Wilhelm-Pieck-Universitaet Rostock

Sektion TE/AS

popFORTH

Bearbeiter: Ralf Neuthe
Stand: 30.05.86

. Konzeption

popFORTH ist ein Abkoemmling von figFORTH mit wesentlichen Veraenderungen in der Massenspeicherarbeit. Diese Veraenderungen zielten darauf hinaus, dass das FORTH-System sich selbst an die unterschiedlichen Diskettenformate anpasst. Ein zweiter Unterschied besteht in der automatischen Anpassung an den vorhandenen Speicherraum, was waehrend des Kaltstartes passiert. Um die Arbeit mit dem Massenspeicher zu ermoeglichen, enthaelt popFORTH ausserdem einen einfachen Editor, dessen Beschreibung noch folgt.

2. Empfehlungen fuer die Belegung der Screens

Wie auch in figFORTH beginnt die Zaehlung der Screens mit der Nummer 0. Es ist bei der Benutzung der Screens zu beruecksichtigen, dass popFORTH die CP/M Dateiarbeit in keiner Weise unterstuetzt. Es wird gnadenlos auf jede Stelle der Diskette ohne Beachtung von schreibgeschuetzten Dateien, Systemspuren, Directorys usw. zugegriffen. Die Fehlermeldungen befinden sich auf Screen 4 und 5 und entsprechen der fig-Empfehlung:

```
Screen # 4
  0  ( Fehlermeldungen )
  1 Stack leer
  2 Dictionary voll
  3 falscher Adressmode
  4 gibt's schon
  5
  6 keine Disk (mehr) verfuegbar
  7 Stack voll
  8 Schreib- oder Lesefehler
  9
 10
 11
 12
 13
 14
 15 WPU Rostock/STE/AS                                  popFORTH
```

```
Screen # 5
  0
  1 nur fuer Compilation zulaessig
  2 Compilation verboten
  3 falsche Struktur
  4 Definition nicht beendet
  5 Wort im geschuetzten Bereich
  6 nur waehrend des Ladens erlaubt
  7 Kursor ausserhalb des Screens
  8 CONTEXT ungleich CURRENT
  9
 10
 11
 12
 13
 14
 15
```

Damit diese Fehlermeldungen im Klartext erscheinen muss die Uservariable 'WARNING' (initialisiert auf 0) auf 1 gesetzt werden.

3. Vertauschung von grossen und kleinen Buchstaben

Falls auf einem Buerocomputer oder einem anderen Geraet gearbeitet werden soll, das eine Tastatur besitzt, die in Grundstellung kleine Buchstaben liefert, ist ein Austauschen der grossen und kleinen Buchstaben sehr empfehlenswert. Folgendes Beispiel soll zeigen, wie das realisiert werden kann:

```
Screen # 12
  0  ( Austausch von grossen und kleinen Buchstaben )
  1  BASE @  HEX
  2 CREATE (KEY)
  3  ' KEY CFA ( from )  HERE 2 - ( to )  6  ( count )
  4      4 ALLOT   CMOVE  SMUDGE
  5
  6 : CONIN  ( ===> c ; Austausch der Buchstaben )
  7   (KEY)
  8   DUP    40 > OVER 5B < AND  IF  20 +        ELSE
  9   DUP    60 > OVER 7B < AND  IF  20 -  ENDIF ENDIF ;
 10  ' CONIN CFA  ' KEY  !
 11  BASE !
 12 ;S
 13
 14
 15
```

4. Erlaeuterung der Befehle des Editors

A) Screen-Editor

Listen von Sceenen:

LIST	(n -->)	Screen n wird gelistet
L	(-->)	listen des aktuellen Screens
NL	(-->)	listen und aktuellsetzen des folgenden Screens
PL	(-->)	listen und aktuellsetzen des vorhergehenden Screens

Loeschen von Screenen:

CLEAR	(n -->)	loeschen und aktuellsetzen von Screen n
CL	(-->)	loeschen des aktuellen Screens
CLEARS	(begin count -->)	loeschen von 'count' Screenen beginnend bei Screen 'begin'

Kopieren von Screenen:

COPY	(from to -->)	Screen 'to' wird mit Screen 'from' ueberschrieben
COPIES	(from to count -->)	analog COPY, aber mit 'count'-facher Ausfuehrung. Ueberlappung von Quell- und Zielbereich ist moeglich.

B) Line-Editor

Die Befehle des Line-Editors beziehen sich auf Operationen mit ganzen Zeilen des aktuellen Screens. Zwischenspeicher fur Zeilen ist der Speicherbereich PAD ...PAD+40H.

D	(n -->)	Vernichten (delete) von Zeile n, Nachruecken der folgenden Zeilen, Speicherung der vernichteten Zeile im PAD
DS	(n count -->)	analog D , aber 'count'-fache Ausfuehrung. Im PAD ist Zeile n+'count'-1
E	(n -->)	Loeschen (erase) von Zeile n, Ersatz durch Leerzeichen, keine Zwischenspeicherung
ES	(n count -->)	Analog E , aber 'count'-fache Ausfuehrung
H	(n -->)	Speicherung (hold) von Zeile n im PAD
I	(n -->)	Einschieben der im PAD befindlichen Zeile n. Zeile n und alle Folgenden werden nach unten verschoben. Zeile 15 geht verloren
P txt	(n -->)	Texteingabe (put) von der Tastatur auf Zeile n und den PAD
R	(n -->)	Ersetzen (replace) der Zeile n durch den Inhalt des PAD
S	(n -->)	Spreizen (spread), auf Zeile n wird eine Zeile mit Leerzeichen einge-

fuegt. Die darunter liegenden Zeilen
werden wie bei Kommando I verschoben

SS	(n count -->)	analog S , aber 'count'-fache Ausfuehrung
T	(n -->)	Anzeige (type) der Zeile n, Zwischenspeicherung im PAD (siehe auch String-Editor)

C) String-Editor

Die Befehle des String-Editors beziehen sich auf den gesamten aktuellen Screen. Die Suche nach Strings beginnt ab der aktuellen Kursorposition im Screen und laeuft bis zu seinem Ende. Die Kursorposition wird in der Variablen R# gehalten. Stringeingaben schliessen sich nach einem Leerzeichen nach dem Befehl an und werden durch CR begrenzt. Als Zwischenspeicher dient der PAD.

TOP	(-->)	Der Kursor wird auf Zeile 0 Spalte 0 des aktuellen Screens gesetzt
T	(n -->)	(type) Der Kursor wird auf Spalte 0 der Zeile n gesetzt (siehe auch Line-Editor)
M	(n -->)	(move) Der Kursor wird um n Zeichen verschoben (auch negativ)
F txt	(-->)	(find) Der Kursor wird hinter den gefundenen Text gesetzt
N	(-->)	(next) analog F , Verwendung des Strings im PAD
B	(-->)	(back) Ruecksetzen des Kursors um die Laenge des Strings im PAD
C txt	(-->)	Einfuegen des Textes ab der aktuellen Kursorposition. Der Kursor wird hinter txt positioniert. Alle Zeichen, die ueber das Zeilenende hinausgeschoben werden, gehen verloren.
X txt	(-->)	Vernichten von txt beim ersten Finden, die Zeichen der Zeile werden nachgeschoben.
TILL txt	(-->)	Vernichten aller Zeichen von der aktuellen Kursorposition bis einschliesslich txt