzukommen: Zweige, die zu Beginn vorgesehen waren, jedoch den Ambitionen der User nicht entsprachen, sind schnell abgestorben, während sich an anderen Stellen, dort wo Bedarf bestand, spontan wild sprießende Triebe bildeten, ohne daß es dazu einer Planung des SysOps bedurft hätte. Ähnlich sind meist die Erfahrungen eines Forth Programmierers, wenn es um das Erstellen einer Applikation geht. Sehr schnell wird die ursprüngliche Planung durch das interaktive Austesten der ersten Software Prototypen ueber den Haufen gewor-

fen, und dann kommt die Fähigkeit von Forth, schnelle Entwicklungszyklen und 'rapid prototyping' durchzuführen, zum Zuge. Eine solche Vorgehensweise ist selbstverständlich ein Graus fuer jeden anständigen Pascal- und Ada-Programmierer, doch welcher Forthler möchte schon zur Dijkstra'schen Ideologie des sturen Top-Down-Designs zurückkehren?

Ausblick in die Zukunft

Zur Zeit des Starts des TREE's meinten einige SysOps des Hamburger Raumes, daß in Deutschland kein öffentliches System mit gehobenen Ambitionen bestehen könnte, da die User hierfür 'noch nicht reif genug' seien. Die bisherigen Erfahrungen scheinen dies erfreulicherweise nicht zu bestätigen. Die Gründe hierfür liegen zum einen sicher in den Vorzügen des Baum-Konzeptes, aber auch in der Zweckorientiertheit des TREE's (als Kommunikationssystem speziell für Forthler), denn die Mailboxszene krankt u.a. daran, daß in zu vielen Boxen das gleiche steht, da zu wenig Spezialisierung erfolgt. Fast alle Mailboxen beklagen sich auch über die geringe Mitarbeit ihrer User. Beim ForthTREE scheint dieses Problem zumindest etwas abgeschwächt zu werden, da für den User die Hemmschwelle, eine eigene Nachricht anzubringen, aufgrund der Baumstruktur gesenkt und wohl auch die Motivation gesteigert wird. Erstrebenswert in o.g. Zusammenhang (Spezialisierung versus Informationsfülle und Redundanz) wäre sicher der Aufbau eines Netzwerkes vieler regionaler TREE's, die ihre lokalen Informationen selbsttätig gegenseitig austauschen. Eine derartige Vernetzung würde die Qualität der vorhandenen Informationen enorm steigern, denn nur die Fülle allein (z.Z. mehrere hundert Boxen in Deutschland) ist kein Garant für Qualität, ja noch nicht einmal für Informationsvielfalt. Es bedarf auch einer flexiblen, dynamischen Ordnung im Chaos, die der Idee von lokaler Autonomie und globaler Koordination gerecht wird.

* * *



ROLF KRETZSCHMAR
ROTE GASSE 7
5112 BAESWEILER
TELEFON 02401/4390

"Gestatten, CommuniTree."
von der 'Mehlschachtel' selbst

Tüüüt...

<CR> <CR>

Willkommen im CommuniTree
der Forth Gesellschaft eV

Letzte Aenderung 3-JUN-86

*** Achtung !
*** Es wurden einige grundlegende Aenderungen vorgenommen, daher
*** sollte die neue Nachricht UPDATES unbedingt gelesen werden !

Der TREE hat am 28-APR-86 seinen Probebetrieb aufgenommen. Auftretende Stoerungen und sonstige Probleme bitte in Form einer Nachricht an den dafuer eingerichteten Zweig TESTPHASE haengen.

Ab sofort sind alle Systemaenderungen in SYS-NEWS im Zweig SYS-INFO dokumentiert.

Neulinge sollten zuerst einmal

READ EINFUEHRUNG

tippen, sie erhalten dann grundlegende Informationen, um mit dem TREE arbeiten zu koennen.

User, die die Grundlagen schon etwas kennen, duerfen dafuer

READ WEITERFUEHRUNG

tippen, dort gibt es fuer sie ein bisschen zusaetzliche Hilfestellung.

Bitte beachten:
Der TREE ist Dienstags von 18-20 Uhr NICHT online. Waehrend dieser Zeit meldet sich eine (freundliche) menschliche Stimme, die Auskuenfte ueber die Forth Gesellschaft eV geben kann.

READ HILFE jederzeit moeglich
READ KONFERENZEN um zu beginnen

<CNTRL-S> stop, <CNTRL-Q> weiter
<C> abbrechen, <K> ueberspringen

Befehl? READ KONFERENZEN

```
*** KONFERENZEN              9-MAR-86
  gelesen:  321

z.Z. laufen folgende Konferenzen :

    +++ Folgenachrichten +++
 FORTH                       2-JUN-86
 GRAFITTI                   11-APR-86
 KOMMUNIKATION              11-MAY-86
 UMWELT                     16-MAY-86
 KONTAKTE                   15-APR-86
 RECHNERECKE                 1-JUN-86
 MARKT                       2-JUN-86
 GAST                       16-APR-86
 PRIVAT                     10-APR-86
 TESTPHASE                   1-MAY-86
 SYS-INFO                    7-MAY-86
 EINFUEHRUNG                 8-MAY-86
 HILFE                       7-MAY-86


*** EINFUEHRUNG              8-MAY-86
neuer Vorgaenger: KONFERENZEN          gelesen:   51
```

Alle Informationen im TREE sind in Form einer Nachricht gespeichert. Diese Einfuehrung hier ist ein Beispiel dafuer.

Die Nachrichten sind nun durch eine Baumstruktur miteinander verbunden. Die Wurzel des Baumes traegt den Namen KONFERENZEN, an ihr haengen u.a. die Nachrichten GAST, KONTAKTE, FORTH.SYSTEME usw. An diesen koennen wiederum andere Nachrichten haengen, so dass sich die fuer einen Baum typische Struktur von sich immer weiter aufspaltenden Aesten und Zweigen ergibt (fuer Freaks: naerer Baum).

Um nun an die Information in den Nachrichten heranzukommen, gibt es eine Anzahl von Befehlen. Der wichtigste von ihnen ist READ. Wenn man den Namen einer Nachricht kennt, so bekommt man mit

READ <Name>

den Inhalt der Nachricht gezeigt. Anschliessend werden noch die Namen der Nachrichten gelistet, die an der eben gelesenen haengen. Auf diese Weise kann man sich durch einen Zweig des Baumes arbeiten. Man muss jedoch keinen festen Weg gehen, sondern kann jede Nachricht im TREE direkt anwaehlen, wenn man ihren Namen kennt.

Um sich Uebersicht im TREE zu verschaffen oder nach bestimmten Informationen zu suchen, gibt es weitere Befehle und Optionen, die alle im Zweig HILFE erklaert werden.

Falls man bereits andere Mailboxen kennt, so ist einige Umgewoehnung erforderlich, doch dafuer bietet der TREE Moeglichkeiten, die sonst nirgends geboten werden. Weitere Informationen ueber das Konzept des TREEs erfaehrt man in der Nachricht TREE? .
Und nun viel Spass im Forth TREE !

*** WEITERFUEHRUNG 9-MAY-86
Vorgaenger: EINFUEHRUNG gelesen: 33

Nach den ersten 10 Tagen des Probebetriebs musste ich feststellen, dass nicht wenige User ausschliesslich den READ Befehl benutzen. Das ist zwar durchaus moeglich, aber sehr umstaendlich und wenig sinnvoll. Daher werde ich im folgenden ein paar Beispiele bringen, wie man die wichtigsten der restlichen Befehle normalerweise einsetzt. Die Namen der Befehle und Optionen sind der Uebersichtlichkeit wegen ausgeschrieben, doch man kann sie natuerlich auch auf den Anfangsbuchstaben abkuerzen.

Angenommen, ich war seit dem 8.Mai nicht mehr im TREE und moechte nun wissen, was es Neues gibt:

```
INDEX KONFERENZEN  STARTING 8-MAY-86
```

Nehmen wir weiter an, dass eine Unzahl von Titeln gelistet worden ist, die ich gar nicht alle lesen kann und will (Telefongebuehren). Mich interessieren aber auf jeden Fall alle Nachrichten, in denen es um PC's geht:

```
READ KONFERENZEN  STARTING 8-MAY-86  FIND PC
```

Ein anderes Beispiel: Ich moechte mir den Zweig FORTH.SYSTEME naeher ansehen. Dazu verschaffe ich mir erst einmal etwas Uebersicht:

```
INDEX FORTH.SYSTEME
```

Aha, einige Nachrichten sind auf jeden Fall neu, bei anderen weiss ich nicht genau, ob ich sie schon gelesen habe. Darum lese ich nun alle:

```
READ FORTH.SYSTEME COMPLETE
```

und wenn ich beim Lesen merke, dass ich eine Nachricht bereits kenne oder sie mich nicht weiter interessiert, so tippe ich einfach kurz auf die 'K' Taste und schon wird zur naechsten Nachricht uebergegangen.

Das sollte eigentlich genuegen, um einen Eindruck von etwas komfortablerer Vorgehensweise im TREE zu bekommen. Um vertraut mit den Befehlen zu werden, muss etwas mit ihnen experimentiert werden.

Ganz besonders moechte ich noch jedermann die Benutzung des Befehls ADDTO empfehlen. Mit seiner Hilfe kann man naemlich eigene Nachrichten an beliebige andere haengen, und sich so vom passiven Konsumenten in einen aktiven Mitgestalter des TREEs verwandeln ! Und das sind schliesslich die Leute, fuer die der TREE in's Leben gerufen wurde !

*** TREE? 10-APR-86
neuer Vorgaenger: EINFUEHRUNG gelesen: 52

Informationen zum Konzept dieses TREE's.
Der TREE unterscheidet sich in seinem Aufbau und seiner Nutzweise

von den ueblichen Mailbox-Systemen.

Die bisher bekannten Boxen bieten den Benutzern nur die Moeglichkeit, ihre Nachrichten in linearer, chronologischer Folge an einen bestimmten Menue-Punkt wie z.B. 'Kontakte' zu haengen. Dadurch ist es kaum sinnvoll moeglich, auf eine aeltere Nachricht eines anderen Benutzers zu antworten, denn der inhaltliche Zusammenhang geht aus der Reihenfolge der Nachrichten nicht hervor. Die Baumstruktur des TREE's ermoeglicht es dem Benutzer, seine Nachricht mit einem Kommentar, einer Ergaenzung oder einer Weiterfuehrung an jede andere im TREE vorhandene Nachricht zu haengen. Somit werden auf einfache Weise zeitlich ungebundene Meinungsaustaeusche und Diskussionen ermoeglicht. Der beabsichtigte Nutzen des TREE's liegt daher nicht so sehr in den sonst ueblichen 'Suche,Biete,Kontakte'-Funktionen oder in persoenlichen und gemeinschaftlichen Mailbox-Faechern, sondern in offenen, dynamischen Diskussionen unter den Benutzern. Natuerlich werden die o.g. bekannten Funktionen dennoch vom TREE angeboten. Der TREE ist also keine Mailbox im ueblichen Sinne, sondern ein recht vielseitiges neues Informations- und Kommunikationsmedium, das in seinem Nutzen allerdings sehr von seinen Benutzern abhaengt. Wenn diese nur zum Lesen der Nachrichten in den TREE kommen, oder ihre eigenen Nachrichten auf das leider verbreitete 'CB-Geschwaetz' beschraenken, so wird der TREE nur eine der unzaehligen 'Mehlschachteln' sein. Sind die Benutzer jedoch interessiert, sich an den laufenden Diskussionen zu beteiligen, so wird der TREE eine interessante und bereichernde Form des Informationsaustausches werden.

In diesem Sinne wuensche ich mir eine rege Teilnahme aller Interessierten an den Moeglichkeiten des TREE's !

Befehl? BYE
Danke fuer Ihren Anruf"

* * *

1986 International Workshop on Forth and Its Applications

Vom 31.10. - 2.11.86 findet im National Taiwan Institute of Technology in Taipei ein Forth-Workshop statt. Die R.O.C. Forth Interest Group und verschiedene Sponsoren der Industrie sind beteiligt. Arbeiten zu folgenden Themen können eingereicht werden:

1. Forth Computer and Forth Systems
2. Forth Applications
   Artificial intelligence, Real time control, Business, Space-based, Laboratory, Personal Systems, Others.
3. Forth Technology
   Finite state machines, Data structures, Hybrid hardware/software systems

Abstracts und die Arbeit selbst einreichen an W.T. Huang, Institute for Information Industry, 12th fl., 116 Nanking East Road, Sec. 2 Taipei (104), Taiwan, Republic of China. Oder an C.H. Ting, 156 14Ave. San Mateo, Ca. 94402 U.S.A.

Bernd Pennemann
Hamburg

8.Juni 1986

EIN TERMINALPROGRAMM IN FORTH

0.) Vorwort

Nachdem der "Tree" der Forth Gesellschaft an das Netz gegangen ist, möchte ich an dieser Stelle ein Terminal- und Kommunikationsprogramm vorstellen. Es ist in Forth-83 geschrieben, wobei gängige Erweiterungen verwendet wurden. Als Implementationsbeispiel wurde der Atari mit Volksforth83 gewählt, wobei auch die Versionen ab 3.71 mit Fileinterface unterstützt werden. Das Programm sollte leicht auf andere Rechner übertragen werden können, weil lediglich die Primitives RX RX? TX TX? und MSEC rechnerabhängig sind.

Im folgenden werde ich beschreiben, was unser Terminalprogramm kann, die Funktionsweise erläutern und schließlich die Implementation möglicher Erweiterungen skizzieren.

1.) Massenspeicherphilosophie

In Forth kennen wir zwei gängige Verfahren, den Massenspeicher (Floppy) zu organisieren. Die einfache Aufteilung des gesamten Massenspeichers in Blöcke zu 1 Kbyte oder die Anwendung dieser Blockufteilung innerhalb von Files des Betriebssystems des Rechners. Das volksFORTH83 unterstützt jetzt beide Verfahren, daher stelle ich im folgenden zwei Versionen des Programms vor, eine mit und eine ohne Files.

Ein Forthblock wird gewöhnlich, wenn er Texte enthält, in 16 Zeilen zu 64 Zeichen unterteilt und "Screen" genannt. Am einfachsten machen wir uns das Leben, wenn wir uns an diese Konvention halten; empfangene Texte werden in diesem Format gespeichert und zu sendende Texte sollten ebenfalls so aufgebaut sein. Der Vorteil dieser Lösung besteht darin, daß wir alle "klassischen" Werkzeuge des Forthsystems benutzen können, um mit diesen Texten zu arbeiten. Zu sendende Texte werden mit dem Forth-Editor editiert, empfangene können mit LIST ausgegeben werden usw. Insbesondere können empfangene Texte dann auch ohne weiteres kompiliert werden ! Damit ist der Programmaustausch zwischen verschiedenen Rechnern über den Tree möglich.

Der Nachteil dieser Lösung besteht darin, daß der Tree normalerweise auf 80 Zeichen pro Zeile eingestellt ist, so daß man bei jedem Anruf dort erst mal mit dem Tree-Befehl TERMINAL die Zeilenlänge ändern muß, da sonst bei längeren Texten jede zweite Zeile eines Screens fast leer ist.

Mein Terminalprogramm gestattet das Downloaden eines oder mehrerer Screens , die vorher mit dem Editor erstellt wurden. Dazu tippt man während des Terminalbetriebs " ESC d " ein. Das System fragt dann nach der Nummer des ersten und die Anzahl der zu sendenden Screens.
Beim Upload muß vor dem Aufruf der erste Screen, auf dem der Text abgelegt werden soll, mit START festgelegt werden. START wird in der folgenden Form benutzt :

```
n START    ( n ist der erste Screen fürs Uploaden)
```

Ist der erste Screen voll, wird automatisch auf den nächsten übergegangen. Dieser übergang ist nicht ganz leicht und wird weiter unten erklärt.
Bei Benutzung des Fileinterfaces gestaltet sich die Sache etwas einfacher. Nach dem Tippen von " ESC d " muß man nur den Filenamen eingeben. Es werden dann alle Screens dieses Files übertragen. Auch START wird anders benutzt:

START <filename>

Auf START muß der Name des Files folgen, in das der ge-uploadete Text geschrieben wird. Dieses File wird vom Forth automatisch verlängert, wenn es voll ist.

2.) Empfangen und Nichtempfangen

Es ist nicht ganz einfach zu verhindern, daß keine Zeichen beim Empfang verlorengehen.
Unsere Lösung sieht folgendermaßen aus: Das Programm wartet darauf, daß der Bediener eine Taste drückt. Während es das tut, guckt es andauernd nach, ob ein Zeichen empfangen wurde, das dann angezeigt und evtl. auf den Massenspeicher geschrieben werden muß. Wenn das Anzeigen und Abspeichern nicht länger als 30 msec (bei 300 Baud) dauert, kann so kein Zeichen verlorengehen, sofern kein Zugriff auf die Diskette erfolgt (der dauert nämlich typisch etwa 1 Sekunde).

Es klingt zunächst wiedersprüchlich, daß der empfangene Text auf dem Massenspeicher gespeichert wird und doch kein Zugriff erfolgen darf. Das Rätsel wird durch den Blockmechanismus von Forth gelöst. Wenn wir erst einmal einen Block, auf den durch Upload geschrieben wird, im Speicher haben, können wir bis zu 1024 Zeichen speichern, ohne daß ein erneuter Diskettenzugriff erfolgen muß. Diskettenzugriffe sind also nur kritisch, wenn beim Upload von einem auf den nächsten Screen übergegangen wird. In diesem Fall aktiviert das Programm ein automatisches XON/XOFF-Protokoll. Empfängt nämlich eine Station das Zeichen XOFF (d.h. ein <CTRL>-S), so muß sie sofort mit dem Senden aufhören und warten, bis ein XON (d.h. ein <CTRL>-Q) eintrudelt. Nun können aber, nachdem wir XOFF gesendet haben, durchaus noch einige Zeichen ankommen, denn es dauert ja eine gewisse Zeit, bis unser XOFF übertragen und von Gegenseite erkannt wurde. Für diesen Fall müssen wir ebenfalls Vorsorge treffen.

Unser Terminalprogramm macht nun folgendes: Wenn ein Diskettenzugriff erfolgen soll, wird erst einmal XOFF gesendet. Alle noch folgenden Zeichen werden empfangen und zwischengespeichert, bis nach einem Zeichen eine Pause von 0,25 Sekunden eintritt. Dann erfolgt der Diskettenzugriff. Anschließend werden die zwischengespeicherten Zeichen "richtig empfangen", also angezeigt und ge-uploaded. Wenn das geschehen ist, senden wir XON (besser gleich zweimal, denn es geht so manches verloren...) und es geht ganz normal weiter.

Dieser Mechanismus für eine allgemein gehaltene Version ist notwendig, weil viele Rechner heutzutage eben nicht gleichzeitig auf der RS232-Schnittstelle Zeichen empfangen und Diskettenzugriffe durchführen können (der Atari kann das und sogar noch tollere Sachen..)

Diese Strategie wird auch verfolgt, wenn beim download der zu sendende Bereich von Screens (bzw. der Name des zu sendenden Files...) eingegeben oder ein Screen gelesen werden soll.

Das Zwischenspeichern der Zeichen, die nach einem XOFF empfangen wurden, geschieht mit Hilfe eines zirkularen Puffers (auch Queue oder FIFO genannt) . Dessen Konzept soll getrennt beschrieben werden.

3.) LF ja oder nein ?

Ganze Glaubenskriege werden um die Frage geführt, ob ein LF (Zeilenvorschub) auf ein CR (Wagenrücklauf) beim Senden folgen soll. Ich schlage vor, keine LF zu senden und empfangene zu ignorieren. Wer LF senden will, muß einfach vor dem Start des Programms

```
LF? ON
```

eingeben.
Ähnlich ist es mit den Nullen. Manche Programme benötigen eine Pause nach jedem CR, die mit Nullen (hex 00) aufgefüllt wird. Braucht der Partner Nullen, so gibt man vor dem Start des Programms

```
n NULLS
```

ein, wobei n die Zahl der zu sendenden Nullen ist.

4.) Bedienung des Programms

Das Programm wird durch Eingabe von

```
TERMINAL
```

gestartet. (Benutzung von START nicht vergessen !)
Jede Taste, die jetzt gedrückt wird, führt dazu, daß das entsprechende Zeichen ausgesandt wird. Die Ausnahme ist ESC; nach Drücken von ESC befindet man sich im Kommandomodus. Folgende Kommandos stehen zur Verfügung :

ESC + ESC Senden eines ESC-Zeichens

ESC + D Download eines Textes. Wir werden je nach Version entweder nach dem ersten und der Anzahl der zu übertragenden Screens gefragt, oder nach dem Namen des Files.

ESC + U Upload an- oder ausschalten. Es wird angezeigt, ob Upload an- oder ausgeschaltet wurde.

ESC + X Terminalprogramm beenden.

Ein wichtiger Trick sei verraten : Man kann die Gegenseite mit CTRL-S anhalten, das Programm mit ESC X verlassen und sich z.B. den Text, der ge-uploaded wurde, ansehen oder LF? ändern oder...
und anschließend die Kommunikation mit TERMINAL und CTRL-Q fortsetzen, ohne das irgendetwas schiefläuft.
Diese Möglichkeit ist übrigens auch der Grund, warum am Ende von TERMINAL kein SAVE-BUFFERS ausgeführt wird. SAFE-BUFFERS ruft beim Sichern der Screens das Wort R/W auf. Wir haben R/W gerade so geändert haben, daß es zunächst ein XON und anschließend ein XOFF sendet. Damit würde die Wirkung des CTRL-S aufgehoben, die Gegenstation würde wieder senden dürfen und es könnten Zeichen verloren gehen.

5.) Erweiterungen

Einer der Vorteile von Forth ist die Ergänzungsfreundlichkeit. Eine Vielzahl von Erweiterungen sind bei diesem Programmkonzept möglich :

-) Massenspeicher
Der Upload- und Download-Mechanismus kann an alle möglichen Textprogramme angepaßt werden. Ändert man z.B. bei Verwendung des Atari mit Fileinterface UPCR wie folgt :

```
: atari.upcr   #cr upemit  #lf upemit ;
```

so bekommt man statt Forth-Screens ganz normale Textfiles beim Upload. Am besten ändert man jedoch nicht das vorhandene Format, sondern fügt neue hinzu.

-) Zeichenkonversion
Umlaute sollten entsprechend der DIN-Norm ausgesandt werden. Man kann eine Variable DIN vorsehen, die, wenn sie gesetzt ist, dazu führt, daß ein "[" in ein "Ä" umgewandelt wird ... Man kann dann vor Aufruf von TERMINAL

```
DIN on
```

eingeben und sich dann in deutsch unterhalten. Wichtig ist auch, daß C64-Benutzer nicht vergessen, von und nach ASCII zu wandeln !

-) Funktionstasten
Man kann Funktionstasten einbauen, indem man, wenn F1 gedrückt wird, die Zeile 3 eines bestimmten Screens sendet, bei F2 die Zeile 4 usw.
Diese Lösung gestattet es dann, die Funktionstastenbelegung mit dem Forth-Editor zu editieren.

-) Statusanzeigen
Man kann Statusanzeigen (siehe Vierte Dimension Vol II, No.1) einführen, die anzeigen, ob ESC gedrückt wurde, ob ein Upload erfolgt, ob XOFF empfangen wurde, ob Fehler auftraten usw...

-) Modusauswahl
Man kann eine schöne Syntax für die Modusauswahl basteln, etwa von der Form :

```
2 stopbit noparity 300 baud
```

oder ähnliches. Diese Funktionen sind natürlich Rechnerabhängig.

-) Drucker
Nützlich wäre es, den Drucker zum Protokollieren ein- und ausschalten zu können. Am besten steckt man die Zeichen, die ausgedruckt werden sollen, in eine Queue (mindestens 128 Zeichen lang), weil viele Drucker, wenn sie eine Zeile drucken, nicht schon die nächste empfangen können. Die Details, wann der Drucker zu testen ist und was man tut, wenn die Queue voll oder leer ist, seien dem Leser überlassen...

-) Multitasking
Viele Klimmzüge, die in diesem Programm erforderlich wurden, damit kein Zeichen verlorengeht, lassen sich vermeiden, wenn man den Multitasker benutzt. Man würde dann je einen Prozeß zum Empfangen, Senden, Drucken, Uploaden und für die Tastaturbedienung verwenden. Auch ein Echo-Modus wäre dann sehr einfach aufzubauen.

Viele der genannten Erweiterungen sind natürlich rechnerabhängig, aber für den Atari würde ich gerne solche Dinge sehen.
Ich hoffe, mit diesem Programm exemplarisch gezeigt zu haben, wie man in Forth Strukturen aufbaut, aus denen Programmteile und daraus wieder Programme zusammengesetzt werden können. Viele der genannten Änderungen sind mit den vorgestellten Bauteilen einfach zu programmieren (andere auch nicht...).

6.) Literaturhinweise

In dieser (oder der folgenden) Ausgabe der Vierten Dimension erscheint ein Artikel über die Benutzung des Trees. Seine Hilfstexte sind aber so umfangreich und gut gemacht, das es keine großen Probleme bei der Benutzung geben sollte.

Typischerweise wird man einen Akkustikkoppler an der RS232-Schnittstelle des Rechners benutzen. Deren Funktionsweise wird in Büchern über Microcomputer Interface Techniken und Artikeln in verschiedenen Zeitschriften beschrieben. Man kann natürlich auch auf die Handbücher der Hersteller zurückgreifen.
Die Programmierung der RS232-Schnittstelle des Atari wird im Entwicklungspaket, aber auch in Büchern verschiedener Firmen beschrieben (z.B. "Atari ST intern", Düsseldorf, 1985).

Das Volksforth für Atari ST- und C64-Computer kann vom Verfasser bezogen werden (siehe Vierte Dimension Vol.II No.1).

7.) Hinweise zum Listing

Im folgenden erkläre ich noch kurz in Form eines Glossars die weniger gebräuchliche Worte, die im Quelltext vorkommen.

```
\          ( -- )                  " skip-line "
```
Ignoriere den auf dieses Wort folgenden Text bis zum Ende der Zeile. Man kann dieses Wort durch die Kommentarklammern ersetzen. Im volksFORTH83 wie folgt definiert:
```
: \     >in @ c/l /  1+ c/l *  >in ! ;  immediate
```

```
\\         ( -- )                  " skip-screen "
```
Ignoriere den auf dieses Wort folgenden Text bis zum Ende des Blockes. Man kann dieses Wort durch die Kommentarklammern ersetzen. Im volksFORTH83 wie folgt definiert :
```
: \\    b/blk >in ! ;  immediate
```

```
?exit      ( f --)                 " question-exit "
```
Führt EXIT aus, falls das Flag f wahr ist. Ist das Flag falsch, so geschieht nichts. Man kann dieses Wort durch die folgende Sequenz ersetzen :
```
IF  exit  THEN
```
Eine mögliche Definition ist :
```
: ?exit   ( f --)   IF  r> drop  THEN ;
```

```
NUMBER     ( adr -- d)
```
Wandelt den counted String (eine Zeichenkette, in deren erstem Byte ihre Länge steht) bei der Adresse adr in die doppelt genaue Zahl d um.

?DO (n1 n2 --) " question-do"
Wird wie DO benutzt. Beginnt eine Schleife. Der Unterschied zu DO besteht darin, daß, falls n1 gleich n2 ist, der Schleifenrumpf gar nicht durchlaufen wird. In diesem Fall würde bei Verwendung von DO die Schleife 65536-mal durchlaufen. Die Sequenz:

```
?DO ... LOOP
```

kann ersetzt werden durch:

```
2dup - IF DO ... LOOP ELSE 2drop THEN
```

CAPITAL (c1 -- c2)
Das Zeichen c1 wird, sofern es ein Kleinbuchstabe ist, in den Großbuchstaben c2 umgewandelt. Ist c1 kein Kleinbuchstabe, so ist c2 gleich c1. Eine Definition, die auf allen Rechnern mit ASCII-Zeichensatz funktioniert, ist im Listing angegeben.

KEY? (-- f) " key-question"
Das Flag f ist wahr, falls eine Taste gedrückt wurde, sonst ist das Flag falsch. Die Definition dieses Wortes ist systemabhängig

KEY (-- c)
Die Definition von KEY entspricht, mit einer Einschränkung, dem Standard. Ist KEY? wahr, so darf KEY nicht länger als ca. 10 msec Ausführungszeit beanspruchen. Insbesondere darf KEY nicht warten, bis die gedrückte Taste wieder losgelassen wird.

Die folgenden Worte sind die "Primitives" des Terminalprogramms:

RX? (-- f)
Das Flag f ist wahr, falls ein Zeichen durch die serielle Schnittstelle empfangen wurde. Andernfalls ist das Flag falsch.

RX (-- c)
Das Zeichen c wurde von der seriellen Schnittstelle empfangen.

TX? (-- f)
Das Flag ist wahr, falls ein Zeichen von der seriellen Schnittstelle übertragen werden kann. Kann kein Zeichen übertragen werden, so ist das Flag falsch.

TX (c--)
Das zeichen c wird von der seriellen Schnittstelle abgeschickt. Dieses Wort darf nicht länger als ca. 10 msec zur Ausführung benutzen, sofern TX? vor dem Aufruf von TX wahr lieferte.

MSEC (n --)
Dieses Wort benötigt ca. n Millisekunden Ausführungszeit und tut sonst nichts.

Die übrigen Worte werden, soweit sie nicht im Forth-83-Standard enthalten sind, auf Screen 1 des Listings definiert.
Für ein Terminalprogramm ohne Fileinterface sind die Screens 5,6 und E überflüssig, für ein Programm mit Fileinterface die Screens 7 und D. Die Screens 11,12 und 13 sind systemabhängig und müssen auf anderen Rechnern angepasst werden.

0

```
0 ######  How to use  TERMINAL  #########          bp 9Jun86
1
2
3 Benutze nach dem Laden :
4                  <n>  start / start <filename>
5                  <n>  nulls !
6                  lf?  on/off
7
8 TERMINAL
9   verarbeitet <esc> + <esc>   : send einmal ESC
A                   +   u      : Schalte UPLOAD um
B                   +   d      : starte Download
C                   +   x      : beende das Programm
D
E
F
```

1

```
0 \ Terminal loadscreen                          bp 13Jun86
1
2 \ : case?   ( n1 n2 -- n1 ff \ tf )
3 \       over  =  dup  IF  swap drop  THEN  ;
4 \ -1 Constant  true
5 \  0 Constant  false
6 \ : on      ( adr --)                  true  swap ! ;
7 \ : off     ( adr --)                  false swap ! ;
8 \ : bounds  ( start len -- end+1 start)  over + swap ;
9 \ : capital ( c1 -- c2 )       \ converts a key to upper case
A \           dup  Ascii a <   over Ascii z > or  ?exit
B \           [ Ascii z  Ascii Z - ] Literal - ;
C \     key?  key  msec   rx  rx?   tx  tx?
D
E $11 $12 +thru     \ System dependent stuff
F   1 $10 +thru
```

2

```
0 \ nice constants                               bp 9Jun86
1
2 : Ctrl    ( -- )          \ define control codes for ASCII
3    bl word  1+ c@ $1F and ;
4
5 Ctrl S  Constant #xoff      \ # means "number"
6 Ctrl Q  Constant #xon
7 Ctrl M  Constant #cr
8 Ctrl J  Constant #lf
9   $1B   Constant #esc
A Ctrl H  Constant #bs
B   $7F   Constant #del
C
D
E
F
```

14

```
bp 9Jun86 \\ comments for volksforth                bp 9Jun86
```

Die XON/XOFF-Geschichte ist auf dem Atari überflüssig, ich habe davon in +UPLOAD auch Gebrauch gemacht.

Dennoch habe ich sie eingebaut, um zu zeigen, was man tun muß, wenn man nur eine ACIA hat.

C64-Benutzer sollten nicht vergessen, daß ihr "Kernal" zwar auch einen Interruptmechanismus für den Empfang hat, ihn aber nicht während eines Diskzugriffes bedient (glaub ich ...)

15

```
bp 13Jun86 Einige Definitionen, die nützlich sind  bp 13Jun86
```

CASE? ist das einfachste CASE, daß ich kenne !

CAPITAL (c1 -- c2)
wandelt einen Kleinbuchstaben c1 in einen Großbuchstaben c2 um.

key? (-- f) f ist wahr, falls eine Taste gedrückt wurde.
key (-- c) c ist die gedrückte Taste, key darf nicht warten bis die Taste wieder losgelassen wurde !
msec (n --) wartet einfach n Millisekunden
Wird mit einer Stoppuhr und DO .. LOOP gebaut.
rx? rx tx? tx siehe weiter hinten ...

16

```
bp 10Jun86
```

CTRL liest ein Zeichen ein, analog zu ASCII
Wie die Bits maskiert werden müssen, hängt vom Zeichensatz ab. Diese Definition sollte aber meist funktionieren.

Anschließend werden einige benötigte Konstanten definiert, viele sind in Forthsystemen vorhanden.

Page No volksFORTH83 der FORTH-Gesellschaft eV TERMINAL.SCR

3

```
0 \ Create Queue for storing characters                bp 16Jun86
1
2 Variable queue                \ this Variable points to Queue
3 : Createqueue   ( len -- )
4    Create  here 6 + dup , dup , over + 1- ,  allot
5    Does>   queue ! ;
6
7 : next          ( adr -- adr' )   \ adr is pointer in data field
8    1+  queue @ 4 + @  over u<  IF drop queue @ 6 + THEN ;
9 : advance       ( adr-- ) dup @ next  swap ! ;
A
B : qempty?       ( -- f)   queue @  dup  @ next  swap  2+ @  = ;
C : qfull?        ( -- f)   queue @  dup  @       swap  2+ @  = ;
D : qc@           ( -- c)   queue @ 2+  dup @ c@  swap advance ;
E : qc!           ( c --)   queue @   under @ c!       advance ;
F : qreset        ( --)     queue @  dup 2+ @ swap ! ;
```

17

```
bp 16Jun86 Auf diesem Screen wird eine Queue-Struktur defini   bp 16Jun86

QUEUE ist eine Variable, die auf die aktuelle Queue zeigt
   In Multitasking-Systemen sollte das eine Uservariable sein.
CREATEQUEUE erzeugt eine Queue der Länge len. Wird der Name
   der neuen Queue genannt, macht sie sich zu aktuellen Queue.

NEXT      adr ist die Adresse eines Bytes in der Queue. Adr' ist
   die Adresse des folgenden Bytes
ADVANCE   Der Zeiger adr wird ein Byte weitergestellt.

QEMPTY?   f ist wahr, falls die Queue leer ist.
QFULL?    f ist wahr, falls die Queue voll ist.
QC@       das Zeichen c wird aus der Queue gelesen...
QC!       das Zeichen c wird in die Queue geschrieben...
QRESET    der Inhalt der Queue wird gelöscht
```

4

```
0 \ read and write on FIFO                             bp 3jun86
1
2 $40  Createqueue receiverbuffer       \ declare FIFO
3 receiverbuffer
4
5 : readbuffer    ( -- c)         \ read char from buffer
6       receiverbuffer
7       qempty? abort" buffer empty"    qc@ ;
8
9 : writebuffer   ( c --)         \ write char to buffer
A       receiverbuffer
B       qfull? abort" buffer full"      qc! ;
C
D
E
F
```

18

```
bp 9Jun86

Diese Queue wird benutzt, um empfangene Zeichen während eines
Diskzugriffs zwischenzuspeichern.

READBUFFER liest ein Zeichen aus der obigen Queue, stoppt aber,
  falls kein Zeichen mehr da ist.

WRITEBBUFFER schreibt ein Zeichen in die obige Queue, stoppt
  aber, falls die Queue keinen Platz mehr bietet.
```

5

```
0 \\ managing upload onto file blocks                  bp 13Jun86
1
2 \ if you have a file interface  ( volksforth rev. no 3.71 ) ,
3 \ use this screens
4
5 Variable upload           \ true if uploading, else false
6
7 : upload?       ( -- f)  upload @ ;
8
9 file upfile
A
B : start         ( --)     \ use BEFOR terminal, eats a name !
C     0 scr !   r# off
D     upfile make           \ assigns a empty file to upfile
E     1 more ;
F
```

19

```
bp 10Jun86 Dieser Screen ist nur für das Volksforth
mit Fileinterface

Hier ist nur START anders. Es erzeugt mit MAKE
 ein File, dessen Name in den File Control Block UPFILE
 eingetragen wird.
 FILE ist ein definierendes Wort, daß FCB's erzeugt.
 1 MORE verlängert schließlich das leere (neu erzeugte) File
 um einen Screen, so daß für den Anfang etwas Platz vorhanden
 ist.
```

6

```
0 \\                                                    bp 13Jun86
1
2 \ The word  MORE  eats up much time, so in principle it must
3 \ use the  XON ... XOFF  mechanism. We have ignored that
4 \ and use the TOS receiver buffer which is present on every
5 \ volksFORTH83 with fileinterface .... ( 'till now ...)
6
7 : +upblock   ( n -- )        \ add n to #upblock
8    scr @ +   dup  scr !
9    upfile capacity  <  ?exit    \ exists block already ?
A    capacity  scr @ -  1+    more ;
B
C : upblock       ( n -- adr )
D    upfile scr @  block ;
E
F
```

1A

bp 10Jun86

+UPBLOCK schaltet von einem Block auf den nächsten um (wenn n gleich eins ist) . Bei Files ist dieser Block normalerweise nicht vorhanden, da das File ja neu erzeugt wurde. Daher müssen Blöcke mit MORE an das File angehängt werden. Eigentlich hätte ich schlicht 1 MORE programmieren können, da das File sowieso voll sein sollte, aber dieses Wort ist universeller und auch außerhalb des Terminals einsetzbar (defensive Programmierung) MORE sollte eigentlich mit XON/XOFF abgesichert werden, auf dem Atari ist es aber nicht nötig.

UPBLOCK n ist die Nummer des Blocks im up-File, adr dessen Adresse.

7

```
0 \  managing upload onto direct blocks                bp 13Jun86
1
2 Variable upload                \ true if uploading
3
4 : upload?       ( -- f)  upload @ ;
5
6 : start         ( n --)        \ use BEFOR terminal
7     scr !   r# off ;
8
9 upload off     1 start
A
B : +upblock    ( n --)          \ increment #upblock by n
C     scr +! ;
D
E : upblock       (   --addr )   \ addr is adr. of the "upblock"
F     scr @ block ;
```

1B

bp 9Jun86

UPLOAD	diese Variable ist gesetzt, falls ge-uploaded wird.
UPLOAD?	f ist wahr, falls ge-uploaded wird.
START	n ist der erste Screen, auf den ein Upload erfolgt. Als Zeiger während des Uploads werden die Variablen SCR und R# verwendet. Geht man nach TERMINAL in den Editor, dann sollte man sich an der Upload-Stelle befinden.
+UPBLOCK	springt während des Uploads n Blöcke weiter.
UPBLOCK	n ist die Nummer des Blocks im up-File, adr dessen Adresse.

8

```
0 \ store chars during upload                          bp 3jun86
1
2 : +upblockptr   ( n --)        \ increment r# by n
3   r# @ +     b/blk /mod   +upblock   r# ! ;
4
5 : upemit        ( c --)        \ "emit" char to upblock
6   upblock   r# @ + c!   update    1 +upblockptr ;
7
8 : updel         ( --)          \ delete last char from upblock
9   -1 +upblockptr   bl upemit  -1 +upblockptr ;
A
B : upcr          ( --)          \ do a "carriage return"
C   c/l  r# @  over mod -  ?dup 0= ?exit
D   upblock   r# @ +  over blank   update
E   +upblockptr ;
F
```

1C

bp 9Jun86

+UPBLOCKPTR	schaltet die Zeiger SCR und R#, die die aktuelle Upload-Position angeben, um n Zeichen weiter.
UPEMIT	speichert das Zeichen c und gehe ein Zeichen weiter
UPDEL	löscht das zuletzt mit UPEMIT gespeicherte Zeichen
UPCR	geht in die nächste Zeile eines Screens über, wobei der Rest der alten Zeile gelöscht wird.

9

```
0 \ receive and decode characters                    bp 3jun86
1
2 Variable wait?   wait? off      \ transmit disabl-
3
4 : rdecode       ( c --)         \ decode and display characters
5    \ #### enter code conversion for C64 or DIN here #####
6   #xon    case?  IF  wait? off                    exit  THEN
7   #xoff   case?  IF  wait? on                     exit  THEN
8   #cr     case?  IF  cr  upload?   IF  upcr THEN  exit  THEN
9   #lf     case?  IF                               exit  THEN
A   #bs     case?  IF  del upload?   IF  updel THEN exit  THEN
B   #del    case?  IF  del upload?   IF  updel THEN exit  THEN
C    upload?  IF  dup upemit  THEN   emit  ;
D
E : receive       ( --)           \ receive and decode chars
F   rx?  IF  rx  rdecode  THEN ;
```

A

```
0 \ transmit chars                                   bp 13Jun86
1
2 : wait          ( c -- c)       \ wait until  xon
3   receive    dup #xon =   over #xoff =  or  ?exit
4   BEGIN  receive  wait? @  0=  UNTIL ;
5
6 : send          ( c --)         \ send a character
7   BEGIN  wait  tx?  UNTIL  tx ;
8
9 Variable lf?       lf? off      \ send a line feed after cr ?
A Variable nulls     0 nulls !    \ send nulls after cr/lf ?
B
C : sendcr        ( -- )          \ send a cr
D   #cr send   lf? @  IF  #lf send  THEN
E   nulls @  0  ?DO  0 send  LOOP ;
F
```

B

```
0 \ steal enough time ...                            bp 13Jun86
1
2 : xoff          ( -- )       \ stop other system in safe way...
3   #xoff send
4   &10 BEGIN  rx?  IF  rx  writebuffer  drop  &10
5                       #xoff send   THEN
6              25 msec    1-  ?dup 0= UNTIL ;
7
8 : xon           ( -- )        \ ...  and restart it
9   BEGIN  qempty? 0=  WHILE  readbuffer rdecode  REPEAT
A   #xon dup  send  send ;
B
C
D
E
F
```

1D

bp 9Jun86

WAIT? ist gesetzt, falls ein XOFF empfangen wurde.

RDECODE überprüft, was mit dem empfangenen Zeichen geschehen soll und verarbeitet es entsprechend.

RECEIVE Das zentrale Wort des Empfangs; liest, falls nötig, ein Zeichen ein und verarbeitet es.

1E

bp 13Jun86

WAIT wartet, falls ein XOFF empfangen wurde, bis wieder gesendet werden darf. XOFF und XON werden jedoch immer übertragen, damit sich die Rechner nicht gegenseitig blockieren.

SEND schickt ein Zeichen an die serielle Schnittstelle, wobei ein evtl. empfangenes XOFF beachtet wird.

LF? ist gesetzt, falls auf ein CR ein LF folgt.

NULLS enthält die Zahl der Nullen nach einem CR

SENDCR sendet ein CR mit/ohne LF und Nullen.

1F

bp 9Jun86

XOFF dient dazu, unseren Rechner von seiner Lauscherpflicht zu entbinden. Der Gegenstation wird ein XOFF geschickt und gewartet, bis sich nichts mehr rührt.

XON verarbeitet ewaige oben angefallene Daten und schickt der Gegenstation dann ein XON.

11

Page No 5 volksFORTH83 der FORTH-Gesellschaft eV TERMINAL.SCR

C

20

```
0 \ transmit a string                                   bp 3jun86
1
2 : sendstring    ( adr len -- )  \ send a string
3   -trailing   1 max
4   bounds  DO   I c@ send   LOOP  sendcr ;
5
6 b/blk  c/l  / Constant l/s
7
8
9
A
B
C
D
E
F
```

bp 9Jun86

SENDSTRING sendet eine Zeichenkette an die Gegenstation.

D

21

```
0 \ download text , for direct access                   bp 13Jun86
1
2 : sendline      ( blk n --)     \ send a line on a block
3   c/l *  swap block +   pad  c/l  cmove   pad c/l sendstring ;
4
5 : sendblock     ( n --)         \ send a block in single lines
6   l/s 0  DO  dup I sendline  LOOP  drop ;
7
8 : getnumber     ( -- n)         \ input number from keyboard
9   query  bl word  number  drop  ;
A
B  : download      ( -- )          \ send a block range
C xoff    cr ." Input start block  "   getnumber
D         cr ." Input no. of blocks "  getnumber   xon
E over + swap ?DO  I dup 4 .r sendblock  LOOP cr ;
F
```

bp 9Jun86

SENDLINE sendet die Zeile n des Screens blk an die Gegenstation. Der Text wird zuerst nach PAD verlagert, da er sich auf einem Screen befinden kann. Dieser Screen kann während des Sendens durch das Uploaden wieder verschwinden.

SENDBLOCK sendet den Screen n aus.

GETNUMBER liest Zahl vom der Tastatur ein.

DOWNLOAD fragt nach dem zu sendenden Bereich von Screens und schickt ihn weg. Der Screen, der gerade unterwegs ist, wird angezeigt.

(Debuged in letzter Sekunde,

E

22

```
0 \\ download text , for file access                    bp 13Jun86
1 file downfile
2 : sendline      ( blk n --)     \ send a line on a block
3   c/l *  swap  downfile block  +   pad  c/l  cmove
4   pad c/l sendstring ;
5
6 : sendblock     ( n --)         \ send a block in single lines
7   l/s 0  DO  dup I sendline  LOOP  drop ;
8
9 : getfile       ( -- n)         \ input filename from keyboard
A   cr ." File name for download :  " query   cr
B   downfile assign ;
C
D : download      ( -- )          \ send a file
E    xoff   getfile  xon
F    capacity 0  DO   I   dup 4 .r   sendblock  LOOP cr ;
```

bp 10Jun86

Beim Fileinterface gibt es einen FCB, der das zu sendende File enthält.

SENDLINE nimmt darauf Rücksicht, daß es mehrere Files gibt und einen zweiten Prozeß, der ein anderes File benutzt.

SENDBLOCK wie gehabt.

GETFILE liest einen Filenamen von der Tastatur ein, der mit ASSIGN an das DOWNFILE zugewiesen wird.

DOWNLOAD fragt nach dem zu sendenden File und schickt es weg.

12

Page No 6 volksFORTH83 der FORTH-Gesellschaft eV TERMINAL.SCR

F 23

```
0 \ input keys                                      bp 9jun86
1
2 : key          ( -- c )       \ get a key
3   BEGIN  receive  key?  UNTIL  key ;
4
5 : esc-keys     ( f -- f )     \ special action after esc
6     key  capital
7     #esc  case?  IF  #esc send                    exit  THEN
8   Ascii U  case?  IF  upload?   0= dup  upload  !
9                       ." UPLOAD 0"
A                       IF ." N"  ELSE ." FF" THEN  exit THEN
B   Ascii D  case?  IF  download                     exit  THEN
C   Ascii X  case?  IF  drop true                   exit  THEN
D           drop ;
E
F
```

KEY wird so redefiniert, daß, während es auf eine Taste wartet, gleichzeitig empfangen wird.

ESC-KEYS interpretiert die Kommandokeys. f ist ein Flag, daß, falls es gesetzt wird, dazu führt, das TERMINAL beendet wird.

10 24

```
0 \ decode keys from keyboard                       bp 10Jun86
1
2 : sdecode      ( f c -- f )  \ decode keys, terminate if true
3   #cr  case?  IF  sendcr          exit  THEN
4   #esc case?  IF  esc-keys        exit  THEN
5                           \  ### add function keys here ###
6   send  ;
7
8 : terminal     ( -- )        \ main word
9   BEGIN  false  key sdecode  UNTIL ;
A
B
C
D
E
F
```

bp 9Jun86

SDECODE interpretiert die Taste c , schaltet bei ESC in den Kommandomodus um und kann auch Funktionstasten verarbeiten. Für f gilt dasselbe wie bei ESC-KEYS Normalerweise wird die Taste einfach losgeschickt.

TERMINAL wartet auf eine Taste und interpretiert sie. Sonst tut TERMINAL nichts...

11 25

```
0 \ patch r/w for save receiving                    bp 13Jun86
1
2 \ R/W costs much time, so change your R/W in the foll. manner
3     \ : saver/w       ( sys -- sys )
4     \      xoff  oldr/w  xon ;
5     \ patch  saver/w  for  oldr/w
6
7 \ for volks/ultraforth :
8     ' r/w >body @  Alias oldr/w
9
A     : saver/w       ( adr blk file f -- f')
B       xoff    oldr/w   xon ;
C
D     ' saver/w Is r/w
E
F
```

SYSTEMABHÄNGIG bp 10Jun86

Hier wird R/W so umgebaut, daß es, falls es aufgerufen wird, erstmal XOFF macht. Dann macht es seine Arbeit. Hinterher gibts noch ein XON.

Ein Beispiel für das ultra/volksFORTH83..

13

Page No 7 volksFORTH83 der FORTH-Gesellschaft eV TERMINAL.SCR

12 26

```
0 \ primitiv modem words for volksFORTH on the Atari    bp 31May86 SYSTEMABHÄNGIG                    bp 9Jun86
1
2 1 Constant rs232
3
4 : rx?          ( -- f )  rs232 bconstat ;                RX?        f ist wahr, falls ein Zeichen abgeholt werden muß
5 : rx           ( -- c )  rs232 bconin ;                  RX         c ist ein empfangenes Zeichen
6
7 : tx?          ( -- f )  rs232 bcostat ;                 TX?        f ist wahr, falls ein Zeichen gesendet werden kann
8 : tx           ( c -- )  rs232 bconout ;                 TX         c ist das zu sendende Zeichen
9
A : msec         ( n -- )  &110 * 0 ?DO LOOP ;             MSEC       das volksforth schafft mehr als 0.1 Megaloops/sec.!
B
C : key          ( -- c )  key $FF and ;                   KEY        unser Key liefert mehr als nur ein Zeichen...
D
E
F
```

13 27

```
0 \ configure rs 232                                bp 1jun86 SYSTEMABHÄNGIG                    bp 9Jun86
1
2 Code rsconf  ( baud ctrl ucr -- )                            RSCONF        nimmt obskure Parameter, um die Schnittstelle
3    .w  -1 # D2 move                                             auf eine normale Einstellung zu bringen.
4    D2 A7 -) move    D2 A7 -) move   D2 A7 -) move               Definition der Zahlen : einschlägige "Literatur"
5    SP )+ A7 -) move            \ ucr
6    SP )+ A7 -) move            \ ctrl
7    SP )+ A7 -) move            \ baud
8     $F # A7 -) move   $E trap    $E # A7 adda   Next  end-code
9
A 9   1  %10001000   rsconf
B   \ 8 Bit, no parity, one stopbit, Xon/Xoff-mode, 300 Baud
C
D
E
F
```

0 0

```
0 ######   How to use   TERMINAL  #########          bp 9Jun86 ######   How to use - TERMINAL  #########          bp 9Jun86
1
2
3 Benutze nach dem Laden :                                     Benutze nach dem Laden :
4                  <n>  start / start <filename>                                <n>  start / start <filename>
5                  <n>  nulls                                                   <n>  nulls
6                  lf?  on/off                                                  lf?  on/off
7
8 TERMINAL                                                     TERMINAL
9   verarbeitet <esc> + <esc>   : send einmal ESC                verarbeitet <esc> + <esc>   : send einmal ESC
A                  +   u     : Schalte UPLOAD um                               +   u     : Schalte UPLOAD um
B                  +   d     : starte Download                                 +   d     : starte Download
C                  +   x     : beende das Programm                             +   x     : beende das Programm
D
E
F
```

Teaching Forth:

Let's Keep It Simple

Ronald E. Apra
San Jose, California

For as long as I have taught beginning programming to elementary and secondary students, the IF THEN construct has been a source of confusion and frustration for beginning students. The problem is further compounded when you introduce the ELSE statement along with IF THEN. Students who have taken geometry before taking programming seem to handle this construct much better due to their experience with IF THEN statements in formal proofs. However, if geometry is made a prerequisite for programming, you limit the number of students who can take programming.

I feel that the source of this confusion lies partly in the syntax of the IF THEN ELSE phrase (BASIC, Logo and Pascal) or the more mind-boggling IF ELSE THEN expression used in Forth. Apple Logo and IBM Logo offer two interesting ways to deal with the problem. In the first method, the operation IF is followed by a test that produces a true or false flag followed by one or two instruction lists. If the flag is true the first instruction is executed, and if the flag is false the second instruction is run. This method eliminates the words THEN and ELSE. In the following Logo procedure,

```
TO DECIDE
OUTPUT IF 0 = RANDOM 2 ["YES]
  ["NO]
END
```

the 0 = RANDOM 2 after the IF produces a "TRUE flag if RANDOM 2 is 0 and the procedure will OUTPUT the message YES. If 0 = RANDOM 2 is "FALSE, then the procedure will run the second list, which is equivalent to the ELSE phase of a conditional. In the second method, the Logo commands TEST, IFTRUE and IFFALSE can be used to write the procedure DECIDE in the following way:

```
TO DECIDE
TEST 0 = RANDOM 2
IFTRUE [OUTPUT "YES]
IFFALSE [OUTPUT "NO]
END
```

where a very readable procedure is produced. TEST yields a "TRUE or "FALSE flag for IFTRUE and IFFALSE.

Since some of my students do program in Forth, it would be a good exercise for them to see if they could come up with a construct in Forth that would work similarly to the above Logo procedure. I am not at this time proposing that the standard IF ELSE THEN construct in Forth be changed, but I challenge the Forth community to see what they can come up with.

In playing around with this problem in Forth, I have written the four simple words **TESTIT, IFTRUE, IFFALSE** and **ENDIT**. **TESTIT** is defined simply as ,

```
: TESTIT  ( n -- n n ) DUP ;
```

and can be used to duplicate a flag or some number on the stack that is about to be tested. In **EXAMPLE1** on screen 95, **TESTIT** is duplicating the flag produced by **0=** and in **EXAMPLE4** on screen 96, **TESTIT** is duplicating the number on the stack that is about to be tested by **1 =**. **IFTRUE** is a new name for **IF**, and **ENDIT** is a new name for **THEN** (see *Forth Dimensions* VI/1, page 26, for a different version of **ENDIT**). I defined **IFTRUE** and **IFFALSE** as follows:

```
: IFTRUE ( flag -- ) [COMPILE] IF ; IMMEDIATE
: IFFALSE ( flag -- ) [COMPILE] THEN ; IMMEDIATE
```

On screen 95, **EXAMPLE1** resolves an IF ELSE THEN condition with a **TESTIT IFTRUE IFFALSE** structure. By the nature of the syntax, the student can point out the **0=** test of n and knows, if the flag is true, where the true condition will be executed. For beginning students, the IF ELSE THEN syntax does not leave enough clues to where the parts of the conditional should go. When a student gets some practice with the **TESTIT IFTRUE IFFALSE** construct, he can better understand the **IFTRUE ELSE ENDIT** or **IFFALSE ELSE ENDIT** structure in **EXAMPLE2** and **EXAMPLE3** of screen 95.

On screen 96, **EXAMPLE4** can produce some interesting results where **TESTIT** is duplicating the input to be tested by **1 =**. See if you can explain why **3 EXAMPLE4** outputs "twothreefour" and then create your own crazy **EXAMPLE**. I bet there are some interesting things that can be done with **TESTIT** and multiple **IFTRUE** and **IFFALSE** statements. In **EXAMPLE5**, the words **TESTIT, IFTRUE** and **ENDIT** seem to improve the readability of the nesting, but I try to encourage students to avoid nesting if at all possible.

I stress "keep it simple" in my programming philosophy, but most beginning students are overwhelmed by the articles that appear in *Forth Dimensions*. For example, the "Techniques Tutorial" department seemed to be a showcase for the skills of some truly great programmers, but it was over the heads of many beginners. I hope this article will stimulate more thought along the lines of "keeping it simple."

```
SCR #95
  0 ( TESTIT, IFTRUE, IFFALSE, ENDIT                    ra/Oct/1985 )
  1
  2 : TESTIT ( n --  n n ) DUP ;
  3 : ENDIT [COMPILE] THEN ; IMMEDIATE
  4 : IFTRUE ( flag -- ) [COMPILE] IF ; IMMEDIATE
  5 : IFFALSE ( flag -- ) COMPILE NOT [COMPILE] IF ; IMMEDIATE
  6
  7 : EXAMPLE1 ( n -- )
  8     0= TESTIT
  9           IFTRUE ." true" ENDIT
 10           IFFALSE ." false" ENDIT ;
 11
 12 : EXAMPLE2 ( n -- ) 0= IFTRUE ." true" ELSE ." false" ENDIT ;
 13
 14 : EXAMPLE3 ( n -- ) 0= IFFALSE ." false" ELSE ." true" ENDIT ;
SCR #96
  0 ( TESTIT, IFTRUE, IFALSE, ENDIT        ra/0ct/85 )
  1
  2 : EXAMPLE4 ( n -- )
  3       TESTIT 1 = IFTRUE ." one" ENDIT
  4       TESTIT 2 = IFFALSE ." two" ENDIT
  5       TESTIT 3 = IFTRUE ." three" ENDIT
  6       TESTIT 4 = IFFALSE ." four" ENDIT
  7               DROP ;
  8
  9 : EXAMPLE5 ( n -- )
 10       TESTIT 1 = IFTRUE ." one" ELSE
 11       TESTIT 2 = IFTRUE ." two" ELSE
 12       TESTIT 3 = IFTRUE ." three" ELSE
 13       TESTIT 4 = IFTRUE ." four" ELSE
 14         ENDIT ENDIT ENDIT ENDIT DROP  ;
OK
1 EXAMPLE4 onetwofour
2 EXAMPLE4 four
3 EXAMPLE4 twothreefour
4 EXAMPLE4 two
```

(Aus: Forth Dimensions, Volume VII, No.6, S.21)

FORTH GRUPPEN

Hamburg
  Treffen jeden vierten Samstag im Monat ab 16:00Uhr in der Berufsfachschule für Radio- und Fernsehtechnik, Eimsbüttelerstr.64-66.
  Kontakt: Bernd Pennemann, 040-6900539
Karsruhe
  Treffen jeden dritten Mittwoch im Monat ab 19:00Uhr im Jugend- und Begegnungszentrum, Krohnenplatz.
  Kontakt: Michael Weiss, 0721-854994
Paderborn
  Treffen in der Gruningerstr.20 nach Verabredung.
  Kontakt: Thomas Asche, 05251-26496
Wuppertal
  Treffen jeden vierten Samstag in Monat ab 20:00Uhr im Bahnhof Ottenbruch, Funckstrasse, W tal-Elberfeld.
  Kontakt: Michael Kalus, Präsidentenstr.40, 5830 Schwelm, 02336-82204

Amateurfunker & Forth
  Kontakt: Bernd Zimmermann, 04105-52068

Darmstadt (*)
  Kursus in der Volkshochschule. Kontakt: Andreas Soeder, 06257-2744
Berlin (*)
  Kontakt: Hans Madlung, 030-4141831
Münchner Raum (*)
  Kontakt-1: Heinz Schnitter, privat 089-3103385, Labor 089-32095276.
  Kontakt-2: Jürgen Helbig und Robert Schörghuber, Waldvögeleinstr.6a, 8000 München 50, 089-2283531
  Kontakt-3: Ekkehard Flögel, 08021-8414
Braunschweiger Raum (*)
  Kontakt: Eckhard Heyne, 05352-58087
Moers (*)
  Kontakt: Hans Chrapla, 0203-3793274

Villach östereich (*)
  Forth-Club in Kärnten. Treffen nach Verabredung. Kontakt: Heinz Klambauer, 04242-33566

Belgien FIG Chapter
  Kontakt: Luk van Loock, Lariksdreef 20, 2120 Schoten, Telefon 03-658-6343

Holland FIG Chapter
  Kontakt: Adriaan van Roosmalen, Heusden Houtsestraat 134, 4817 We Breda, Telefon 31-76-713104

Schweiz FIG Chapter
  Kontakt: Max Hugelshofer, ERNI & Co., Elektro-Industrie, Stationsstrasse, 8306 Bruttisellen, Telefon 01-833-3333

(*) = Forth-Enthusiasten oder Gruppen, die noch nicht offiziell als Gruppe der FG anerkannt sind bzw. zu dem Zweck noch Verbindungen zu weiteren Forthlern suchen. Wenn Sie interessiert sind, kann hier auch ihr Name stehen. Schreiben Sie an die VIERTE DIMENSION.