File: 'API-M-DE.DOK' (42 pages) Date: September 18, year of the CPC 2023

File for documentation of the programming interface (API) of the Future Operating System (FutureOS) in the expansion ROM of the M4 expansion.
Functions that are important for programmers and that are located in the mass storage ROM of the FutureOS are described here.
Attention: The M4 XROM (ROM M) is only available if the FutureOS Installer installed it in the M4 expansion!
The OS functions are described in the following form:

1. Brief description: The functioning of the OS function is briefly described in one sentence.

2. Label: This label is used for the OS function in the source code. This label is also used in the supplied label library (#EQU-API.ENG) with the current ROM addresses. In your own programs you should always address all OS functions (and system variables) with these labels. This makes the source code uniform. In addition, the start addresses of the OS functions COULD change in future OS versions. See file '#EQU-API.ENG'.

3. ROM number: this is the logical number of the OS ROM in which the corresponding OS function can be found. However, the OS functions described in this file only occur in ROM M. Since the ROM number is to be understood logically, it should not be automatically equated with the physical ROM select.

4. Start address: indicates the entry address of the OS function.

5. Entry conditions: the entry conditions of the OS function are described and both register contents and RAM variables are discussed.

6. Exit conditions: the exit conditions of the OS function are described and both register contents and RAM variables are discussed.

7. Manipulated: all manipulated or destroyed registers and RAM variables are displayed. Sometimes manipulated peripheral components are also specified.

8. Description: A complete explanation of the functions of the described OS function follows.

9. Please note: some important details are briefly touched upon. This is particularly important because all OS functions have been uncompromisingly trimmed for maximum speed, and incorrect handling can lead to system stability problems.

The OS functions described below are used to manage the SD card of the M4 expansion. They are all located in the FutureOS ROM M.

The current version of this file 'API-M-DE.DOC' is also available on the Internet. See FutureOS homepage:
http://www.FutureOS.de (under Downloads)

M4: Calling an OS function of the M4 XROM via function number

Brief description: An OS function of the XROM M is called using its function number. Afterwards ROM A is switched on.

Label: API_M4

ROM Number: M

Starting address: &C0F7

Entry conditions: Register L contains the function number

Exit conditions: The function was called when the carry flag was set. All other parameters depend on the called OS function of the M4 XROM.

Manipulated: F, H, ROM A is active, and what was changed by the called OS function

Description: This entry point is used to be able to call a function in ROM M from the other OS ROMs A-D. However, it can also be usefully used by the user.
The function number of the function to be called is transferred in ROM M using the L register. Permissible values are a multiple of three. So e.g.: 0, 3, 6, 9, 12, 15, 18 etc.
After calling a function in ROM M, the carry flag is set and then the OS ROM A is switched active.

Please note: FutureOS ROM A was set active before jumping back.

M4: Read the main/root directory of the M4 SD card

Brief description: The main/root directory of the M4 SD card is read.

Label: GET_M4D

ROM Number: M

Start address: &FC81

Entry conditions: A free 16 KB expansion RAM (E-RAM) must be available. It is reserved for buffering the M4 SD card DIRectories.
The variable FN_LR (=&BE11) decides whether to read into the first (=&00) or into the second (=&20) E-RAM directory buffer.

Exit conditions: If the RAM variable 'M4_ERAM' contains the value 0, no directory could be read in because there is no 16 KB E-RAM buffer free. Otherwise the main/root directory was read and buffered into the M4 E-RAM buffer.
The (newly) reserved E-RAM block of the M4 SD card is banked in.

Manipulated: AF, BC, DE, HL, AF', BC', DE', HL', IX, REG16_2-4, L_RAM and the E-RAM status.
The E-RAM block of the M4 SD card has been manipulated.

Description: This OS function allows to reset one of the two buffers for M4 SD card directories to the main/root directory (aka root). The directory is both read and sorted.
The RAM variable FN_LR, which must contain either the value &00 or &20, decides which of the two buffers is reset to the main/root DIRectory.
If a 16 KB E-RAM buffer has not been reserved for the M4 DIRs so far, then this is done automatically.

Please note: A 16 KB E-RAM for buffering the M4 SD-DIR must be available before calling this OS function.

M4: Reading a sub-directory of the M4 SD expansion

Brief description: A sub-directory (Sub-DIR) of the M4 SD card is read. An E-RAM must have already been reserved for the M4 DIR buffer (i.e. a DIRectory has already been read, e.g. the main/root directory using the OS function GET_M4D).

Label: GET_M4S

ROM Number: M

Starting address: &FC84

Entry conditions: A 16 KB expansion RAM (E-RAM) has already been reserved for working with the M4 SD card, e.g. using GET_M4D.
The variable FN_LR (=&BE11) decides whether to read into the first (=&00) or into the second (=&20) E-RAM directory buffer.
The complete path name of the sub-DIR is located beginning at address &4100 (read into buffer 1) or &6100 (read into buffer 2) of the M4 E-RAM.

Exit conditions: A sub-directory was read into the M4 E-RAM and sorted.
The E-RAM block of the M4 SD card is banked in.

Manipulated: AF, BC, DE, HL, AF', BC', DE', HL', IX, REG16_2-4, L_RAM and the E-RAM status.
The E-RAM block of the M4 SD card was banked in and updated.

Description: This OS function reads a sub-directory of the M4 SD card. This buffered directory is also sorted.
The RAM variable FN_LR, which must contain either the value &00 or &20, decides in which of the two buffers the Sub-DIR is read.
If FN_LR contains the value &00, the path name of the sub-DIR must also be in the M4 E-RAM starting at address &4100. However, if the second buffer is read, FN_LR contains the value &20 and the path name is located beginning at address &6100 in the M4 E-RAM.
The E-RAM select of the M4 E-RAM is in RAM variable M4_ERAM.
Values from &C4, &C5 to &FF are usable, just as is usual with the 16 KB E-RAM banking of the CPC.

It can be banked in by: LD B,&7F:LD A,(M4_ERAM):OUT (C),A

The path name must be located inside this E-RAM.
Example of a name: '/FutureOS'

Please note: A 16 KB E-RAM for buffering the M4 SD-DIR must already have been reserved before this OS function is called. This E-RAM select must have been written in RAM variable M4_ERAM (usable values are &C4...&FF).

M4: Change the M4 (Sub-)DIRectory

Brief description: Change one of the two M4 (Sub-)DIRectories.

Labels: M4_CDIR (DIR name in M4 E-RAM) or M4_CD (DIR name at HL)

ROM Number: M

Start addresses: &FC87 (M4_CDIR) or &FC8A (M4_CD)

Entry conditions: The main directory (root) of the M4 SD card must already have been read.
The RAM variable FN_LR (=&BE11) defines whether the directory for buffer 1 (=&00) or buffer 2 (=&20) should be changed.
- When called using M4_CDIR, the full name of the new M4 DIR is located at address &4100 (buffer 1) or &6100 (buffer 2).
- When called using M4_CD, the full name of the new M4 DIRs is in memory beginning at register HL.

Exit conditions:
A = &00 if successful
A = &FF if the M4 DIR was not found

Manipulated: AF, BC, DE, HL and the current M4 DIR
The memory from &BE50 to &BE61 has been manipulated.

Description: These two OS functions of the M4 ROM can be used to change the directory of the M4 SD card.
RAM variable FN_LR decides whether the first (&00) or second (&20) buffered M4 directory should be changed and activated. The new name is either passed in the M4 DIR E-RAM (M4_CDIR) or HL points to it (M4_CD). The name is always terminated by byte &00.

Note: The directory is changed, but not read nor sorted. You would use the OS function GET_M4S for this.

Please note: E-RAM must already have been reserved for buffering the M4 DIRs. This is done, for example, by reading the main DIR of the M4 SD card once.
The directory is changed, but NOT read!

M4: Show the header of the 1st tagged M4 file on screen

Brief Description: The file header of the first tagged file of the M4 SD card will be displayed on the screen.

Label: DHED_M4

ROM Number: M

Starting address: &FC8D

Entry conditions: A file of the M4 SD card should be tagged.

Exit conditions:
XL = &B3: A header is present and displayed. MODE 1 and the lower ROM are now active.
XL = &8C: No header was found in RAM from &B400 onwards. An error message was issued and it was switched to MODE 2, the lower ROM is active.

Manipulated: AF, BC, DE, HL, AF', BC', DE', HL', IX, REG08_1-2, REG16_6 to REG32_1, REG_PC, the RAM areas from &B400 to &B5FF, &B600 to &B6FF and from &BE20 to &BE36. The lower ROM has been switched active. The screen mode is 1 or 2.

Description: This OS function reads the first 512 bytes of the tagged marked file on the M4 SD card to address &B400 into RAM. Then the file header is displayed in MODE 1.
When calling this OS function, a file on the SD card should be tagged, otherwise DHED_M4 returns with error code XL = &8C. In this case, MODE 2 has been switched on and the lower ROM is active.
If DHED_M4 returns with XL = &B3, the file header is displayed on the screen. It is switched to MODE 1 and the lower ROM is switched active.

Please note: After calling this OS function, the lower ROM is switched on. The call must therefore be made from an address above &4000. It's also necessary to reset the screen mode.

M4: Read the header of the first tagged file from M4 SD card

Brief description: The file header of the first tagged file on the M4 SD card is read and stored in RAM from &B400 onwards.

Label: LADAH_M4

ROM Number: M

Starting address: &FC90

Entry conditions: An M4 SD card DIR must be read. A file should be tagged.

Exit conditions:
A = &FF: The file header is in RAM beginning at &B400.
In this case the OS variables (REG08_1) and (REG_PC+1) contain the number of the source medium: &0D for SD card directory 'N' or &0E for the second SD card directory 'O'. Furthermore, the OS variable REG08_2 contains the E-RAM I/O number of the SD card directory E-RAM buffer &C4...&FF. And beginning at REG16_6 the name of the SD card file is given in the format: 'N--:FileNameExt' or 'O--:FileNameExt'.
A = &00: Either no tagged file was found on the SD card. Or there was an error opening the file.

Manipulated: AF, BC, DE, HL, AF', BC', DE', HL', IX, REG08_1-2, REG16_6 to REG32_1, REG_PC, the RAM areas from &B400 to &B5FF, &B600 to &B6FF and from &BE20 to &BE36.

Description: This OS function reads the first 512 bytes of the first tagged file of the M4 SD card to address &B400 into RAM. If the file contains a header, it's now in RAM beginning at &B400.
When calling this OS function, a file on the SD card should be tagged, otherwise LADAH_M4 returns with error code A = &00.
If LADAH_M4 returns with A = &FF, a potential header was successfully read to &B400. However, whether it is really a file header must be checked using the checksum (see TST4HED in this ROM M or TST_HED in ROM B).

Please note: Some ASCII files on M4 SD card may contain an additional file header that has nothing to do with the actual file. This can be examined using the OS function T_HED_A (in ROM M).

M4: Read the header of a file from M4 SD card

Brief description: The file header of a file defined by its name on the M4 SD card is read and stored in RAM from &B400 onwards.

Label: M4_R_HED

ROM Number: M

Starting address: &FC93

Entry conditions: HL = name of the file
An M4 DIRectory must be read. The file is in the current directory.

Exit conditions: A = &00: The file header is located in RAM at &B400.
A = &FF: There was an error opening the file.

Manipulated: AF, BC, DE, HL, AF', BC', DE', HL', the RAM from &B400 to &B5FF and the RAM from &BE20 to &BE36

Description: This OS function reads the first 512 bytes of a file from the M4 SD card into RAM starting at address &B400. If the file contains a header, it is in RAM from there on.
To call this OS function, register HL must point to the name of the file. Furthermore, the directory in which the file is located must be active (see e.g. M4_CDIR or M4_CD).
If M4_R_HED returns with A=&00, a potential header has been read. However, whether it's really a header still has to be checked using the checksum (see TST4HED in this ROM M or TST_HED in ROM B).

Please note: Some ASCII files on M4 SD card may contain an additional file header that has nothing to do with the actual file. This can be examined using the OS function T_HED_A (in ROM M).

M4: Test if a file contains a header

Brief description: It's tested whether a file contains a header.

Label: TST4HED

ROM Number: M

Starting address: &FC96

Entry conditions: DE = pointer to potential header = &XX00/80!

Exit conditions: With the zero flag set, a classic file header was found, as used by the CPC firmware and FutureOS. In this case register BC contains the checksum from the file header data itself.
If the zero flag was deleted, no file header could be recognized.

Manipulated: AF, BC, DE, HL and XL

Description: This OS function can be used to test whether a file contains a file header. Before this OS function TST4HED is called, the potential header must be read. This can be done e.g. using the OS function M4_R_HED. Furthermore, register DE is to be loaded with the pointer to the file header (in this example &B400). The header must always be within a 256-byte page (&NN??)!
After the return of this OS function, the zero flag provides information about the existence of a file header. If it's deleted, the header is missing.
However, if the Zero flag is set after the end of TST4HED, a classic file header was found, as used by the CPC firmware / Amsdos and FutureOS. In this case register BC contains the checksum from the file header.

Please note: Caution! If register BC contains the value &0000 after the return, a header was recognized but does not exist!
If a file contains only &00 bytes in its first 128 bytes (this occurs e.g. with images), then a header is incorrectly recognized. In this case, however, BC = &0000!

M4: Test if an ASCII file contains an M4 specific header

Brief description: It's tested whether an ASCII file contains a header that was additionally generated by the M4 expansion.

Label: T_HED_A

ROM Number: M

Starting address: &FC99

Entry conditions: HL points to the beginning of the potential header

Exit conditions: An additional header generated by the M4 expansion was found when the zero flag was set.
If the zero flag was deleted, no additional M4-specific header was found. However, this file could have a real header, as known from the firmware or FutureOS.

Manipulated: AF and HL

Description: Some ASCII files written to the M4 expansion SD card contain an additional &80 bytes header which is not generated on other storage media.
In order to be able to work with these ASCII files, it's important to know whether such an M4-specific header has been placed in front of a file.
This OS function T_HED_A is used for this. It tests whether there is such an M4 header at the beginning of the file. Before calling 'T_HED_A' it's necessary to read in the potential header using e.g. OS function 'M4_R_HED' in order to be able to examine it.
Before calling T_HED_A, register HL must be loaded with the pointer to the header (&B400 in this example).
After this function returns, the zero flag provides information about the existence of an M4-specific header. If it's deleted, the additional header is missing. In this case, however, the file can contain a regular file header; this must be tested separately, e.g. using the OS function TST4HED in this ROM M or using TST_HED on ROM B.
However, if the zero flag is set after the return of T_HED_A, you should ignore the first 128 bytes of the ASCII file being examined, since they contain this additional M4-specific header.

Please note: A file that does not contain an additional M4-specific header can contain a regular file header, as is usual with the normal firmware of the CPC or FutureOS.

M4: Load a file defined by its name

Brief description: A file whose name is known is loaded from the M4 SD card. Two directories are available (N/O).

Label: LADE_M4

ROM Number: M

Starting address: &FC0C

Entry conditions: The directory of the M4 SD card, in which the file to be loaded is located, should be available and buffered in E-RAM.

A' = Medium (13-14 = &0D-&0E) from which the file is to be loaded. If the 8th bit is set (&8D / &8E), the file header is ignored.

DE = Pointer to the M4 SD card file name. The name can contain up to 27 characters. They are followed by the dot '.', then three characters extension and byte &00.
Or the file name is floppy compatible; it then consists of the user number (1 byte), name (8 bytes) and the extension (3 bytes) of the file: 'UFileNameExt'.

If the file to be loaded has no header or the header is to be ignored, the following data must also be specified:

REG08_4 = Load type (0,1,2,3) - see OS function LADEN4 in ROM M
REG16_3 = Load address in RAM or E-RAM, 16 bits
AKT_RAM = physical E-RAM number, &7FC4, &C5 .. &7FFF .. &78FF
RAMCHAR = Screen MODE during loading

Exit conditions:

The register A provides information about the success of the loading process:

- A = &00 ==> The directory E-RAM of medium N or O is not available, therefore no file can be loaded.
- A = &01 ==> The medium from which the file is to be loaded is not active. The file cannot be loaded.
- A = &02 ==> The file does not exist in the current directory of the specified medium. Or there was an error loading.
- A = &FF ==> The file was loaded properly.

(REG_PC+1) = Number (13 or 14) of the medium (SD card directory N or O) from which the file was loaded.
(FN_LR) = &00 / &20 depending on whether from medium N or O was loaded.

Manipulated: AF, BC, DE, HL, AF', BC', DE', HL', IX, IY and the RAM variables REG08_0-3, 6 REG16_1, REG_PC, AKT_RAM and FN_LR. Also the RAM areas &B400-&B6FF, &BE20-&BEAF and &BC00-&BC7F, as well as the RAM status and the screen mode.

Description: This OS function is used to load any file. The file is uniquely identified by its source medium and its name. The name can either be in regular FutureOS format or in M4 format, with conversion if required.
The source medium (&0D or &0E for the current directory in N or O) is specified in A'. If the 8th bit is set (A' = &8D / &8E), any file header is ignored. Otherwise the header decides

where the file is loaded. Without a file header, the following RAM variables decide at which address, in which RAM block, and in what way the file is loaded.
REG08_4 = Load type: The load type &02 loads the file into main memory. Load type &03 loads the file into E-RAM (see LADEN4).
REG16_3 = Load address in RAM or E-RAM, 16 bit
AKT_RAM = physical E-RAM number: &7FC4 .. &7FFF .. &78FF (4 MB!).
If the file was loaded correctly, the OS function returns with &FF in the accumulator. Otherwise, A contains the error code.
REG_PC+1 contains the number of the medium (&0D / &0E) from which the file was loaded. &0D / &0E correspond to media N / O.

Please note: This function is used by the ROM C function 'LOAD'

M4: Reading a file of up to 64 KB

Brief description: Reading a file of up to 64 KB in length from the SD card of the M4 expansion into the current RAM configuration.

Label: M4_R_64K

ROM Number: M

Starting address: &FC9C

Entry conditions: The current RAM configuration is used
HL = pointer to the complete path+filename of the file to be read
The path+name is terminated by byte &00.
DE = Target address of the file to be read from the M4 SD card

Exit conditions:

- A = &00: The file was read successfully. In this case, the file header is located in RAM starting at &BC00.
- A = &FD: Error opening file (DE is preserved)
- A = &FE: The file is larger than 64 KB, so too large to load
- A = &FF: The file does not exist or was not found

Manipulated: AF, BC, DE, HL, AF', BC', DE', HL', the RAM from &BC00 to &BC7F and the RAM from &BE20 to &BE7F

Description: This OS function is used to read a file of up to 64 KB from the M4 SD card into RAM. The current RAM configuration is used.
A pointer to the path+filename is transferred in register HL and the target address of the file is transferred in register DE.
After successfully loading the file, register A is set to the value &00. If the accumulator has a different value, an error has occurred during loading.
To load a file larger than 64 KB, you can use the OS function 'M_R_X16'.

Please note: only files up to a maximum of 64 KB can be used

M4: Reading a file of up to 4 MB

Brief description: Reading a file of up to 4 MB in length from the SD card of the M4 expansion into the expansion RAM (E-RAM).

Label: M4_R_X16

ROM Number: M

Starting address: &FC9F

Entry conditions: AKT_RAM: First E-RAM block to be read into.
HL = pointer to the complete path+filename of the file to be read. The path+name is terminated by byte &00.
DE = Target address of the file to be read from the M4 SD card, less than &8000.

Exit conditions:

- A = &00: The file was read successfully
- A = &FD: Error while opening the file (DE is preserved)
- A = &FE: The file is larger than 4 MB, so too big to load it
- A = &FF: The file does not exist or was not found

Manipulated: AF, BC, DE, HL, AF', BC', DE', HL', IX, the OS variable AKT_RAM and the RAM area from &BE20 to &BEAF.

Description: This OS function is used to read a file of up to 4 MB from the M4 SD card into the expansion RAM. The E-RAM configuration from the OS variable AKT_RAM is used for this. It specifies the first 16 KB E-RAM block to be read into.
A pointer to the path+filename is passed in register HL. And the target address of the file is passed in register DE. The target address must be less than &8000. It's usually in the range from &4000 to &7FFF.
After successfully loading the file, register A is set to the value &00. If the accumulator has a different value, an error has occurred during loading.
The OS function 'TEILA4' is useful for reading files larger than 4 MB.

Please note: only files up to a maximum of 4 MB can be used

M4: (Partial) loading a marked file into short-term storage

Brief description: If possible, a file is loaded completely from the M4 SD card into all free 16 KB short-term memory E-RAMs (KZS).

Label: TEILA4 (first loading) // TEILB4 (reload remainder(s))

ROM Number: M

Start address: &FCA2 (TEILA4) // &FCA5 (TEILB4)

Entry conditions:
TEILA4: System and XRAM_?? variables contain correct values.
TEILB4: The content of the RAM variables REG08_2, REG_IY, REG_SP, REG_PC+1 has not been changed since TEILA4 was called!

Exit conditions: (applies to TEILA4 and TEILB4)

- A = &00 ==> The file was completely loaded into the KZS E-RAMs.
- A = &F0 ==> The file was partially loaded into the KZS E-RAMs, all KZS E-RAMs are fully occupied by loaded data. The rest of the file can be loaded using TEILB4.
- A = &FC ==> Error loading 16 KB block from SD card
- A = &FD ==> Error opening file from SD card
- A = &FE ==> No (KZS) E-RAM free, so nothing can be loaded
- A = &FF ==> No file tagged on M4 SD card

REG08_1 = REG_PC+1 = Source medium number = &0D / &0E for 'N'/'O'.
REG08_2 = E-RAM block that buffers the two SD card directories
REG_BC1 = / bits 0-15 / 32 bits file length of the loaded file
REG_DE1 = / bits 16-31 /
REG_SP+1 = file ID = file descriptor of the opened file (SD card)

Manipulated: AF, BC, DE, HL, AF', BC', DE', HL', IX, IY, the OS RAM variables REG08_0-5, REG16_1, REG16_6...REG32_1 (only TEILA4), AKT_RAM, REG_IY , REG_SP, REG_PC, XRAM_C4..FF and the RAM status

Description: This OS function makes it possible to load a tagged file of any length from the M4 SD card into the short-term memory (KZS) blocks (= 16 KB E-RAMs). If the number of existing KZS E-RAM blocks is not sufficient, the file is only loaded until all 16 KB KZS E-RAM blocks are filled with the files data. The rest of the file can then be subsequently loaded, if necessary in several steps. The first tagged file is loaded.
If the file fits completely in memory, the OS function returns with A=&00. So there is no need to load the rest of the file. If the file was too large to be loaded completely, all 16 KB KZS E-RAM blocks were loaded with the first part of the file and the OS function returns with A=&F0.
(If the accumulator contains a different value, an error has occurred).
After loading a part of the file, care must be taken that the RAM variables REG08_2, REG_IY, REG_PC+1 and REG_SP remain unchanged.
Only then can the remainder of the file be loaded with TEILB4. The exit conditions of TEILB4 correspond to those of TEILA4. Under certain circumstances, another part of the file has to be loaded, this depends on the total length of the file.

Please note: In order to call up TEILB4 correctly, the RAM variables REG08_2, REG_IY, REG_SP and REG_PC+1 must remain unchanged after calling TEILA4!

M4: Load file according to its header / No header --> OS

Brief description: A file with a file header is loaded according to the data in its file header. Namely from the M4 SD card.
If the file to be loaded does not contain a header, the system jumps to the OS.

Label: X_LADEN_NO

ROM Number: M

Starting address: &FCA8

Entry conditions: The E-RAM buffer of the M4 SD card is banked in between &4000 and &7FFF (see OS variable M4_ERAM).
HL = pointer to the name of the file to be loaded
FN_LR = &00 (load from SD card 'N') or &20 (load from SD card 'O')
YH = &FF -------> No header --> jump to KLICK in ROM D
YH = &00-&FE -------> No header --> Jump to the M4 shell in ROM M
RAMCHAR = Screen MODE during loading

Exit conditions: If, contrary to expectations, the file to be loaded does not have a header, this OS function either jumps to KLICK in ROM D (YH was &FF) or to the shell of the M4 XROMs M (YH was less than &FF).
A = &00 --> The file was loaded according to its header
A > &00 --> An error occurred while loading the file
The screen MODE specified in RAMCHAR has been activated

Manipulated: AF, BC, DE, HL, AF', BC', DE', HL', IX, REG08_3, REG16_1, AKT_RAM, REG_PC, the target RAM / E-RAM of the loaded file, the RAM areas of &B400-&B6FF, &BC00-&BC7F and from &BE20 to &BEAF. Also the screen mode and the RAM / E-RAM configuration

Description: This OS function is to read a file from SD card of M4 expansion. The file is loaded according to its file header data. Either in the main memory or in the E-RAM. However, if the file does not contain a header, the system jumps back to the OS, either via 'KLICK' (ROM D) or to the M4 Shell. This depends on Z80 register YH.
Before calling this OS function, HL points to the file name of a file on the M4 SD card. The M4 E-RAM (see OS variable M4_ERAM) must already be banked in because the file names are stored in there.
The OS variable FN_LR indicates whether the first or second buffered directory of the M4 SD card should be used. So whether it should be loaded from medium N or O. Accordingly, FN_LR must contain the value &00 or &20. To define which screen MODE should be active during loading, the MODE (0-3) can be written to the lower two bits of the OS variable RAMCHAR.
After the file has been loaded, the accumulator provides information about the success: If A = &00, the loading was successful. Otherwise an error occured.

Please note: This OS function jumps back to the OS if the file to be loaded has no header!

M4: Load selected file into main memory or E-RAM

Brief description: The first tagged file is loaded into the main memory or the expansion RAM (E-RAM).

Label: LADEN4

ROM Number: M

Starting address: &FCAB

Entry conditions: A = &FF --> File tag byte unchanged
REG08_3 defines the loading type of the file:
= 0 --> Load into main memory beginning at &0000
= 1 --> Load into E-RAM at address &0000 and the first E-RAM (&C4)
= 2 --> Load into main memory at address in OS variable (REG16_1)
= 3 --> Load into E-RAM of OS variable (AKT_RAM) at address (REG16_1)
REG16_1 = destination address - only for loading types 2 and 3!
AKT_RAM = First E-RAM block to be used - only with loading type 3!
RAMCHAR = Screen MODE during loading

Exit conditions: OS variable (REG_PC+1) contains the source medium
A = &00 --> The file was loaded according to its header
A > &00 --> An error occurred while loading the file
The screen MODE specified in RAMCHAR has been activated

Manipulated: AF, BC, DE, HL, AF', BC', DE', HL', IX, REG08_1-3, REG16_1-2, REG16_6..REG32_1, AKT_RAM, REG_PC, the target RAM / E-RAM of the loaded File, the RAM from &B600-&B6FF, from &BC00-&BC7F and from &BE20 to &BEAF. Also the screen mode, the file tagging bytes and the RAM / E-RAM configuration

Description: This OS function is to read the first tagged file from M4 expansion SD card. The file is loaded into the main memory or into the E-RAM, this depends on the content of the OS variable REG08_3. If the accumulator contains the value &FF when LADEN4 is called, the file tagging byte is retained, otherwise the status of the file is set from 'tagged' to 'used' (regular case). Additional parameters must be specified for loading types 2 and 3 (see above).
To define which screen MODE should be active during loading, the MODE (0-3) can be written to the lower two bits of the OS variable RAMCHAR.
After the file has been loaded, the accumulator provides information about the success: If A = &00, the loading was successful. Otherwise an error occured.

Please note: The file is loaded according to the loading type defined in REG08_3, the file header has no influence.

M4: save a file to M4 SD card

Brief description: A defined memory content (main memory or E-RAM) is saved as a file on the SD card of the M4 expansion.

Label: SICHRE

ROM Number: M

Starting address: &FC12

Entry conditions: The SAVE mode is entered in REG08_3:
(REG08_3 = &32 (="1"+1) ==> save foreground program) / currently
(REG08_3 = &33 (="2"+1) ==> save background program) / inactive!
REG08_3 = &34 (="3"+1) ==> Save from main memory
REG08_3 = &35 (="4"+1) ==> Save from expansion RAM
REG16_6+1 = Target medium to be saved to: This medium is symbolized by an ASCII character, either 'N/n' (=&4E/&6E) or 'O/o' (=&4F/&6F) for one of the SD card directories.
For SAVE mode 3(&34) or 4(&35) additional data must be specified:
REG16_8 = Beginning of the name of the file to be backed up. Either these are 12 bytes: the user number (1 byte), file name (8 bytes) and expansion (3 bytes). The User# is ignored!
Or the first byte (user number) is &FF, then REG16_9 contains the pointer to a regular M4 SD file name: file name (up to 27 characters), '.'-character and the extension (3 characters).
REG_IX = Source address from which to save (16 bit)
AKT_RAM = Source block from which to save: &7FC0/&7FC4-&78FF
REG_IY = file length in KB (16 bits). File length: maximum 4 MB

Exit conditions: File was saved on the M4 SD card and added to the appropriate directory.
Or there is a jump to the OS if an error has occurred.
In any case, the main memory is banked in (AKT_RAM = &7FC0).

Manipulated: AF, BC, DE, HL, AF', BC', DE', HL', IX, IY and the RAM variables REG08_1, AKT_RAM, REG_R, FN_LR, the RAM area from &B600 to &B70F. In addition, the E-RAM configuration and one of the SD card directories have been changed.

Description: This OS function allows a memory area to be written as a file in one of the two buffered directories of the M4 SD card (and noted in the directory).
The variable REG08_3 specifies how to save, either a program that has already been read in can be saved or any RAM / E-RAM area.
In any case, drive 'N' or 'O' must be specified in REG16_6+1 as an ASCII value of &4E/&6E (N/n) or &4F/&6F (O/o).
From REG16_8 either the name begins in the classic format: User(1) + Name(8) + Extension (3). Or REG16_8 contains the byte &FF, in this case REG16_9 points to an M4 SD card file name in the format Name(x) + '.'-character + extension(3). The name should contain a maximum of 27 characters. The extension may contain a maximum of 3 characters.

Please note: It can only be saved if an E-RAM has been reserved for the M4 SD card and a target directory has been read.
The target medium is specified in REG16_6+1 as an ASCII value.
If an error occurs, this OS function jumps to the desktop!

M4: Save a file of up to 64 KB

Brief description: A file of up to 64 KB in length is written to the SD card of the M4 expansion. Therefore the current RAM configuration is used.

Label: M4_W_64K

ROM Number: M

Starting address: &FCAE

Entry conditions: The current RAM configuration is used
HL = Pointer to the complete path+filename of the file to be written. The path+name is terminated by byte &00.
DE' = Length of the file to be saved in bytes (max. 64 KB)
HL' = Source address of file to save to the M4 SD card

Exit conditions: A = &00: The file was written successfully.
In this case the zero flag is set and A' = file descriptor.
A = &FF: An error occurred while creating the file.

Manipulated: AF, BC, DE, HL, AF', BC', DE' and HL'

Description: This OS function is used to save a RAM area of up to 64 KB in a file on the M4 SD card. The current RAM configuration is used.
Before this OS function is called, a pointer to the path+filename is given in the HL register. DE' contains the file length in bytes and HL' the start address from where to write.
After successfully writing the file, register A contains the value &00. If the accumulator contains a different value (A=&FF), an error has occurred during writing.

To write a file larger than 64 KB, you can use the OS function 'M4_W_X16'.

Please note: Only files up to a maximum of 64 KB can be saved.

M4: Write a file of up to 4 MB

Brief description: A file of up to 4 MB in length is written from the E-RAM to the SD card of the M4 expansion.

Label: M4_W_X16

ROM Number: M

Start address: &FCB1

Entry conditions: AKT_RAM: First E-RAM block that is written
HL = Pointer to the complete path+filename of the file to be written. The path+name is terminated by byte &00.
DE' = Length of the file to be saved in KB (max. 4096)
HL' = Source address of file to save for M4 SD card

Exit conditions: The file was successfully written if the accumulator contains a value less than &FF, this value then corresponds to the file descriptor
A = &FF: An error occurred while creating the file

Manipulated: AF, BC, DE, HL, AF', BC', DE', HL' and AKT_RAM

Description: This OS function is used to save an expansion RAM area in a file of up to 4 MB in length on the SD card of the M4 expansion.
Before the call, the E-RAM configuration from which writing is to start must be written to the OS variable AKT_RAM. It specifies the first 16 KB E-RAM block to be saved from. Pass a pointer to the path+file name in register HL. DE' contains the file length in KB (maximum 4096). And HL' contains the start address from which to write.
After the file is successfully written, register A contains the value of the file descriptor. However, if the accumulator contains the value &FF, an error has occurred during writing.

To write a file larger than 4 MB, one can use the OS function 'TEISI4'.

Please note: Only files up to a maximum of 4 MB can be saved.

M4: (Partially) saving a file from the short-term storage E-RAMs

Brief description: A file is (partially) saved from the KZS E-RAMs.

Label: TEISI4 (start of file) and TEISK4 (rest of file)

ROM Number: M

Start address: &FCB4 (TEISI4), &FCB7 (TEISK4)

Entry conditions:
TEISI4: A = &0D or &0E, depending on whether to save to the first or second buffered directory of the M4 SD card, N or O.
REG16_8 = Pointer to the file name (WITHOUT path specification!) plus byte &00
REG_BC1 = / bits 0-15 / 32 bits Length of the file to be saved
REG_DE1 = / bits 16-31 / specified in bytes (max. 4 GB)

TEISK4: The RAM variables REG_AF1, REG_BC1 and REG_DE1 must have been preserved since TEISI4 was called.
Exit conditions: The accumulator provides information about the success of the operation:

- A = &FF ==> The file was completely written to the SD card
- A = &F0 ==> The file was only PARTIALLY written
  Further parts can now be saved using TEISK4
- A = &00 ==> There is no E-RAM buffer for the M4 SD card
- A = &01 ==> No directory was read/buffered from the SD card
- A = &02 ==> The file has a length of 0 KB
- A = &05 ==> There are no KZS free in the E-RAM, E-RAM used differently
- A = &06 ==> The file could not be created for writing

Manipulated: AF, BC, DE, HL, AF', BC', DE', HL', IX, REG08_1, REG_AF1, REG_BC1, REG_DE1, FN_LR, the XRAM_?? Variables, the RAM areas &B600-&B6FF and &BE20-&BE31
In addition, the RAM configuration and the target directory have been changed

Description: This OS function is the counterpart to TEILA(4). While TEILA(4) partially loads a file, TEISI4 / TEISK4 are used to save the loaded file again. Whole or in pieces, it depends on the E-RAM available. Free blocks are used as KZS, that are 16 KB E-RAMs used for short-term storage.
Initially, TEISI4 is called. TEISI4 opens a file for writing and saves all short-term storage (KZS) E-RAMs on SD card. The success of the action is passed in the accumulator.
If TEISI4 returns with the byte &FF in the accumulator, then the file has been completely saved. However, if the accumulator contains the value &F0, all KZS E-RAMs were saved, but the file is longer than the E-RAM available. The RAM variables REG_AF1, REG_BC1 and REG_DE1 must now be preserved. The remainder of the file must first be loaded with TEILB(4), then this remainder can be written to disk with TEISK4. This process shall be repeated several times if the file is very long or E-RAM is low.

Please note: After calling TEISI4, the RAM variables REG_AF1, REG_BC1 and REG_DE1 must be preserved, as they are required when saving other parts of a very large file using TEISK4.

M4: Open a file to read and determine the file length

Brief description: A file of the M4 expansion SD card is opened for reading and its file length is determined.

Label: M4_O_S

ROM Number: M

Starting address: &FCBA

Entry conditions: HL = Pointer to the complete path+filename of a file on the SD card of the M4 expansion. The path+name is terminated by byte &00.

Exit conditions: A = &00: The file was opened and its file length was determined. In this case, A' contains the file descriptor and registers HL and DE contain the 32-bit file length.
A = &FD: An error has occurred -> HL and DE are meaningless

Manipulated: AF, BC, DE, HL, AF', BC', DE', HL' and the RAM area from &BE20 to &BE4F

Description: The OS function M4_O_S is used to open a file from an SD card for reading and to determine its file length.
To do this, a pointer to the entire path+filename is given in HL. This path+filename ends with byte &00.
After the successful call of M4_O_S, the value &00 is written into the accumulator. A' contains the file descriptor, this file ID is important to access or close the file.
In addition the registers HL and DE contain the precise file length in bytes. HL contains bits 31-16 and DE bits 15-0.
However, if M4_O_S returns with A = &FD, an error has occurred.

Please note: An open file should be closed again after usage. The OS function M4_DSL can be used for this.
The opened file can only be read, not written.

M4: Close open file of the M4 SD card

Brief description: A previously opened file of the SD card of the M4 expansion is closed again.

Label: M4_DSL

ROM Number: M

Starting address: &FCBD

Entry conditions: A' = file descriptor = file ID

Exit conditions: BC = &FC00 // DE = &4304

Manipulated: BC, DE and the file was closed

Description: Open files should also be closed after usage. Because it's only possible to keep a certain number of files open at the same time.
The OS function M4_DSL is used to close a file. When called, register A' must contain the file descriptor of the previously opened file.

Please note: There is no feedback about the success.

M4: Reading 16 KB of an open file to address &4000

Brief description: A block of 16 KB is read to &4000.

Label: M4_R16

ROM Number: M

Starting address: &FCC0

Entry conditions: A file must already have been opened for reading. The current E-RAM configuration is used for reading.
A' = file descriptor of opened file of M4 SD card

Exit conditions:
A = &00: Data was read successfully
A > &00: An error has occurred!

Manipulated: AF, BC, DE, HL, BC', DE', HL', the RAM area from &4000 to &7FFF and the RAM area from &BE50 to &BE7F

Description: This OS function is used to read 16 KB from a file and write it to RAM from &4000 onwards. The file on the M4 SD card must already have been opened for reading. Its file descriptor is passed to M4_R16 in register A'. The 16 KB read data are written to the block from &4000 to &7FFF of the current E-RAM configuration. This makes it possible to transfer 16 KB data blocks directly into 16 KB E-RAM blocks.
After the return, the accumulator contains the value &00 if reading was successful, otherwise the accumulator has a value greater than &00.

Please note: Before calling this OS function, an M4 SD card file must have been opened for reading! This can be done e.g. using the OS function 'M4_O_S'.

M4: Write 16 KB from RAM to a file open for writing

Short Description: Write 16 KB of data to an open file.

Label: M4_W16

ROM Number: M

Starting address: &FCC3

Entry conditions: A file must already have been opened for writing. The current E-RAM configuration is used to write the data into the file.
A' = file descriptor of opened file of M4 SD card

Exit Conditions: 16 KB were written to an open file from the current RAM configuration. The 16 KB block was written from &4000 to &7FFF.

Manipulated: AF, BC, DE, HL and the file to be written got 16 KB longer

Description: This OS function is used to write 16 KB of data from RAM beginning at &4000 to &7FFF to a file. The file on the M4 SD card must already have been opened for writing. When M4_W16 is called, the file descriptor of the file being open for writing must be given in register A'. The 16 KB to be written are fetched from the RAM block from &4000 to &7FFF of the current E-RAM configuration. This makes it possible to write complete 16 KB E-RAM blocks in one go to an open file.

Please note: Before calling this OS function, an M4 SD card file must have been opened for writing!

M4: Delete file from M4 SD card

Brief Description: A file is deleted from M4 SD card.

Label: M4_ERA

ROM Number: M

Starting address: &FCC6

Entry conditions: HL = Pointer to path+filename of the file to be deleted. The path+filename ends with byte &00.

Exit conditions:
A = &00: File deletion was successful
A = &FF: An error has occurred

Manipulated: AF, BC, DE, HL and the RAM area from &BE50 to &BE61

Description: This OS function is used to delete files or (empty) directories. To do this, a pointer to the entire path+filename of the file/directory is provided in HL.
If M4_ERA returns with A = &00, then the deletion was successful. If an error occurs, the accumulator contains value &FF.
Caution: If a directory is to be deleted, all files and subdirectories must first be deleted. To be able to delete a directory, it must be empty.

Please note: When deleting directories, they must be empty.

M4: Creating a new (sub)directory on SD card

Brief description: Create a (sub)directory on the SD card of the M4 expansion.

Label: M4_MKD

ROM Number: M

Starting address: &FCC9

Entry conditions: HL = Pointer to the name of the new directory to be created. The name ends with byte &00.

Exit conditions:
A = &00: The new directory was created
A = &FF: An error has occurred, e.g. directory already exists

Manipulated: AF, BC, DE, HL and the RAM area from &BE50 to &BE61

Description: If you want to create a new (sub)table of contents on the SD card of the M4 expansion, you use this OS function M4_MKD.
The new directory is created in the currently valid directory. The current directory can be changed with the OS functions M4_CDIR or M4_CD (see above).
Before calling this function, a pointer to the name of the new directory must be provided in register HL. The name must end with byte &00. After calling M4_MKD, the accumulator either contains the value &00, then the new DIR was created successfully. Or the accumulator contains the value &FF, in which case an error has occurred. Such an error can be, for example, an unusable name, or the directory to be created already exists.

Please note: If an attempt is made to create an already existing directory again, this OS function aborts with the error code &FF in register A.

M4: Rename a file on the M4 SD card

Brief description: A file of the M4 SD card is renamed.

Label: M4_REN

ROM Number: M

Starting address: &FCCC

Entry conditions:
HL = Pointer to the old path+name of the file / Both names end
DE = Pointer to the new path+name of the file / with the byte &00!

Exit conditions:
A = &00: The file was successfully renamed
A = &04: The file does not exist (old file name or path incorrect)
A = &08: A file with the new name already exists
A = &FF: Another error has occurred

Manipulated: AF, BC, DE, HL and the RAM area from &BE50 to &BE61

Description: The OS function M4_REN allows renaming files.
To do this, pointers to the old and new path+filename of the file must be transferred in HL and DE.
If the function returns with &00 in the accumulator, the renaming was successful.
Otherwise, the content of A provides information about the error.

Please note: If errors occur, please check the two path+filenames

M4: Determine the file length of a file on SD card

Brief description: The size of a file on the SD card is determined.

Label: M4_G_SIZE

ROM Number: M

Starting address: &FCCF

Entry conditions: HL = Pointer to the complete path+filename of a file on the SD card of the M4 expansion. The path+name is terminated by byte &00.

Exit conditions:
A = &00: The file length (bytes) was determined.
It is transferred in registers HL and DE.
A = &FD: An error has occurred -> HL and DE are meaningless

Manipulated: AF, BC, DE, HL, AF', BC', DE', HL' and the RAM area from &BE20 to &BE4F

Description: The OS function M4_G_SIZE is used to determine the length of a file from an SD card in bytes.
To do this, a pointer to the entire path+filename is provided in HL. This path+name ends with byte &00.
After the successful call of M4_G_SIZE, the value &00 is returned in the accumulator. And registers HL and DE contain the precise file length in bytes. HL contains bits 31-16 and DE bits 15-0.
However, if M4_G_SIZE returns with A = &FD, an error has occurred. In this case HL and DE are meaningless.

Please note: The checked file was closed again.

M4: Generate a path+filename from current path and filename

Short description: A string containing path and name + 0 is generated from the current path of the M4 SD card and a file name.

Label: ER_PN or EP_B6 (path+name starts from &B600)

ROM Number: M

Start address: &FCD2 (ER_PN) or &FCD5 (EP_B6)

Entry conditions: A directory of the M4 SD card must have been read. And the buffer E-RAM of the SD card directories must be banked in.
DE = Target address from which path + name + byte &00 should be generated. DE does not have to be loaded when EP_B6 is called, because DE is automatically set to &B600 in this case.
HL = Pointer to a file name in one of the two buffered SD card directories in the SD DIR buffer E-RAM.
The RAM variable FN_LR contains either the value &00 or &20. This determines whether the path from the cached SD card directory of SD card media 'N' or 'O' should be used.

Exit conditions: Path + file name + byte &00 are in RAM beginning at the specified target address. If EP_B6 was called, this is &B600.
DE = Pointer to the byte after path+name+&00. A = &00.

Manipulated: AF, BC, DE, HL and the RAM area of path + name + &00

Description: When using the M4 SD card, one or two directories can be buffered. The variable FN_LR indicates whether the first SD card drive 'N' or the second 'O' should be used. Therefore FN_LR (=&BE11) contains either &00 or &20.
The OS function ER_PN now uses the selected SD card directory to compile the path + name + byte &00 from the selected path of this directory and a file name (HL). Register DE contains a pointer to the destination address. When using EP_B6, DE does not have to be loaded because &B600 is set as the target address.
After calling one of the two functions, DE points to the byte after the newly generated 'path + filename + byte &00'. The byte &00 serves as an end marker.

Please note: If ER_PN or EP_B6 is called with HL = &FFFF, then only the selected path of the selected SD card medium is copied.

Find the first tagged file of all media (A to O)

Brief description: All media from A to O (floppy disc drives, HD20 hard disk and both buffered M4 SD card directories) are searched for the first tagged file. This OS function corresponds to FMD32 in ROM C.

Label: FMD_AO

ROM Number: M

Starting address: &FCD8

Entry Conditions: See FMD32 in ROM C

Exit Conditions: See FMD32 in ROM C

Manipulated: See FMD32 in ROM C

Description: See FMD32 in ROM C
The OS function 'FMD_AO' of the ROM M calls the OS function 'FMD32' in ROM C. Please read there.

Please note: See FMD32 in ROM C

M4: Find first tagged file of SD card

Brief description: All file tagging bytes of media N and O (of the M4 SD card) are searched for a tagged file.

Label: FMD_NO

ROM Number: M

Starting address: &FC03

Entry Conditions: The file tagging bytes of media 'N' and 'O' must be correct. They are located in the E-RAM of the M4 expansion.
A < &FF ==> regular function
A = &FF ==> File status is NOT changed!

Exit conditions:
HL = &0000 and zero flag = 1 ==> No file tagged on M4 SD card!
HL > &0000 and zero flag = 0 ==> HL points to the 'address' of the name of the first tagged file in the DIRectory E-RAM of the M4 SD card.
REG08_1 = A = Medium on which the file is located, this is either &0D for medium 'N' or &0E for medium 'O'.
REG08_2 = E-RAM select of the M4 SD card DIRectories &C4-&FF. This E-RAM is already banked in.
REG16_6 to REG32_1 = Short file name (16er) in the format 'N--:FileNameExt' or 'O--:FileNameExt'. The point between name and extension has been removed!
FN_LR was set to &00 (for medium N) or to &20 (for medium O) depending on which medium the first tagged file was found on.

If A was not equal to &FF when FMD_NO was called:
The file status is changed from Tagged (marked) to Retagged (used), such a file is shown with a strikethrough.

Manipulated: All registers except IY. E-RAM status, file tagging bytes, REG08_1..2, REG16_6..REG32_1

Description: This subprogram is used to search for a tagged/marked file in media 'N' and 'O' of the M4 SD card. At entry, the system variables must contain correct values.
This OS function provides the address of the file name of the first tagged file in HL. In addition this file name is converted to 16er short format and stored in RAM from REG16_6 onwards to remain compatible with 'FMD32'.
If register HL contains the value &0000 and the zero flag is set, then no tagged file was found.
With HL > &0000, the medium on which the first marked file is located is specified in the accumulator and in REG08_1. Either &0D or &0E for medium 'N' or medium 'O' respectively. Both are buffered directories of M4 SD card.
In addition, the physical E-RAM select from &C4 - &FF is provided in REG08_2, in which the two directories of the M4 SD card are buffered. This E-RAM is already banked in.

Please note: The system variables must be correct, otherwise corrupt data could be returned.
Caution: If register A has the value &FF before the OS function is called, the status of the file found will NOT be changed. If you called FMD32 again, you would find the same file again!

Instead of loading the accumulator with &FF, you can also call this OS function via label EMD_NO, at address: &FC00.

M4: find tag byte of first tagged file of 'N' or 'O'

Brief Description: All file tagging bytes of media N or O (of M4 SD card) are searched for the 'Tagged' mark. The number of the first marked file is returned.

Label: FTBNO

ROM Number: M

Starting address: &FCDB

Entry Conditions: The file tagging bytes of media 'N' and 'O' must be correct. They are located in the E-RAM of the M4 expansion. This E-RAM must be banked in.
HL = Pointer to start of file tagging bytes
HL = &4400 for medium 'N' / N and O are the two buffered M4's
HL = &6400 for medium 'O' / SD card directories

Exit conditions:
HL = &0000 ==> No file tagged on M4 SD card!
HL > &0000 ==> HL contains the number of the marked file &0001-&03FF and DE points to the currently marked tag byte --> DE = &4400-&47FF (Medium N) or = &6400-&67FF (Medium O)

Manipulated: AF, C, DE and HL

Description: This OS function searches for the first marked file tagging byte of either medium 'N' or medium 'O'. Both media are buffered directories of the M4 SD card.
When the function is called, HL must be loaded with either &4400 or for &6400. This address corresponds to the start of the tagging/marker bytes of medium N or O.
After the return, HL provides information about the success of the operation. If HL is loaded with &0000, no tagged file was found. However, if HL contains a value between &0001 and &03FF, this value corresponds to the number of the tagged file of the medium being searched.
In this case, register DE also contains a pointer to the current (tagged/marked) file tag byte.

The OS function FNOFN can be used to convert the file number of the first tagged file found into a pointer to its name.

Please note: Before calling this function, the buffer E-RAM of the M4 expansion must be banked in.

Example: LD B,&7F:LD A,(M4_ERAM):OUT (C),A ;Bank in M4 E-RAM

M4: Calculate pointer to filename from file tag byte number

Brief description: The number of a file tag byte is converted into the address of the name of this file on the M4 SD card.

Label: FNOFN

ROM Number: M

Starting address: &FCDE

Entry conditions: The E-RAM of the M4 SD card must be banked in
DE = Number of (tagged) file &0001-&0400. Attention: Do NOT use 0!
HL = Pointer to the 1st name of the medium 'N' or 'O' of the M4 SD card
HL = &4800 --> Start 1. File name of medium 'N'
HL = &6800 --> Start 1. File name of medium 'O'

Exit conditions: HL = Pointer to the name sought

Manipulated: AF, DE and HL

Description: This OS function is used to convert the number of a file into a pointer to its name. This is a file on the M4 expansion SD card.
The number of the first tagged file can be calculated with the previously described OS function 'FTBNO'. And with the OS function 'FNOFN', the address in the buffer E-RAM can be calculated from the number of the file.
When calling 'FNOFN', the buffer E-RAM of the directories of the SD card of the M4 expansion must already be banked in. Furthermore, register HL contains a pointer to the first character of the first name of the medium used. HL = &4800 is to be set for medium 'N' and HL = &6800 is to be set for medium 'O'. The number of the file you are looking for is provided in register DE.
After the return of this OS function, HL points to the first character of the searched file name in the E-RAM buffer of the M4 expansion.

Please note: Before calling this function, the buffer E-RAM of the M4 expansion must be banked in if it's not already.

Example: LD B,&7F:LD A,(M4_ERAM):OUT (C),A ;Bank in M4 E-RAM

Attention: DE must not contain the value &0000! 0 would become &FFFF!

M4: Convert SD card filename to OS format

Brief description: A filename of M4 SD card is converted to FutureOS format for filenames.

Label: FNOCN

ROM Number: M

Starting address: &FCE1

Entry conditions: A = 'N' or A = 'O' for medium N or O
HL = Pointer to the M4 SD card filename
The buffer E-RAM for the M4 SD card is present and banked in.

Exit conditions: The newly generated file name is located from REG16_6 to REG32_1 in the usual FutureOS format:
For Media N: 'N:--FileNameExt'
For Media O: 'O:--FileNameExt'
There is no user number any longer, and there is no period in front of the extension. The format follows the usual style of how filenames are displayed under FutureOS.

Manipulated: AF, BC, DE, HL and REG16_6 to REG32_1

Description: The SD card of the M4 expansion differs significantly from the usual CPC / FutureOS file names in two respects:
- There are no user numbers
- Names can contain more than eight characters before the period

This OS function 'FNOCN' converts a filename of the M4 SD card to the usual format 'MUU:FileNameExt':

M = medium (here N or O)
UU = user number (here '--')
FileName = Eight characters of the file name
ext = extension

The period present in SD card file names is omitted here.
A maximum of eight characters are used in the file name, excess characters are not copied. If the name is shorter, blanks are used for padding.
This OS function is used by the OS function 'FMD_NO'.

Please note: If the file name of the SD card is too long, only the first eight letters before the period will be copied!

M4: Remove all spaces from a filename/string

Short description: All spaces are removed from a file name of the M4 SD card (or a string).

Label: LZEN

ROM Number: M

Starting address: &FCE4

Entry conditions: HL = Pointer to the beginning of the file name

Exit Conditions: All spaces have been removed

Manipulated: AF, BC, DE, HL and the filename/string

Description: This OS function removes all spaces (&20) from a file name of the M4 SD card or from any string. The file name or string must end with the byte &00.
When using the M4 expansion SD card, file names cannot contain spaces (= character &20 = 32). This OS function eliminates the spaces and thereby shortens the length. The end is still marked by byte &00. The start address of the file name or string is not changed.

Please note: The end of the file name or string is indicated by a &00 byte. Processing ends after the first byte &00 found.

M4: Select the following E-RAM block, set DE = &4000

Brief description: The subsequent E-RAM block is selected and banked in within the maximum possible 4 MB expansion RAM. And register DE is set to &4000.

Label: EX_ERAM

ROM Number: M

Starting address: &FCE7

Entry conditions: OS variable AKT_RAM contains current E-RAM

Exit conditions: DE = &4000
AKT_RAM contains the subsequent banked in 16 KB E-RAM block

Manipulated: F, DE and AKT_RAM

Description: A maximum of 4 MB expansion RAM (E-RAM) can be connected to the CPC. Under FutureOS, this E-RAM is addressed in blocks of 16 KB, these appear between &4000 and &7FFF.
The E-RAM selection of the current E-RAM block should be stored in the OS variable AKT_RAM (&7FC4-&7FFF, &7EC4-&7EFF, ... &78C4-&78FF).
The OS function EX_ERAM reads the current value from AKT_RAM and switches to the next 16 KB block of the expansion RAMs. The following E-RAM is banked in and AKT_RAM is updated. Furthermore, register DE is set to &4000, this is the beginning of the new E-RAM block.

Please note: It's not checked whether the newly selected E-RAM block is physically available/existing.

M4: Select the following E-RAM block, set HL = &4000 / BC = &FE00

Brief description: The subsequent E-RAM block is selected and banked in within the maximum possible 4 MB expansion RAM. And register HL is set to &4000. And BC = &FE00, the M4 data port.

Label: NX_ERAM

ROM Number: M

Starting address: &FCEA

Entry conditions: OS variable AKT_RAM contains current E-RAM

Exit conditions: HL = &4000 // BC = &FE00
AKT_RAM contains the subsequent banked in 16 KB E-RAM block

Manipulated: F, BC, HL and AKT_RAM

Description: A maximum of 4 MB expansion RAM (E-RAM) can be connected to the CPC. Under FutureOS, this E-RAM is addressed in blocks of 16 KB, they are banked in between &4000 and &7FFF.
The E-RAM selection of the current E-RAM block should be stored in the OS variable AKT_RAM (&7FC4-&7FFF, &7EC4-&7EFF, ... &78C4-&78FF).
The OS function NX_ERAM reads the current value from AKT_RAM and switches to the next 16 KB block of expansion RAMs. The following E-RAM is banked in and AKT_RAM is updated. Furthermore, register HL is set to &4000, this is the beginning of the new E-RAM block. And register BC is set to &FE00, which is the data port of the M4 expansion.

Please note: It's not checked whether the newly selected E-RAM block is physically available/existing.

M4: Determine the number of entries in an M4 table of contents

Short description: The number of all sub-directories and files in the current M4 directory is determined.

Label: ZDN

ROM number: M

Start address: &FCED

Entry conditions: The buffer E-RAM for the M4 SD card exists and is banked it. An M4 directory was read.
HL = Pointer to the 1st name of the directory to be examined. Typically &4800 (medium 'N') or &6800 (medium 'O').

Exit conditions:
DE = Number of entries (SubDIRs + file names) of the examined M4 directory.

Manipulated: AF, BC, DE and HL

Description: This function determines the number of all entries in the current directory of M4 Medium 'N' or 'O'. Both, sub-directories and files are counted.

When calling the function: the M4 E-RAM (= directory buffer) must have been banked in and register HL must point to the start of the current directory to be examined.
Normally: HL must be loaded with &4800 for medium 'N' or with &6800 for medium 'O'.

After the function returns, the DE register contains the number of entries found (sub-directories + file names).

Please note: The M4 DIR buffer E-RAM must be banked in / switched on when calling this function.
Both, file names and sub-directories are counted.

M4: Convert an M4 filename to 9P3 format

Short description: A file name from an M4 directory is converted into 9P3 format, i.e. 9 character name + period + 3 character extension.

Label: KM4OS

ROM number: M

Start address: &FCF0

Entry conditions: If the source name is located in the E-RAM, it must be banked in.
HL = Source name = Pointer to the 1st character of the M4 SD card file name
DE = Target name = Pointer to 13 bytes of free memory (within page &nn??)

Exit conditions:
HL = Pointer to the 1st character of the subsequent SD card file name

Manipulated: AF, BC, DE and HL

Description: In order to be able to display the file names from the SD card of the M4 expansion under FutureOS formatted, they must be put into a defined format. This function is provided by 'KM4OS'.

The name of an M4 SD card entry is converted into the 9P3 format. An SD card entry can be up to a few dozen characters long. The 9P3 format consists of: 9 characters name, then the period, and finally: 3 characters file extension. An example is: 'File-Name.Ext'.

However, the names of sub-directories are not converted into 9P3 format, in this case only the first 13 characters are copied.

When calling 'KM4OS', HL must contain a pointer to the source name entry to be converted (in M4 SD card format). And DE points to a memory area of 13 bytes into which the new file-name is written in 9P3 format. These 13 bytes must lie within a code page (&nn??).

After the function returns, HL points to the beginning of the next file-name. This means the following name in the source directory.

Please note: Names of sub-directories are not converted into 9P3 format, only the first 13 characters are copied.