FDOS-Driver MCS for Alpha 1 and Beta 8
(source is loader41.asm)

Der FDOS-Driver bietet dem Benutzer den Zugang zu sämtlichen Disk-Funktionen ohne Zuhilfenahme der FDOS-Exekutive. Das Driver-Programm enthält Routinen zum sequentiellen Schreiben/Lesen eines Bytes. Der Benutzer muß lediglich vorher die entsprechende Datei "eröffnen".

Die Kommunikation erfolgt durch fünf Subroutinen und eines Vektors im RAM, der die benötigten Parameter enthält:

a) Subroutines:

(1) CLEAR (Adresse $B800)

initialisiert den FDOS-Driver und löscht evtl. offene Files.

(2) ENTRY (Adresse $B803)

Einsprung für FDOS-Driver.

Der "Übergabevektor" ($B480-$B48A) bestimmt die auszuführende Funktion.

(3) CHKCNT (Adresse $B806)

prüft beim sequentiellen Lesen, ob das File zu Ende ist oder nicht.

(Setzt Status Bit Z = 1, wenn letztes Byte)

(4) RDCHDK (Adresse $B809)

liest ein Byte von der Diskette in den Akkumulator.

(5) WRCHDL (Adresse $B80C)

schreibt den Akkumulatorinhalt (1Byte) auf Diskette.

(6) "Übergabevektor":

Name	Adresse (Hex)	Bedeutung
TEMPA	B480	legt die auszuführende Funktionen fest.
SVDVT	B481	Drive # (0 oder 1)
FILN	B482-B487	Dateiname (6ASCII'Zeichen)
FLAG	B488	Dateikennung, Programm Daten usw.
L	B489	
H	B48A	Hilfsregister 'HL'

Mögliche Werte für TEMPA und ihre Bedeutung:

TMP - UEBERGABE-WERTE:

F1 EROEFFNE FILE ZUM SCHREIBEN

E1 SCHLIESSE FILE VOM SCHREIBEN UND ERNEUERE DIRECTORY

F0 EROEFFNE FILE ZUM LESEN

E0 SCHLIESSE FILE VOM LESEN

F2 POSITIONIERE RECORD ZUM LESEN BEI GEOEFFNETEM FILE

E2

F3 POSITIONIERE RECORD ZUM SCHREIBEN BEI GEOEFFNETEM FILE

E3 SCHLIESSE FILE OHNE DIRECTORY-EINTRAG

F4 EROEFFNE DIRECOTRY ZUM LESEN

F5 EROEFFNE ALTES FILE ZUM RENAME

F6 EROEFFNE NEUES FILE ZUM RENAME

E5 SCHLIESSE ALTES FILE VOM RENAME

E6 SCHLIESSE ALTES FILE VOM RENAME

F7 EROEFFNE ZIEL FUER COPY

F8 LOESCHE FILE

F9 LAENGENANGABE FÜR CREATE

Loader41 is a 2K file system in ROM at $B800. It handles two drives with 40 tracks of 16 sectors of 128 bytes, about 80 KB per disk. A program can have one file open for reading and one for writing at the same time, and moves data a byte at a time.

Disk layout. The directory is on track 0, read into $B500. Each entry is 12 bytes, ten to a sector: a 6-character name, a flag byte (bit 7, probably write protect) and the start and size of the file. Files are stored in consecutive sectors.

Entry points.

Address	Name	Function
$B800	CLEAR	Reset: close all files
$B803	ENTRY	Execute the command in $B480
$B806	CHKCNT	Test for end of file
$B809	RDCHDK	Read the next byte from the open file into A
$B80C	WRCHDL	Write the byte in A to the open file

Calling a command. Put the command code in $B480, the drive number (0 or 1) in $B481 and the 6-character file name in $B482–$B487, then JSR $B803. In most codes an even value works on the read channel and an odd value on the write channel.

Code	Function
$F0 / $F1	Open a file for reading / writing (probable)
$E0 / $E1	Close the read / write file; closing for writing flushes the last sector
$F2 / $F3	Open at a record position (probable)
$F4	Open from the start of the disk (probable)
$F5	Find or create a directory entry (probable)
$F6 / $F8	Write a directory entry; $F8 also sets the flag bit
$E3	Write out the current sector
$F7	Copy a file between drives (in drdisv4, $EE4E)
$F9	Set the file position for random access (in drdisv4, $EE19)
$E2, $E5–$E9	Accepted, do nothing

The functions marked "probable" are read from the code, not from documentation.

Errors. On an error the loader restores the caller's stack and returns with an error code in $B48B; 0 means no error.

Code	Meaning
1	File not found
2	Sector read or write failed
3	Position beyond the end of the file, or a restore or seek failed
5	Disk full
6	Directory full

Code	Meaning
7	Unknown command
8	Disk write-protected
9	Channel already open

Codes 4 and 15 can also come out of the controller error table for other failed commands; their exact meaning is unclear.