Filesystem Interpreter

The FS_Interpreter was written do test the AVR-DOS function at developing. But it can used generally to test the AVR-hardware with the drive (CF-Card, HardDisk, MMC, SD and others). It is a BASCOM-AVR program, which accepts user inputs from the serial interface (RS232) and perform the desired DOS actions. You can get an impression, what you can do with AVR-DOS. Connect your AVR with a RS232 to a Terminal-program on a PC. You can use the following listed commands. Separate Parameters with comma ",". No commas are allowed in texts. Maximum length of input line is 80 characters. Maximum length of a text is 70 characters.

Function	Command	Parameters	Remark
File System			
Init File System	**FS**	[<Partition#>]	default partition# is 1 ; use 0 for Drive without a Master Boot Record (MBR)
Size of Disk	**DISKSIZE**		KBytes
Size of free Disk Space	**DISKFREE**		KBytes
Show File System info	**FSINFO**		
Show system info of directory buffer	**DIRINFO**		
Show system info of FAT Buffer	**FATINFO**		
Directory			
Show Directory	**DIR**	[<Filename>]	wildcard supported
Print (ASCII) File	**TYPE**	<Filename>	print file sector by sector
Dump File	**DUMP**	<Filename>	
Get Filename from Directory	**DIR$**	[<FileNamePattern>]	With Parameter the first matching file is searched, without parameter the next matching file is searched
Get Length of a file	**FILELEN**	[<FileName>]	Without paramter info of last searched file is returned
Get change date/time of a file as string	**FILEDATETIMES**	[<FileName>]	Without parameter info of last searched file is returned
Get change date/time of a file as discrete time date values	**FILEDATETIMEB**	[<FileName>]	Without parameter info of last searched file is returned
Get File attribute	**GETATTR**	[<FileName>]	Without parameter info of last searched file is returned Format: 00ADVSHR (A=Archiv, D=Directory, V=Volume ID, S=System, H=Hidden, R=Read Only)

Function	Command	Parameters	Remark
Delete File	**DEL**	<Filename>	Wildcard not supported
Rename a file	**NAME**	<OldFileName>, <NewFileName>	
Change Directory	**CHDIR**	<PathName>	Change to subdirectory <Pathname> \ changes to root directory . changes to parent directory
Make new directory	**MKDIR**	<PathName>	
Remove a directory	**RMDIR**	<PathName>	Removes only empty directories
File open, read and write			
Get free file handle number	**FREEFILE**		
File Open for Input	**FOI**	<Filename> [,<File#>]	
File Open for Output	**FOO**	<Filename> [,<File#>]	
File Open for Append	**FOA**	<Filename> [,<File#>]	
File Open for Binary	**FOB**	<Filename> [,<File#>]	
Write Line to File	**WL**	<File#>, <Text>	
Write multiple Line to File	**WLM**	<File#>, <CounterStart>, <CounterEnd>, <Text>	Write Text and appending Line-number as line from Counter start to counter end
Read Line from File	**RL**	<File#>	
Put Text to File	**PUTT**	<File#>, <Text>, [<Position>]	only binary opened files (FOB) length fixed to 20, text without blanks
Get Text from File	**GETT**	<File#> [,<Position>]	only binary opened files (FOB)
Put Byte to File	**PUTB**	<File#>, <Byte> [,<Position>]	Byte is numeric, only binary opened file
Get Byte from File	**GETB**	<File#> [,<Position>]	only binary opened file
Put Integer to File	**PUTI**	<File#>, <Integer> [,<Position>]	only binary opened file
Get Integer from file	**GETI**	<File#> [,<Position>]	only binary opened file
Put Word to File	**PUTW**	<File#>, <Word> [,<Position>]	only binary opened file
Get Word from file	**GETW**	<File#> [,<Position>]	only binary opened file
Put Long to File	**PUTL**	<File#>, <Long> [,<Position>]	only binary opened file
Get Long from file	**GETL**	<File#> [,<Position>]	only binary opened file
Put Single to File	**PUTS**	<File#>, <Single> [,<Position>]	only binary opened file
Get Single from file	**GETS**	<File#> [,<Position>]	only binary opened file
File length	**LOF**	<File#>	
File position last read/write	**LOC**	<File#>	

Function	Command	Parameters	Remark
Get End of File Status	**EOF**	<File#>	0 = not EOF; 255 = EOF; Error (Invalid file#) ➔ 255
File position next read/write	**SEEK**	<File#>	
Set next read/write position	**SEEK**	<File#>, <Position>	
File Attribut	**FILEATTR**	<File#>	return file open mode
Flush file(s) to disk	**FLUSH**	[<File#>]	Without parameter all open files are flushed
Close File	**CLOSE**	<File#>	
Save SRAM to File	**BSAVE**	<Filename>, <Start>, <Length>	
Load SRAM from File	**BLOAD**	<Filename>, <Start>	Handle this command with care, if you overwrite Stack or other important system information the program will crash and can produce unwanted results even on the disk
Show System Info of an opened file	**FILEINFO**	<File#>	
Drive/Card and sector			
Show Compact Flash Identity	**CFI**		Dump 512 Bytes of CF-Card Identity Information
Reset CF Card	**CFRESET**		
Init CF Card	**CFINIT**		
Read Master Boot Record	**MBR**		Show sector 0 (MBR) and partition table
Dump Sector	**SD**	<First Sector#> [,<Last Sector#>]	Dump 512 Bytes of Sector, without second parameter only one sector is dumped; last show sector is stored in Buffer; at sectors written with SWM the sector# is shown too
Write Sector (from Buffer)	**SW**	<First Sector#>, <Count of Sectors> [,<SRAM Address<]	Without parameter <SRAM Address> buffer is written
Write multiple sectors	**SWM**	<First Sector#>, <Count of Sectors> [,<Start-Byte>]	For speed test; Write <count of sectors> starting at <first sector#> with a bit pattern (increasing byte values) starting with <Start-byte> Last four bytes of sector contains sector# as long binary value preceeding by a ":" On system with a running clock the time for one sector is shown.
Read multiple sector	**SRM**	<First Sector>, <Count of Sectors>	For speed test; On system with a running clock the time for one sector is shown.

Function	Command	Parameters	Remark
SRAM Buffer and ERAM			
Show SRAM (Memory)	**MD**	[<SRAMStart>] [,<SRAMEnd>]	Dump SRAM Memory, without <SRAMStart> the buffer is shown.
Show ERAM (EEPROM)	**ED**	<ERAMStart> [,<ERAMEnd>]	Dump ERAM Memory
Adjust buffer pointer	**MP**	<MemoryPointer>	Adjust the SRAM- Buffer Pointer for next writing with MB, MT or MF The actual Memory-Pointer is shown with the Prompt
Adjust ERAM pointer	**EP**	<ERAMPointer>	Adjust the ERAM- Pointer for next writing with EB, ET or EF The actual Memory-Pointer is shown with the Prompt
Write single byte(s) to buffer	**MB**	<Byte1> [,<Byte2>] [,<Byte2>] ..... [,<Byte8>]	Write single Bytes to the Buffer. For <Byte> the ASCII –Code must be typed. For exampe 65 or $41 for Letter „A“. Writing to Buffer starts at address of MemoryPointer. Up to 8 Bytes are possible with one MB command.
Write single byte(s) to ERAM	**EB**	<Byte1> [,<Byte2>] [,<Byte2>] ..... [,<Byte8>]	Write single Bytes to the Buffer. For <Byte> the ASCII –Code must be typed. For exampe 65 or $41 for Letter „A“. Writing to ERAM starts at address of ERAMPointer. Up to 8 Bytes are possible with one MB command.
Fill Buffer with multiple bytes	**MF**	<Byte> [,<BufferStart>] [,<BufferEnd>]	Fill the Buffer with <Byte> from <BufferStart> to <BufferEnd>. For <Byte> the ASCII –Code must be typed. For exampe 65 or $41 for Letter „A“. Without <BufferStart> and <BufferEnd> the buffer from the MemoryPointer in the Buffer to End of the Buffer is filled. You can start also filling at <BufferStart> and end at <BufferEnd>.
Write Text to Buffer	**MT**	<Text>	Fill the SRAM- Buffer with <Text> starting at the position of the memorypointer.
Write Text to ERAM	**ET**	<Text>	Write Text to internal EEPROM starting at ERAM-Pointer; check command EP

Function	Command	Parameters	Remark
Others			
Print last Error code	**ERROR**		
Show and change date	**DATE**	[<Date>]	Use date format configured in the Config Date statement
Show and change time	**TIME**	[<Time>]	
Reset the AVR	**RESET**		restart program from 0