Kassettenarbeit

FORTH83 nutzt einen Teil des Hauptspeichers als RAM-Disk, um hier Screens abzulegen. Die Adressen SOD und EOD definieren Anfang und Ende dieses Speicherbereichs.

```
ORIGIN +
--------
17H  DW  TOP  ;oberste vom FORTH nutzbare Adresse
19H  DW  SOD  ;erste Adresse, die die RAM-Disk belegen kann
1BH  DW  EOD  ;letzte Adresse, die die RAM-Disk belegen kann
```

Die Routinen zum Laden bzw. Abspeichern der RAM-Disk auf Kassette müssen für jeden Rechner speziell entwickelt werden. Selbstverständlich sollten dazu die Routinen benutzt werden, die das jeweilige Rechner-Betriebsprogramm dem Nutzer bietet. Aus Vereinheitlichungsgruenden werden dafür die Befehle PUT und GET empfohlen, die als nicht initialisierte Vektoren schon im Wörterbuch enthalten sind. Die entsprechenden Routinen sollten ins Vokabular I/O gelegt werden.

Mit **1 GET** wird das Laden ab Screen 1 (Block 1) gestartet. Der Filename wird abgefragt. Am Ende des Ladens wird die Nummer n des letzten Screens angezeigt. Falls nicht mit –> programmiert wurde (der Normalfall), werden die Screens mit **1 n THRU** kompiliert, andernfalls mit 1 LOAD. Am Ende des Kompiliervorgangs kommt die Meldung „ok“.

Glossar der Kassettenbedienworte

GET (n –) Voc FORTH
liest ab Screen n über den Header von Kassette.

PUT (n1 n2 –) Voc FORTH
schreibt von Screen n1 bis Screen n2 einschließlich auf Kassette.

SAVE-SYSTEM (–) Voc FORTH
schreibt das gesamte aktuelle System einschließlich eingebrachter Erweiterungen auf Kassette.

MORE (–) Voc FORTH
wiederholt den vorangegangenen Lese/Schreibvorgang von/auf Kassette mit dem gleichen Kopf.

Get/Put Orig. Br.

Kassettenroutinen für Z1013.

Die Variante Get/Put Alpha unterscheidet sich nur in der Definition von PRST. Es wird die Adresse 200 genommen (200 CONSTANT PRST), s. Installation.

```
Screen # 01 ----------------------

00 ( RAM-SAVE ueber Header 5.95)
```

```
01 ONLY FORTH ALSO I/O
02 ALSO ASSEMBLER
03 I/O DEFINITIONS HEX
04 1B CONSTANT ANFA \ Z. anfadr
05 1D CONSTANT ENDA \ Z. endadr
06 23 CONSTANT STA \ Z. stadr
07 200 CONSTANT PRAN \ Prog.anfang
08 320 CONSTANT PRST \ Prog.start
09 \ muss angpasst werden 320
10 : SAVA ( von bis BLnr ---)
11     1+ 'RAM 1- ENDA !
12        'RAM    ANFA !
13        PRST    STA ! ;
14 \ legt Adressen fuer Header
15 \ fest

Screen # 02 -----------------------

00 ( RAM-SAVE ueber Header 5.95)
01 CODE JUMP
02    F5 C, C5 C, D5 C, E5 C,
03    DD C, E5 C, FD C, E5 C,
04    FD C, 21 C, 4300 , \ C-Eintr.
05    3E C, 00 C,
06  \ pusht Register
07    CD C, FFF1 , \ Sprungadr
08    FD C, E1 C,
09    DD C, E1 C, E1 C, D1 C,
10    C1 C, F1 C, C3 C, 422 ,
11  \ popt Register
12          END-CODE
13
14
15

Screen # 03 -----------------------

00 ( RAM-SAVE ueber Header 5.95)
01 FFF4 CONSTANT SARUF \ UP-save
02 FFF1 CONSTANT LORUF \ UP-load
03 \ b. 16K 3C00 LOR. 3C03 SAR.
04 : JUAD ['] JUMP +  ; \ Anspr
05 : NEW 00 0F JUAD C! ; \ neudef
06 : OLD 3A 0F JUAD C! ; \ wiederh
07 : FILE 46 0D JUAD C! ; \ F eint
08 : COM  43 0D JUAD C! ; \ C eint
09 : SARUF! SARUF 11 JUAD ! JUMP ;
10 : LORUF! LORUF 11 JUAD ! JUMP ;
11 \ traegt UP ein ueber Sprungver
12 DEFER EXPAN  \ vekt Erweiterung
13   : ILL ;  ' ILL IS EXPAN
```

```
14 \S ermoeglicht zusaetzliche
15    Erweiterungen fuer Check

Screen # 04 ----------------------

00 ( RAM-SAVE ueber Header 5.95)
01 : RAM-LOAD ( ab nscr ---) CR
02    'RAM ANFA ! NEW FILE LORUF!
03    EXPAN EMPTY-BUFFERS ;
04 : RAM-SAVE ( n1 n2 --)  CR SAVA
05     NEW FILE EXPAN SARUF! ;
06
07 ALSO FORTH DEFINITIONS
08 : SAVE-SYSTEM    ( ---)
09    PRAN ANFA ! HERE ENDA !
10    PRST STA ! NEW COM EXPAN
11    SARUF! ;
12 : MORE  OLD JUMP ; \ wiederh.UP
13
14 ' RAM-LOAD IS  GET ( n --)
15 ' RAM-SAVE IS  PUT ( von bis --)
```

JUMP macht einen allg. Headersave-Funktionsaufruf. Es werden die Einsprungadressen aus dem Sprungverteiler genutzt.

Mit JUAD wird das Wort JUMP modifiziert. NEW und OLD patchen das Wiederholungsflag; FILE und COM den Filetyp, SARUF! und LORUF! die eigentliche Funktion.

```
JUMP is
2A6A:    PUSH  AF
2A6B:    PUSH  BC
2A6C:    PUSH  DE
2A6D:    PUSH  HL
2A6E:    PUSH  IX
2A70:    PUSH  IY
2A72:    LD    IY,4300    ; offs 0Dh H(IY)=Typkennzeichen
2A76:    LD    A,4E       ; offs 0Fh A=0 ohne, A=4EH mit Typ und
Namensabfrage, 3EH Wiederholen mit gleichem Kopf
2A78:    CALL  FFF4       ; offs 11h FFF1H - JMP LORUF, FFF4H - JMP SARUF
2A7B:    POP   IY
2A7D:    POP   IX
2A7F:    POP   HL
2A80:    POP   DE
2A81:    POP   BC
2A82:    POP   AF
2A83:    JP    0422
2A86: END

nb: Aus Forth-Sicht müsste man nicht alle Register retten; BC und IY
reichen.
```

Anpassung

Kennung 'N' für Datei nutzen (statt 00h), d.h. Namensabfrage

```
NEW 4E 0F JUAD C! ; \ neudef
```

Z9001

```
Screen # 01 -----------------------

00 \ 7 Kassettenroutinen VP-08JUL90
01
02 I/O DEFINITIONS HEX
03
04 0300 CONSTANT PANF
05
06 001B CONSTANT DMA
07 005C CONSTANT FCB  DECIMAL
08 FCB 15 + CONSTANT BLNR
09 FCB 17 + CONSTANT AADR
10 FCB 19 + CONSTANT EADR
11 FCB 21 + CONSTANT SADR
12 FCB 23 + CONSTANT SBY
13
14
15

Screen # 02 -----------------------

00 \ Systemruf           VP-08JUL90
01
02 HEX CODE CBOS   ( nr -- RegA f)
03  0E1 C, 0C5 C, 0E5FD , 4D C,
04  0CD C, 5 , 0E1FD , 0C1 C, 26 C,
05  0 C, 6F C, 0E5 C, 62ED ,
06  0E5 C, 0C3 C, >NEXT ,
07 END-CODE
08
09 CODE MAREK
10  0C5 C, 0E5FD ,
11  0CD C, 0FF59 , 0CD C, 0FAE3 ,
12  0E1FD , 0C1 C, 0C3 C, >NEXT ,
13 END-CODE DECIMAL
14
15

Screen # 03 -----------------------
```

```
00 \ Systemroutinen      VP-08JUL90
01
02 : OPENW ( -) CR 15 CBOS 2DROP ;
03 : CLOSW ( -) 16 CBOS 2DROP ;
04 : WRITS ( -) 21 CBOS 2DROP ;
05 : OPENR ( - f) CR 13 CBOS NIP ;
06 : READS ( - f1 f2) 20 CBOS ;
07
08
09 : FTYP   ( adr # -- )
10    3 MIN FCB 8 + SWAP CMOVE ;
11
12 : FNAME  ( name ( -- )
13    FCB 8 ERASE
14    BL WORD COUNT 8 MIN
15    FCB SWAP CMOVE ;

Screen # 04 -----------------------

00 \ Abspeichern 1       VP-08JUL90
01
02 \ saven von AADR bis EADR
03 : FILESAVE   ( -- )
04    EADR @ 1+
05    AADR @ DUP DMA ! -
06    128 /MOD DUP
07    IF SWAP 0= IF 1- THEN
08      0 ?DO WRITS SPACE LOOP
09    ELSE 2DROP THEN
10    CLOSW SPACE ;
11
12
13
14
15

Screen # 05 -----------------------

00 \ Laden               VP-08JUL90
01
02 : STOPKEY   ( --)
03    KEY 3 = IF QUIT THEN ;
04
05 : BLOCKLOAD  ( -- f)
06    BEGIN READS
07     WHILE DROP STOPKEY REPEAT ;
08
09 : CLOAD ( name ( von --)
10    FLUSH 128 DMA !  FNAME
11    " F83" FTYP   BEGIN OPENR
12     WHILE STOPKEY REPEAT
```

```
13    'RAM DMA !
14    BEGIN BLOCKLOAD UNTIL
15    CR ." scr# " SADR ? ;

Screen # 06 ----------------------

00 \ Speichern           VP-08JUL90
01
02 : CSAVE   ( name ( von bis --)
03    1+ 2DUP SWAP - SADR !
04    'RAM 1- EADR ! 'RAM AADR !
05    0 SBY C!
06    FNAME " F83" FTYP
07    OPENW SPACE FILESAVE ;
08 ALSO FORTH DEFINITIONS
09 : .FILES \ Anz. Files von Kass.
10    CR BEGIN 0 36 C!
11       128 DMA ! MAREK BLNR C@
12       0= IF 128 8 TYPE ASCII .
13       EMIT 136 3 TYPE CR THEN
14    36 C@ IF STOPKEY THEN AGAIN ;
15

Screen # 07 ----------------------

00 \ Vektorinit.         VP-08JUL90
01
02
03 : SAVE-SYSTEM ( name ( --)
04    HERE EADR ! PANF DUP AADR !
05    SADR ! 0 SBY C!
06    FNAME " COM" FTYP
07    OPENW SPACE FILESAVE ;
08
09 ' CSAVE IS PUT
10 ' CLOAD IS GET
11
12 ONLY FORTH ALSO DEFINITIONS
13
14 8 LIST
15

Screen # 08 ----------------------

00 \S GLOSSAR
01 n GET fname
02    Lesen File fname.F83 in RAM-
03    Disk ab Screen n
04
05 n1 n2 PUT fname
06    Abspeichern Screens n1 bis n2
```

```
07    auf Kassette mit Namen fname
08
09 SAVE-SYSTEM fname
10    Abspeichern FORTH als File
11    fname.COM
12
13 .FILES
14    Anzeige der Filenamen von Kas
15    sette, Abbruch mit >STOP<
```

MC Routinen dürfen BC (IP) und IY (RP) nicht verändern!

```
CBOS: Aufruf Sprungverteiler, auf Stack wird die Funktionsnummer erwartet)
2A62  E1              POP     HL      ; Funktionsnummer
2A63  C5              PUSH    BC
2A64  FD E5           PUSH    IY
2A66  4D              LD      C,L
2A67  CD 05 00        CALL    0005H   ; Systemruf BOS
2A6A  FD E1           POP     IY
2A6C  C1              POP     BC
2A6D  26 00           LD      H,00H
2A6F  6F              LD      L,A     ; Rückgabe A -> HL
2A70  E5              PUSH    HL      ; auf Stack legen
2A71  ED 62           SBC     HL,HL   ; Cy als Flag
2A73  E5              PUSH    HL      ; auf Stack legen
2A74  C3 22 04        JP      0422H
MAREK: Aufruf OS-Systemfunktion Lesen eines bel. Blocks von Kassette
2A81  C5              PUSH    BC
2A82  FD E5           PUSH    IY
2A84  CD 59 FF        CALL    0FF59H  ; KARAM
2A87  CD E3 FA        CALL    0FAE3H  ; INIT
2A8A  FD E1           POP     IY
2A8C  C1              POP     BC
2A8D  C3 22 04        JP      0422H
```

Beispiel

Screens zählen ab 1. Je nach Größe der RAM-Disk unterscheidet sich die Maximal-Zahl an Screens. Ein Screen umfasst 512 Byte (1/2 KByte). z.B. Z9001 RAM-Disk Speicher 6000h-BFFFh entspricht 48 Screens (1..48)

```
I/O
HEX SOD @ . 6000 ok
EOD @ U. BFFF ok
DECIMAL MAX# . 47 ok      ( EOD = C000h würde hier 48 bringen)
```

```
1 EDIT
    Screen editieren, Kommandos s.
https://hc-ddr.hucki.net/wiki/doku.php/forth/fgforth/sedit
```

```
    Ende mit CTR C - sichert Screen im RAM und verlaesst Editor

1 LOAD
    übersetzt Screen in Forth-Wörterbuch
1 1 PUT TEST1
    sichert Screen 1 bis 1 auf Massenspeicher als Datei "TEST1.F83"

1 GET TEST1
    lädt Datei "TEST1.F83" nach Screen 1 ff.
```